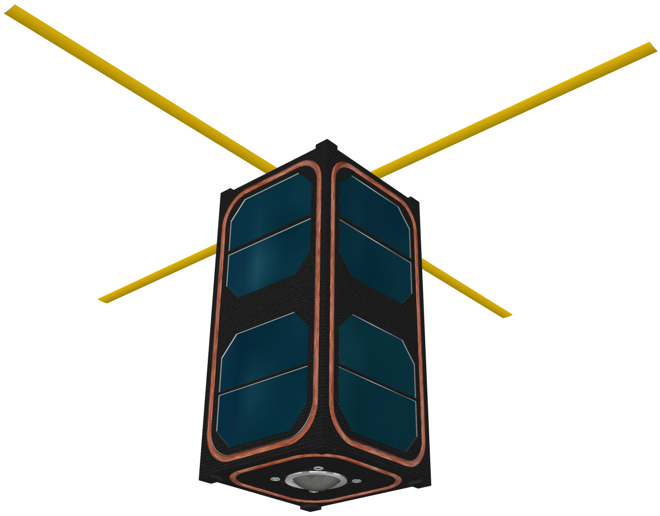

Satellite Attitude Control System

Fredrik Alvenes

Department of Engineering Cybernetics

2012

Abstract

This reports gives an introduction to attitude control of the NTNU Test Satellite (NUTS). It gives an insight to mathematical modeling of satellite attitude dynamics for 3 degrees of freedom. By the different limitations of how the NUTS operates, these models are adjusted accordingly.

A set of strategies for controlling the attitude is presented. Through an explanation of the magnetic actuators, the control laws are also adapted to work with the NUTS satellite.

Combined, these findings are put to use in the form of a complete Simulink simulator for the satellite in orbit. Results with different control strategies and environmental models are given. Some thoughts of utilizing the control algorithms as part of the final embedded computer system are pointed out throughout the text.

Preface

This project thesis is the result of the work done in TTK4550 at the Norwegian University of Science and Technology (NTNU), Department of Engineering Cybernetics. I would like to thank my supervisor, Thor Inge Fossen for guidance and support. The enthusiasm in the NUTS project, and especially Roger Birkeland, has been greatly appreciated. Thank you for letting me be a part of your team. Gaute Braathen on the NUTS-ADCS team has been most motivating and helpful. I wish him the best of luck on his Master Thesis.

Also, a big thank you to my brother Karl for his companionship between my studies, and to Tina for always believing in me.

The NUTS is planned to launch by the year 2014, and our work will reach the final frontier.

List of Figures

Contents

Chapter 1

Introduction

1.1 Motivation

The NUTS project has been an ongoing project for two years. During this time many Project- and Master-Thesis's has been written evolving several parts of the attitude controller and ADCS. Not only the controller software and algorithms, but also different kinds of sensors, thrusters and embedded platforms.

The goal of this thesis is to do a study evolving the NUTS satellite project. This includes learning the basics of how a satellite operates in space. For example satellite orbits, environmental factors, coordinate systems, limits of the actuator system and avoid singularities with the use of quaternions. All of this can be both interesting and challenging with a background from earth based control systems.

1.2 Outline of the thesis

From the study different nonlinear controllers will prepared. This will be the basis for further studies and extensions, that can include more advanced implementations, like attitude estimation.

1.3 The NUTS satellite

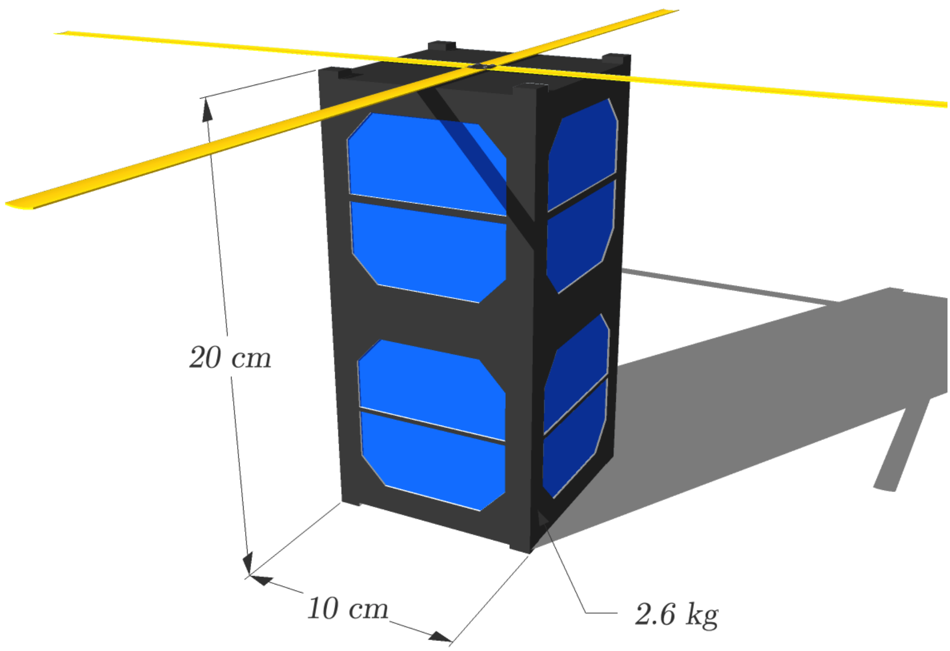

Figure 1.1: CAD model of the NUTS satellite

The NUTS satellite (NTNU Test Satellite) is a continuation of two previous attempts by NTNU to launch a space vehicle. The first project never made it to space do to a failure in the launch rocket. The second attempt made it to space, but no radio signals where ever recorded, and the status of the project is still "unknown". The Department for Electronics and Telecommunications (IET) now aims for a third try with a launch window within 2014.

The satellite follows a standard called CubeSat. This is a low-cost alternative for space related research. The idea is to build a Satellite of 10x10x10 cm (a volume of exactly one liter) with a maximum weight of 1.33 kilograms. The NUTS satellite is however twice the length and weight, a so called 2 unit (2U) CubeSat. Like stacking two CubeSats on top of each other. Because of this standard, it is possible for a "mother-rocket" to carry several CubeSats, releasing them from $10x10$cm chambers.

The orbit used will be a *sun synchronous orbit*; going across the globe from crossing the poles.

- The satellite will go in orbit from the North Pole to the South Pole
- The satellite will be over the same location on Earth at the exact same solar time (making it easier to collect data via radio signals, or make observations like pollution from morning rush-trafikk over a specific city).
- The plane created by the orbit will always face the sun with same *inclination angle.*
- The amount of sunlight hitting the satellite (and it's solar panels) can be accurately calculated ("daylight" between $50 - 100\%$ of the time

during one orbit).

- One orbit will take ≈ 5800 seconds

The main mission of the satellite to analyze gravity with the help of an infrared camera. This camera will be mounted on the bottom of the NUTS, always pointing towards Earth. Without an attitude controller making sure this orientation is held, the mission will be completely compromised.

1.4 Attitude controllers

An ADCS (Attitude Determination Control System) gives a sattelitte 3 degrees of freedom (3DOF) in space and is a very common system to include on satellites. Because the NUTS needs specific observation from Earth's surface, an attitude controller is absolutely necessary. Bigger and more expensive satellites uses 4DOF (surge velocity control) for the ability to either maintain or change it's orbit. The International Space Station (ISS) is one example.

As with any Cybernetic task, there are several approaches for an attitude controller. In scientific papers and previous master thesises, most solutions uses a combination of nonlinear controllers. Linearizations attempts have also been made, but is mostly dependant on a satellite quite near it's desired attitude.

State estimation is also an important part of the ADCS. Measurements can be noisy, discontinuous and biased. However, this is out of the scope of this project thesis, and will be left for a possible future review.

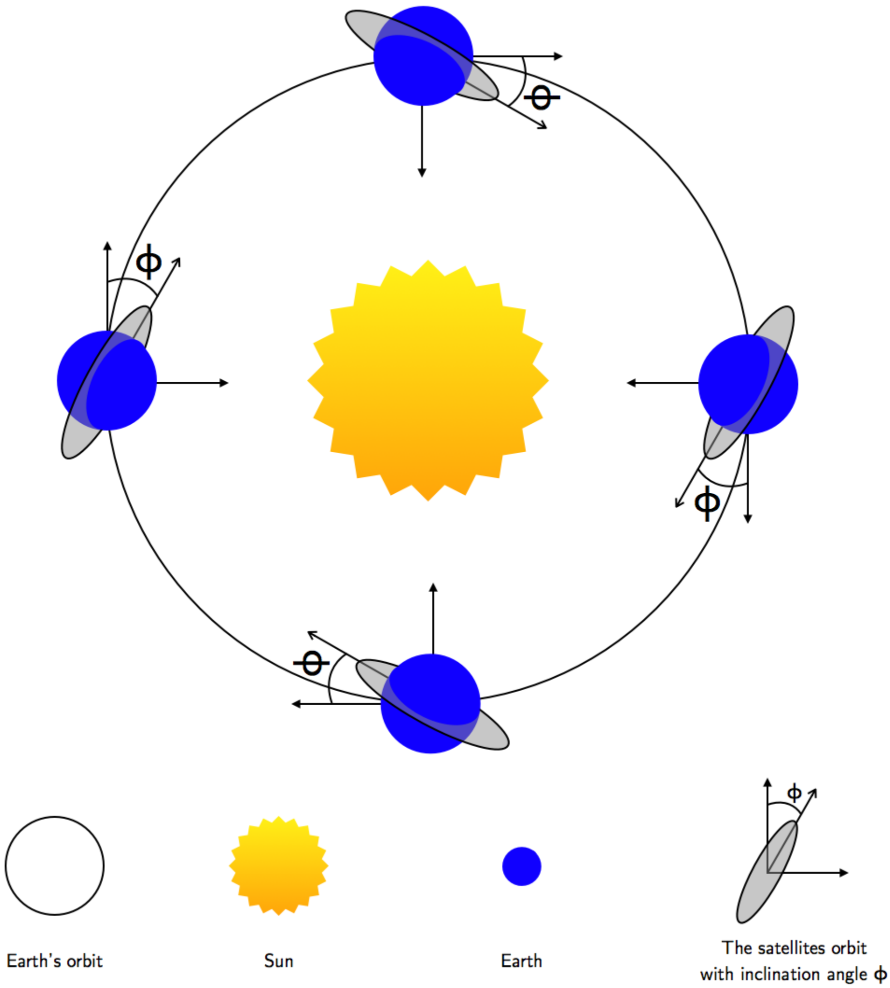

Figure 1.2: Sun synchronous orbit

Chapter 2

Theory and definitions

This chapter aims to give an overview of different mathematical theory and definitions used to model the satellite and the controller. Here is also an overview of sensor systems.

2.1 Coordinate Frames

Three coordinate frames are used for the satellite:

- **Earth-centered inertial (ECI)** Origin at the centre of Earth, z-axis pointing out of the north-pole, x and y fixed in space. Non accelerated frame.

- **Earth-centered Earth-fixed** Origin at the centre of Earth and the z-axis pointing out of the north pole. x and y rotates with Earth at $\omega_{Earth} = 7.2921 \cdot 10^{-5} rad/s$.

- **North-East-Down (NED)** Origin at the Earth's reference ellipsoid (WGS 1984) and z-axis pointing towards the Earth's centre (nadir direction). x and y follows the tangent lines on Earth's ellipsoid pointing towards true north and east respectively.

- **Orbit**, this frame is located at a distance $r_{Earth} + h_{altitude}$ from the centre of ECI at the centre of the satellite. It's z-axis always points towards the centre of ECEF and the x-axis points in the velocity direction. It rotates around the Earth at $\omega_{orbit} \approx \sqrt{\frac{GM}{r^3}}$.

- **Body-fixed (BODY)** Origin at the center of mass in the satellite, where x, y and z moves and rotates with the satellite. y is the axis of maximum inertia, see chapter 4. z is the axis of minimum inertia (where the camera points out). x is defined as direction of travel.

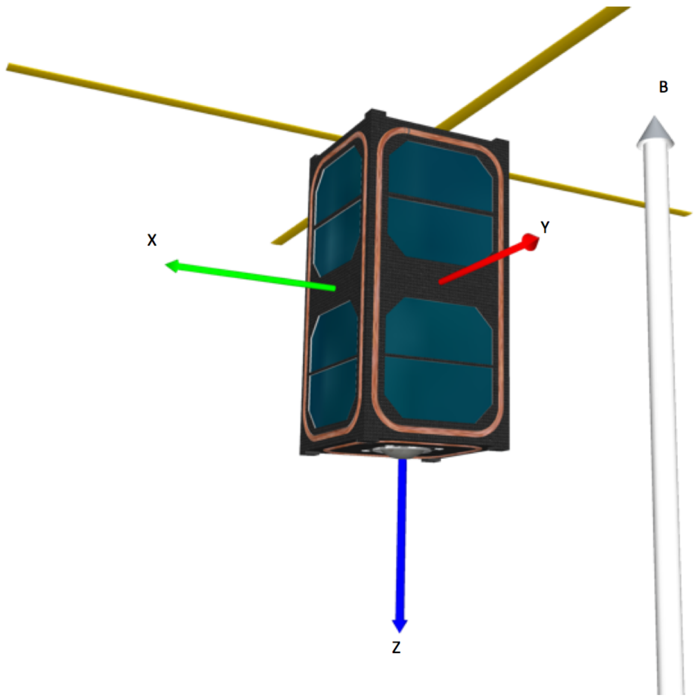

Figure 2.1: NUTS with BODY coordinate system and magnetic field B

2.2 Magnetic field

The satellite will be falling around the Earth's magnetic field at an altitude of around 600 kilometers. At a larger scale it is possible to look at this magnetic field as a dipole. This means that one basically consider the Earth as a large dipole magnet, with the magnetic vector field exiting the south pole and entering the north pole. Simple as it may seem, this interpretation is has proven very useful over hundred of years for navigating with instruments like a compass and a printed map.

The NUTS satellite will not only use the magnetic field for navigation, but also utilize it as medium for actuation (see section 2.3). For this purpose one should instead look at a more detailed model then a dipole. Every five years the International Association of Geomagnetism and Aeronomy updates their International Geomagnetic Reference Field (IGRF). It is constructed using data from different satellites and geomagnetic observatories. The calculation is based on:

$$V(r, \phi, \theta) = a \sum_{\ell=1}^{L} \sum_{m=0}^{\ell} \left(\frac{a}{r}\right)^{\ell+1} \left(g_\ell^m \cos m\phi + h_\ell^m \sin m\phi\right) P_\ell^m \left(\cos\theta\right) \quad (2.1)$$

This is the Gauss coefficients defining a spherical harmonic expansion of the magnetic scalar potential.

r Radial distance from the Earth's center

L Maximum degree of the expansion

ϕ East longitude

Θ Polar angle

a Earth's radius

g_ℓ^m, h_ℓ^m Gauss coefficients

Where the new and updated coefficients are published each 5 years, making it easier to update implementations. With the Aerospace Toolbox in Matlab, this can be calculated with

```
[mag_field_vector, hor_intensity, declination, ...
    inclination, total_intensity, mag_field_sec_variation, ...
```

```
    sec_variation_horizontal, sec_variation_declination, ...
    sec_variation_inclination, sec_variation_total] = ...
    igrf11magm(height, latitude, longitude, decimal_year)
```

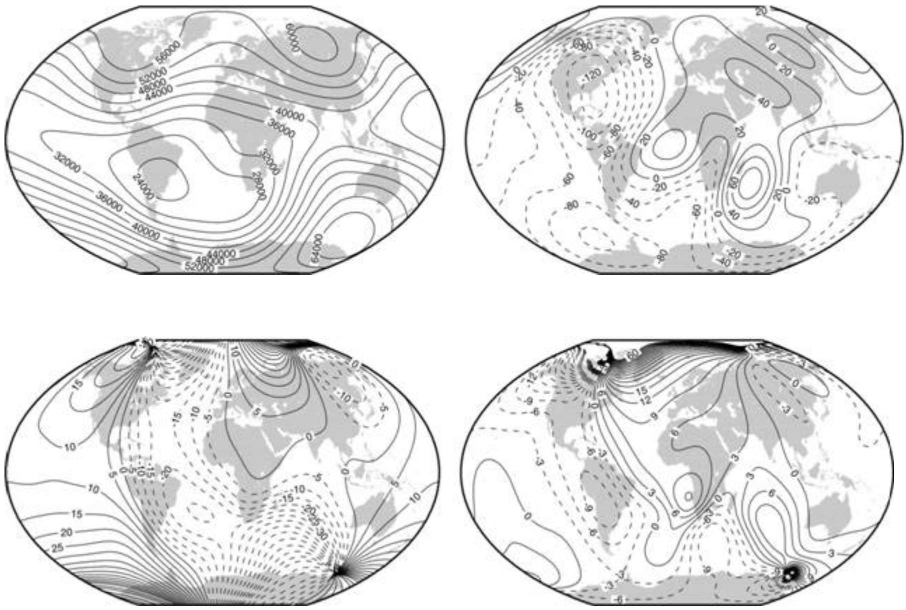

Figure 2.2: IGRF magnitude *Image courtesy of British Geological Survey*

As one can observe from the the figures, it is quite clear that the Earth's magnetic field is far from a standard dipole. Both regarding the magnetic force, and the field itself. This is important, as the satellite's ADCS system is bound to know the resultant vector from the magnetic field in order to produce a suitable current for the coil actuators (see section 4.4).

There is a main practical difference between a dipole field and the IGRF in low earth orbit (160 - 2000 km). If the Earth had a the magnetic field of a perfect dipole magnet, the field would have zero magnitude in the orbit-frame y-axis. x and y would have almost perfect sinusoidal values with period of one orbit (try to picture this as a satellite moving around a dipole like figure 2.4). Contrary, the real IGRF field has somewhat larger variations of magnitude in the x- and y-axis, and even the z-axis has a varying magnitude.

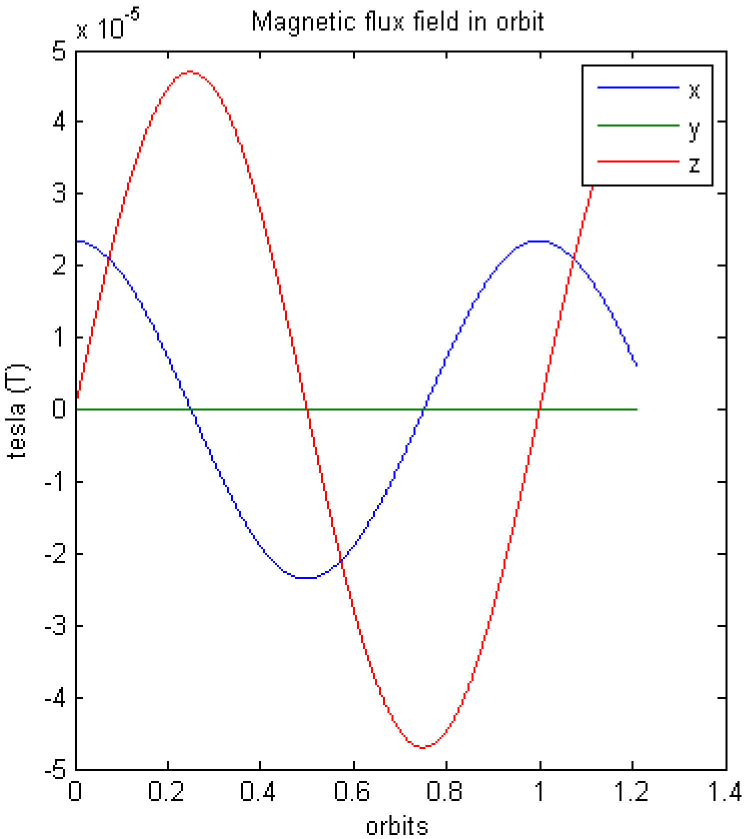

Figure 2.3: Magnetic flux from Earth when modeled as a dipole. Measured in the orbit coordinate system

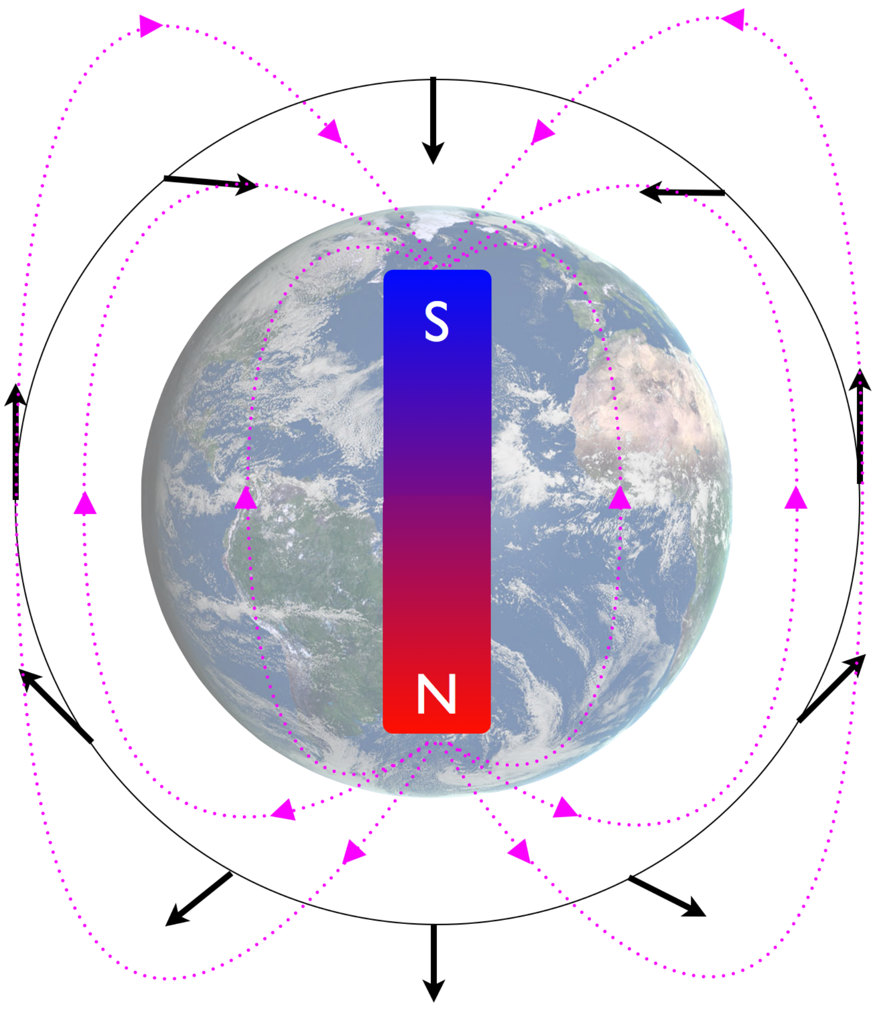

Figure 2.4: Earth as a dipole with experienced magnetic flux in orbit (black) and total magnetic flux field (pink)

2.3 Magnetic coils and allignment

Larger, commercial, or military satellite uses different kinds of actuators to move in space. Some satellites even have so powerful actuators that they can correct orbit and change their altitude. There also exists so called ”grave-yard orbits” where satellites, by the help of powerful thrusters, can be brought out when they are otherwise not functional. This can be realized by using for example monopropellant hydrazine thrusters.

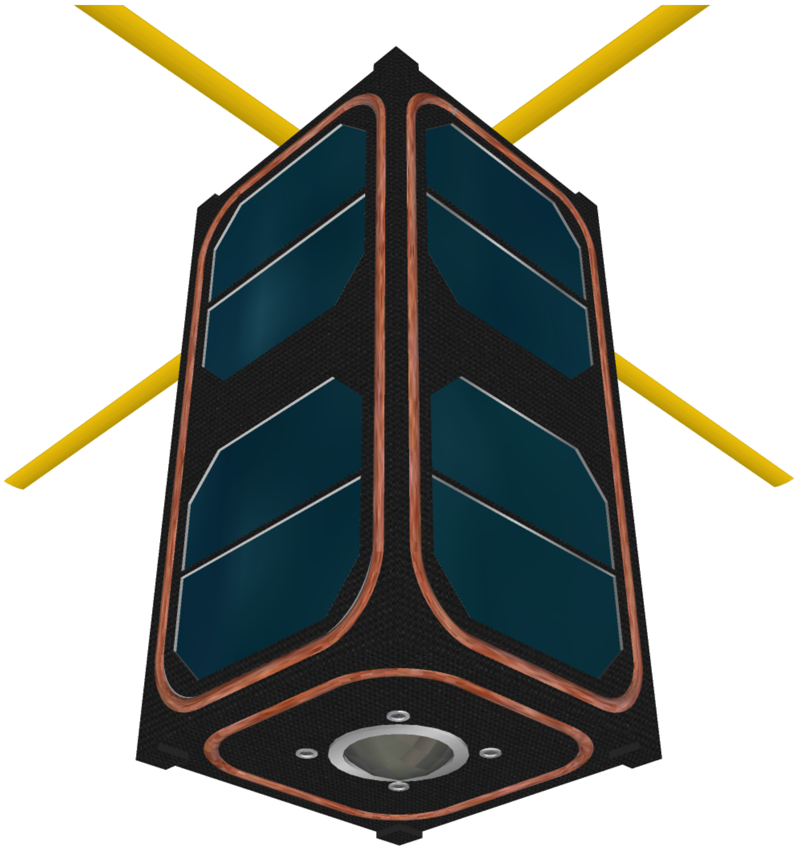

Figure 2.5: The three electromagnetic coils exposed

The NUTS satellite is a very small satellite in the CubeSat family, and therefore it has limited storage space for actuators. Large hydrazine thrusters is definitely off the table, unless the sole purpose of the satellite is to test such a system. Other variants include ion-thrusters, torque wheels and electro-magnetic coils. The primary actuator for CubeSats is often torque-wheels with magnetic coils as a backup. The NUTS satellite will however use magnetic coils as the main and only actuators. This is chosen for several reasons. Primarily it is very simple and reliable. The footprint is also far smaller then any other system, combined with a very low power consumption.

The main drawback is only a limited amount of torque can be produced in the Earth's magnetic field. This is the cause of many factors, and a more mathematical approach can be found in section 4.4.

The coils on the NUTS will be realized by spinning copper thread inside the frame, making three coils normal to the BODY's x, y and z axis, as seen in figure 2.5. The magnetic field generated by the thrusters will be given in tesla (T) by the formula:

$$m_b = \frac{N \cdot A}{R} \cdot V \tag{2.2}$$

- **N** number of turns with copper in the coil
- **A** area the coil is covering (≈10x20 and ≈10x10 cm)
- **V** voltage over the coil
- **R** resistance in the coil

2.4 Sensors for attitude estimation

At the time of this writing there is still discussion on the team for what kind of sensors that should be included for attitude estimation. The following options are given:

- **Gyroscope**

  Most likely included, but has bias and should be used together with a state estimator or filter.

- **Accelerometer**

  Will give relatively low measurement values in orbit (≈zero local gravity) that might only produce a noisy signal with very little amplitude.

- **Magnetometer**

  The Honeywell HMC5983 has already been chosen as part of the final hardware assembly. Using the magnetometer requires good knowledge on Earth's magnetic field (like an onboard IGRF look-up-table (LUT)). It also requires testing of the radio-communication system and other onboard electronics, in order to estimate biases and noise affecting this instrument.

- **Commercial Sun sensor**

  A sun sensor can be find the direction-vector to the Sun. It can be thought of as a replacement for an accelerometer. The assembly is quite simple (shelf component), but might occupy too much area on the outside of the NUTS, otherwise used for solar panels.

- **Sun sensor based on solar panel measurements**

  Martin Nygren on the NUTS-ADCS team is currently writing his Project Thesis based on this research.

- **Military grade GNSS (GPS)**

A GNSS system can be used to calculate the magnetic field in orbit, if the satellite carries an onboard IGRF-LUT based on for example GPS-coordinates. The GPS system is however not intended for civil space applications. A request would probably be bureaucratic and in economically challenging for the NUTS project (prices of $\approx$ 10.000 USD).

A good sun-sensor and a gyroscope should be enough for attitude estimation, with the proper filtering. The controller is also dependent of a measurement of the magnetic field, and for this a magnetometer is chosen. The final hardware assembly is yet to be determined by the NUTS project. However, the simulator used in this thesis will rely on full state feedback from the mathematical model. To mimic the output of the magnetometer, noise (based on the HoneyWell data sheet [11]), can be added.

Chapter 3

Hardware platform

Through this Project Thesis it was always considered an option to make the controller algorithm available on an embedded platform. I study was done to investigate different options, that would suit the workflow of the development.

In the recent years many different low cost hardware modules has reached the marked. Some of these are complete System on a Chip (SoC) solutions. Examples are the famous Raspberry Pi, which consists of an 1GHz ARM CPU, 256MB of RAM, WiFi, USB and network controllers. It even has an HDMI output, allowing you to run a full-sized 1080p media center out of the box. All of this at market prize of 35 USD, and it consumes less then 3.5 watts of power.

Other SoC solutions includes the The Panda Board, The Beagle Board and the Norwegian (Trondheim based) Cotton Candy. All of these products are suitable for extensive implementation of Attitude controllers. However, the main purpose of the attitude controller is now to be fully implemented as part of the NUTS satellite project. The hardware has already been chosen as the Atmega2561. This, combined with the very low footprint and resource friendly implementation, the choice was made for an Arduino 2560. The Arduino mega is well known, trusted, tested and open source. It evolves around Atmega 2560, which is essentially the same chip used in the NUTS project, only it has more I/O pins. The specification is a 16 MHz 8-bit AVR micro controller, which in turn, should be more then sufficient to run the attitude controller.

3.1 Simulink Coder

Simulink Coder, formally known as Real-Time Workshop, is a Toolbox for the Matlab Simulink environment. Students at NTNU have previously been given very limited access to this toolbox. But starting from the 2012 autumn semester, it is now part of the Matlab License accessible to all students.

Simulink Coder, in the very basic, is a tool to help you realize Simulink diagrams as C or C++ code. It has built in support for the Lego Mindstorms NXT, Beagleboard and ArduinoMega platforms. By installing a few extra drivers, a complete Simulink-modelled-system can run on these devices with a click of a button. If you where to use the code on different targets, there are options to generate C-code that can later be modified and used in for example Atmel Studio. Practically it means that whatever results you can get from Simulink should be a lot more accessible for later development. The group that will assemble and program the satellite before launch, might not be too familiar with Matlab- or Simulink-code. Probably they will have a background with more computer programming, and C-code is a very universal language in their domain.

A lot of good and thorough work has already been done by students, by simulating ADCS systems in Matlab. Some embedded boards has also been custom made. By utilizing the Simulink Coder toolbox, it is now possible to work directly with Simulink diagrams on the Atmel Atmega2560 (part of the Arduino Mega Kit) that will be used on the satellite going to space.

3.2 Hardware and software specifications

- **ATmega 2561** The code must be able to run on this micro controller;

  8-bit AVR

  16 MHz

  256 Kbytes Flash

- **32-bit floating point precission**

  This is a limitation to both the Arduino and the 8-bit AVR.

  The controller will store decimal values within 32-bit (4-byte).

  Precision is not maintained.

  "Floats" demand a lot more CPU-time compared to integers.

Practically, "floats" should be avoided as much as possible.

- **Discrete algorithms** All algorithms must be discrete, like for example a Kalman filter.

In this particular instance, the code will be generated directly from Simulink. When the code is compiled and ran directly within Simulink, by using the *Run on target* feature, there is very little control to what Simlink actually outputs. This actually works like any other simulation done in Simulink, except the Arduino is chosen in a drop down menu. Performance is simply done by observing how fast the system is running in real-time, to act as an estimate on how well the code performs.

The other alternative is to compile the code (whole system or "block by block") with certain parameters and code specifications in Simulink. Then there will be full access to the code, and this can be implemented in a bigger system (like the final NUTS satellite). One can use any IDE (integrated development environment) of choice that supports C/C++. Most likely Atmel's own tools, known as Atmel Studio, will be used to prepare the final code on the satellite. As this supports all AVR-hardware on all levels.

As with all programming routines, it is important that the code is readable/understandable for a third party. Good documentation and logically named functions and values is mandatory. The code should also be as modular and scalable as possible. After the code is generated from Simulink, it is therefore important to review the code before handing it off to a third party for implementation. Simulink does however provide fairly good starting point with it's exported code. Although out of the scope of this thesis, it is worth mentioning that the ADCS system cannot be delivered as a plug-and-play solution directly on the satellite. This has to do with how the hardware and software is coupled together.

Chapter 4

Attitude Model

By incorporating theory from Fossen [2], Hughes [5] and Tudor [3] one can derive an attitude model for the satellite. Usually one would use for example Fossen's vectorial setting, but the satellite only has three degrees of freedom (3 DOF), and the forces acting on the body are fare fewer then for example a 6 DOF marine craft (at least in our simplification). The resemblance is still high, and the model can be interpreted as a simplified version.

4.1 Rigid body kinetics

$$\boldsymbol{I}_{CG}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\boldsymbol{I}_{CG}\boldsymbol{\omega}) = \boldsymbol{\tau}_m + \boldsymbol{\tau}_g + \boldsymbol{\tau}_a \tag{4.1}$$

$\boldsymbol{\omega} = \begin{bmatrix} p & q & r \end{bmatrix}^\top$ is the angular velocities in roll, pitch and yaw.
$\boldsymbol{I}_{CG} = \boldsymbol{I}_{CG}^\top > 0$ is the inertia tensor about the centre of gravity. When we assume homogenous displacement of mass along the principal axises, we can reduce it to a diagonal matrix:

$$I_{CG} = \begin{bmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{xy} & I_y & -I_{yz} \\ -I_{xz} & -I_{yz} & I_z \end{bmatrix} = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \tag{4.2}$$

Since the NUTS satellite is a 2 unit (2U) CubeSat, one can use the following inertia tensor for a cuboid (with the body frame located at the center):

$$I_{CG} = \begin{bmatrix} \frac{1}{12}m(h^2+d^2) & 0 & 0 \\ 0 & \frac{1}{12}m(w^2+d^2) & 0 \\ 0 & 0 & \frac{1}{12}m(w^2+h^2) \end{bmatrix}$$

According to Tudor [3] it is important that $I_y > I_x > I_z$ in order to maintain the equilibriums of the satellite in the magnetic field (it is the result of how the gravity will effect the moment of inertia). This has been thoroughly stressed to the team in charge of the frame and component assembly.

By using a skew-symmetric matrix $\boldsymbol{S}(\boldsymbol{\omega}) = -\boldsymbol{S}(\boldsymbol{\omega})^\top = \begin{bmatrix} 0 & I_z r & -I_y q \\ -I_z r & 0 & I_z p \\ I_y q & -I_x p & 0 \end{bmatrix}$ the kinetics can be rewritten:

$$\boldsymbol{I}_{CG}\dot{\boldsymbol{\omega}} + \boldsymbol{S}(\omega)\omega = \boldsymbol{\tau}_m + \boldsymbol{\tau}_g + \boldsymbol{\tau}_a \tag{4.3}$$

4.2 Kinematics

$$\dot{\boldsymbol{q}} = \boldsymbol{T}_q(\boldsymbol{q})\boldsymbol{\omega}^b_{b/n} \tag{4.4}$$

To represent the attitude we use unit quaternions $\boldsymbol{q}$. It represents attitude with a vector containing one real part η and three imaginary parts $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2, \epsilon_3]^\top$. Together $\boldsymbol{q} = [\eta, \epsilon_1, \epsilon_2, \epsilon_3]^\top$. It also satisfies $\boldsymbol{q}^\top \boldsymbol{q} = 1$.

The reason for using unit quaternions is to avoid singularities in the rotational matrices. This is a must have since one have to assume the satellite can take any arbitrary attitude.

By using the angular velocity transformation from chapter 2 in [1]:

$$\boldsymbol{T}_q(\boldsymbol{q}) = \begin{bmatrix} -\epsilon_1 & -\epsilon_2 & -\epsilon_3 \\ \eta & -\epsilon_3 & \epsilon_2 \\ \epsilon_3 & \eta & -\epsilon_1 \\ -\epsilon_2 & \epsilon_1 & \eta \end{bmatrix}, \quad \boldsymbol{T}_q^\top(\boldsymbol{q})\boldsymbol{T}_q(\boldsymbol{q}) = \frac{1}{4}\boldsymbol{I}_{3x3} \tag{4.5}$$

A deeper explanation on quaternions can be found in for example Fossen [1] and Vik [13].

4.3 Environmental torques

The satellite is exposed to several environmental torques on its path around Earth. The most dominant ones is gravity from Earth and air-resistance from the upper layers of the atmosphere. The satellite will also be affected by gravity from both the moon, the sun and all other objects in the universe. "Solar wind/pressure" is also part of the big equation. But apart from earth-gravity and atmospheric-drag, other environmental torques are estimated to be small enough to neglect, within the specifications of how robust an attitude controller should be.

According to [3] the gravity vector $\boldsymbol{\tau}_g$ can be modeled as:

$$\boldsymbol{\tau}_g = 3\left(\frac{\mu}{R_c^3}\right) z_o \times \boldsymbol{I} \cdot z_o \tag{4.6}$$

μ Earth's gravitational constant multiplied with the Earth's mass

R_c^3 distance from the Earth centre to the satellite mass

z_o z-axis in the orbit frame

A decision has also been made to include a disturbance based on the atmospheric drag. This has limited coverage in previous master thesis's regarding the attitude control system. Aerodynamic drag on satellites is not necessarily an intuitive concept, unless one is familiar with space technology.

Rawashdeh et al. [6] shows simulations that a 3U CubeSat can be stabilized with "fins" for altitudes below 450km. However this can reduce the lifespan of the satellite, because it slows it down in orbit. It is important to remember that the aerodynamic drag sets the absolute limit on the lifecycle for any satellite without heave/altitude compensation (like the NUTS satellite). The orbit speed will eventually decrease so much, the satellite will fall towards Earth an burn up in the atmosphere. An estimate for the NUTS satellite is 4 years at an altitude of around 600km. At these altitudes the drag is however considerably lower then the ones simulated in [6]. According to Wertz and Wiley [12] the particle density more then 10 times higher at altitudes of 450km compared to 600km ($\approx 1.13 \times 10^{-12} {}^{kg}/_{m^3}$ versus $\approx 1.04 \times 10^{-13} {}^{kg}/_{m^3}$). The aerodynamic drag is therefore most important considering the surge velocity. But it also has an effect on the satellite's attitude.

Hughes [5] suggests an expression for the aerodynamic drag;

$$\boldsymbol{\tau}_a = \rho_a V_{orbit}[V_{orbit} A_{drag} c_p \hat{\boldsymbol{V}}_{orbit} - (\boldsymbol{I}_{CG} + \hat{\boldsymbol{V}}_{orbit} \boldsymbol{J})\boldsymbol{\omega}^b_{ob}] \tag{4.7}$$

ρ_a density of the atmosphere in ${}^{kg}/_{m^3}$

V_{orbit} magnitude of orbit velocity vector (constant)

A_{drag} projected 2-dimensional surface area facing the velocity direction

c_p centre of pressure

$\hat{\boldsymbol{V}}_{orbit}$ unit velocity direction

$\boldsymbol{I}_{CG}$ moment of inertia

$\boldsymbol{J}$ new moment of inertia for drag

The idea is basically to create a new moment of inertia matrix $\boldsymbol{J}$ that is slightly off centre from the normal $\boldsymbol{I}$. This will contribute to the moment of inertia when the satellite is spinning. Aerodynamic theory also suggests that the centre of pressure might not be aligned with the centre of mass, generating a torque even when the satellite is not spinning. The constant $\boldsymbol{\tau}_a$ is simulated as a "worst case" scenario, with the largest possible area facing the surge direction. That in turn gives the highest possible disturbance torque from the atmospheric drag (based on Hughes expression (4.7)). It can also be seen that if the satellite spins, the torque will be higher.

4.4 Actuator dynamics

The choice of magnetic actuators for the NUTS satellite has several advantages. It is cheap, solid-state and uses very little power. Since the coils can be winded around the frame components, it virtually does not take up any space. Weight is overall not considered an issue with any of the components, since the satellite frame is relatively small given the weight limitations. There is simply not enough space to fit 2.6 kg of components.

On the other hand, modeling the dynamics and control algorithm can be challenging, since the magnetic field where the satellite is moving changes constantly. An analogy could be an air-plane moving through a various-density atmosphere, suddenly leaving you with limited control of either roll, pitch or yaw.

The satellite falls around the Earth surrounded by a changing magnetic flux field in the body frame $\boldsymbol{B}_b = [B_x, B_y, B_z]^\top$. In order for the satellite to change it's attitude, it generates a magnetic field of it's own with one of the three electric coils:

$$\boldsymbol{m}_b = \begin{bmatrix} m_x \\ m_y \\ m_y \end{bmatrix} = \begin{bmatrix} \frac{N_x A_x V_x}{R_x} \\ \frac{N_y A_y V_y}{R_y} \\ \frac{N_z A_z V_z}{R_z} \end{bmatrix} \tag{4.8}$$

$\boldsymbol{m}_b$ magnetic field in the BODY axis/coordinate system

N number of turns with copper thread in the coil

A area of the coil

V voltage over the coil

R resistance in the coil

The torque produced is given as the cross product of $\boldsymbol{m}_b$ and $\boldsymbol{B}_b$;

$$\boldsymbol{\tau}_m = \boldsymbol{m}_b \times \boldsymbol{B}_b = \boldsymbol{S}(-\boldsymbol{B}_b) \cdot \boldsymbol{m}_b = \begin{bmatrix} 0 & B_z & -B_y \\ -B_z & 0 & B_x \\ B_y & -B_x & 0 \end{bmatrix} \begin{bmatrix} \frac{N_x A_x V_x}{R_x} \\ \frac{N_y A_y V_y}{R_y} \\ \frac{N_z A_z V_z}{R_z} \end{bmatrix} \tag{4.9}$$

This relation can be observed from figure 4.1. Her is Earth's magnetic field $\boldsymbol{B}^b$, the magnetic field set up by the actuators $\boldsymbol{m^b}$ and the resulting cross product of the two as the torque $\boldsymbol{\tau}_m$. Remember that the torque vectors must not be interpreted as force vectors. This is torque *around* the vector. The vector $\boldsymbol{\tau}_d$ will be further explained in section 4.5.

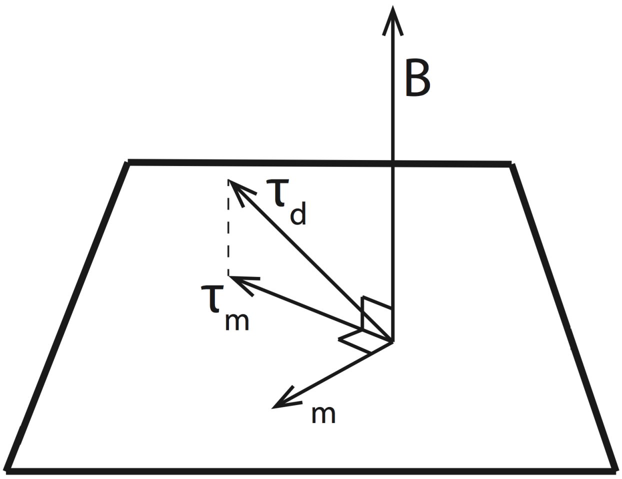

Figure 4.1: Magnetic and torque vector relations - *Image courtesy of [3]*

From figure 4.1 it can be observed that $\boldsymbol{m}_b$ can be derived as the following crossproduct:

$$\boldsymbol{m}_b = \boldsymbol{\tau}_m \times \boldsymbol{B}_b \tag{4.10}$$

By writing the full expression for $\boldsymbol{m}_b$, a *control allocation matrix* based on the voltage $\boldsymbol{V}$ can be derived;

$$\begin{bmatrix} \frac{N_x A_x V_x}{R_x} \\ \frac{N_y A_y V_y}{R_y} \\ \frac{N_z A_z V_z}{R_z} \end{bmatrix} = \boldsymbol{\tau}_m \times \boldsymbol{B}$$

Now separating the the voltage vector

$$\begin{bmatrix} \frac{N_x A_x}{R_x} & 0 & 0 \\ 0 & \frac{N_y A_y}{R_y} & 0 \\ 0 & 0 & \frac{N_z A_z}{R_z} \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = \boldsymbol{\tau}_m \times \boldsymbol{B}$$

By solving the expression for $\boldsymbol{V}$, the control allocation matrix $\boldsymbol{K}_{coil}$ is found

$$\begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = \begin{bmatrix} \frac{N_x A_x}{R_x} & 0 & 0 \\ 0 & \frac{N_y A_y}{R_y} & 0 \\ 0 & 0 & \frac{N_z A_z}{R_z} \end{bmatrix}^{-1} \boldsymbol{\tau}_m \times \boldsymbol{B}$$

$$\boldsymbol{V} = \boldsymbol{K}_{coil}^{-1}(\boldsymbol{\tau}_m \times \boldsymbol{B}) \tag{4.11}$$

This is done because a real world satellite only accepts voltage signals as inputs, not torque. Keep in mind that $\boldsymbol{B}$ will change throughout the orbit. On the contrary, $\boldsymbol{N}$, $\boldsymbol{A}$ and $\boldsymbol{R}$ are constants that will be final after the assembly of satellite.

By using this in equation 4.9, an expression for $\boldsymbol{\tau}_m$ can also be found[1];

$$\boldsymbol{\tau}_m = \begin{bmatrix} 0 & B_z & -B_y \\ -B_z & 0 & B_x \\ B_y & -B_x & 0 \end{bmatrix} \begin{bmatrix} \frac{N_x A_x}{R_x} & 0 & 0 \\ 0 & \frac{N_y A_y}{R_y} & 0 \\ 0 & 0 & \frac{N_z A_z}{R_z} \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix}$$

$$\boldsymbol{\tau_m} = \boldsymbol{S}(-\boldsymbol{B}) \cdot \boldsymbol{K}_{coil} \cdot \boldsymbol{V} \tag{4.12}$$

One can observe that since the magnetic field is cyclic, two of the components in $\boldsymbol{B}_b$ can be zero at the exact same time in space. Meaning the satellite will not be able to induce a torque around one axis. At this point the satellite will indeed be under-actuated. By comparing equation (4.9) with figure 2.3 it can be seen that this will happen approximately four times during one orbit (if the satellite is following the orbit frame, and the Earth's magnetic field is modeled as a dipole). This is however not that big of an issue, since the magnetic field is cyclic, and the satellite will eventually move out of this state (see figure 2.3).

4.5 Scaling and limitations

It is important to understand the relationship between $\boldsymbol{\tau}_m$ and $\boldsymbol{\tau}_d$. This can be seen as the *available torque* and the *desired torque.* The desired torque is the torque that the controllers optimally would like to produce. This could be any arbitrary torque in the BODY coordinate system. Because $\boldsymbol{\tau}_m$ is given as the cross product (4.9), $\boldsymbol{\tau}_m$ will always be in the plane

[1] $\boldsymbol{V} \neq \boldsymbol{K}_{coil}^{-1}\boldsymbol{S}(-\boldsymbol{B})^{-1}\boldsymbol{\tau}_m,$ because $\boldsymbol{S}(-\boldsymbol{B})^{-1}$ is singular

perpendicular to $\boldsymbol{B}_b$. This can be seen in figure 4.1. By further inspection it can also be seen that the torque from the controller $\boldsymbol{\tau}_d$ might not be in this plane.

To deal with this issue, one must simply acknowledge that $\boldsymbol{\tau}_m$ *is as close to* $\boldsymbol{\tau}_d$ *it is possible to get*. That is $\boldsymbol{\tau}_m$ will only be able to represent the component of $\boldsymbol{\tau}_d$ lying in the plane perpendicular to $\boldsymbol{B}_b$. Mathematically the length of $\boldsymbol{\tau}_d$ should be scaled down to match this restriction, to avoid unnecessary use of electricity. To achieve this a scaling function is multiplied with equation (4.10):

$$\boldsymbol{m}_b = f(\cdot)\boldsymbol{\tau}_m \times \boldsymbol{B}_b \tag{4.13}$$

By combining this with equation (4.9), a new expression for $\boldsymbol{\tau}_m$ is formed

$$\boldsymbol{\tau}_m = f(\cdot)(\boldsymbol{\tau}_d \times \boldsymbol{B}_b) \times \boldsymbol{B}_b \tag{4.14}$$

By the definition of a cross product, it is possible to express the length of $\boldsymbol{\tau}_m$ with a function of the angle α between $\boldsymbol{B}$ and $\boldsymbol{\tau}_d$

$$|\boldsymbol{\tau}_m| = |\boldsymbol{m}_b||\boldsymbol{B}_b| = f(\cdot)|\boldsymbol{B}_b||\boldsymbol{\tau}_d||\boldsymbol{B}_b|sin(\alpha)$$

Now one can solve for $f(\cdot)$ by the fact that $|\boldsymbol{\tau}_m| = |\boldsymbol{\tau}_d|sin(\alpha)$ (right-angled triangle):

$$f(\cdot) = \frac{1}{|\boldsymbol{B}_b|^2} \tag{4.15}$$

The final expression for $\boldsymbol{m}_b$[2]:

$$\boldsymbol{m}_b = \frac{1}{|\boldsymbol{B}_b|^2}(\boldsymbol{B}_b \times \boldsymbol{\tau}_m) \tag{4.16}$$

and the final expression for $\boldsymbol{\tau}_m$:

$$\boldsymbol{\tau}_m = \frac{1}{|\boldsymbol{B}_b|^2}(\boldsymbol{\tau}_d \times \boldsymbol{B}_b) \times \boldsymbol{B}_b \tag{4.17}$$

[2]By the cross product definition; $\boldsymbol{B}_b \times \boldsymbol{\tau}_m = -\boldsymbol{\tau}_m \times \boldsymbol{B}_b$. This is depends on what direction $\boldsymbol{B}_b$ is defined positive.

Chapter 5

Controller

Based on literature studies, an attitude controller will here be derived. The controllers is based on previous master thesis's and papers.

5.1 Passive controller

A passive controller for a CubeSat can be set up quite easy using strong rare-earth magnets like for example neodymium magnets. This can be set up as part of a semi-passive system, to act as more damping to the system, or it can work completely on its own.

Reasons for choosing such a system is many. First of all a CubeSat has limited amount of space for actuators. A neodymium magnet can produce a much more powerful magnetic field then an electromagnet for the same size (given the limited power supply). The extra space can be used for other equipment. Some missions, unlike the NUTS project, might not have any preference to the attitude of the satellite, other then for example a slow spin rate. A constant (powerful) magnet can be added to work on its own, or act as gain in one direction:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{neodymium} \quad \text{or} \quad \boldsymbol{\tau} = \boldsymbol{\tau}_{controller} + \boldsymbol{\tau}_{neodymium} \tag{5.1}$$

A passive controller can also work very well for geostationary satellites (not moving relative to the NED frame). For such a purpose, one can simply mount a magnet aligned with the constant, non-changing, magnetic field in this orbit.

The NUTS project however, have strict specifications regarding attitude. Since magnetic field will be changing constantly as product of the polar orbit, a passive controller might even disturb the satellites position even further.

One the contrary, all electric components that induce a current will generate a magnetic field. Also batteries can be magnetic. When the satellite is fully assembled, it is possible to measure the sum of the total magnetic field generated by the NUTS. If this turns out be above the saturation limit of the controller, one can install a constant magnet in the frame, compensating in the opposite direction:

$$\boldsymbol{\tau}_{magnet} = \boldsymbol{m}_{magnet} \times \boldsymbol{B}_b = -(\boldsymbol{m}_{NUTS} \times \boldsymbol{B}_b) = -\boldsymbol{\tau}_{NUTS} \tag{5.2}$$

$$\implies \boldsymbol{\tau}_{magnet} + \boldsymbol{\tau}_{NUTS} = 0 \tag{5.3}$$

The NUTS is bound to create a local magnetic field. Even if it is within the "robustness limits" of the controller, it should be measured, as it will act as bias on the magnetometer (see section 2.4).

5.2 Detumbling

A detumbling controller will most likely be part of the final ADCS system. As the name states, the only purpose is to slow down unknown initial tumbling/spin in the attitude. Two alternatives has previously been proposed in the NUTS project, *Bdot* and *Dissipative controller*. The B-dot controller shows excellent results in the detumbling phase after the satellite has ejected from the carrying rocket [3]. It is not particularly useful for correcting small deviations. This is because the Bdot controller, as the name suggests, only reacts to rate of change in the geomagnetic field ($\dot{\boldsymbol{B}}$) in the BODY coordinate system. In practice, this controller will never rest and constantly drain power from the satellite. As previously stated, Bdot can do an excellent job for de-tumbling the satellite, and then switching to a different controller.

Another approach is to use the angular velocity $\boldsymbol{\omega}_{ob}^{b}$ in the BODY frame relative to the orbit frame. (Keep in mind that the orbit coordinate system always points its z-axis towards Earth. This gives a constant rotation in the BODY y-axis). This measurement can be taken from a sun-sensor and/or a gyroscope to avoid the use of the magnetometer. The controller could take the following form:

$$\boldsymbol{\tau}_{controller}^{b} = -k_d \boldsymbol{\omega}_{ob}^{b}, \quad k_d > 0 \tag{5.4}$$

Setting up an expression for voltage

$$\begin{aligned}
\boldsymbol{m}^b &= \boldsymbol{\tau}^b_{controller} \times \boldsymbol{B}^b \\
\boldsymbol{V} &= \boldsymbol{K}^{-1}_{coil}\boldsymbol{m}^b \\
\boldsymbol{V} &= \boldsymbol{K}^{-1}_{coil}(\boldsymbol{\tau}^b_{controller} \times \boldsymbol{B}^b) \\
\boldsymbol{V} &= \boldsymbol{K}^{-1}_{coil}(-k_d\boldsymbol{\omega}^b_{ob} \times \boldsymbol{B}^b) \\
\boldsymbol{V} &= -k_d\boldsymbol{K}^{-1}_{coil}(\boldsymbol{\omega}^b_{ob} \times \boldsymbol{B}^b)
\end{aligned}$$

Now the scaling factor from section 4.5 is taken into account

$$\boldsymbol{V} = -\frac{k_d}{|\boldsymbol{B}^b|}\boldsymbol{K}^{-1}_{coil}(\boldsymbol{B}^b \times \boldsymbol{\omega}^b_{ob}), \quad |\boldsymbol{V}| < 5\,\text{volts} \tag{5.5}$$

This will act as the detumbling controller of choice in further testing and simulations in chapter 7. The limit on voltage as ± 5 volts is set by the hardware specification on the NUTS power supply system.

5.3 Model-independent Control Law

Wen and Kreutz-Delgado describes in [7] a model-independent control law:

$$\boldsymbol{\tau} = k_p\boldsymbol{\epsilon} - k_d\tilde{\boldsymbol{\omega}} \tag{5.6}$$

It is stated that the controller is ”*...globally stabilizing for a class of desired trajectories*” by a Lyapunov analysis. However, this controller cannot be *globally* stabilizing. Both due to the fact that the system is not controllable in parts of the orbit, and that the system has multiple equilibriums caused by gravity.

τ torque on any arbitrary axis

k_p constant proportional controller gain

$\boldsymbol{\epsilon}$ imaginary part of the quaternion $\boldsymbol{q} = [\eta\, \boldsymbol{\epsilon}]^\top = [\eta\, \epsilon_1\, \epsilon_2\, \epsilon_3]^\top, \quad \boldsymbol{q} \to \pm[1\,0\,0\,0]^\top$

k_d constant derivative controller gain

$\tilde{\boldsymbol{\omega}}$ angular velocity minus desired velocity $\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega}_b - \boldsymbol{\omega}_{desired}, \quad \tilde{\boldsymbol{\omega}} \to 0$

This is a PD-controller that can be seen as an extension to the detumbling controller;

$$\boldsymbol{\tau}^b_{controller} = -k_p\boldsymbol{\epsilon} - k_d\boldsymbol{\omega}^b_{ob}, \quad k_d, k_p > 0 \tag{5.7}$$

With the same approach as in the previous section, the controller (with voltage as output) can be derived

$$\boldsymbol{V} = -\frac{k_p}{|\boldsymbol{B}^b|}\boldsymbol{K}^{-1}_{coil}(\boldsymbol{B}^b \times \boldsymbol{\epsilon}) - \frac{k_d}{|\boldsymbol{B}^b|}\boldsymbol{K}^{-1}_{coil}(\boldsymbol{B}^b \times \boldsymbol{\omega}^b_{ob})$$

$$\boldsymbol{V} = -\frac{\boldsymbol{K}^{-1}_{coil}}{|\boldsymbol{B}^b|}\left(-k_p(\boldsymbol{B}^b \times \boldsymbol{\epsilon}) - k_d(\boldsymbol{B}^b \times \boldsymbol{\omega}^b_{ob})\right), \quad |\boldsymbol{V}| < 5\,\text{volts} \tag{5.8}$$

This is the primary controller that will maintain the attitude of the NUTS. The idea is to switch this controller on when the satellite has travelled a finite number of orbits, and it can be assumed that most of the spin is gone. This controller is the basis for many quaternion feedback satellite attitude controllers. A short stability analysis is found in appendix A.

Chapter 6

Simulator

In order to test the controllers, one need off course a simulator. There are several ways to make a simulator. It can be built from ground up, with the advantage of having an in depth knowledge of the entire systems. But it is time consuming and difficult to implement with all complexities evolving a full sized simulator. Complexities like environmental torques and satellite kinetics. Errors (in for example the Simulink environment) can also be very time consuming debug.

Another approach is to test controller algorithms on already existing simulators. These simulators can be previously made systems from graduated students or commercial companies. If the simulator has well documented code, and has been approved by a third party, less time is needed to evaluate the simulator. Instead it is possible to start evaluating the controller design right away.

Prevoius studies and developments of simulators was done. Inspiration, testing and benchmarking was investigated in order to complete the simulator. Both Zdenko Tudor's [3] and Eli Jerpseth Overby's [9] simulators where used as inspiration. An extension of Tudor's simulator is also used by other members NUTS. Results from their work, has been discussed during this development. A direct performance comparison between the simulators is not given here, since their work is still in writing.

The Princeton Satellite Systems CubeSat Toolbox was acquired by NUTS as a commercial alternative. But given the limited timeframe, the results have not been compared with this simulator.

6.1 Nonlinear Simulink Simulator

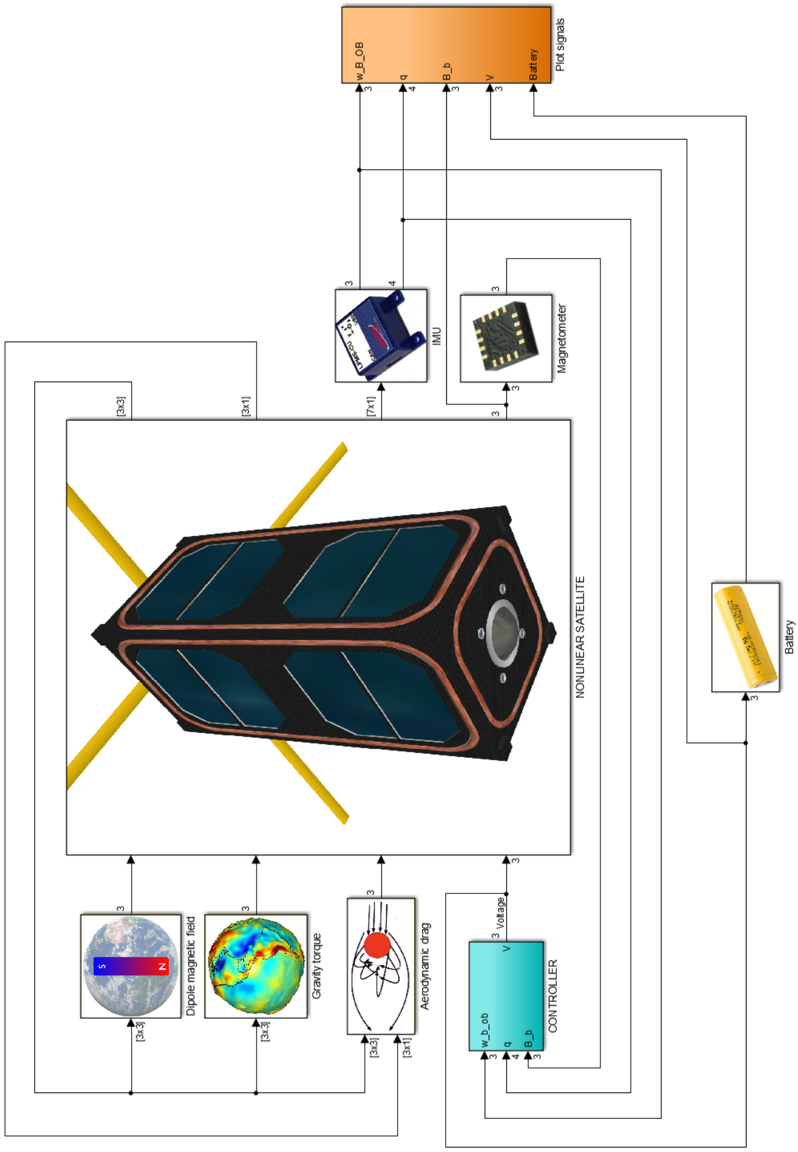

Figure 6.1: Nonlinear NUTS Simulink model with controller

Previous simulators has mostly been developed in Matlab code. It can be very time-consuming to carry on such projects, since it is all written code. It was also known from the beginning that the controller had to be compiled for a specific embedded platform (see section 3.2). Going from Simulink to C-code with the Simulink Coder environment was consider a great advantage (see section 3.1). For this reason the development of a Simulink simulator was begun. This simulator was made from ground up, with inspiration from [9].

It includes a nonlinear satellite model exactly like the one described in chapter 4. But instead of the IGRF model, a dipole is used. This choice was mainly done because it is was considered too time-consuming to develop an IGRF model, and the controller performance should still be valid for testing with a "dipole-environment".

The main *Nonlinear Satellite* block consists of the satellites kinematics and kinetics. Except from the voltage input from the controller, the satellite block has three inputs; *Dipole magnetic field*, *Gravity torque* and *Aerodynamic drag*. These are transformed to the BODY-coordinate system using $\boldsymbol{R}_o^b$ outputted from the Satellite block. The aerodynamic drag is also dependent on the angular velocity, so this is feedback as well.

The goal was initially to create a simulator with full-state-back to the controller. The output from the Satellite block is therefore $\boldsymbol{q}$, $\boldsymbol{\omega}_{ob}^b$ and $\boldsymbol{B}_b$, that are the attitude, angular velocity and magnetic resultant vector in BODY. The NUTS team have however decided on a specific magnetometer to use. The data resembling this instrument has been implemented in the *IMU* block, but can be turned on and off via a switch in the block. More results on this feature can be found in chapter 7.

State feedback is sent to the *Controller* block that in turn calculates a voltage to correct the attitude. The controller block has been developed using Matlab function blocks. This makes it somewhat easier to make radical changes, but still maintain an acceptable simulation time. The controller also has an so called orbit propagator, that simply calculates the number of orbits based on time. This is tuned to trigger the PD-controller, when the detumbling controller most likely has finished it's job.

All initial states and constants are set by running an Init.m file. Such as weight, initial attitude and coil parameters. After simulation, all relevant data is sent to a *Plot signals* block, that can be plotted in Matlab using the plot script.

Last, but not least, is a block containing data on the *Battery*. It is popular to show how much energy the satellite uses, but it was found more interesting to give this as a function of the satellites battery.

A full breakdown of the model can be found in appendix C.

Chapter 7

Results

This chapter will present the results from the different controller approaches. The following specifications is equal to all simulations:

- **Simulator** The Simulink Simulator from section 6.1
- **Simulation time** 30.000 seconds (≈ 6 orbits)
- **Initial attitude** (except for the constant controller)

  $\boldsymbol{\omega}_{ob}^{b}$ inital angular velocity given in rad/sec

  $\boldsymbol{\omega}_{ob}^{b} = [0.1 \quad -0.2 \quad 0.05]^{\top}$

  $\boldsymbol{\Theta}_{ob}$ inital attitude orienation in *roll, pitch, yaw* in degrees

  $\boldsymbol{\Theta}_{ob} = [10^{\circ} \quad 5^{\circ} \quad -2^{\circ}]^{\top}$

- **Altitude** 600 kilometers Low Earth Orbit

- **Mass** 2.6 kg
- **Coil parameters** (found by the model built in [3])

  $N_x = 355 \quad N_y = 800 \quad N_z = 800$

  $A_x = 144\, cm^2 \quad A_y = 144\, cm^2 \quad A_z = 64\, cm^2$

  $R_x = 110\,\Omega \quad R_y = 110\,\Omega \quad R_z = 110\,\Omega$

- **Battery** 3.7 V   1800 mAh
- **Magnetometer noise is OFF** except for the specific simulation 7.6
- **Aeordynamic drag is OFF** except for the specific simulation 7.5

The initial angular velocity and orientation was based a "worst case plausible" scenario. Since the satellite is spinning relatively fast after deployment, the initial orientation won't effect the results much. To limit the amount of presented data, only the most import figures are here presented. Also be aware that there is no difference between $\pm 180°$ in the attitude plots. This is a result of how the quaternions are converted.

7.1 Passive controller

The passive controller discussed in section 5.1, was implemented with data found from a commercial product.

- K&J Magnetics, Inc.
- Neodymium-Iron-Boron

- Cylinder dipole
- 13.6 grams
- Surface field 0.6487 Tesla

This was realized as simply disconnecting the state feedback from the controller, and adding a constant magnetic field in the satellite block. Initial attitude and angular-velocity was *zero*.

From the plots one can observe that the satellite goes into a completely uncontrolled spin.

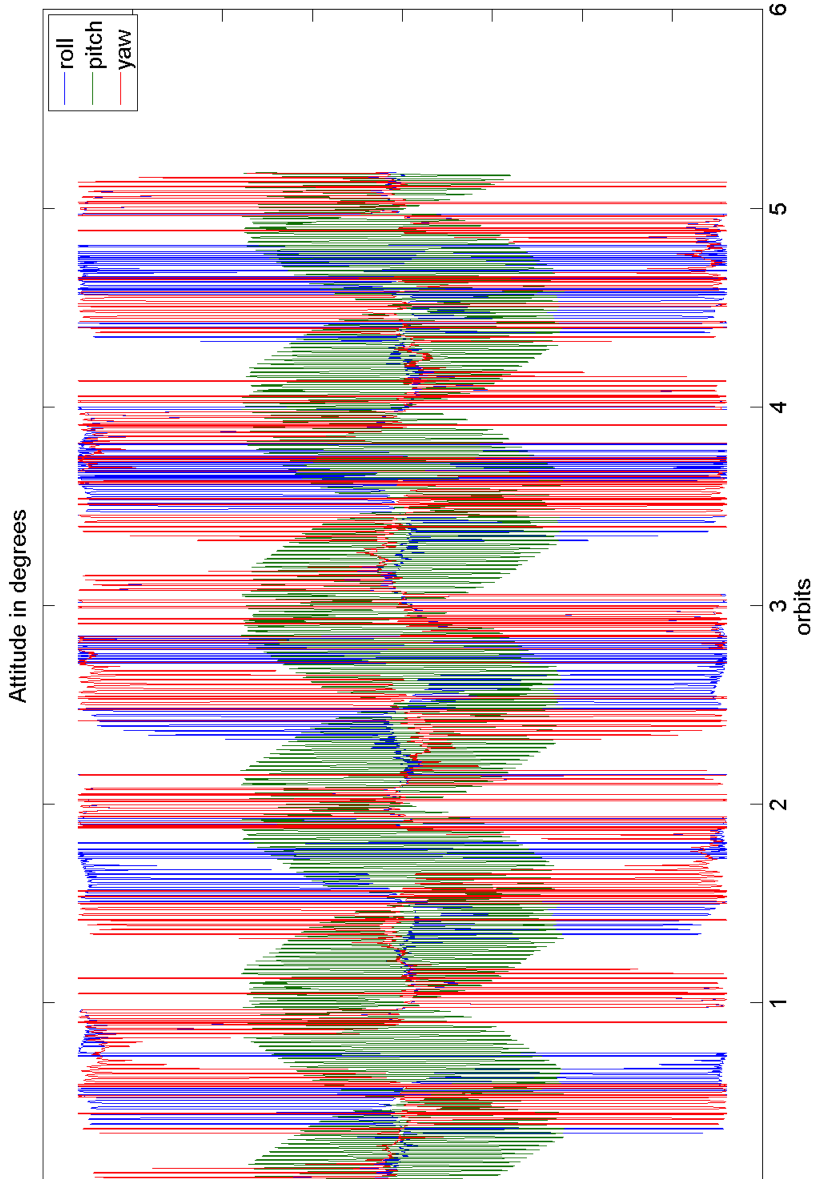

Figure 7.1: Attitude with constant magnet

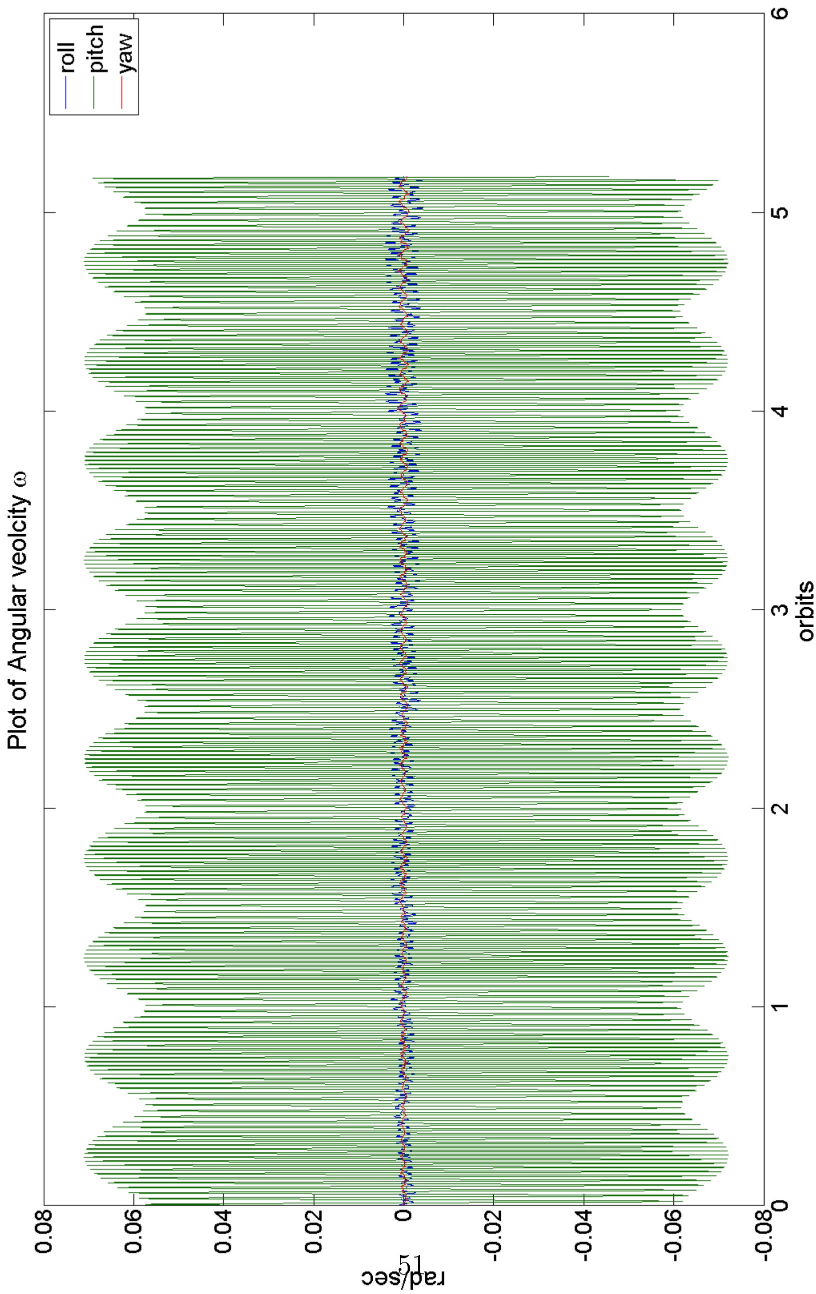

Figure 7.2: Angular velocity with constant magnet

7.2 Detumbling controller

$$
\begin{aligned}
\boldsymbol{V} &= -\frac{k_d}{|\boldsymbol{B}^b|}\boldsymbol{K}_{coil}^{-1}(\boldsymbol{B}^b \times \boldsymbol{\omega}_{ob}^b), \quad |\boldsymbol{V}| < 5\,\text{volts} \\
k_d &= 4 \cdot 10^{-5}
\end{aligned}
$$

Here, the detumbling controller from section 5.2 is simulated. The value for k_d was first set quite high ($k_d \approx 2$), but this resulted in a very aggressive and unstable controller. After some discussion with the NUTS-ADCS team, it was tuned down a lot., and the satellite finally calmed down, as intended.

It is clear from the plots that the angular velocity calms down almost completely after just a couple of orbits. The attitude is however way off.

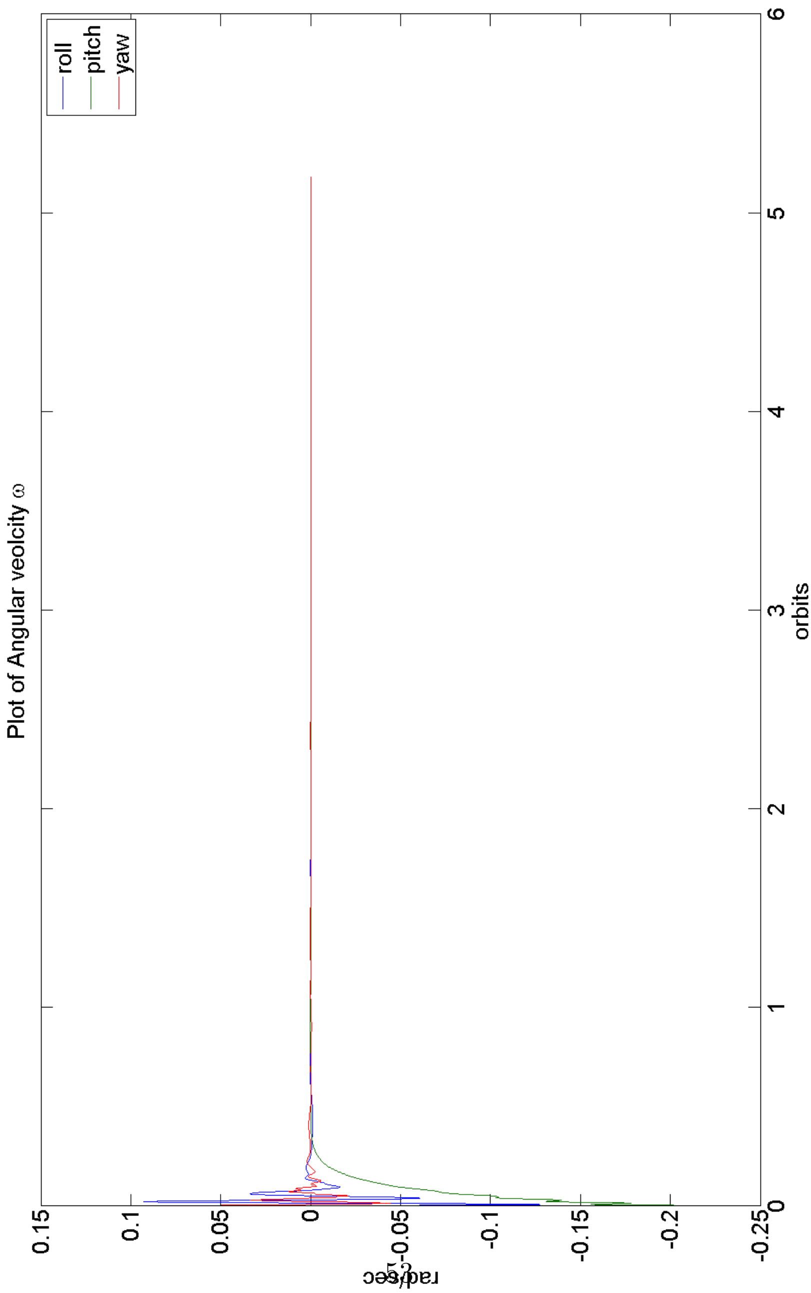

Figure 7.3: Angular velocity with the detumbling algorithm

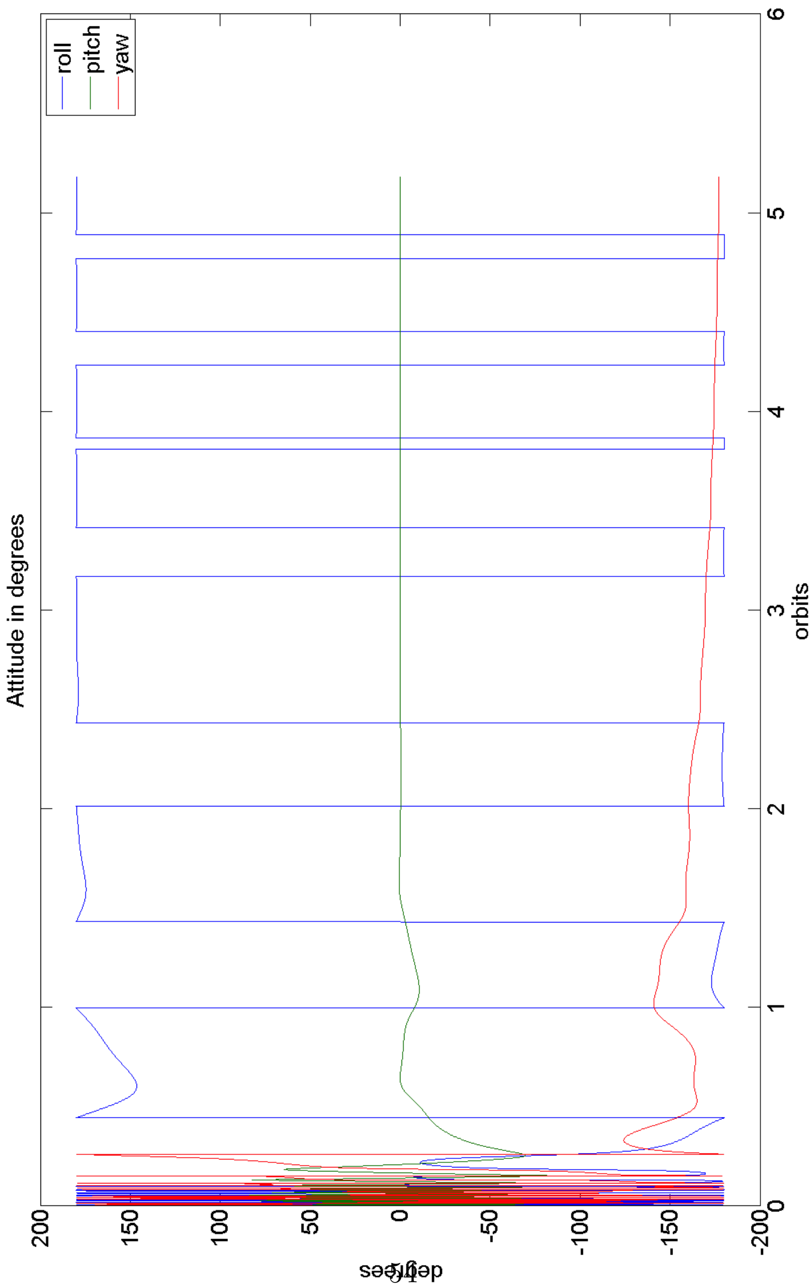

Figure 7.4: Attitude with the detumbling algorithm

7.3 PD-controller

The PD-controller from section 5.3 is simulated with the following controller gains;

$$\boldsymbol{V} = -\frac{\boldsymbol{K}_{coil}^{-1}}{|\boldsymbol{B}^b|}\left(-k_p(\boldsymbol{B}^b \times \boldsymbol{\epsilon}) - k_d(\boldsymbol{B}^b \times \boldsymbol{\omega}_{ob}^b)\right), \quad |\boldsymbol{V}| < 5\,\text{volts}$$
$$k_d = 4 \cdot 10^{-5}$$
$$k_p = 5 \cdot 10^{-8}$$

Again, k_p and k_d is found by tuning. k_p is also above the threshold found in the stability analysis in appendix A.

From the plots it can be seen that the angular velocity is slightly less stable then what the detumbling controller alone could manage. The attitude is however better, but has somewhat high oscillations.

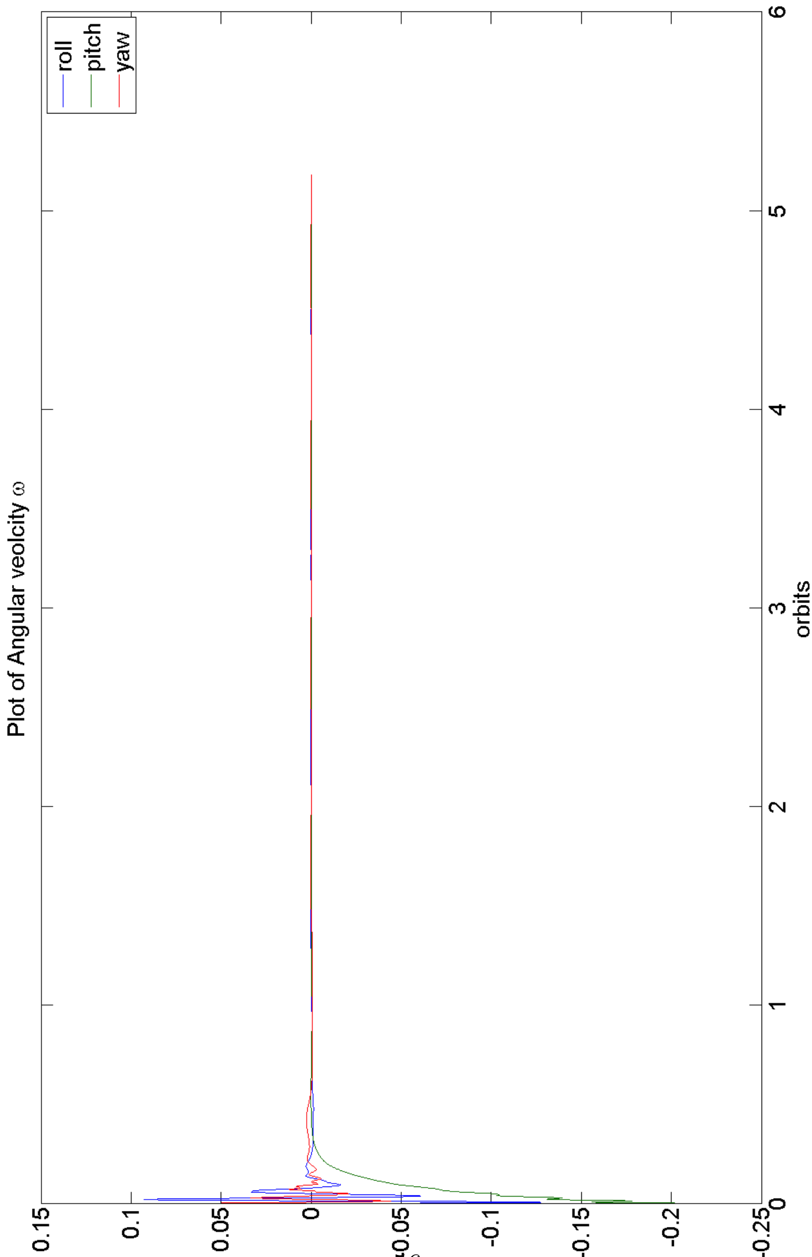

Figure 7.5: Angular velocity with only the PD-controller

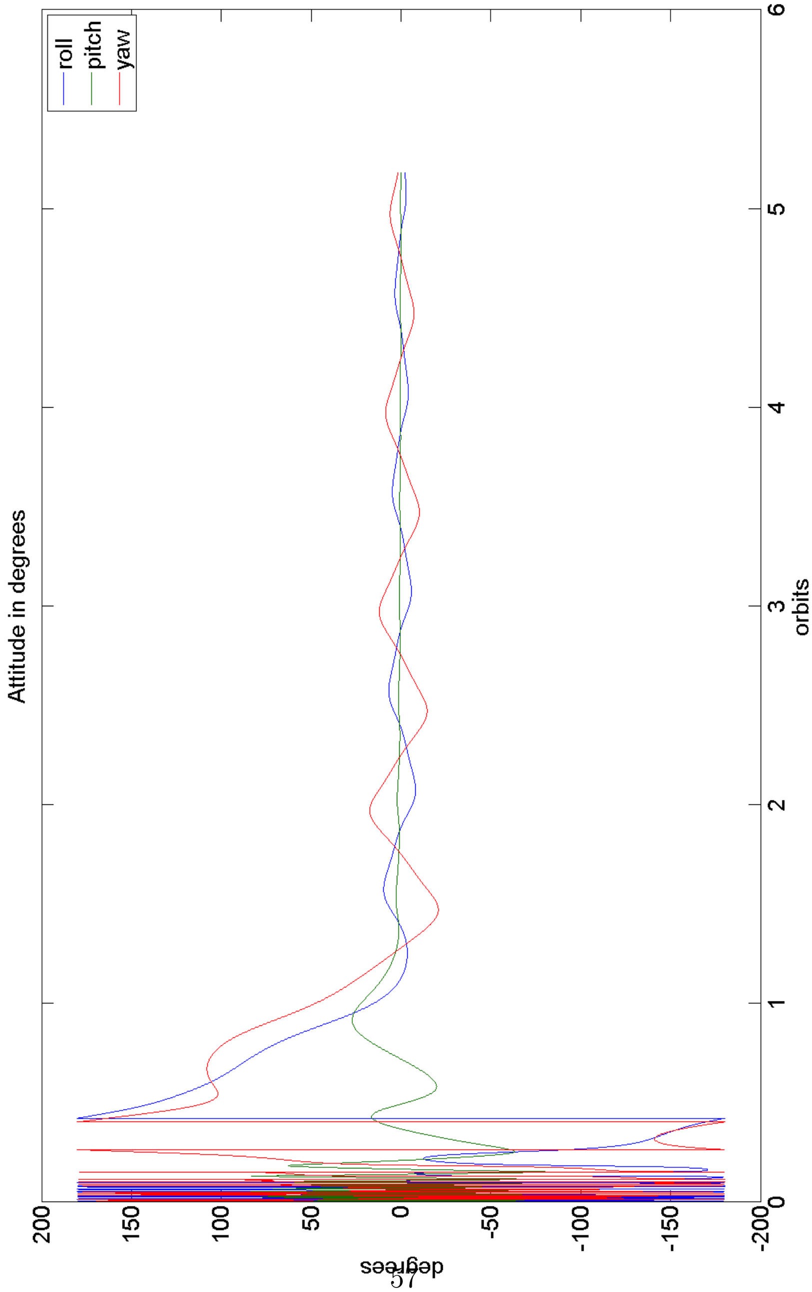

Figure 7.6: Attitude with only the PD-controller

7.4 Detumbling and PD-controller combination

$$
\begin{aligned}
\boldsymbol{V} &= -\frac{k_d}{|\boldsymbol{B}^b|}\boldsymbol{K}_{coil}^{-1}(\boldsymbol{B}^b \times \boldsymbol{\omega}_{ob}^b), \quad \text{orbits} \leq 2 \\
\boldsymbol{V} &= -\frac{\boldsymbol{K}_{coil}^{-1}}{|\boldsymbol{B}^b|}\left(-k_p(\boldsymbol{B}^b \times \boldsymbol{\epsilon}) - k_d(\boldsymbol{B}^b \times \boldsymbol{\omega}_{ob}^b)\right), \quad \text{orbits} > 2 \\
k_d &= 4 \cdot 10^{-5} \\
k_p &= 5 \cdot 10^{-8} \\
|\boldsymbol{V}| &< 5\,\text{volts}
\end{aligned}
$$

In this simulation a combination of the Detumbling- and PD-controller was tested. After trial and error the PD-controller was set to ”take over” after 2 orbits. Otherwise, the parameters are the same.

One can easily spot that after two orbits the controller starts to consider controlling the attitude. The angular velocity is very calm, and the attitude almost zero by a deviation of one degree.

This time the quaternion plot is also included, and one can observer how $\boldsymbol{q} \rightarrow\approx [1\,0\,0\,0]^\top$.

It is also interesting to observer how the controller pushes the limits on the voltage output and how this calms down in synergy with the attitude. The battery is on the contrary nearly not effected.

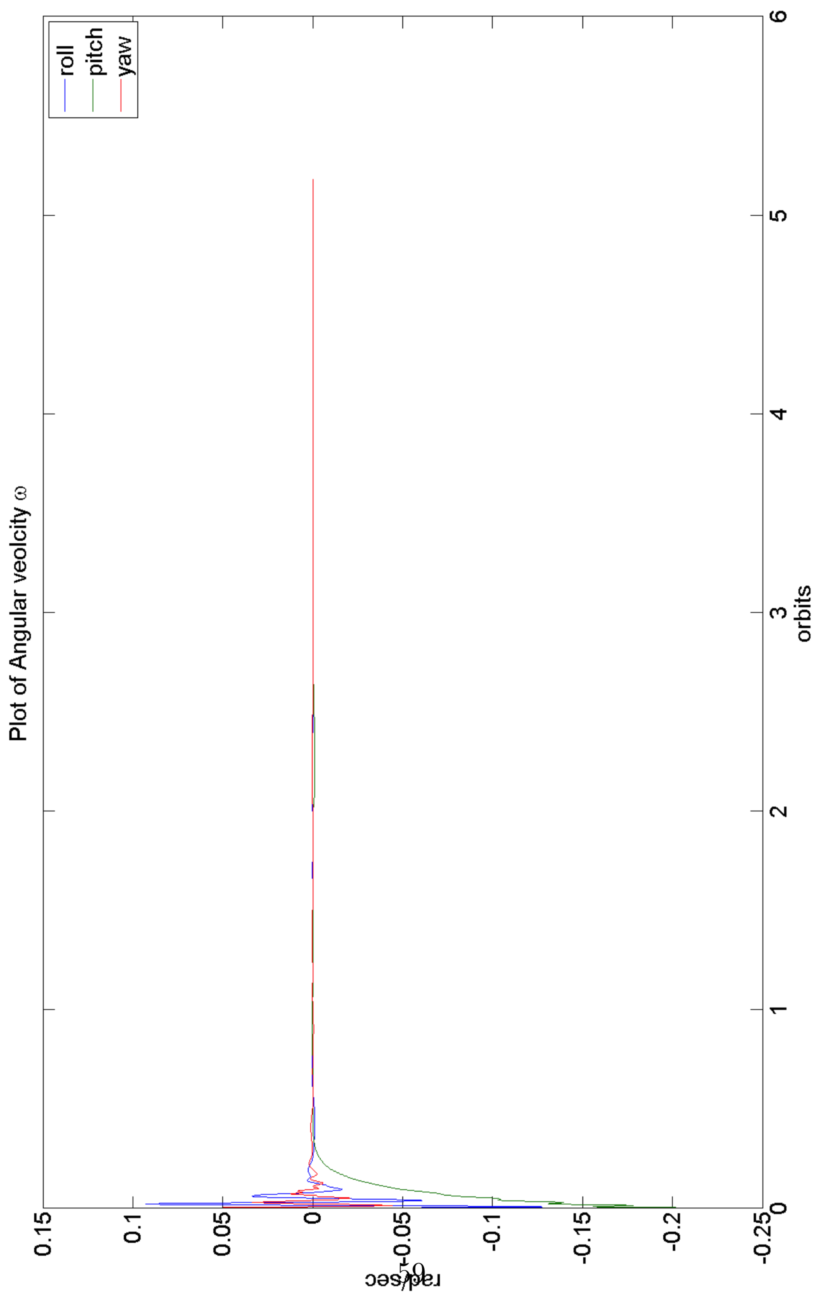

Figure 7.7: Angular velocity with detumbling and PD-combination

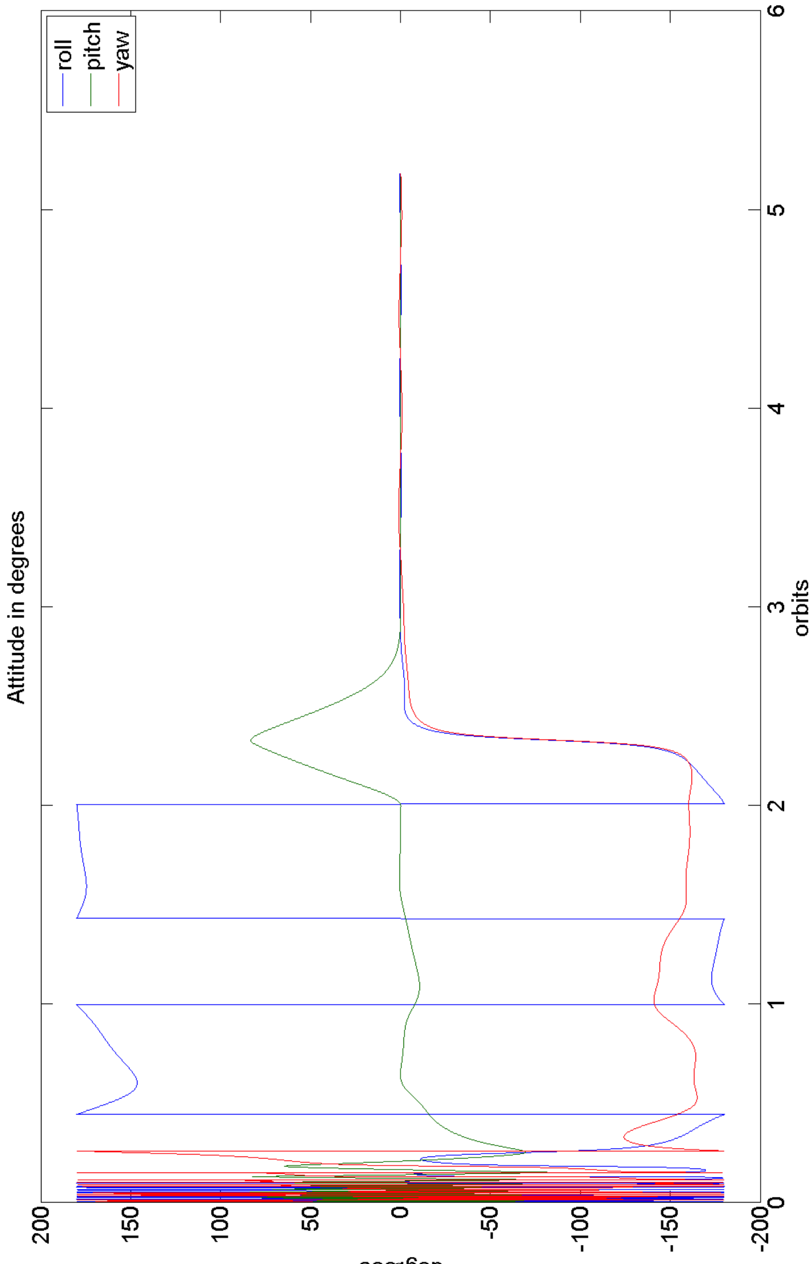

Figure 7.8: Attitude with detumbling and PD-combination

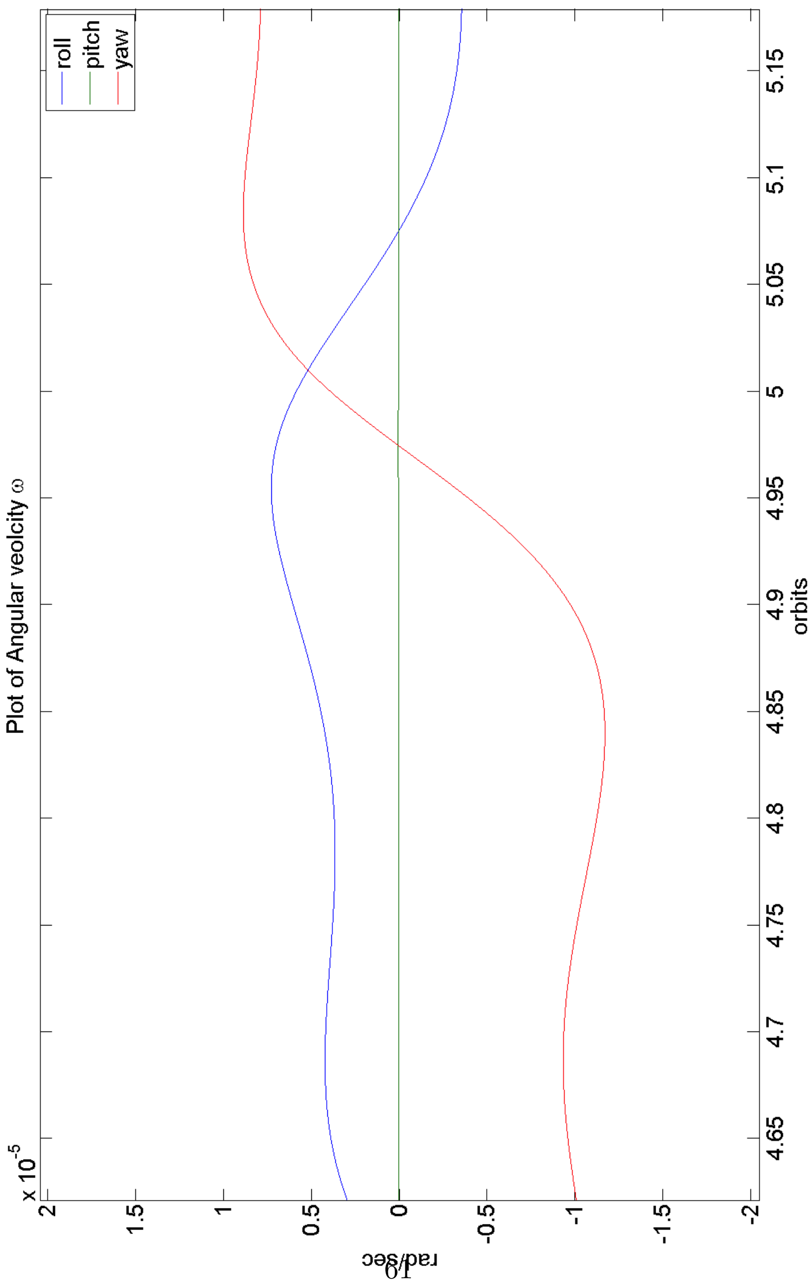

Figure 7.9: Angular velocity with detumbling and PD-combination Zoomed

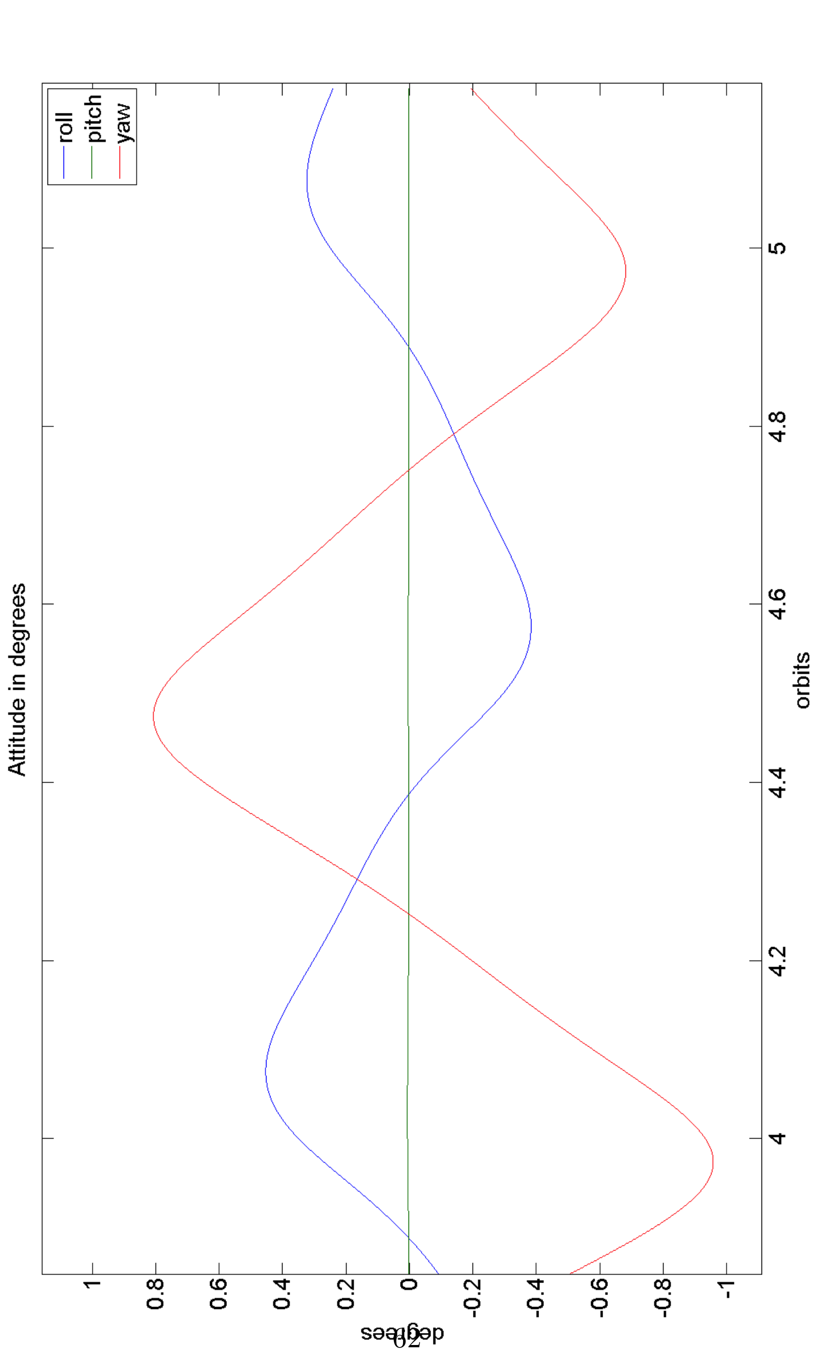

Figure 7.10: Attitude with detumbling and PD-combination Zoomed

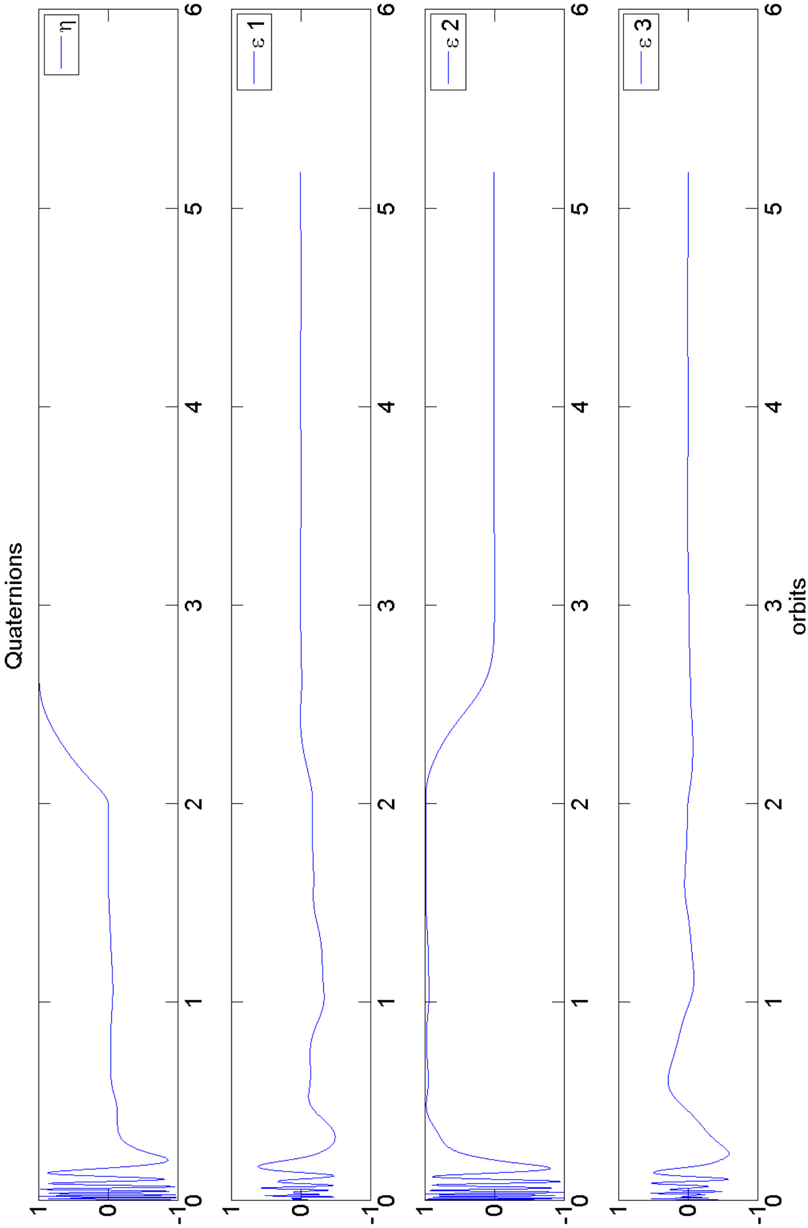

Figure 7.11: Quaternions with detumbling and PD-combination

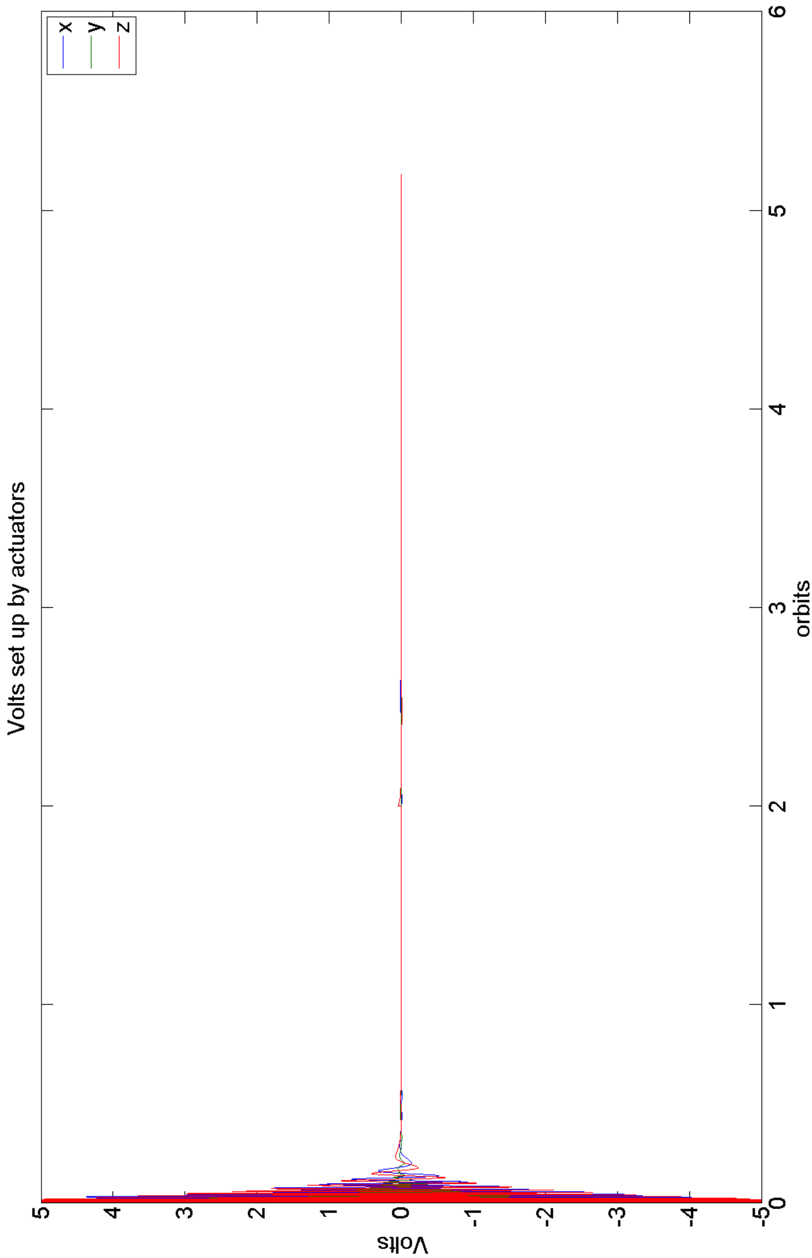

Figure 7.12: Volts set up by the detumbling and PD-combination

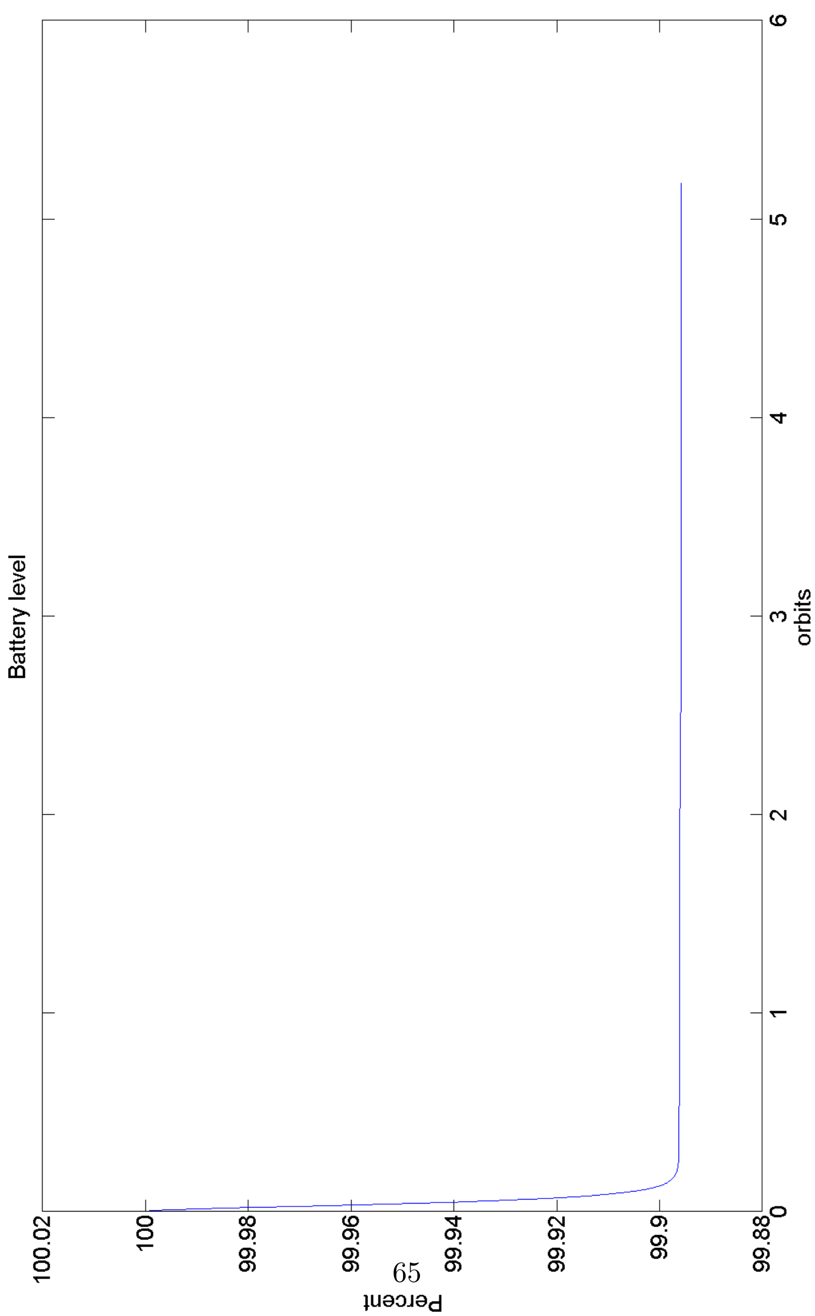

Figure 7.13: Battery use with detumbling and PD-combination

7.5 Aerodynamic drag

$$
\begin{aligned}
\boldsymbol{V} &= -\frac{k_d}{|\boldsymbol{B}^b|}\boldsymbol{K}_{coil}^{-1}(\boldsymbol{B}^b \times \boldsymbol{\omega}_{ob}^b), \quad \text{orbits} \leq 2 \\
\boldsymbol{V} &= -\frac{\boldsymbol{K}_{coil}^{-1}}{|\boldsymbol{B}^b|}\left(-k_p(\boldsymbol{B}^b \times \boldsymbol{\epsilon}) - k_d(\boldsymbol{B}^b \times \boldsymbol{\omega}_{ob}^b)\right), \quad \text{orbits} > 2 \\
k_d &= 4 \cdot 10^{-5} \\
k_p &= 5 \cdot 10^{-8} \\
|\boldsymbol{V}| &< 5\,\text{volts} \\
\boldsymbol{I}_{CG}\dot{\boldsymbol{\omega}} + \boldsymbol{S}(\omega)\omega &= \boldsymbol{\tau}_m + \boldsymbol{\tau}_g + \boldsymbol{\tau}_a
\end{aligned}
$$

By adding the aerodynamic torque, the satellite still manages to calm down the angular velocity. The attitude is still however way off at up to 100°, and also suffers from oscillations. These simulations is also consistent with what the rest of the NUTS-ADCS team achieved.

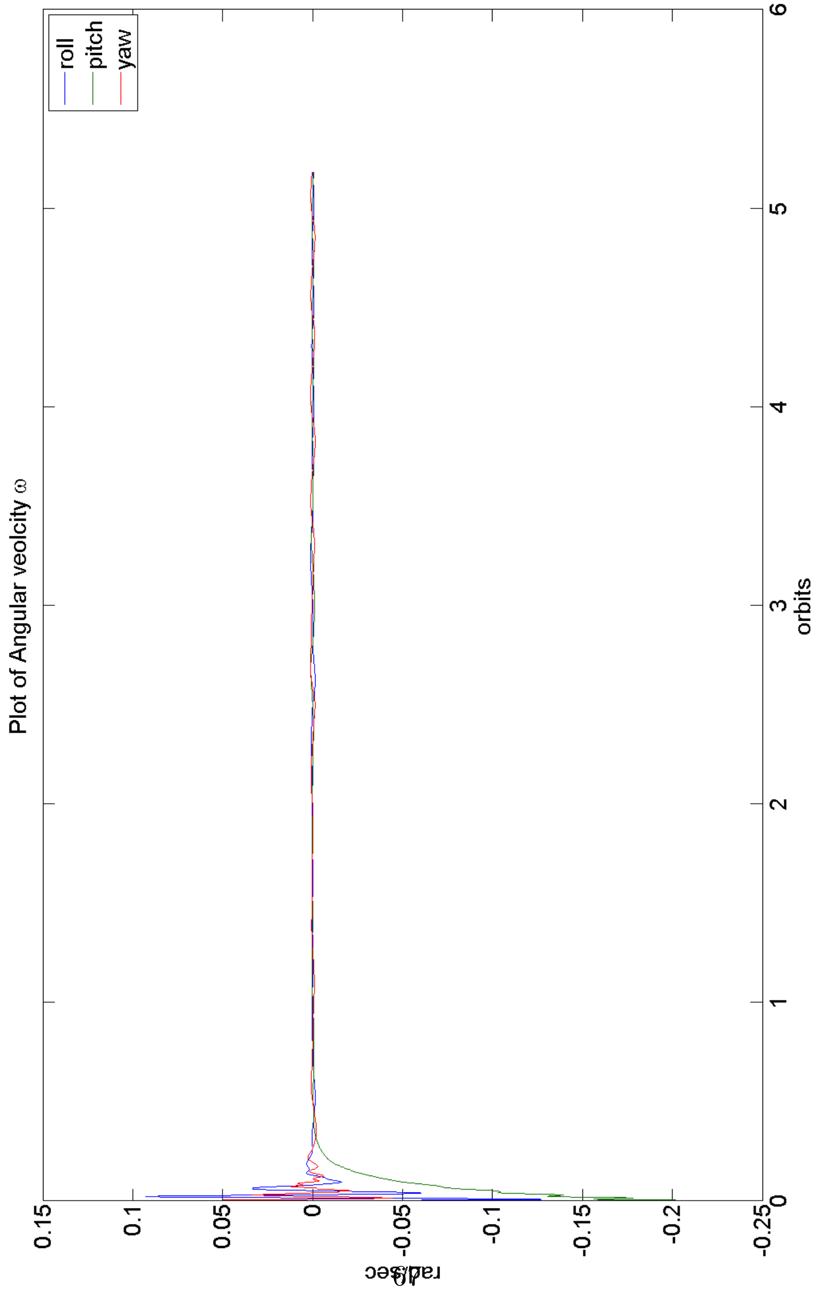

Figure 7.14: Angular velocity with aerodynamic disturbance

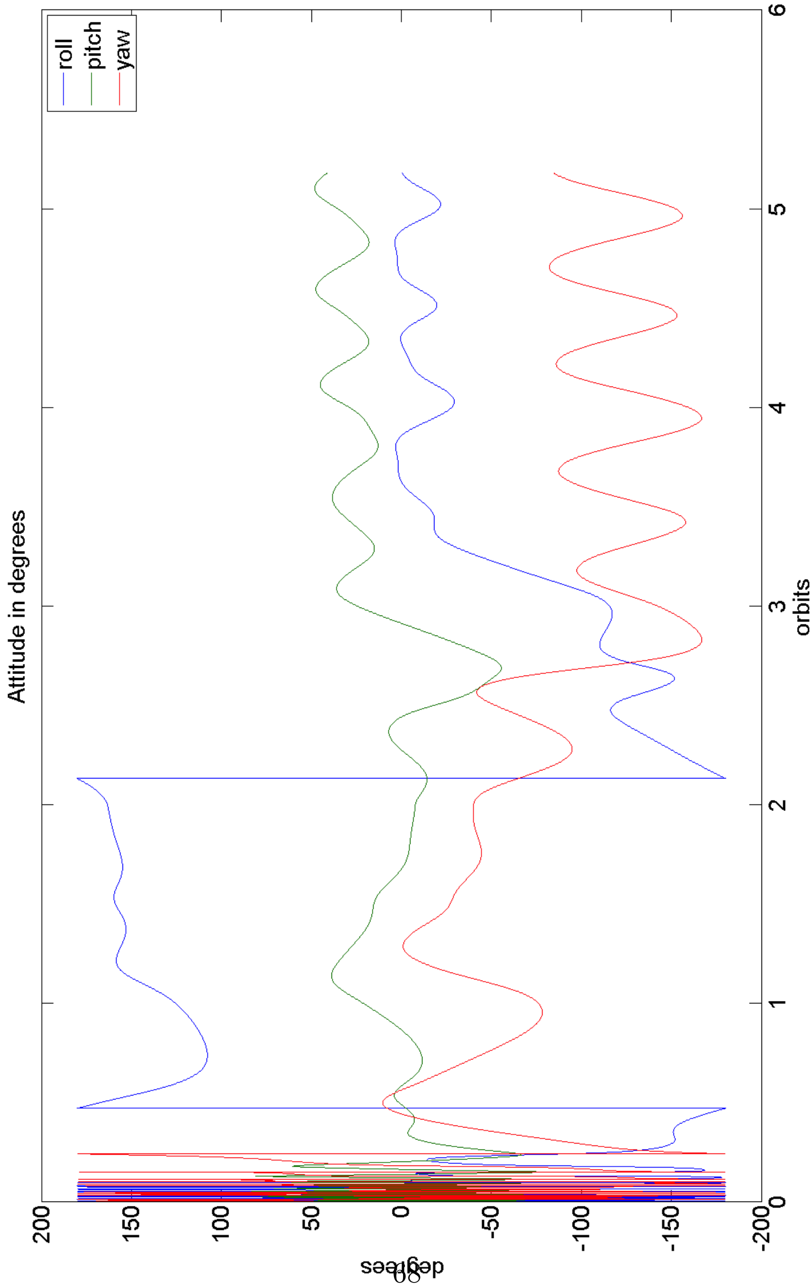

Figure 7.15: Attitude with aerodynamic disturbance

7.6 Simulated magnetometer feedback

Based on the Honeywell HMC5983 specifications, the magnetometer feedback to the controller was given added noise.

$$
\begin{aligned}
\boldsymbol{V} &= -\frac{k_d}{|\boldsymbol{B}^b|}\boldsymbol{K}_{coil}^{-1}(\boldsymbol{B}^b \times \boldsymbol{\omega}_{ob}^b), \quad \text{orbits} \leq 2 \\
\boldsymbol{V} &= -\frac{\boldsymbol{K}_{coil}^{-1}}{|\boldsymbol{B}^b|}\left(-k_p(\boldsymbol{B}^b \times \boldsymbol{\epsilon}) - k_d(\boldsymbol{B}^b \times \boldsymbol{\omega}_{ob}^b)\right), \quad \text{orbits} > 2 \\
k_d &= 4 \cdot 10^{-5} \\
k_p &= 5 \cdot 10^{-8} \\
|\boldsymbol{V}| &< 5\,\text{volts} \\
\boldsymbol{B}^b &= \boldsymbol{B}^b \pm 0.0002\,\text{tesla}
\end{aligned}
$$

The result is still a stable attitude, but a lot more noisy angular velocity. The voltage output from the controller is also changing very rapidly.

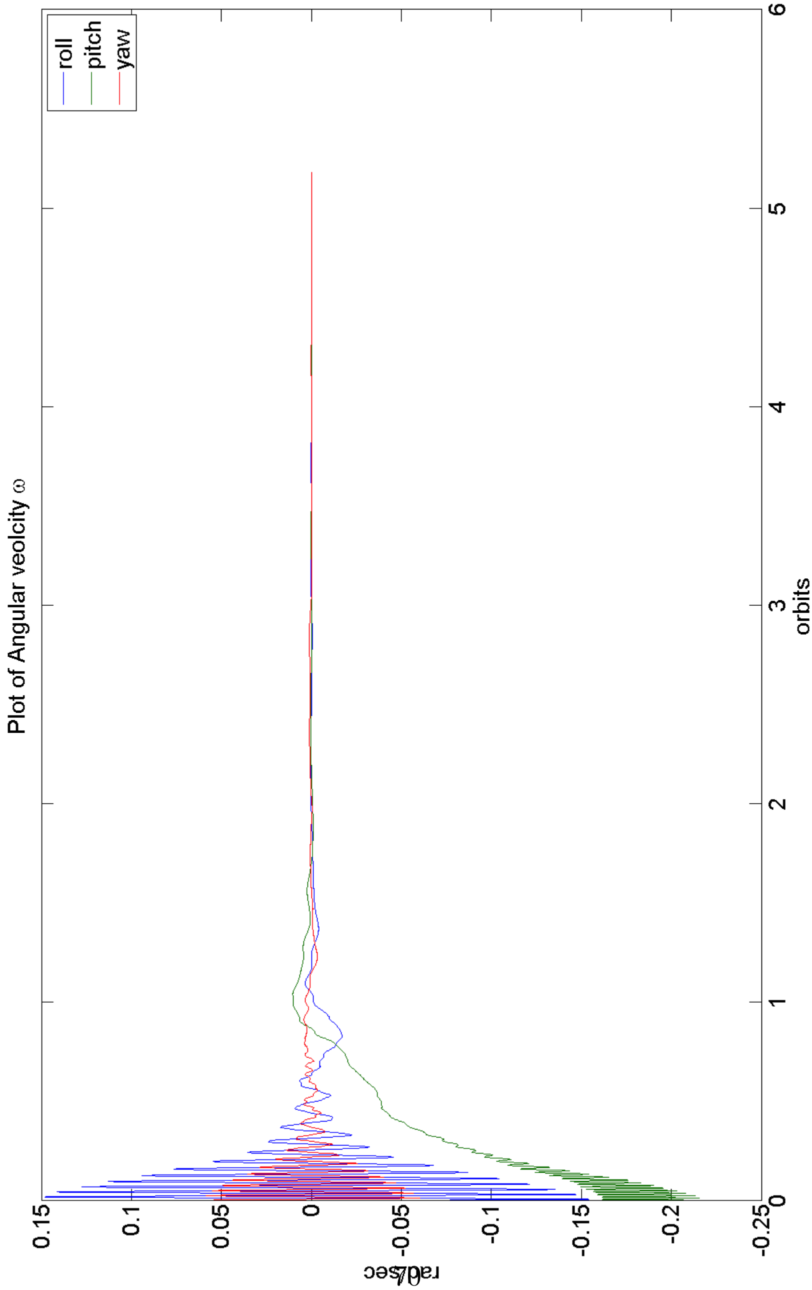

Figure 7.16: Angular velocity with noise on magnetometer feedback

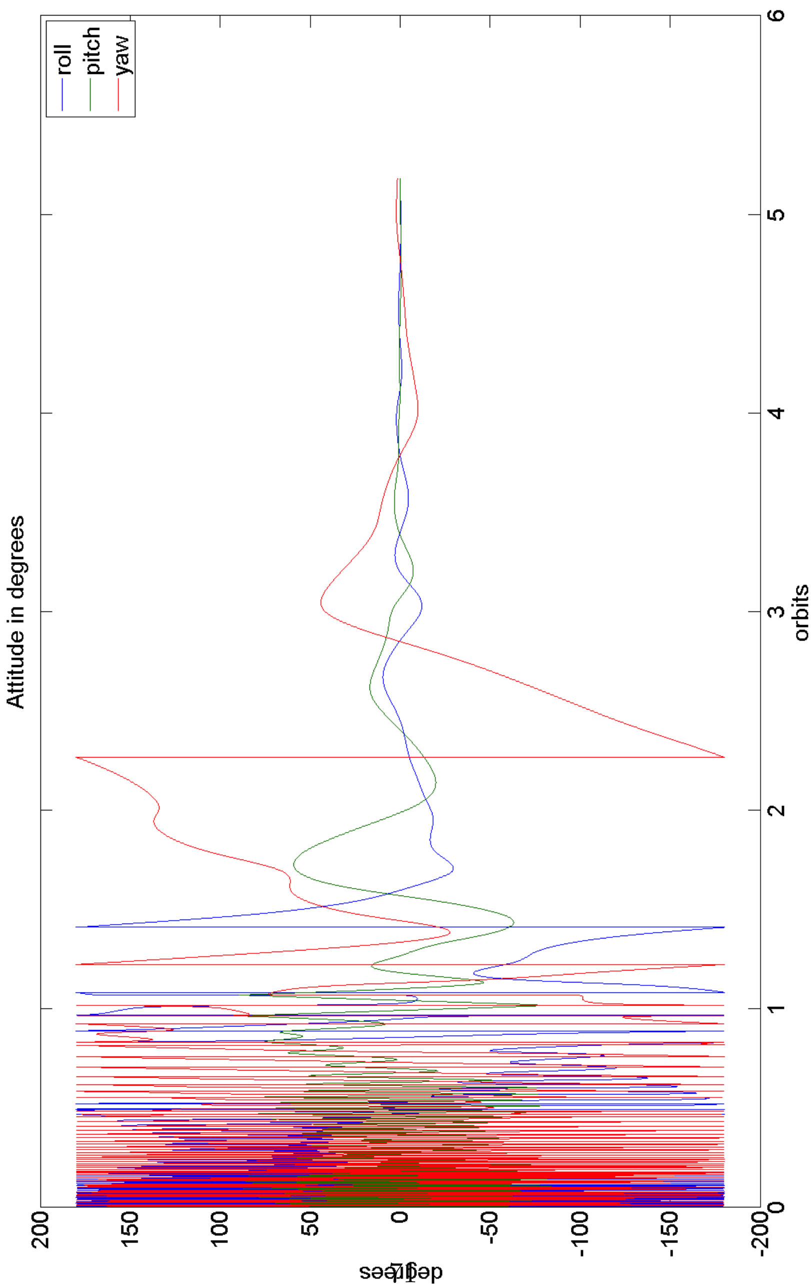

Figure 7.17: Attitude with noise on magnetometer feedback

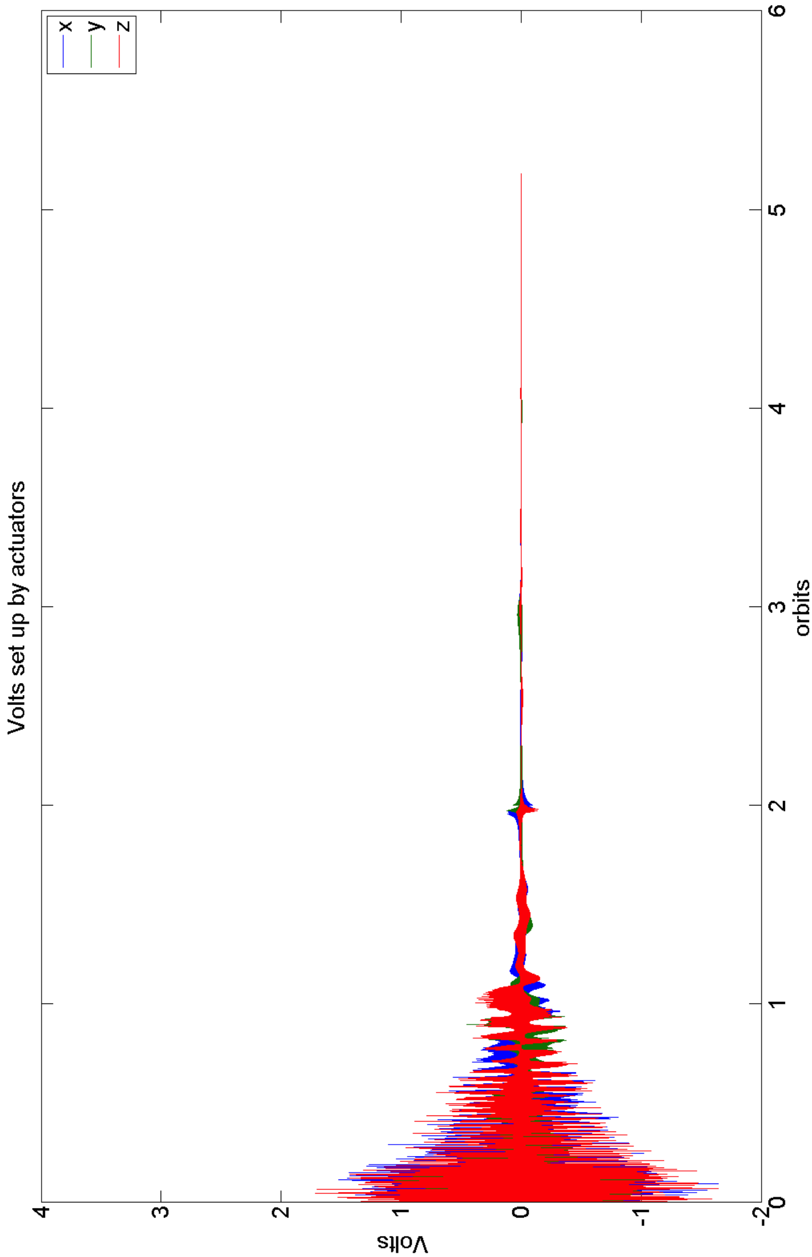

Figure 7.18: Voltage out put from controller with noise on magnetometer feedback

7.7 Controller on Arduino

To run only the controller on the Arduino, while the simulator was ran on the PC, was not realizable. This kind of hardware-in-the-loop (HIL) was not possible with the standard Simulink Coder tools. It is however possible to run the *entire* Simulink system on the Arduino. The result of this showed that the system was running used around one second per time-step, with a fixed time-step of 0.1.

Chapter 8

Discussion and conclusion

It is unknown exactly what the initial attitude and angular velocity will be in real life. It is however reasonable that the initial orientation will be quite "far off" the desired trajectory, but the initial angular velocity will most likely be lower then the simulations.

It is quite clear that the *Passive controller* 7.1 is not working to neither damp the system nor maintaining attitude. This was quite self explanatory before the simulation, but it should not underestimated as a tool to compensate for unwanted constant magnetic fields.

The *detumbling controller* 7.2 does an excellent job at detumbling the satellite. Both for high and low initial angular velocities. Clearly any angular velocity is punished, but the controller fails to correct the attitude. This is however expected, as the angular velocity goes down, the controller simply does not generate much torque.

By instead using *PD-controller* 7.3 it was able to correct the attitude much better. However not sufficiently. This might be a result of somewhat higher angular velocities that is not compensated for.

The theory suggested a *combination of detumbling and PD-control* 7.4. After several simulations it was found that the detumbling controller would have damped the angular velocity sufficiently after two orbits. The orbit propagator was then used to switch to the PD-controller. This demonstrates relatively good results. The angular velocity is extremely close to zero ($\pm 1 \cdot 10^{-5}$ rad/sec). The attitude is also very close with a deviation of $\pm 1°$. This was discussed with the NUTS-group in charge of the camera system. They can tolerate deviations up to $\pm 50°$ in x- and y-axis, so the attitude is well within limits.

The quaternions is behaving exactly the way the controller was intended. Slowly converging $\boldsymbol{q} \rightarrow\approx [1\,0\,0\,0]^\top$. The regulator is also within the limits of voltage output. It might look like this would consume a lot of battery, but the results clearly shows this is not the case.

When adding the disturbance caused by *aerodynamic drag* 7.5, the controller will manage to detumble, but it cannot correct the attitude. The first solution to this was to ad more gain to the controller, but this resulted in a too aggressive behavior and eventually an uncontrolled spin of the satellite. It is obvious that a different control strategy should have been used. The deviations is however tolerated by the camera system, as they are within the $\pm 50°$ limit in the x- and y-axis. But from a cybernetic perspective, such deviations should try to be avoided.

On the contrary, it is not necessary that the aerodynamic torque is correct. One also have to take into account that the torque is constant-worst-case.

For example it is more likely that the aerodynamic torque will decrease when the satellite closes in on the reference attitude, since the area facing the velocity is smaller, and the centre of pressure might be closer to the center of gravity.

In the last simulation 7.6 *noise from a simulated magnetometer* was added to the state feedback. Both the attitude and angular velocity shows nearly the same results as the combined detumbling and PD-controller. But the voltage generated by the controller tries to "follow" the magnetic feedback. This also results in a more noisy angular velocity. Clearly the regulator needs a filter. But real issue is that there is no limitation to how fast the voltage from the controller change with time. This is simply a flaw with the simulator.

As an extra simulation the whole system was ran on the *Arduino* 7.7. Research was done to perform both hardware-in-the-loop (HIL) and processor-in-the-loop. This was not supported with the regular procedures of Simulink Coder. More as an interesting test, the entire Simulink model was ran on the Arduino. The code was able to compile and run without any errors. Simulation time was however very long (about 0.1 time-step = 1 second). It is however well known that a PD-controller can run on a micro controller of this type 3.2. So the test is more interesting if the code should later be extended with more complex functions for the controller.

8.1 Conclusion

Designing a satellite attitude control system for the NUTS requires an extensive knowledge of how the satellite project will be carried through.

The environment of operation has strict boundaries, and the satellite has a limited amount of space for both actuators and sensor systems. Knowing these limitations it is still possible to use standard theory and definitions for both mathematical modeling and controller design.

The Simulink Simulator is a sufficiently good "sandbox" to test different control strategies. It has some simplifications in terms of the magnetic field, and it has some flaws in the voltage output. All of this can however be fixed, since the simulator is quite modular and scalable.

The final full state feedback PD-controller shows adequate results, except for the inclusion of aerodynamic torque. This reason for this is considered a combination of the controller design and a too harsh aerodynamic torque.

Using the Arduino platform for performance testing was not conclusive, as HIL-testing was not achieved. The Simulink code is however considered valid to be compiled for this platform.

8.2 Future work

The Simulink simulator should first of all be expanded to include more realistic environment models like; IGRF look-up-tables, a more dynamic aerodynamic torque, final sensor assembly of NUTS satellite (and the noise/bias data of these) and more realistic disturbance on magnetic sensors when using the actuators and radio systems. If the simulator is expanded to include such realistic measurements, a filter or state estimator should be part of the system. A real feedback from the actuators affecting the magnetometer should also be taken into account. A possible solution to run a HIL test with the Arduino, is to acquire real sensors that can act as input, instead of the simulator. Finally, the PD controller should also be expanded (possibly integral action) to compensate for the large deviations caused by aerodynamic torque.

Bibliography

[1] Thor I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*, John Wiley and Sons, 1st ed. 2011 (ISBN 9781119991496).

[2] Thor I. Fossen, *Mathematical models for control of aircraft and satellites*, Department of Engineering Cybernetics NTNU, 2nd ed. 2011, http://www.itk.ntnu.no/fag/TTK4190/lecture_notes/2012/Aircraft%20Fossen%202011.pdf

[3] Zdenko Tudor, *Design and implementation of attitude control for 3-axes magnetic coil stabilization of a spacecraft*, Institute of Engineering Cybernetics NTNU, 2011.

[4] Kristian Lindgrd Jenssen and Kaan Huseby Yabar, *Development, Implementation and Testing of Two Attitude Estimation Methods for Cube Satellites*, Department of Engineering Cybernetics NTNU, 2011.

[5] Peter C. Hughes, *Spacecraft Attitude Dynamics*,
John Wiley and Sons, 1986 (ISBN 0-471-81842-9).

[6] Samir Rawashdeh, David Jones, Daniel Erb, Anthony Karam and James E. Lumpp, Jr., *Aerodynamic attitude stabilization for a ram-facing cubesat*,
University of Kentucky, Lexington, 2009.

[7] John Ting-Yung Wen and Kenneth Kreutz-Delgado, *The Attitude Control Problem*,
Transactions on automatic control, IEEE, 1991.

[8] Arati S. Deo and Ian D. Walker, *Overview of Damped Least-Squares Mehtods for Inverse Kinematics of Robot Manipulators*,
Department of Electrical and Computer Engineering, Rice University, Houston, 1993.

[9] Eli Jerpseth Overby, *Attitude control for the Norwegian student satellite nCube*,
Department of Engineering Cybernetics, NTNU, 2004.

[10] Per Kolbjorn Soglo, *3-aksestyring av gravitasjonsstabilisert satellite ved bruk av magnetspoler*,
Institutt for Teknisk Kybernetikk, NTH, 1994.

[11] Honeywell, *3-Axis Digital Compass IC HMC5983 advanced information*,
Honeywell International Inc., 2012.

[12] James R. Wertz and Wiley J. Larson, *Space Mission Analysis and Design*,
W. J. Larson and Microcosm, Inch, Third Edition, 1999.

[13] Bjornar Vik, *Integrated Satellite and Inertial Navigation Systems*,
Department of Engineering Cybernetics, NTNU, 2012.

Appendix A

Stability analysis

Lyapunov theory can be used to both to test stability of a system, and to arrive at a stabilizing control law. Both Soglo [10] and Tudor [3] uses this to different extents, and some of their results will here be presented.

$$
\begin{aligned}
V &= \frac{1}{2}\boldsymbol{\omega}_{ob}^{b\top}\boldsymbol{I}\boldsymbol{\omega}_{ob}^{b} + \frac{3}{2}\omega_o^2\boldsymbol{c}_3^\top\boldsymbol{I}\boldsymbol{c}_3 - \frac{1}{2}\omega_o^2\boldsymbol{c}_2^\top\boldsymbol{I}\boldsymbol{c}_2\frac{1}{2}\omega_o^2(\boldsymbol{I}_y - 3\boldsymbol{I}_z)\\
\frac{\partial V}{\partial t} &= \dot{V} = \boldsymbol{\omega}_{ob}^{b\top}\boldsymbol{\tau}_m^b \leq 0 \quad \forall\,\boldsymbol{\tau}_m = 0
\end{aligned}
$$

This means the system is stable in a Lyapunov sense (with zero input)

With controller, the Lyapunov function can be extended to

$$
V = \frac{1}{2}\boldsymbol{\omega}_{ob}^{b\top}\boldsymbol{I}\boldsymbol{\omega}_{ob}^{b} + \frac{3}{2}\omega_o^2\boldsymbol{c}_3^\top\boldsymbol{I}\boldsymbol{c}_3 - \frac{1}{2}\omega_o^2\boldsymbol{c}_2^\top\boldsymbol{I}\boldsymbol{c}_2\frac{1}{2}\omega_o^2(\boldsymbol{I}_y - 3\boldsymbol{I}_z) + k[\boldsymbol{\epsilon}^\top\boldsymbol{\epsilon} + (1-\eta)^2]
$$

Here $k > 0$. Because of the quadratic expression, the extension will be; zero for the correct attitude ($\boldsymbol{\epsilon} = 0 \wedge \boldsymbol{\eta} = 1$), positive definite for other attitudes.

Soglo and Tudor are also using the fact that $\boldsymbol{\epsilon}^\top \boldsymbol{\epsilon} + \eta^2 = 1$. This means it is possible to rewrite the last term as $2k(1-\eta)$. The time derivative of this is $-2k\dot{\eta}$. Accordingly, the the new expression for $\dot{V}$ is

$$\dot{V} = \boldsymbol{\omega}_{ob}^b[k\boldsymbol{\epsilon} + \boldsymbol{\tau}_m^b]$$

with the expression for the control law

$$\boldsymbol{\tau}_m^b = -d\boldsymbol{\omega}_{ob}^b - k\boldsymbol{\epsilon} \quad \text{with gain} \quad d > 0,\, k > 8\omega_0^2(I_y - I_z) > 0$$

That is the same controller as found chapter 5.[1]

[1]For the values used in the simulator, the k-gain should be 8×10^{-9} or higher.

Appendix B

Matlab Simulink code

The *init.m* script for setting up all the parameters and defining initial values.

```
clc
clear all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parameters for earth and orbit %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

m  = 2.6;                    % [kg] Satellite mass
dx = 0.11;                   % X-axis length
dy = 0.1;                    % Y-axis length
dz = 0.2;                    % Z-axis length
Re = 6371.2e3;               % [m] Earth radius
Rs = 600e3;                  % [m] Satellite altitude
Rc = Re+Rs;                  % [m] Distance from earth ...
```

```
    center to satellite
G = 6.67428e-11;              % Earth gravitational constant
M = 5.972e24;                 % Earth mass
w_O = sqrt(G*M/Rc^3);         % Satellite angular velocity ...
    relative Earth
w_O_IO = [0 -w_O 0]';         % Velocity of orbit coordinate ...
    system


%%%%%%%%%%%%%%%%%%%%%%
% Moments of inertia %
%%%%%%%%%%%%%%%%%%%%%%

Ix = (m/12)*(dy^2+dz^2);      % X-axis inertia
Iy = (m/12)*(dx^2+dz^2);      % Y-axis inertia
Iz = (m/12)*(dx^2+dy^2);      % Z-axis inertia

% The inertia matrix
Inertia = diag([Ix Iy Iz]);


%%%%%%%%%%%%%%%%%%%%%%%
% Initial orientation %
%%%%%%%%%%%%%%%%%%%%%%%

% Set the satellites inital attitude in degrees (set by user)
roll_degrees_0       = 10;
pitch_degrees_0      = 5;
yaw_degrees_0        = -2;

%roll_degrees_0       = 0;
%pitch_degrees_0      = 0;
%yaw_degrees_0        = 0;

```

```
% Converting to radians
roll = (pi/180)*roll_degrees_0;
pitch = (pi/180)*pitch_degrees_0;
yaw = (pi/180)*yaw_degrees_0;

% Converting to quaternions (set as inital value in ...
    Kinematics integrator)
q_0 = euler2q(roll, pitch, yaw);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initial angular velocity %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Initial angular velocity in radians per second (set by user)
%w_B_OB_0 = [0 0 0]';
%w_B_OB_0 = [0.0005 0.0003 -0.003]';
w_B_OB_0 = [0.1 -0.2 0.05]';

%Calculates inital rotation matrix from body to orbit
%(This rotation matrix = I if q_0 = [+-1 0 0 0])
R_o_b = Rquat(q_0);
R_b_o = R_o_b';

%c1 is the first column vector in R_b_o
%This can be used to calculate the inital
%angular velocity relative intertia frame
c2 = R_b_o(:,2);
c1 = R_b_o(:,1);
w_B_IB_0 = w_B_OB_0+w_O*c1


%%%%%%%%%%%%%%%%%%%%
% Regulator gains %
```

```
%%%%%%%%%%%%%%%%%%%

%K_p = 0;
%K_d = 4e-5;

K_p = 5e-8;
K_d = 4e-5;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Coil parameters and controll allocation matrix %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Number of turns with copper thread in each coil
Nx = 355;
Ny = 355;
Nz = 800;

% Area of each coil
Ax = 0.08*0.18;
Ay = 0.08*0.18;
Az = 0.08^2;

% Resistance in each coil
Rx = 110;
Ry = 110;
Rz = 110;

% K matrix - controll allocation matrix for regulator
K = [(Nx*Ax)/Rx 0 0; 0 (Ny*Ay)/Ry 0; 0 0 (Nz*Az)/Rz];


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The dipole megnetic field of Earth %
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Permeability of vacuum
my = 1e17/(4*pi);

% Magnitude of magnetic field
B_null = my/(Rc^3);
```

The *detumbling* algorithm inside the controller block

```
function V = fcn(K,w_b_ob,B_b)
% Controller gain
d = K_p;

% Moment set up by controller
m_B = -(d/norm(B_b,2)^2)*cross(B_b,w_b_ob);

% Torque set up by coils
V = (K^-1)*m_B;
```

The *reference controller* algorithm inside the controller block

```
function V = fcn(K,B_b,w_b_ob,q)

eps = q(2:4);

% Controller gains
d = K_d;
k = K_p;

% Moment set up by controller
```

```
m_B = ...
    (1/norm(B_b,2)^2)*(-d*cross(B_b,w_b_ob)-k*cross(B_b,eps));

V = (K^-1)*m_B;
```

The *Aerodynamic torque* algorithm as input to the satellite

```
%CODED BY GAUTE BRAATHEN 2012

function tau_a = fcn(R_B_O,w_B_OB,w_O,I)


m = 2.6;                    % [kg] Satellite mass
jx = 0.03;  %0.05           % X-axis length
jy = 0.03;  %0.049          % Y-axis length
jz = 0.083;                 % Z-axis length
Jx = (m/12)*(jy^2+jz^2);    % X-axis inertia
Jy = (m/12)*(jx^2+jz^2);    % Y-axis inertia
Jz = (m/12)*(jx^2+jy^2);    % Z-axis inertia
J = diag([Jx Jy Jz]);       % Inertia matrix

x = 0.1;
y = 0.1;
z = 0.2;
A_xo = x*z;                 % Outer satellite area, x-panel
A_yo = y*z;                 % Outer satellite area, y-panel
A_zo = x*y;                 % Outer satellite area, z-panel
A_drag = z*sqrt(x^2 + y^2); % Maximum area for calc ...
    disturbance drags

Re = 6371.2e3;              % [m] Earth radius
Rs = 600e3;                 % [m] Satellite altitude
```

```
Rc = Re+Rs;                    % [m] Distance from earth ...
    center to satellite

orbitPeriod = (2*pi)/(w_O); % For defining simulation length

vel = 2*pi*Rc/orbitPeriod;  % Orbit velocity

rho_a = 4.89e-13;

c_p = [0.02 0.02 0.02]';
c_p_x = Smtrx(c_p);
%Vr = R_B_O*[-1 0 0]';
Vr = R_B_O*[0 -1 0]';
Vr_x = Smtrx(Vr);

tau_a = rho_a*vel*( vel*A_drag*c_p_x*Vr - (I + ...
    Vr_x*J)*w_B_OB );
```

Appendix C

Simulink model

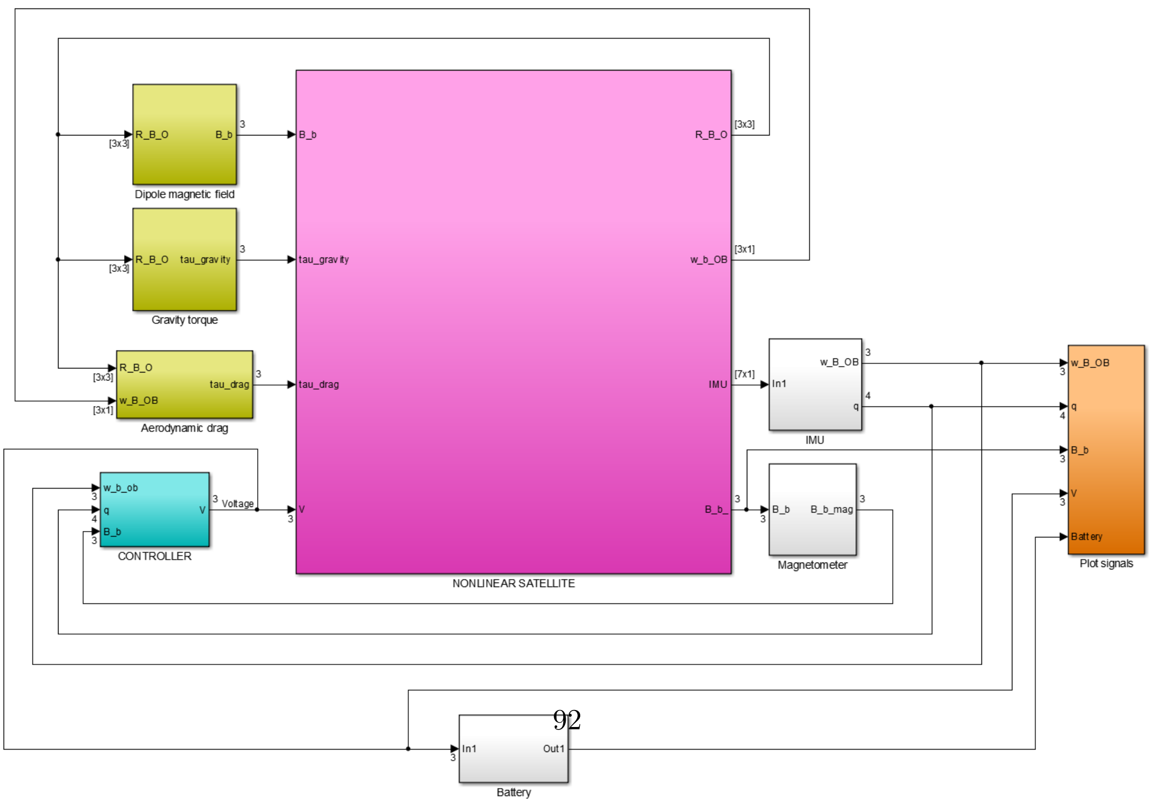

Figure C.1: The complete Simulink model unmasked

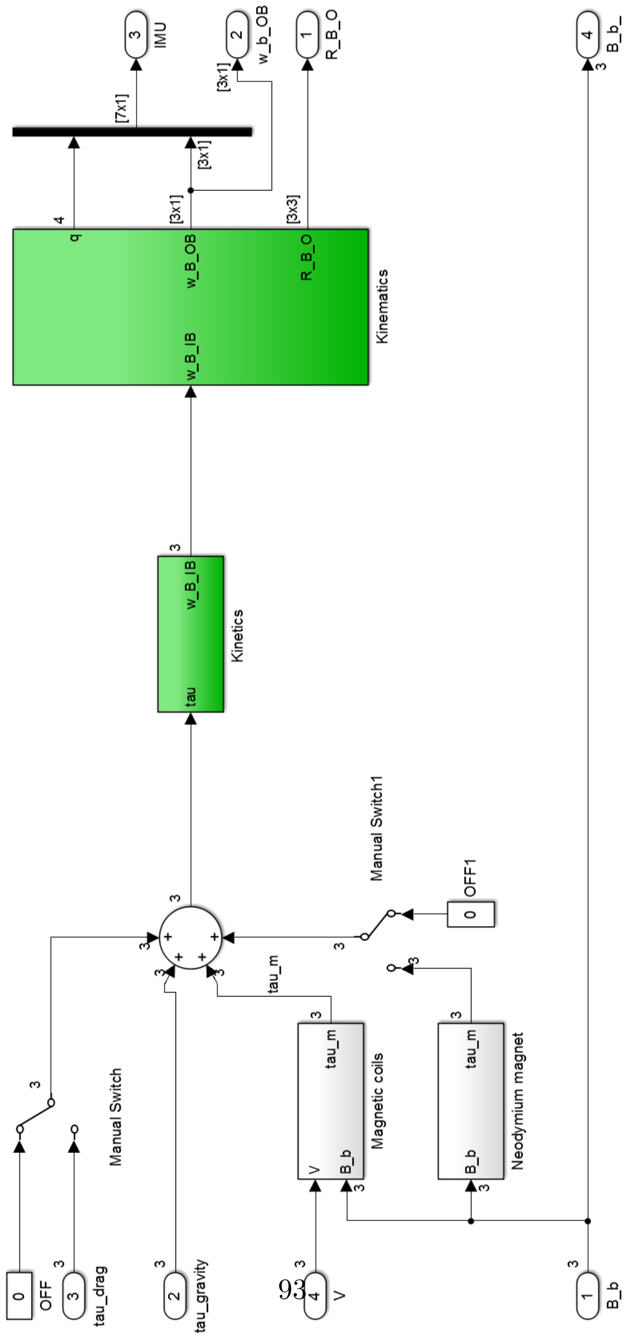

Figure C.2: The Nonlinear Satellite block

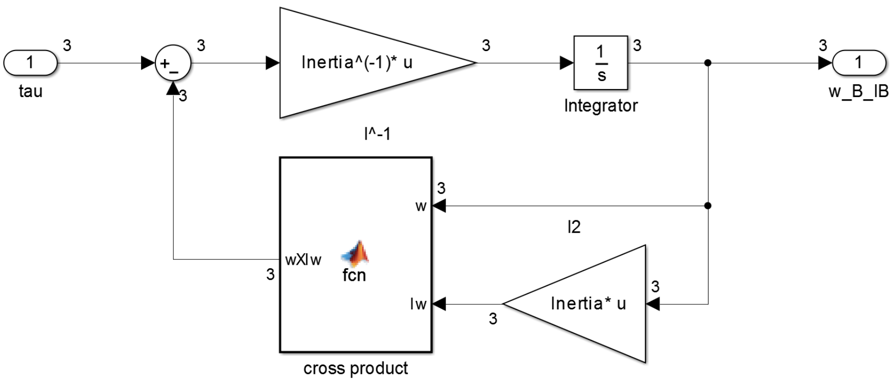

Figure C.3: The Kinetics block

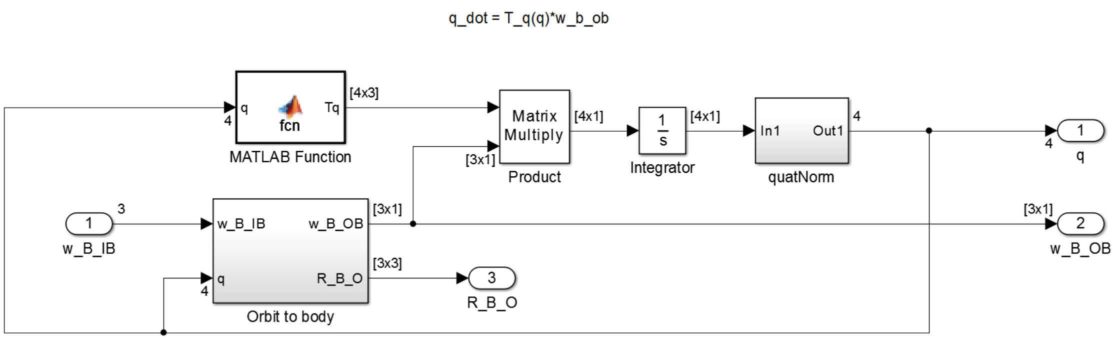

Figure C.4: The Kinematics block

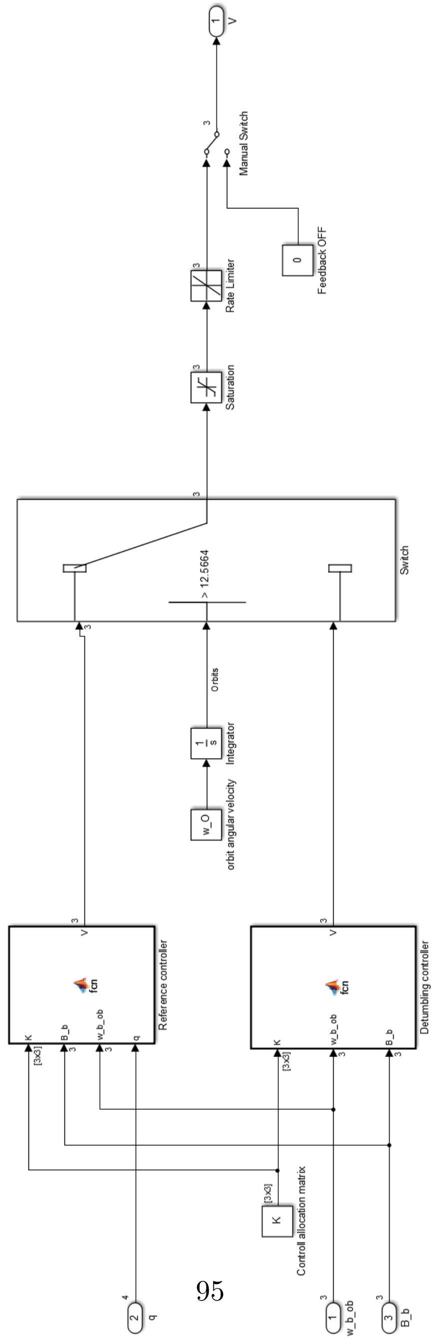

Figure C.5: The Controller block