NTNU – Trondheim
Norwegian University of
Science and Technology

Attitude Controller-Observer Design for the NTNU Test Satellite

Fredrik Alvenes

Master of Science in Engineering Cybernetics
Submission date: June 2013
Supervisor: Thor Inge Fossen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Faculty of Information Technology,
Mathematics and Electrical Engineering
Department of Engineering Cybernetics

MSC THESIS DESCRIPTION SHEET

Name:	Fredrik Alvenes
Department:	Engineering Cybernetics
Thesis title (Norwegian):	Attityde Reguleringssystem og Observer design for NTNU Test Satellitt
Thesis title (English):	Attitude Controller-Observer Design for the NTNU Test Satellite

Thesis Description: The purpose of the thesis is to develop and simulate a satellite attitude control system. The control system should use physical measurements and unmeasured states should be estimated using an observer.

The following items should be considered:

1. Literature study on attitude estimators and control systems for satellites.
2. Construct a satellite simulator in Matlab for generation of physical measurements and include realistic noise models. The simulator should also include actuator dynamics.
3. Include options for simulation of failure modes in the simulator.
4. Design a nonlinear attitude controller using output feedback. Unmeasured states should be estimated using a nonlinear observer.
5. Simulate the controller-observer in Matlab and include experimental results (if possible).
6. Conclude your results.

Start date:	2013-01-14
Due date:	2013-06-10

Thesis performed at:	Department of Engineering Cybernetics, NTNU
Supervisor:	Professor Thor I. Fossen, Dept. of Eng. Cybernetics, NTNU

Abstract

This report presents the results from the development and design of an Attitude Controller-Observer for 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 strategy 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. An attitude observer is developed, benchmarked and implemented in both a simulator and hardware.

Combined, these findings are put to use in the form of a complete Simulink simulator for the satellite in orbit. Results from the control strategy, environmental models and attitude estimation are given. A fully working algorithm for the attitude observer, developed on a microcontroller, is also included.

Sammendrag

Denne rapporten presenterer resultatene fra utvikling og design av et attityde reguleringssystem med observer for satellitten NTNU Test Satellite (NUTS). Den gir en innsikt i matematisk modellering av satellitt attitude dynamikk for tre frihetsgrader. Basert på begrensningene i miljøet NUTS opererer, er disse modellene justert tilsvarende.

En strategi for attitude kontroll er presentert. Gjennom en forklaring av de magnetiske aktuatorene, er kontroll-lovene tilpasset NUTS satellitten. En attitude observer er utviklet, testet og implementert i både en simulator og på embedded maskinvare.

Kombinert er disse funnene tatt i bruk i form av en komplett Simulink simulator for satellitt i bane. Resultater fra kontroll strategien, miljø-modeller og attitude estimering er gitt. En fullverdig algoritme for attitude observeren, utviklet på en mikrokontroller, er også inkludert.

Preface

This project thesis is the result of the work done in TTK4900 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. Torleiv Håland Bryne from the Cybernetics 2013 class has been most motivating and helpful. I wish him the best of luck on his Master Thesis and future PhD work.

Bernt-Johan Bergshaven has provided his excellent skills when programming the IMU. Terje Haugen, at the Department of Engineering Cybernetics Workshop, has been most helpful creating the 3D prints. Thank you very much to both of you.

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

List of Tables

Nomenclature

$\boldsymbol{\epsilon}$	Imaginary part of unit quaternion
$\boldsymbol{\omega}$	angular velocity in roll, pitch and yaw
$\boldsymbol{\omega}_{imu}^{b}$	Angular velocity data from gyroscope
$\boldsymbol{\omega}_{mes}$	Observer injection term based on sensor data
$\boldsymbol{\omega}_{orbit}$	Angular velocity of the orbit coordinate system
$\boldsymbol{\tau_a}$	Aerodynamic torque acting on the satellite
$\boldsymbol{\tau_d}$	Desired torque calculated by controller
$\boldsymbol{\tau_g}$	Gravity torque acting on the satellite
$\boldsymbol{\tau_m}$	Actuator torque acting on the satellite
$\boldsymbol{B}_b$	Earth's magnetic field measured in the BODY coordinate system
$\boldsymbol{I}_{CG}$	Inertia tensor about the centre of gravity
$\boldsymbol{K}_{coil}$	Control allocation matrix

$\boldsymbol{m}_{mag}^{b}$	Magnetic field measured from magnetometer
$\boldsymbol{m}_{b}$	Magnetic field vector in BODY generated by actuators
$\boldsymbol{q}$	Attitude represented as 4-dimensional quaternion
$\boldsymbol{R}_{n}^{b}$	Rotational matrix from BODY to NED coordinate system
$\boldsymbol{R}_{o}^{b}$	Rotational matrix from BODY to ORBIT coordinate system
$\boldsymbol{R}_{s}^{o}$	Rotation matrix from ORBIT to Sun
$\boldsymbol{s}^{o}$	Sun sensor output rotated to the BODY frame
$\boldsymbol{s}^{s}$	Sun sensor output normal to the Sun
$\boldsymbol{T}(\boldsymbol{q})$	4-by-3 matrix part of angular velocity transformation
$\boldsymbol{V}$	Voltage over the three actuator coils
$\boldsymbol{v}_{i}^{b}$	Directional sensor data in Mahony Observer
$\boldsymbol{v}_{0i}^{n}$	Directional reference vector in Mahony Observer
$\dot{\hat{\boldsymbol{b}}}_{gyro}^{b}$	Gyroscope bias estimate
η	Real part of unit quaternion
$\hat{\boldsymbol{q}}$	Attitude estimate
$\hat{\boldsymbol{v}}_{i}^{b}$	Estimated vector measurment in body
$\hat{\boldsymbol{v}}_{i}^{b}$	Sensor data reference based on attitude estimate
k_{d}	Derivative gain in controller

k_p Proportional gain in controller

R Resistance in an actuator coil

BODY Reference frame in centre of gravity of the satellite

ECEF Earth-centered Earth fixed reference frame

ECI Earth-centered inertial reference frame

GNSS Global Navigation Satellite System

GPIO General Purpose Input/Output

GPS Global Positioning System

IGRF International Geomagnetic Reference Field

LUT Look-up-table

MCU Microcontroller Unit

NUTS NTNU Test Satellite

ORBIT Reference frame following the orbit of the satellite

RMS Root Mean Square

Contents

Chapter 1

Introduction

1.1 Background and motivation

When applying for a valid Master Thesis Project, an effort was made to find a project regarding attitude estimation of unmanned aerial vehicles (UAV's). This seemed problematic, as flying requires closed airspace and government approved pilots. During this time an initiative towards recruiting cybernetics students was made by the NUTS team. Having completed the two introduction courses on space technology, the choice was an easy choice. The NUTS project is also a great opportunity to apply the knowledge from the *Navigation and Vessel Control Systems* branch of Cybernetics.

Working with systems that actually will be launched into space, and complete a given task from orbit, is motivation enough by itself.

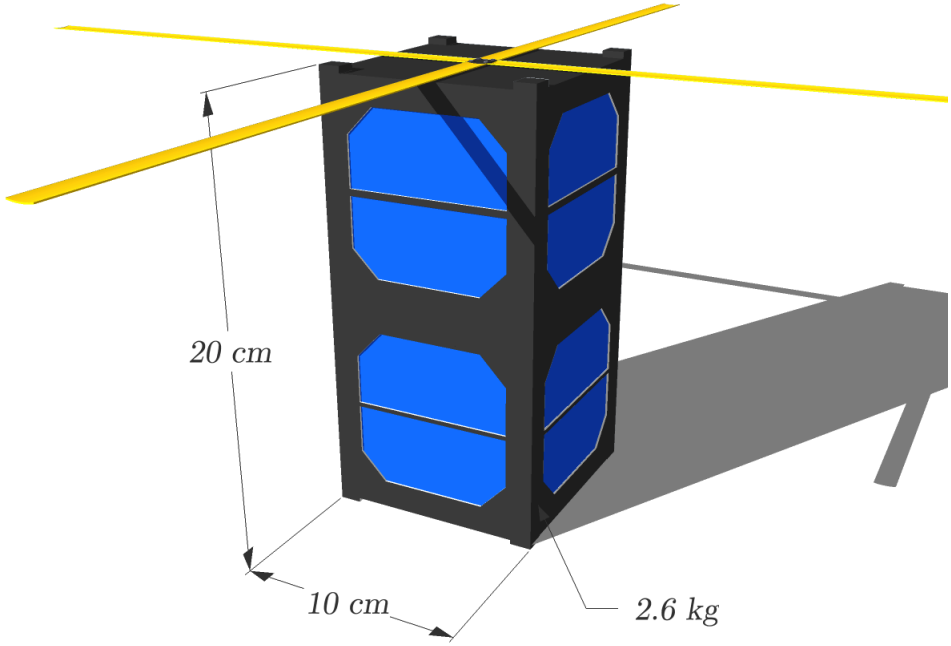

Figure 1.1: CAD model of the NUTS satellite

1.2 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 of 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 (P-pods).

The orbit used will be a *sun synchronous orbit*; going around Earth while 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 every day.
- 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 approximately 5800 seconds

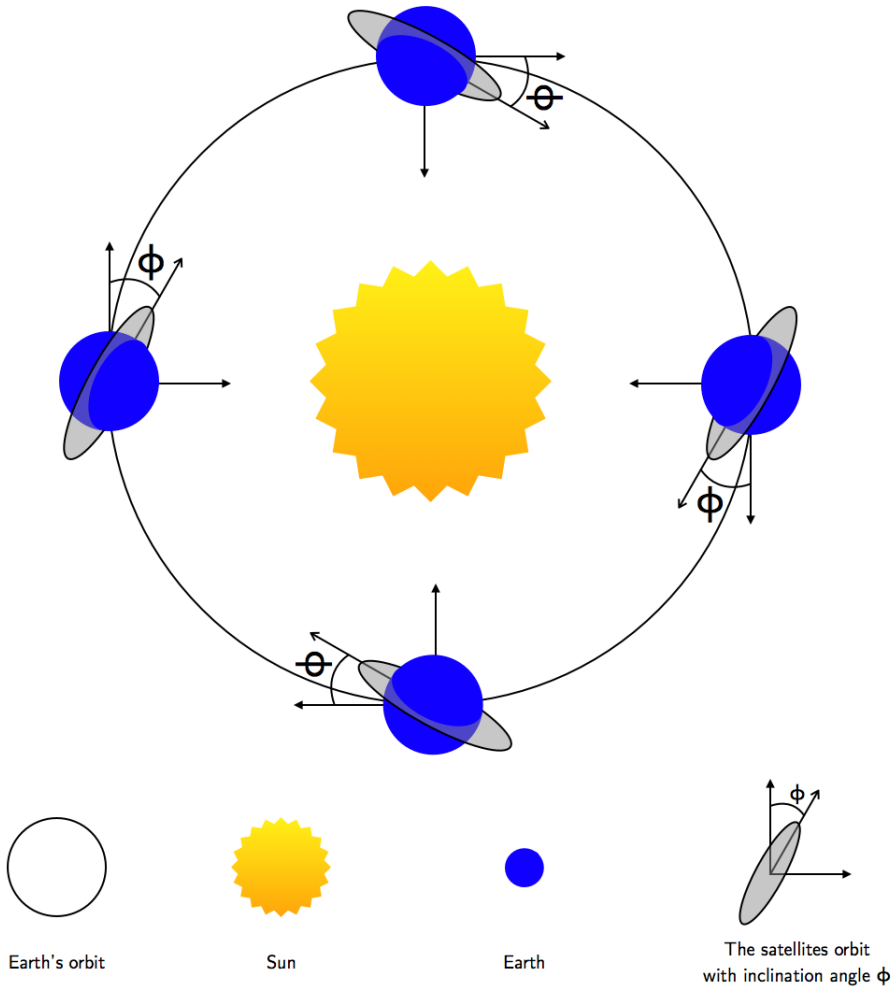

Figure 1.2: Sun synchronous orbit

The main mission of the satellite is 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.

1.3 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 atmosphere, 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.

Attitude estimation is also an important part of the ADCS. Measurements can be noisy, discontinuous and biased.

1.4 PreviousWork

This master is the continuation of the Project work TTK4550 done in 2012. This project picks up the conclusions and results. Partially to be able to dig even further into the experiments, but also to push the simulator, algorithms and tools developed, into real world applications and units.

TTK4550 was concentrated on an introduction to how the NUTS satellite operates in space; in terms of actuators, sensor-systems control theory.

A simulator was built in Matlab Simulink. It includes kinetics, kinematics, environmental models, physical hardware limitations and regulator algorithms. Results from this simulation showed that is is perfectly possible to regulate the attitude of a 2-unit cubesat in orbit with the help of three magnetic coils. Some assumption where however made, and the most important are listed here:

- The attitude PD-controller must receive accurate attitude quaternions from the IMU
- The magnetometer has a deterministic bias from other components and/or actuators
- The atmospheric drag torque is relatively small

Given these assumptions, the PD-controller was shown to be asymptotically stable.

1.5 Status of NUTS ADCS systems

Attitude control systems for the NTNU Test Satellite have been worked on since the first initializations of a satellite program at NTNU. The third and current satellite from NTNU (NUTS) is a 2-unit design with magnetic actuators. Many master thesis's has been published with different proposals on ADCS design, like for example gravity booms. The current focus is to use magnetic actuators, and that is also what the majority of students proposes.

Per Kolbjorn Soglo [10] published in 1994 his master thesis that helps define many of the physical properties with magnetic actuated cubesat design for the NUTS. His topics does however not mention attitude estimation from sensor output.

A full Matlab Simulink simulator for a cubesat was published by Eli Jerpseth Overby [9] in 2004. This simulator was used as a reference design for this Master Thesis' simulator.

In 2011 Kristian Lindgard Jenssen and Kaan Huseby Yabar [4] published one of the most thorough master thesis on the NUTS ADCS to this date. A full development and implementation of the EQUEST algorithm was given. The algorithm was developed to work on a standard Atmel microcontroller and compared with the performance of an EKF. They demonstrate optimistic results, however the EQUEST algorithm increased in complexity when trying to compensate for noise and include gyroscope data.

Toril Bye Rinnan [14] published in 2012 a proposal to "kickstart" an Nonlinear Grip Observer with the EQUEST algorithm from Jenssen and Yabar. The results are very promising, and gives an excellent demonstration of how to get the best of both worlds. This will however add extra complexity to the system, and it is not suggested how to maintain a robust system on the final microcontroller (like for example a real time operating system).

The NUTS projected has an ongoing cooperation with the Aalborg University in Denmark. In 2011 Vinther, Jensen, Larsen and Wisniewski [15] published the workings of the ADCS system running on their recently launched $AAUSAT_3$ cubesat. This is a commercial system based on an Unscented Kalman Filter. This filter uses $50 - 60\%$ of the capacity on an ARM7™ micro controller. This an MCU with almost three times the capacity of the proposed NUTS MCU. Commercial systems are neither an option for the NUTS satellite.

So far the NUTS project does not have any prototypes running the full scale ADCS systems. Only bits and pieces are delivered either in the form of data simulations or as prototype embedded microcontrollers with limited functionality.

1.6 Target goal of Master Thesis

The primary goal of this Master Thesis is to explore, develop and prototype an ADCS system for the NUTS satellite. This requires a full breakdown of the kinetics/kinematics, sensor systems, orbital/environmental parameters and microcontroller/embedded theory. The NUTS will be made almost entirely "in house" by students, so buying commercial systems (apart from sensors and MCU's) is not an option.

1.7 Outline of the thesis

- **Chapter 2** Defines the orbit in which the NUTS will operate. A brief introduction to how the coils will operate in this magnetic environment. A presentation of the kinetics and kinematics gathered from different sources. This theory is also fused with environmental torques such as gravity and aerodynamic drag.
- **Chapter 3** Lists the requirement specification given by the NUTS team, regarding the ADCS system.
- **Chapter 4** Explains the development of the Actuators. This is based on magnetic torque theory. The polar orbit of the NUTS sets limitations on the actuators. This chapter also explains how these challenges where met.
- **Chapter 5** Presents the control strategy for the NUTS satellite and a short stability analysis.
- **Chapter 6** Gives an overview of possible sensors and systems for attitude estimation. A study on different attitude estimation algorithms are provided, with a conclusive choice. The attitude estimation is adapted to the NUTS sun sensor. Stability properties are also given.

- **Chapter 7** An overview of the construction and functionality of the Matlab Simulink Simulator. Together with how the different sensors where modeled.
- **Chapter 8** A breakdown of the details regarding the microcontroller used for attitude estimation. Here are also specifications for the MEMS sensors, together with how the Mahony Observer was written in C-code to work on the embedded platform. Also how the logging of data was carried out.
- **Chapter 9** The simulator with a full ADCS system is tested and verified for a different test cases. Data and verification of the Mahony Observer running on the embedded platform are provided. The microcontroller is also benchmarked.
- **Chapter 10** Discusses the results from the simulation and hardware implementation in chapter 9.
- **Chapter 11** Concludes the work done in this thesis.
- **Chapter 12** Gives advice on further development and improvements.

Chapter 2

Theory and definition

2.1 Coordinate Frames

Different coordinate systems are necessary to represent attitude and position *relative* to each other.

- **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 (ECEF)** 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 to 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 (2.3)). z is the axis of minimum inertia (where the camera points out). x is defined as direction of travel.

The design of the the NUTS satellite mainly invokes the BODY and ORBIT coordinate systems.

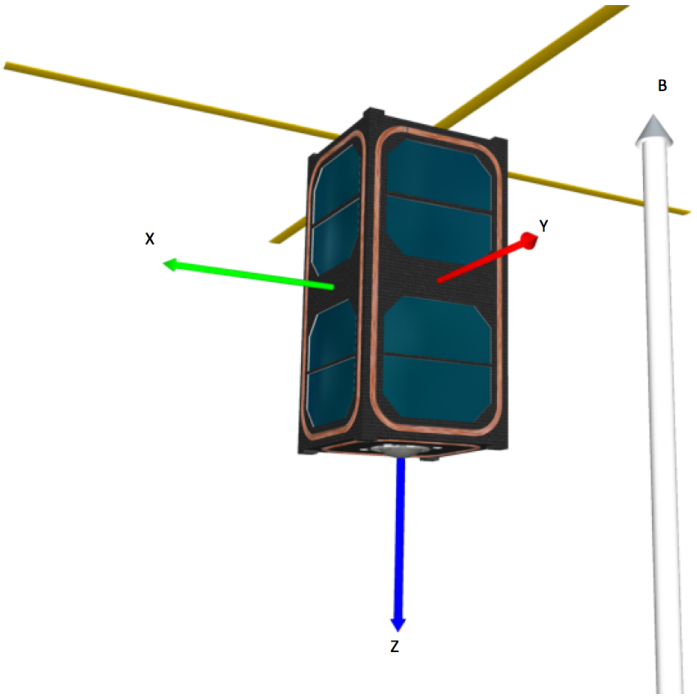

Figure 2.1: NUTS with BODY coordinate system and magnetic field B

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 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 a medium for actuation (see section (4.1)). 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

Θ Polar angle

L Maximum degree of the expansion

a Earth's radius

ϕ East longitude

g_ℓ^m, h_ℓ^m Gauss coefficients

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)
```

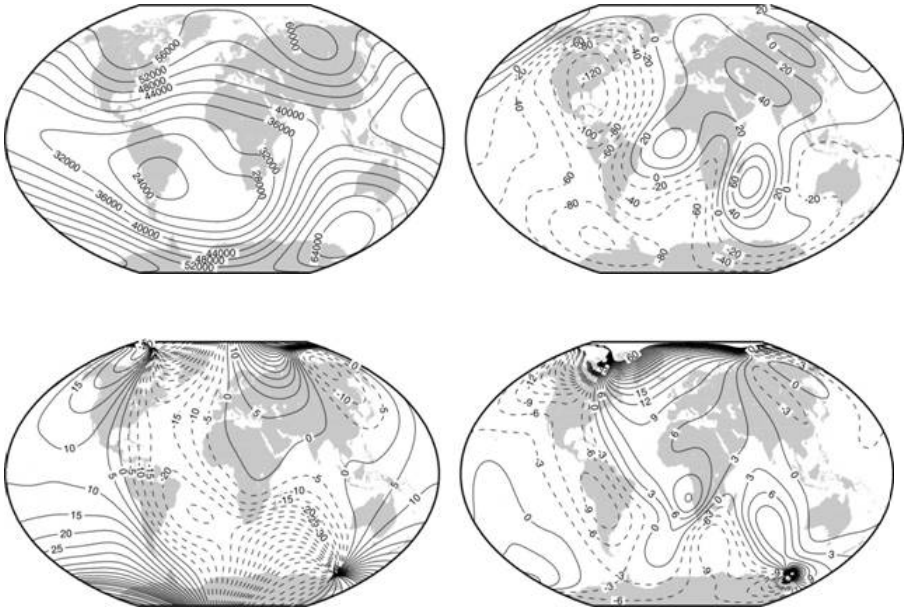

Figure 2.2: IGRF magnitude *Image courtesy of British Geological Survey*

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 (4.2)).

There is a main practical difference between a dipole field and the IGRF

in low earth orbit (160 - 2000 km). If the Earth had 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 the satellite is 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 an uneven magnitude.

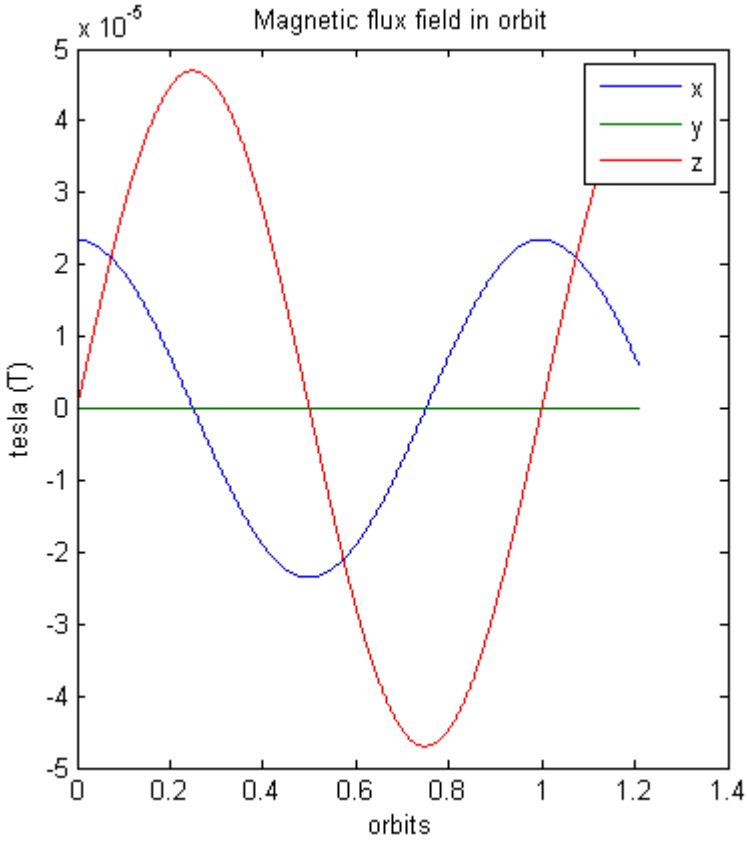

Figure 2.3: Magnetic flux from Earth when modeled as a dipole. Measured in the orbit coordinate system

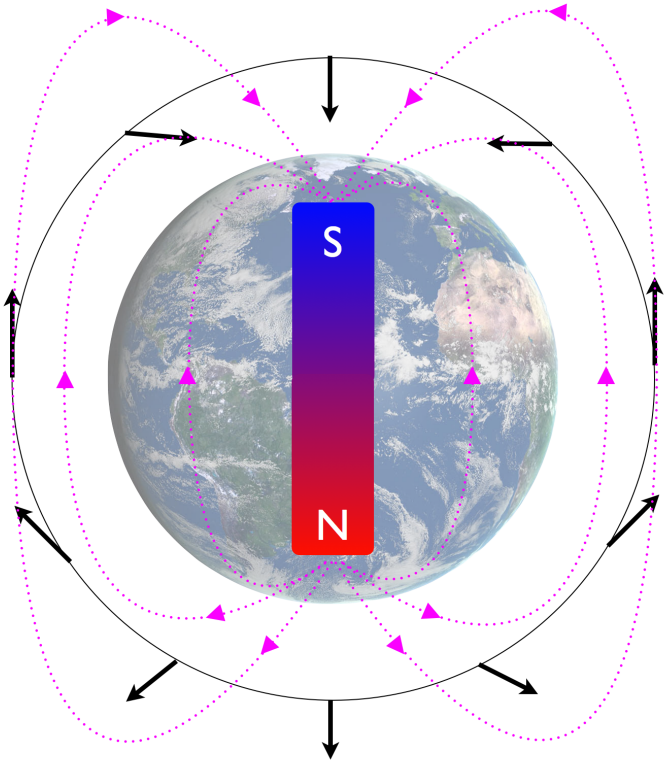

Figure 2.4: Earth as a dipole with experienced magnetic flux in orbit (black) and total magnetic flux field (pink)

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.

2.3 Rigid body kinetics

$$\boldsymbol{I}_{CG}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\boldsymbol{I}_{CG}\boldsymbol{\omega}) = \boldsymbol{\tau}_m + \boldsymbol{\tau}_g + \boldsymbol{\tau}_a \tag{2.2}$$

$\boldsymbol{\omega} = \begin{bmatrix} p & q & r \end{bmatrix}^\top$ 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 one assumes homogenous displacement of mass along the principal axises, it can be reduced to a diagonal matrix:

$$I_{CG} = \begin{bmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{xy} & I_y & -I_{yz} \\ -I_{xz} & -I_{yz} & I_z \end{bmatrix} = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \tag{2.3}$$

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:

$$\boldsymbol{I}_{CG}\dot{\boldsymbol{\omega}} + \boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{\omega} = \boldsymbol{\tau}_m + \boldsymbol{\tau}_g + \boldsymbol{\tau}_a \tag{2.4}$$

2.4 Kinematics

$$\dot{\boldsymbol{q}} = \boldsymbol{T}_q(\boldsymbol{q})\boldsymbol{\omega}_{b/n}^b \tag{2.5}$$

To represent the attitude it is wise to use a unit quaternion $\boldsymbol{q}$. 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. It also makes for a more effective computer implementation (see section (8.3)).

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

$$\boldsymbol{T}_q(\boldsymbol{q}) = \begin{bmatrix} -\epsilon_1 & -\epsilon_2 & -\epsilon_3 \\ \eta & -\epsilon_3 & \epsilon_2 \\ \epsilon_3 & \eta & -\epsilon_1 \\ -\epsilon_2 & \epsilon_1 & \eta \end{bmatrix}, \quad \boldsymbol{T}_q^\top(\boldsymbol{q})\boldsymbol{T}_q(\boldsymbol{q}) = \frac{1}{4}\boldsymbol{I}_{3x3} \tag{2.6}$$

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

2.5 Environmental torques

The satellite is exposed to several environmental torques on its path around Earth. The most dominant ones are 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 (JAXA [26]). 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 Tudor [3] the gravity vector $\boldsymbol{\tau}_g$ can be modeled as:

$$\boldsymbol{\tau}_g = 3\left(\frac{\mu}{R_c^3}\right) z_o \times \boldsymbol{I} \cdot z_o \tag{2.7}$$

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. But a thin part of the atmosphere is actually present at 600km altitude, and will cause some drag.

Rawashdeh et al. [6] shows in 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 is 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 mostly 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;

$$\boldsymbol{\tau}_a = \rho_a V_{orbit}[V_{orbit} A_{drag} c_p \hat{\boldsymbol{V}}_{orbit} - (\boldsymbol{I}_{CG} + \hat{\boldsymbol{V}}_{orbit}\boldsymbol{J})\boldsymbol{\omega}_{ob}^{b}] \qquad (2.8)$$

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 (where the aerodynamic force attacks), 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 (2.8)). It can also be seen that if the satellite spins, the torque will be higher.

Chapter 3

Requirement Specification

These are the relevant requirement specifications for the NUTS ADCS system, provided by the different groups working on the project.

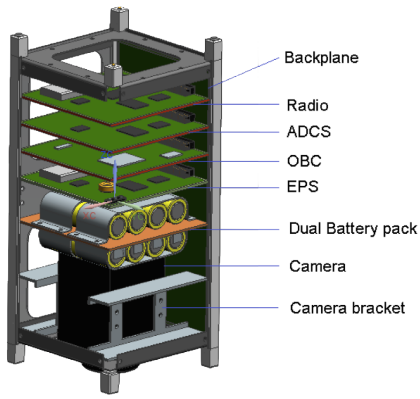

Figure 3.1: Hardware layout *Image courtesy of Christian Nomme*

3.1 Size, weight and operating temperature

Each electronic module in the NUTS will be given room for a PCB card in the dimension of $\approx 8 \times 8 \times 2$ centimeters. This volume should hold all logic, microprocessors and sensors.

During eclipses the satellite will cool down to $-17.5°$ C. In the sunny part the terminal equilibrium should be reached at $18°$ C.

Power consumption

The PCB card will be connected to a backplane that supplies power. Two regulated 3.3 V and 5.5 V sources will be distributed. The power comes from both a series of batteries and solar cells

- 8 x 3.3 V 18650 LiFePO4 batteries

  29.04 watt-hours total

- 18 solar cells of 2.3 V each with 30 % efficiency

  1.3 watt continuous peak output when oriented correctly

The ADCS system has first priority on power after launch. Then it will have second to third priority after beacon and radio systems. When transmitting, these radios consumes 0.7 watts of power. It is not yet determined exactly when and how often the radios will be powered. Neither how much power is available at all times for the ADCS system. A good measurement should however be not to exceed the solar panels charge, except when detumbling.

3.2 Propulsion actutators

The current design of the frame allows for three electromagnetic coils, one i X-, Y- and Z-axis of the body walls. Room is also left for an extra set in the opposite wall for redundancy or extra torque.

The area of each coil is approximately 10×20 centimeters in the long directions, and 10×10 in the short direction.

Power to the coils are provided from the backplane, and must be used within the power budget.

3.3 Attitude estimation

The attitude must be estimated from onboard sensors. As long as the sensors fit the size, weight and power budget they can be used. No commercial systems, except from individual components should be utilized.

The attitude estimator must provide attitude data via the backplane for the camera system. This will later be sent as metadata for further research and image processing.

3.4 Controller

The attitude controller should be able to handle the following cases:

- Detumble the satellite from initial spin after orbit launch

  This must be done within power budget and as soon as possible

  Initial spin is only known to be in "a small scale"

- Point the strap down camera system Z-axis with the centre of Earth (nadir direction)

The radio communication system is also dependent on a correct attitude. At the time of this writing the NUTS team is running simulations of the correlation between bandwidth and pointing accuracy. What *is* know, is that the radio signal are polarized, meaning the satellite can't be "upside-down" to send and receive data. This is however a weaker requirement then the camera system, found in table (3.1).

Table 3.1: Pointing accuracy ADCS system requirement

Axis	**Requirement**
X roll	$\pm 25^\circ$ degrees
Y pitch	$\pm 25^\circ$ degrees
Z yaw	None

3.5 Hardware architecture

The NUTS group is currently using the Atmel AVR® UC3-A3 as their mainframe computer system. This is a 32-bit RISC architecture microcontroller with 66MHz clock frequency; The most powerful AVR microcontroller that Atmel currently supplies. It delivers 91 DMPIS[1], compared to a mid range desktop computer that delivers over 100.000 DMIPS.

The entire ADCS system must be able to run on this microcontroller. Keeping all computer systems on one architecture makes it a lot easier to integrate the ADCS with communication protocols, power supplies, clock signals and so forth. The teams in charge of these functions can modify code and board

[1]CPU benchmark; Dhrystone Million Instructions Per Second

outlays to fit with the mainframe and backbone solution, without in depth knowledge of the ADCS.

Chapter 4

Design: Actuators

4.1 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.

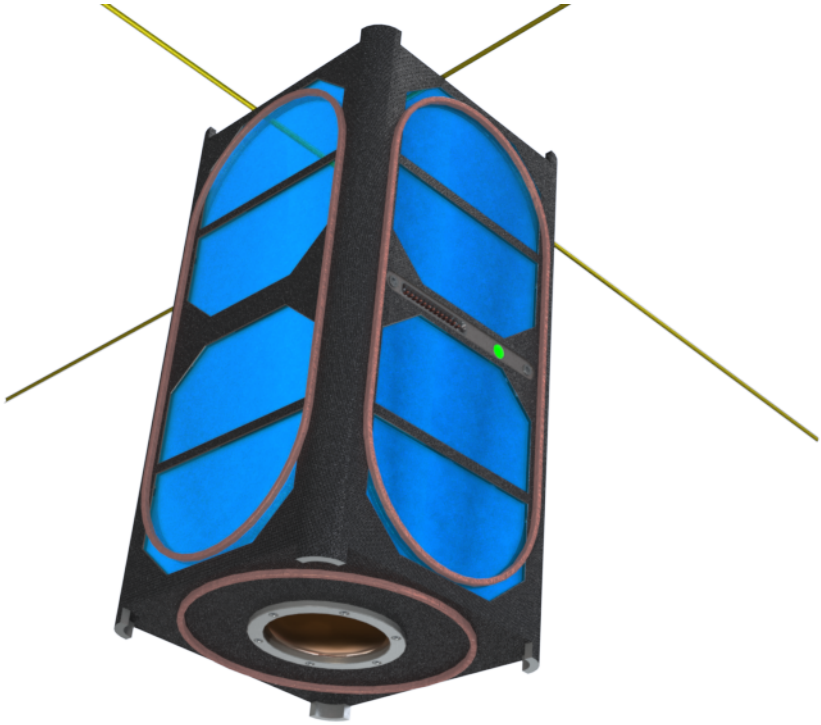

Figure 4.1: The three electromagnetic coils exposed

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 electromagnetic coils. The primary actuator for CubeSats is often magnetic coils. Also the NUTS satellite will be using magnetic coils as the main and only actuators. This is chosen for several reasons; 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 almost not enough space to fit 2.6 kg of components.)

The main drawback is that only a limited amount of torque can be produced in the Earth's magnetic field. This is the result of many factors, and a more mathematical approach can be found in section (4.2).

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 (4.1). The magnetic field generated by the thrusters will be given in tesla (T) by the formula:

$$m_b = \frac{N \cdot A}{R} \cdot V \tag{4.1}$$

- **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

4.2 Actuator dynamics

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:

$$\boldsymbol{m}_b = \begin{bmatrix} m_x \\ m_y \\ m_y \end{bmatrix} = \begin{bmatrix} \frac{N_x A_x V_x}{R_x} \\ \frac{N_y A_y V_y}{R_y} \\ \frac{N_z A_z V_z}{R_z} \end{bmatrix} \tag{4.2}$$

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

$\boldsymbol{N}$ number of turns with copper thread in the coil

$\boldsymbol{A}$ area of the coil

$\boldsymbol{V}$ voltage over the coil

$\boldsymbol{R}$ resistance in the coil

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

$$\boldsymbol{\tau}_m = \boldsymbol{m}_b \times \boldsymbol{B}_b = \boldsymbol{S}(-\boldsymbol{B}_b) \cdot \boldsymbol{m}_b = \begin{bmatrix} 0 & B_z & -B_y \\ -B_z & 0 & B_x \\ B_y & -B_x & 0 \end{bmatrix} \begin{bmatrix} \frac{N_x A_x V_x}{R_x} \\ \frac{N_y A_y V_y}{R_y} \\ \frac{N_z A_z V_z}{R_z} \end{bmatrix} \tag{4.3}$$

This relation can be observed from figure (4.2). Here is Earth's magnetic field $\boldsymbol{B}_b$. The magnetic field is set up by the actuators as $\boldsymbol{m^b}$, 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. Similar to the torque a compass needle experiences before pointing to the North Pole. The vector $\boldsymbol{\tau}_d$ will be further explained in section (4.3).

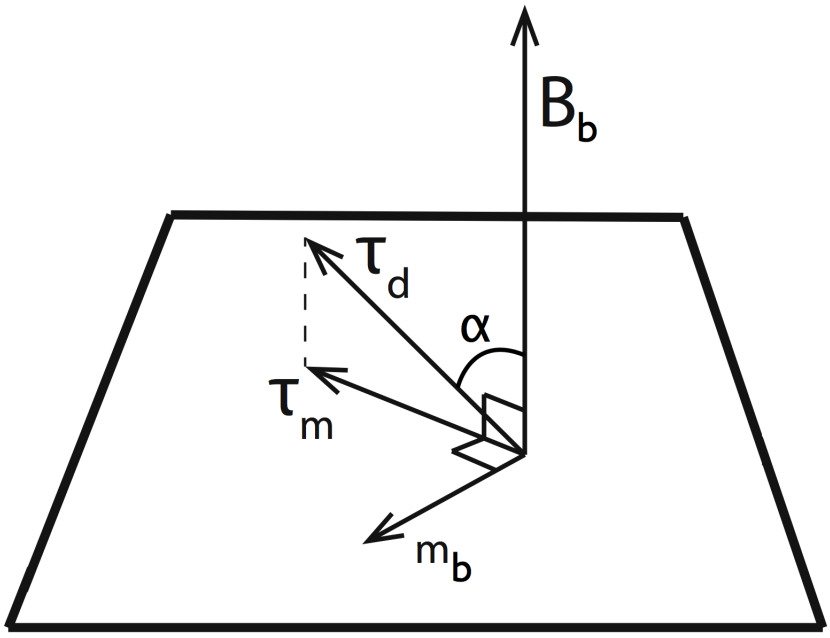

Figure 4.2: Magnetic and torque vector relations - *Image courtesy of [3]*

From figure 4.2 it can be observed that $\boldsymbol{m}_b$ can be derived as the following crossproduct:

$$\boldsymbol{m}_b = \boldsymbol{\tau}_m \times \boldsymbol{B}_b \tag{4.4}$$

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

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

Now separating the the voltage vector

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

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

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

$$\boldsymbol{V} = \boldsymbol{K}_{coil}^{-1}(\boldsymbol{\tau}_m \times \boldsymbol{B}_b) \tag{4.5}$$

This is done because the coils will need volts as input, not torque. Keep in mind that $\boldsymbol{B}_b$ 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 (4.3), an expression for $\boldsymbol{\tau}_m$ can also be found[1];

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

$$\boldsymbol{\tau_m} = \boldsymbol{S}(-\boldsymbol{B}_b) \cdot \boldsymbol{K}_{coil} \cdot \boldsymbol{V} \tag{4.6}$$

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

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

indeed be under-actuated. By comparing equation (4.3) 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 in orbit, and the satellite will eventually move out of this state.

4.3 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 arbritary torque in the BODY coordinate system. Because $\boldsymbol{\tau}_m$ is given as the cross product (4.3), $\boldsymbol{\tau}_m$ will always be in the plane perpendicular to $\boldsymbol{B}_b$. This can be seen in figure 4.2. 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*, i.e. $\boldsymbol{\tau}_m$ 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 (4.4):

$$\boldsymbol{m}_b = f(\cdot)\boldsymbol{\tau}_m \times \boldsymbol{B}_b \tag{4.7}$$

By combining this with equation (4.3), a new expression for $\boldsymbol{\tau}_m$ is formed

$$\boldsymbol{\tau}_m = f(\cdot)(\boldsymbol{\tau}_d \times \boldsymbol{B}_b) \times \boldsymbol{B}_b \tag{4.8}$$

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 $\boldsymbol{B}_b$ 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):

$$f(\cdot) = \frac{1}{|\boldsymbol{B}_b|^2} \tag{4.9}$$

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

$$\boldsymbol{m}_b = \frac{1}{|\boldsymbol{B}_b|^2}(\boldsymbol{B}_b \times \boldsymbol{\tau}_m) \tag{4.10}$$

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

$$\boldsymbol{\tau}_m = \frac{1}{|\boldsymbol{B}_b|^2}(\boldsymbol{\tau}_d \times \boldsymbol{B}_b) \times \boldsymbol{B}_b \tag{4.11}$$

4.4 Physical coil prototype

Together with the team in charge of the frame assembly, some advise was given on building a physical prototype of the NUTS, including functional actuator coils. The coils was 3D printed at the Department of Engineering Cybernetics Workshop. These should be available for full size prototype testing of the ADCS. A model can be seen in figure (4.3).

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

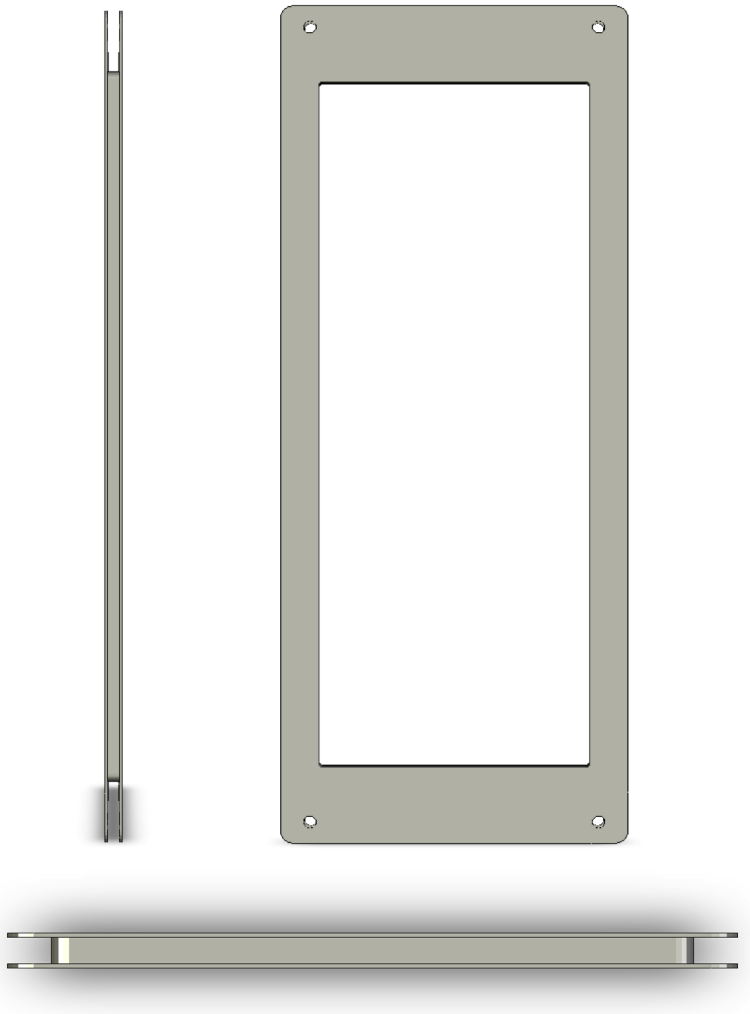

Figure 4.3: 3D model of the x- and y-axis coil before print *Image courtesy of Christian Nomme*

Chapter 5

Design: Controller

5.1 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. This measurement can be taken from a sun-sensor or a gyroscope. For this controller it is preferable to use the de-biased

gyroscope data from the attitude estimation algorithm (see chapter (6.3). The detumbling controller will then take the following form;

$$\boldsymbol{\tau}^b_{controller} = -k_d(\boldsymbol{\omega}^b_{ob} - \hat{\boldsymbol{b}}^b_{gyro}), \quad k_d > 0 \tag{5.1}$$

Setting up an expression for voltage

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

Now the scaling factor from section (4.3) is taken into account

$$\boldsymbol{V} = -\frac{k_d}{|\boldsymbol{B}^b|}\boldsymbol{K}^{-1}_{coil}\left[\boldsymbol{B}^b \times (\boldsymbol{\omega}^b_{ob} - \hat{\boldsymbol{b}}^b_{gyro})\right], \quad |\boldsymbol{V}| < 5\,\text{volts} \tag{5.2}$$

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

5.2 Detumbling fallback

Initially the detumbling controller was set to only be functional in the first orbits after launch. However the Hardware Team in the NUTS project informed that the satellite almost guaranteed would be struck by a "micro-meteorite" at any given time during the mission. These are very small particles traveling through space with extreme velocities.

If such a particle struck the satellite, it would most likely be too damaged

to continue the mission. It was however decided to still let the ADCS be prepared, since an impact would cause spin/tumbling. The detumbling controller is therefore set to trigger on angular velocity. If this reaches a a given limit, the controller will switch to detumbling mode.

5.3 Model-independent Control Law

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

$$\boldsymbol{\tau} = k_p \boldsymbol{\epsilon} - k_d \tilde{\boldsymbol{\omega}} \tag{5.3}$$

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.

$\boldsymbol{\tau}$ 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;

$$\boldsymbol{\tau}^b_{controller} = -k_p \boldsymbol{\epsilon} - k_d(\boldsymbol{\omega}^b_{ob} - \hat{\boldsymbol{b}}^b_{gyro}), \quad k_d, k_p > 0 \tag{5.4}$$

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

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

$$\boldsymbol{V} = -\frac{\boldsymbol{K}_{coil}^{-1}}{|\boldsymbol{B}^b|}\left[-k_p(\boldsymbol{B}^b \times \boldsymbol{\epsilon}) - k_d\left(\boldsymbol{B}^b \times (\boldsymbol{\omega}_{ob}^b - \hat{\boldsymbol{b}}_{gyro}^b)\right)\right], \quad |\boldsymbol{V}| < 5\,\text{volts} \tag{5.5}$$

This is the primary controller that will maintain the attitude of the NUTS. The idea is to switch this controller on when the angular velocity is relatively low.

5.4 Stability analysis of model independent PD-controller

Lyapunov theory can be used to both to test stability of a system, and to arrive at a stabilizing control law. Both Soglo [10], Tudor [3] and Wen [7] uses this to different extents. A brief summary is given here;

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

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 section (5.3)[1], only without the gyroscope bias estimate. Soglo concludes that with the given controller and gain requirements, the controller is uniformly asymptotically stable.

5.5 Passive controller and magnetic disturbance

A passive controller can 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 the magnetic field will be changing constantly as a result of the polar

[1]With the NUTS specification the k-gain should be 8×10^{-9} or higher.

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 itself. If this turns out to be above the saturation limit of the controller, one can install a constant magnet in the frame, compensating in the opposite direction:

$$\boldsymbol{\tau}_{magnet} = \boldsymbol{m}_{magnet} \times \boldsymbol{B}_b = -(\boldsymbol{m}_{NUTS} \times \boldsymbol{B}_b) = -\boldsymbol{\tau}_{NUTS} \tag{5.6}$$

$$\implies \boldsymbol{\tau}_{magnet} + \boldsymbol{\tau}_{NUTS} = 0 \tag{5.7}$$

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.

Bråthen [32] demonstrates that it is possible to shut down the magnetic coils when measuring the magnetic field, to avoid disturbance on the magnetometer. The system will act as a latch between measuring and actuating. Grip et al.[33] shows it is possible to use a fourth-order notch filter to remove the magnetic disturbance from high a intensity light source. These approaches are out of the scope of this thesis, and it is assumed the magnetic coils and magnetometer are independent.

Chapter 6

Design: Attitude Estimation

6.1 Sensors for attitude estimation

A good sun-sensor, magnetometer and gyroscope should be enough for attitude estimation. The final hardware assembly is however yet to be determined by the NUTS project. The following options are considered:

- **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**

  Using a magnetometer requires good knowledge on Earth's mag-

netic field (like an onboard IGRF look-up-table (LUT)). Testing of the radio-communication system and other onboard electronics, is required in order to estimate biases and noise affecting this instrument. It is however a vital component for the controller, so it will be included in the final assembly.

- **Orbit propagator**

  The system will be dependent on knowing where in orbit the satellite is. This is mainly to be able to calculate the magnetic field from the look-up-table. An "orbit propagator" might be realized with an internal system clock, that can be verified, updated and calibrated from ground. Such a clock will be part of the final backbone system available to all the internal electronic components.

- **Commercial Sun sensor**

  A sun sensor can 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 Master Thesis based on such 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).

Considering the above facts (and discussions with the NUTS group), it is most likely that the NUTS will carry a gyroscope, magnetometer and sun sensor (based on solar panel measurements). Also an internal clock for the orbit propagator. Further development and simulations are based on this choice.

6.2 Study on attitude estimation algorithms

There are mainly two paths to follow when doing attitude estimation; Statistical and dynamical.

A statistical tool is based on solving an equation on the form $y = f(x) = 0$. A dynamical tool uses a differential equation, typical $\dot{\hat{x}} = f(\cdot)$. This is more useful since this equation will have memory, and have filter-like properties.

The following sections describes some different algorithms to choose from.

Statistical algorithms

- Wahba's problem
- TRIAD
- QUEST
- EQUEST

TRIAD

The TRIAD algorithm was originally proposed by Harold Black in 1964 [19].

$$[\boldsymbol{a}^n \vdots \boldsymbol{m}^n \vdots (\boldsymbol{a}^n \times \boldsymbol{m}^n)] = \boldsymbol{R}_b^n [\boldsymbol{a}^b \vdots \boldsymbol{m}^b \vdots (\boldsymbol{a}^b \times \boldsymbol{m}^b)] \tag{6.1}$$

This algorithm was used for several years in satellite attitude estimation, but was later surpassed by QUEST and the Kalman filter.

Wahba's problem

The statistician Grace Wahba proposed in 1965 [17] a cost function that combines vector measurement from different coordinate systems;

$$J(\boldsymbol{R}) = \frac{1}{2} \sum_{i=1}^{n} k_i \parallel \boldsymbol{a}_i^n - \boldsymbol{R}_b^n \boldsymbol{a}_i^b \parallel^2 \tag{6.2}$$

By minimizing this problem, one is able to find the rotation matrix $\boldsymbol{R}$ between the $\boldsymbol{a}_i^n$ vector in the reference frame, and $\boldsymbol{a}_i^b$ vector in the body frame. k_i is optional gain for each observation.

QUEST

$$J(\boldsymbol{q}) = \sum_{j=1}^{n} \frac{1}{\sigma_j^2} (\boldsymbol{I} - \boldsymbol{b}_j^\top \boldsymbol{R}_b^i(\boldsymbol{q}) \boldsymbol{r}_j) \tag{6.3}$$

Originally proposed by Shuster and Oh [18] in 1981. The QUEST algorithm resembles the optimization problem of Wahba and Paul Davenport's

q-Method [20]; It minimizes an equation, trying to find the rotation between a set of reference vectors in different coordinate systems.

The algorithm is widely used, maybe because the concept is relatively easy to grasp. Multiple versions of this algorithm also focuses on different ways to solve the minimization problem itself, not necessarily improvements of the attitude estimate.

Like Whaba's problem and TRIAD, the QUEST algorithm can only utilize vectorial data. The correct attitude (given correct vectorial measurements) are found in one iteration, making it incredibly fast. Even by increasing the number of reference vectors, Markley and Mortari [20] points out that QUEST uses very few floating point operations to calculate the attitude. Still, it has no memory of previous attitudes and this won't filter out noise from the measurements. This is clearly demonstrated by Grip et. al. [22] with a direct comparison between QUEST, EKF and a Nonlinear Observer.

EQUEST

In 2011, Jenssen and Yabar [4] (working on their master thesis for the NUTS satellite) used the results from Psiaki's Extended Quaternion Estimation algorithm [21] to utilize gyroscope measurements.

$$J(\boldsymbol{q}) = \frac{1}{2}\sum_{j=1}^{n}\left\{\frac{1}{\sigma_j^2}(\boldsymbol{b}_j - \boldsymbol{R}_b^i(\boldsymbol{q})\boldsymbol{r}_j)^\top(\boldsymbol{b}_j - \boldsymbol{R}_b^i(\boldsymbol{q})\boldsymbol{r}_j)\right\}... \\ ... + \frac{1}{2}(\boldsymbol{q} - \hat{\boldsymbol{q}}_{gyro})\boldsymbol{D}(\boldsymbol{q} - \hat{\boldsymbol{q}}_{gyro})) \tag{6.4}$$

The EQUEST algorithm is an extension to QUEST that incorporates measurements from a gyroscope. Jenssen and Yabar also adds a linear prediction term, based on the slowly varying attitude of the satellite. This will even-

tually reduce the effect of noisy measurements;

$$+\frac{1}{2}(\boldsymbol{q}-\hat{\boldsymbol{q}}_{pre})^{\top}\boldsymbol{S}(\boldsymbol{q}-\hat{\boldsymbol{q}}_{pre}) \tag{6.5}$$

By adding this prediction term, Jenssen and Yabar was able to demonstrate a smoothening effect on the attitude estimate. Rinnan [14] also shows how the EQUEST algorithm can be used to initialize a nonlinear observer, since it converges so fast.

Dynamical algorithms

- Kalman Filter family
- Salcudean Observer
- Mahony Observer
- Grip Observer

Kalman Filter

Tha Kalman Filter was first cited by Rudolph E. Kalman in 1960 [25]. Naturally, it was first given in its discrete form, and later developed into a continuous form. The advantages of the Kalman filter was many, and it helped NASA a great deal on for example the lunar landing with Apollo 11. Because the Kalman filter is optimal with respect to minimum variance, it incorporates all measurements that can be provided. Even data with bad precision.

The Kalman filter is (only) asymptotically stable, it will work as the optimal state estimator (linear and nonlinear) and the estimate is unbiased

and minimum variance. However some assumptions about the process must made; Noise and measurements from the process must be white and Gaussian, initial state must be Gaussian and system must be observable and linear. Our satellite is not linear, and therefore we cannot utilize a Kalman filter in its original form. That is also why the Extended Kalman Filter was developed.

Extended Kalman filter

With the Extended Kalman Filter (EKF) it is also possible to handle nonlinear systems, at a certain cost that is. For each iteration, the equations is linearized. Now the Kalman Filter might no longer be optimal, and if the linearization point gets too far away from the true state, it might diverge. This means stability properties cannot be guaranteed globally.

It's important to mention that in order to utilize quaternions as attitude representation, one has to modify the EKF due to the extra degree of freedom. Otherwise one will eventually experience singularities/gimbal lock. This is known as a Quaternion Kalman Filter [23].

Salcudean Observer

The Salcudean Observer was originally proposed by S. Salcudean in 1991 [31]. This is a totally different approach to attitude estimation compared to the Kalman Filter. The Salcudean Observer is model dependent, in that it utilizes the moment of inertia matrix compared with the torque applied to

the body;

$$\dot{\boldsymbol{\tau}} = \boldsymbol{\tau}\,\frac{1}{2}\,k_p\,\boldsymbol{I}_{xyz}^{-1}\,\boldsymbol{e}\,sgn(e_0) \tag{6.6}$$

$$\boldsymbol{R} = \left[\hat{\boldsymbol{R}}\boldsymbol{R}^\top(\hat{\boldsymbol{\omega}} + k_v\,\boldsymbol{I}_{xyz}^{-1}\,\boldsymbol{e}\,sgn(e_0)\right] \times \hat{\boldsymbol{R}} \tag{6.7}$$

$$e_0 = \beta_0\hat{\beta}_0 + \beta^\top\hat{\beta} \tag{6.8}$$

$$\boldsymbol{e} = \hat{\beta}_0\beta - \beta_0\hat{\beta} - \beta \times \hat{\beta} \tag{6.9}$$

As shown, the Salcudean Observer is not only model dependent, but also requires a direct quaternion mapping from the IMU. It is derived by exploiting the dynamics of rigid body motion and Euler quaternion representation of rotation. Salcudean did not present any stability or performance results.

6.3 Nonlinear Mahony Observer

In 2008 Mahony, Hamel and Pflimlin [16] presented three *Nonlinear Complementary Filters on the Special Orthogonal Group*. The idea was to provide good attitude estimates from low cost IMU's. Typically used in Micro Aerial Vehicles (MAV's). This is also the use case Mahony et al. presents, and where one usually finds this algorithm today.

$$\boldsymbol{\omega}_{mes}^b = -vex\left(\sum_{i=1}^{n}\frac{k_i}{2}(\boldsymbol{v}_i^b(\hat{\boldsymbol{v}}_i^b)^\top - \hat{\boldsymbol{v}}_i^b(\boldsymbol{v}_i^b)^\top)\right) \tag{6.10}$$

$$\dot{\hat{\boldsymbol{b}}}_{gyro}^b = -\frac{1}{2}\boldsymbol{K}_i\boldsymbol{\omega}_{mes}^b \tag{6.11}$$

$$\dot{\hat{\boldsymbol{q}}} = \boldsymbol{T}_q(\hat{\boldsymbol{q}})\left[\boldsymbol{\omega}_{imu}^b - \hat{\boldsymbol{b}}_{gyro}^b + \boldsymbol{K}_p\boldsymbol{\omega}_{mes}^b\right] \tag{6.12}$$

- The $vex()$ function is the inverse cross product operator

  $$vex(S(\boldsymbol{a})) = \boldsymbol{a}$$

- $\boldsymbol{v}_i^b$ is the sensor data from for example;

  $\boldsymbol{v}_1^b$ accelerometer

  $\boldsymbol{v}_2^b$ magnetometer

- $\hat{\boldsymbol{v}}_1^b$ is the estimate corresponding to the accelerometer[1];

  $\hat{\boldsymbol{v}}_1^b = \boldsymbol{R}_n^b(\hat{\boldsymbol{q}})\boldsymbol{v}_{01}^n$

  $\boldsymbol{v}_{01}^n = [0\,0\,1]^\top$

- $\hat{\boldsymbol{v}}_2^b$ is the estimate corresponding to the magnetometer;

  $\hat{\boldsymbol{v}}_2^b = \boldsymbol{R}_n^b(\hat{\boldsymbol{q}})\boldsymbol{v}_{02}^n$

  $\boldsymbol{v}_{02}^n = \frac{1}{\sqrt{m_x^2+m_y^2+m_z^2}}[m_x\, m_y\, m_z]^\top$

- k_1, k_2, K_p and K_i are observer gains

  k_1 and k_2 validity of the accelerometer and magnetometer

  K_p deviation from reference frame

  K_i gyro bias correction

[1]The gravity reference is only present in the positive z-direction in NED

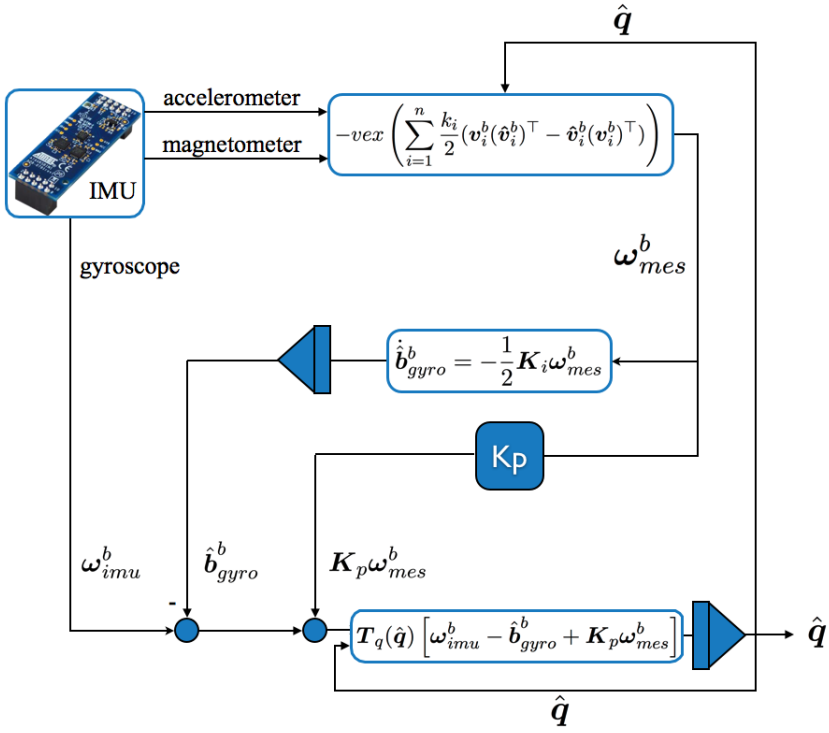

Figure 6.1: Nonlinear Observer state diagram

To explain the functionality of this observer, it might be useful to start with $\hat{\boldsymbol{q}}$ at the right hand side of figure (6.1). This attitude estimate is fed back to the $vex()$ algorithm for a comparison with the accelerometer and magnetometer. $\hat{\boldsymbol{q}}$ is used to create a rotation matrix from NED to BODY, which in turn makes it possible to compare the reference vectors with data from the sensors.

$\boldsymbol{\omega}^b_{mes}$ will be generated as a result and can be interpreted as the estimate error. This signal is then split into two threads; One is gained with $\boldsymbol{K}_p$ and directly subtracted from the $\boldsymbol{\omega}^b_{imu}$ measurement. The other is gained with $-\boldsymbol{K}_i$, integrated and then subtracted. Combined it can be viewed as a PI-controller for the estimate error.

$\dot{\hat{\boldsymbol{b}}}$ is the estimate of the gyroscope *bias*. By subtracting this term, one will make sure the output from the gyroscope can be numerically integrated without causing drift. The second term $\boldsymbol{K}_p\boldsymbol{\omega}^b_{mes}$ (sometimes known as $\boldsymbol{\sigma}$) is the *injection*. This is the part that moves the attitude estimate so it follows the body frame.

The final step is to numerically solve the attitude differential equation (6.12) (while the gyroscope bias is converging) with feedback from $\hat{\boldsymbol{q}}$.

6.4 Choice of attitude estimation algorithm

The different algorithms all have their strengths and weaknesses. QUEST is fast and is easy to use, but limited to vectorial data and overly affected by noise. EQUEST handles both of these issues, but this also makes the algorithm more complicated, which again makes it more computationally heavy for the NUTS onboard computers.

Kalman filters are in general well known and trusted algorithms. Synthetic testing also demonstrates outstanding performance [22]. An implementation still requires a deeper mathematical understanding than most statistical tools. Both EQUEST and EKF has previously been proposed as solutions for the NUTS satellite. But the Mahony Observer should be less computational heavy then EKF and have better noise capabilities then EQUEST. It is also a less complex algorithm to work with then both of them. At the time of this writing no cubesat ADCS system based on a Mahony Observer has been

launched.

The Mahony Observer (with the bias projection algorithm from Grip et al. [27]) combines many advantages into an exclusive package that separates it from the other estimation algorithms;

1. Guaranteed semi-global exponential stability and convergence
2. Low complexity
3. Small source code footprint
4. Developed with low cost MEMS sensors in mind
5. Filter liker properties
6. Model independent

This makes the Mahony observer the attitude estimation algorithm of choice.

6.5 Using the Mahony observer with a sun sensor

As previously mentioned the satellite won't use an accelerometer in the final assembly. Simply because there is no usable gravity to measure in orbit. However, in order for the the Mahony Observer to function properly, it needs a measurement with similar properties to an accelerometer, and this is exactly where the sun sensor will come in. The orbit coordinate system is well defined. The x-axis is positively defined in the surge direction. The z-axis is positively defined towards origo of ECI/ECEF. Since the NUTS follows a sun synchronous orbit, the y-axis will always point towards the sun. But because the satellite is not located at the centre of Earth, the y-axis will miss the centre of the Sun with the radius of Earth + the height of the orbit.

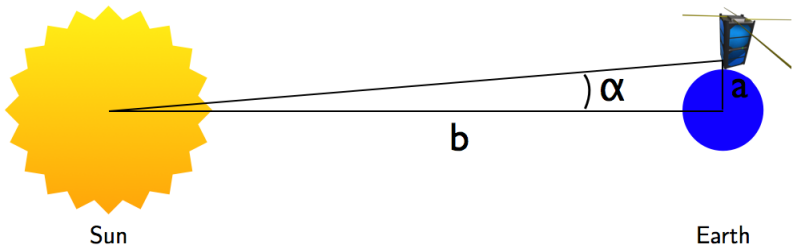

Figure 6.2: Triangle formed by the satellites orbit, Earth and Sun

Because of the enormous distance from Earth to the Sun, the following calculation describes the approximation of $\alpha \approx 0$ as shown in figure (6.2).

$$
\begin{aligned}
a &= r_{Earth} + r_{orbit} \\
a &= 6\,356\,800\,m + 600\,000\,m = 6\,956\,800\,m \\
b &= r_{Earth's\,orbit} \\
b &= 149\,597\,870\,000\,m \\
\alpha &= tan^{-1}\left(\frac{a}{b}\right) = tan^{-1}\left(\frac{6\,956\,800\,m}{149\,597\,870\,000\,m}\right) \\
\alpha &= 0.002664^{\circ} \approx 0
\end{aligned}
\tag{6.13}
$$

Neither an accelerometer nor a sun sensor would even be able to approach this accuracy. Again, $\alpha \triangleq 0$ is assumed for the rest of the calculations.

The offset given by α doesn't tell the whole story. The ORBIT coordinate system is not normal to the Sun if there excises an *inclination angle* (see chapter (1.2) and figure (1.2)).

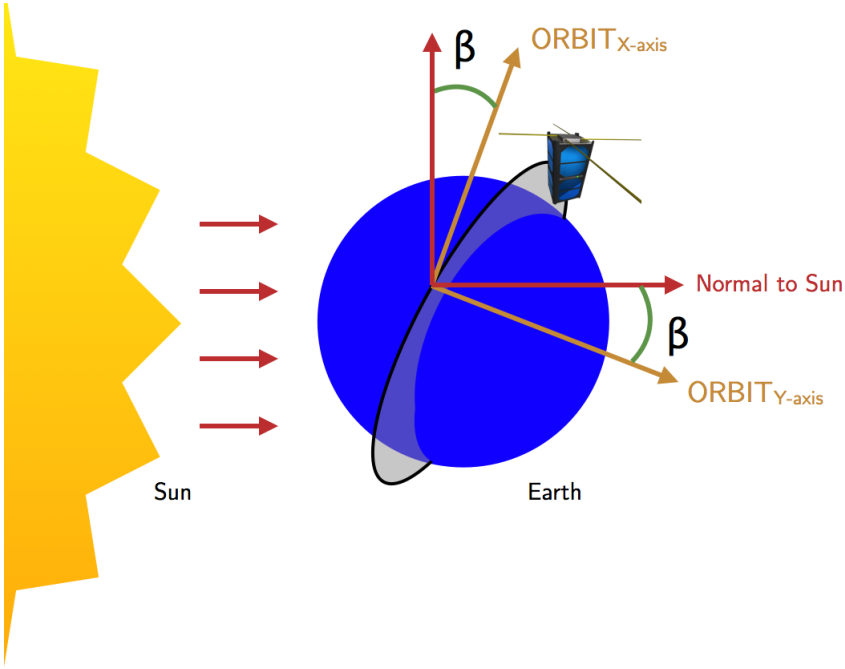

Figure 6.3: Inclination of ORBIT relative to the Sun

As seen by figure (6.3), the orbit coordinate system will be rotated around the z-axis (with an angle β) relative to the vector from the Sun. This angle is not yet defined for NUTS (awaiting launch/orbital parameters). To compensate, the sun vector in BODY will here be defined by including a rotation matrix. When the final inclination angle is known, it must be applied to the equations and pre-calculated before launch.

Looking at the equation for the Mahony Observer from section (6.3), the expression for accelerometer estimate must be changed to fit the Sun sensor

read-out[2];

$$\begin{aligned}\boldsymbol{s}^s &= [s_x\ s_y\ s_z]^\top \\ \boldsymbol{s}^o &= \boldsymbol{R}_s^o \boldsymbol{s}^s \\ \hat{\boldsymbol{v}}_1^b &= \boldsymbol{R}_o^b(\hat{\boldsymbol{q}})\boldsymbol{v}_{01}^o \\ &= \boldsymbol{R}_o^b(\hat{\boldsymbol{q}})\frac{\boldsymbol{s}^o}{\|\boldsymbol{s}^o\|}\end{aligned}$$

With the final rotation in place, the full expression now yields

$$\hat{\boldsymbol{v}}_1^b = \boldsymbol{R}_o^b(\hat{\boldsymbol{q}})\frac{\boldsymbol{R}_s^o \boldsymbol{s}^s}{\|\boldsymbol{R}_s^o \boldsymbol{s}^s\|} \tag{6.14}$$

(where $\boldsymbol{R}_s^o$ is constant and $\boldsymbol{s}^s$are predetermined)

Using this theory, section (6.3) and [1], the Nonlinear observer can now be written;

$$\boldsymbol{\omega}_{mes}^b = -vex\left(\sum_{i=1}^{2}\frac{k_i}{2}(\boldsymbol{v}_i^b(\hat{\boldsymbol{v}}_i^b)^\top - \hat{\boldsymbol{v}}_i^b(\boldsymbol{v}_i^b)^\top)\right) \tag{6.15}$$

$$\boldsymbol{\omega}_{mes}^b = -vex\left(\frac{k_1}{2}(\boldsymbol{v}_1^b(\hat{\boldsymbol{v}}_1^b)^\top - \hat{\boldsymbol{v}}_1^b(\boldsymbol{v}_1^b)^\top) + \frac{k_2}{2}(\boldsymbol{v}_2^b(\hat{\boldsymbol{v}}_2^b)^\top - \hat{\boldsymbol{v}}_2^b(\boldsymbol{v}_2^b)^\top)\right) \tag{6.16}$$

Collapsing the equation with the cross product operator

$$\boldsymbol{\omega}_{mes}^b = \left(\frac{k_1}{2}(\boldsymbol{v}_1^b \times \hat{\boldsymbol{v}}_1^b) + \frac{k_2}{2}(\boldsymbol{v}_2^b \times \hat{\boldsymbol{v}}_2^b)\right) \tag{6.17}$$

[2]Three coordinate systems are used; Normal to the Sun s, Orbit o and BODY b

Inserting the estimates and normalized measurements to gather the complete expression for $\boldsymbol{\omega}^b_{mes}$

$$\boldsymbol{\omega}^b_{mes} = \Bigg[\frac{k_1}{2} \left(\frac{\boldsymbol{s}^b_{sensor}}{\|\boldsymbol{s}^b_{sensor}\|} \right) \times \left(\boldsymbol{R}^b_o(\hat{\boldsymbol{q}}) \frac{\boldsymbol{R}^o_s \boldsymbol{s}^s}{\|\boldsymbol{R}^o_s \boldsymbol{s}^s\|} \right) + \cdots$$
$$\cdots \frac{k_2}{2} \left(\frac{\boldsymbol{m}^b_{mag}}{\|\boldsymbol{m}^b_{mag}\|} \right) \times \left(\boldsymbol{R}^b_o(\hat{\boldsymbol{q}}) \frac{1}{\sqrt{m_x^2 + m_y^2 + m_z^2}} \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} \right) \Bigg] \tag{6.18}$$

The reader may note that unless the orbit has no inclination angle relative to the Sun, the satellite will at $\approx$ 40-50 % of one orbit period don't receive valid data from the sun-sensor. This challenge is does not restrict the use of the Mahony Observer. Simulations of this phenomenon can be found in chapter (9).

6.6 Mathematical stability of Mahony Observer

Mahony states [16] that the filter is *...ensuring almost global stability of the observer error* and *...the filter remains well conditioned in the case where only a single vector direction is measured.* The loss of a direction vector might be the case for the NUTS, if the sun-sensor is covered by shadow from Earth.

However the proof of stability for the Mahony observer requires that the reference vectors $\boldsymbol{v}^b_i$ are constant [16]. Normally this holds for non-accelerated vehicles (on the Earth's surface), since neither the magnetic field nor the gravity changes. For the NUTS satellite, only the sun-vector is known to be constant, as the magnetic field changes throughout the orbit.

Grip, Fossen, Johansen and Saberi [28] suggests an extension to the Ma-

hony Observer, by adding a projection algorithm to the gyroscope bias estimate;

$$\dot{\hat{\boldsymbol{b}}}^b_{gyro} = Proj(\hat{\boldsymbol{b}}_{gyro}, -\boldsymbol{K}_i \boldsymbol{\omega}^b_{mes}) \tag{6.19}$$

Mathematically this method ensures that *...the equilibrium point of the quaternion error dynamics is semi-global exponential stable for* $n \geq 2$ *independent inertial directions* v^n_{0i} [29].

Practically this addition ensures that the gyroscope bias estimate don't leave a pre defined region. It can be considered as "wild point" signal processing.

Chapter 7

Design: Simulator

In order to test the controllers and state estimation algorithm, 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 result in hefty debugging.

Another approach is to test the controller- and state estimation 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.

Previous 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 was also developed by Gaute Bråthen [32]. During Bråthen's development, the results where

discussed, and verified with the simulator given here.

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.

7.1 Nonlinear Simulink Simulator

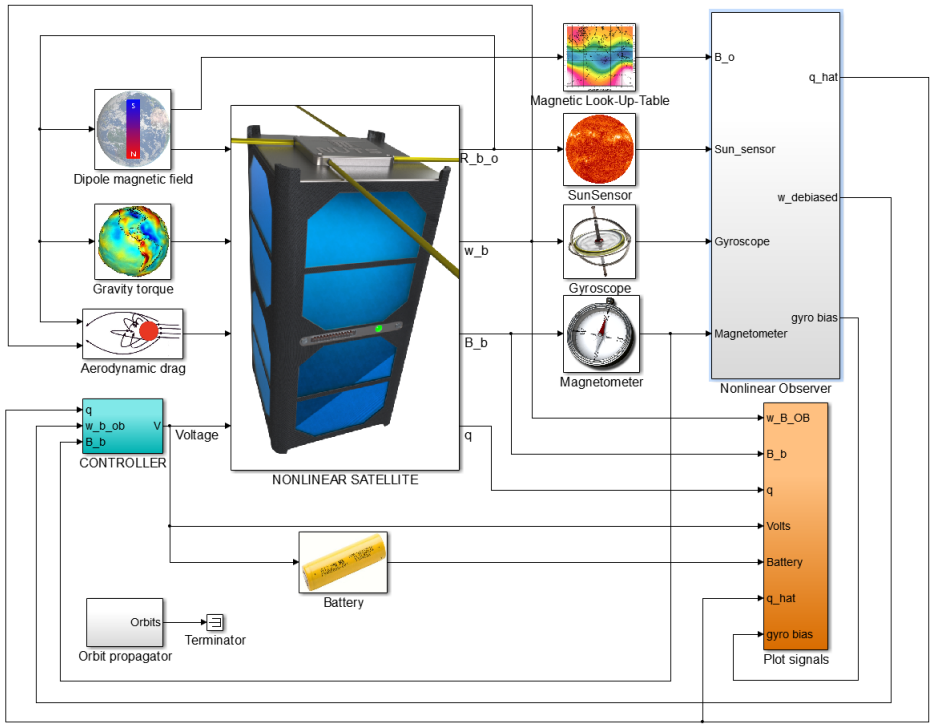

Figure 7.1: Nonlinear NUTS Simulink model with controller

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. Simulink has the ability to generate C-code with the Simulink Coder environment. For this reason the development of a Simulink simulator was begun. This simulator was made from ground up, with inspiration from [9]. However, it was later decided to write C-code directly to the embedded platform, as the code Simulink provided was challenging to carry on.

The simulator includes a nonlinear satellite model exactly like the one developed in section (2.3) and (2.4). 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 in 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 fed back as well.

The output from the satellite block is used to make "sensors". Details are described in the next section. Sensor data is received by the *Nonlinear Observer*. This includes a sun sensor, gyroscope, magnetometer and magnetic reference in orbit. The nonlinear observer is assembled with Simulink blocks very similar to how it is described in [29]. See figure (7.2). Output from the nonlinear observer is $\hat{\boldsymbol{q}}$ and $\boldsymbol{\omega}_{debiased} = \boldsymbol{\omega}_{gyroscope} - \hat{\boldsymbol{b}}_{bias}$.

The attitude estimate, along with the de-biased gyroscope, 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 a *detumbling trigger* that switches between the reference controller and detumbling controller

based on angular velocity.

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 interesting to give this as a function of the satellites battery as well.

7.2 Generating sensor data

The simulator have three sensors mainly modeled after the sensors used in the hardware platform (see section (8.2))

- Sun sensor

  Modelled after the Solar Mems Technology SSOC-A60

  $$\begin{bmatrix} 0.4 \\ -0.3 \\ 0.5 \end{bmatrix} \text{ offset}$$

  6.32510^{-3} RMS noise

The Sun sensor uses the $\boldsymbol{R}_o^b$ rotation matrix from the mathematical model. This is then multiplied with a vector given in the orbit coordinate system; $\boldsymbol{s}^o = [0\,1\,0]^\top$. The supplier does not provide noise or bias characteristics. Noise similar to the Bosch BMA-150 accelerometer was instead added. Also

a small bias offset (physical mounting deviation). The output will be;

$$\boldsymbol{s}_{sun}^{b} = \boldsymbol{R}_{o}^{b} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + \boldsymbol{\delta}_{offset} + \gamma_{noise}$$

Much like an accelerometer, but with the "gravity vector" in the y-axis.

- Gyroscope

  Modelled after the InvenSense ITG-3200

  $\begin{bmatrix} -30 \\ 40 \\ 25 \end{bmatrix}$ $^{\circ}/_{s}$ bias

  0.38 $^{\circ}/_{s}$ RMS noise

The Gyroscope uses the angular velocity vector $\boldsymbol{\omega}^{b}$ from the mathematical model. Bias is added as a worst case, based on data from the manufacturer. This is then added noise The final output is;

$$\boldsymbol{\omega}_{gyr}^{b} = \boldsymbol{\omega}^{b} + \boldsymbol{b}_{bias} + \gamma_{noise}$$

- Magnetometer

  Modelled after the AKM AK8975

  10^{-8} Tesla RMS noise

The magnetometer uses uses the magnetic field vector in the body coordinate system from the mathematical model $\boldsymbol{B}^b$. The supplier of the magnetometer does not provide data on noise. MEMS sensors of this kind was found to have noise characteristics in the nanoTesla range. Ten times this "noise roof" was then added. Final expression;

$$\boldsymbol{m}^b_{mag} = \boldsymbol{B}^b + \gamma_{noise}$$

The Nonlinear Observer needs a reference to the correct magnetic field in orbit. This has to be stored as a look-up-table on the on-board computer. To mimic the output of such a look-up-table, a *zero-order hold* block was used. This can be set to sample and hold the magnetic orbital values for any given number of seconds. Technically this just limits the resolution of the signal.

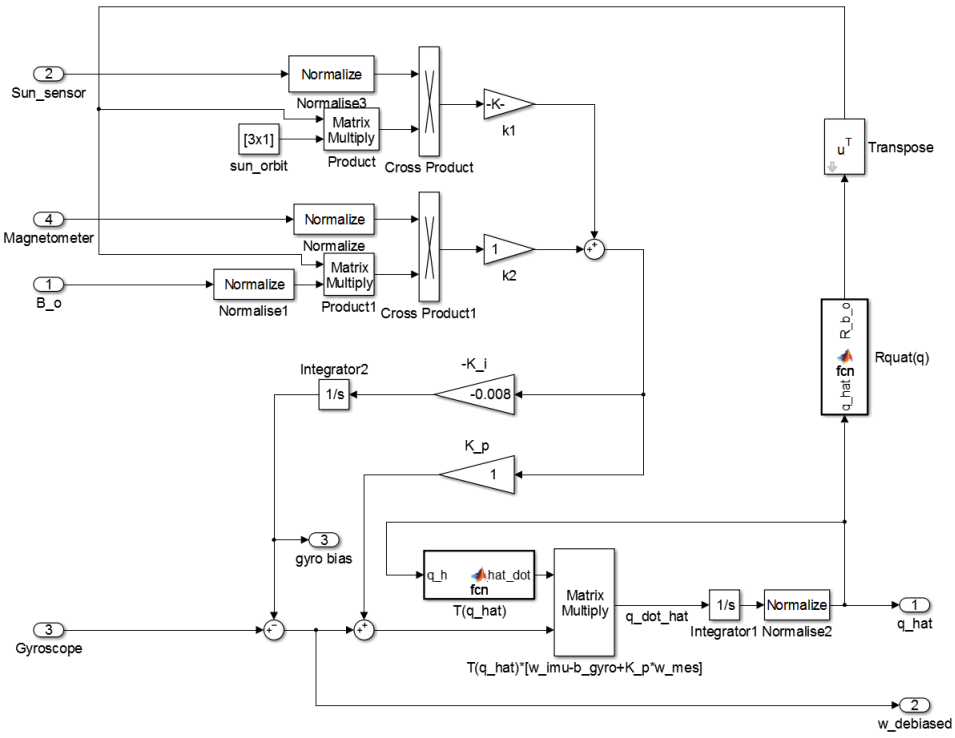

Figure 7.2: Nonlinear Observer realized in Simulink

Chapter 8

Design: Hardware

8.1 Micro controller unit

Atmel is a marked leader in both microcontroller units (MCU) and integrated development platforms (IDE). They supply 8-bit AVR and 32-bit ARM microcontrollers. Atmel also provides their own IDE (Atmel Studio) which is available for free to anyone.

After a discussion with Atmel's Applications Engineer team in Trondheim, a set of hardware to begin the development was chosen;

- XMEGA-A3BU Xplained development kit
  Evaluation platform kit with USB, screen and buttons
  Covers the exact same area as a credit card
  Unit price of $29 USD
- AVR® ATxmega256A3BU
  High performance low power 8-bit MCU
  16KB of RAM and 256KB programmable storage

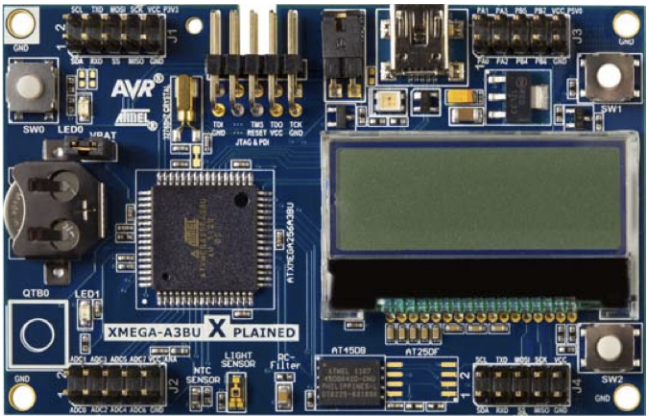

Figure 8.1: XMEGA-A3BU Xplained 1:1 scale

- 32MHz clock frequency
- Power consumtion $<$ 1 Watt for the entire kit
- 128×32 pixels LCD display
- External AVR JTAGICE3 programmer/debugger

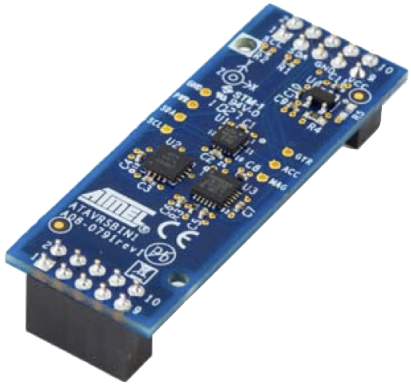

Figure 8.2: Inertial One (ATAVRSBIN1) 9-DOF sensor board

8.2 Sensor unit

The Atmel Xplained kits can be extended with several sensor boards. A so called "9-DOF" unit (3 sensors with 3 DOF each), the AVR Inertial One was chosen for this project. This board directly connects on top of the A3BU using two (out of four) GPIO connection headers.

- Inertial One (ATAVRSBIN1)
  Unit price of $24 USD
- Bosch Sensortec Three-Axis Accelerometer (BMA-150)

  4 mg resolution (±2 g range)

  ±60 mg zero-g offset (bias, can be pre-calibrated)

  6.3 mg RMS noise at 100Hz

  −40°C ... +85°C temperature range
- InvenSense Three-Axis Gyroscope (ITG-3200)

  ±2000 $^\circ/_s$ full scale range

  ±40 $^\circ/_s$ bias

  0.38 $^\circ/_s$ total RMS noise at 100Hz

  −40°C ... +85°C temperature range
- AKM Three-Axis Electronic Compass (AK8975)

  ±1229 μT range

  0.3 μT/LSB sensitivity

  −30°C ... +85°C temperature range

The reader might note that the satellite will not use an accelerometer in the final design. Instead the NUTS will rely on a sun sensor (section (6.5)). The accelerometer is part of the hardware setup in order to provide a fully function system on the Earth's surface (where the development takes place). This will make it possible to benchmark and test the stability of the system.

8.3 Development

The code is written in the C programming language. Atmel Studio provides a Software Framework (ASF) that supplies the programmer with routines such as timers, interrupts and drivers for the LCD and sensors. An example reading accelerometer data:

```
1 sensor_read(&accelerometer, SENSOR_READ_ACCELERATION, ...
      &accel_data);
2             int32_t x_acc = accel_data.axis.x;
3             int32_t y_acc = accel_data.axis.y;
4             int32_t z_acc = accel_data.axis.z;
```

The accelerometer data is now stored as separate integers in milli-g. A similar function call can be done for both the magnetometer and gyroscope.

In order to realize the the attitude observer from section (6.3) there is also a need for an efficient integration algorithm. The well know Forward Euler algorithm was implemented (Fossen [1] p. 545);

```
1 struct vector Forward_euler_vector_integration(struct ...
      vector *b_dot, float stepsize) {
2
```

```
3       float h = stepsize;
4
5       b_hat_stored_euler.x = b_hat_stored_euler.x + h*b_dot->x;
6       b_hat_stored_euler.y = b_hat_stored_euler.y + h*b_dot->y;
7       b_hat_stored_euler.z = b_hat_stored_euler.z + h*b_dot->z;
8
9       return b_hat_stored_euler;
10  }
```

Instead of creating routines for vector and matrix multiplication, the non-linear observer equations was written out in a more "variable-by-variable" fashion. Then these results where combined as "structs" too keep the code more readable. For example, gaining the vector $\boldsymbol{\omega}_{mes} = [x\, y\, z]^\top$ with the scalar $-K_i$ (to create $\hat{\dot{\boldsymbol{b}}}^b_{gyro}$) is done as follows;

```
1  struct vector w_mes_Ki_gain(struct vector *w){
2       struct vector b_dot;
3           b_dot.x = w->x * (-Ki);
4           b_dot.y = w->y * (-Ki);
5           b_dot.z = w->z * (-Ki);
6       return b_dot;
```

The whole code has a structure similar to figure (8.3), where each function call is numerated. This observer utilizes quaternions for attitude representation. This is very efficient for realization on a computer. The amount of differential equations is undoubtably reduced compared to a Kalman filter (Ricatti Equations). A reduction of code less prone to errors and bugs is also possible. For micro controllers it's a great advantage to avoid the use of trancedental functions like sine, cosine and tangent. This either has to be realized as look-up-tables or for example a tailor expansion. Both with limited accuracy. A full hardware implementation require a multitude of registers. This is complicated, expensive and the consistency is hard to verify if exposed to space radiation compared to software (according to the

NUTS-team in charge of code verification).

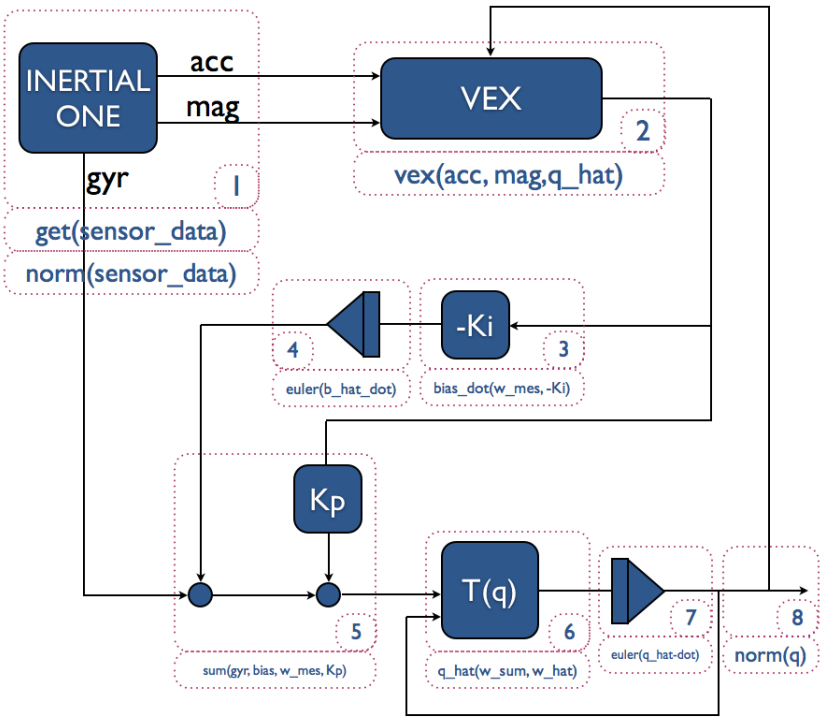

Figure 8.3: Nonlinear Observer divided into function calls

Written in pseudo code:

```
1       global q_hat;
2       global ForwardEuler_bias;
3       global ForwardEuler_q_hat;
4
5       while (true) {
6
7           sensor_read(gyroscope);
8           sensor_read(accelerometer);
```

```
9           sensor_read(compass);
10
11          acc_norm = normalize(acc);
12          mag_norm = normalize(mag);
13
14          omega_mes = vex(acc, acc_norm, mag, mag_norm, q_hat);
15
16          bias_dot = w_mes_Ki_gain(omega_mes, Ki);
17
18          bias = ForwardEuler(bias_dot);
19
20          omega_mes_sum = sum(gyr, bias, omega_mes, Kp);
21
22          q_hat_dot = generate(omega_mes_sum, q_hat);
23
24          q_hat = ForwardEuler(q_hat_dot);
25
26          q_hat = normalize_q(q_hat);
27
28          print_LCD(q2euler(q_hat));
29      }
```

The program does not follow any clock or synchronization mechanism. It iterates through the code as fast as possible. This can result in faulty step-sizes through the integrators. To avoid this problem, the time-stamp from the sensors was initially used as reference. But after benchmarking, a constant step size was implemented.

8.4 Logging of data

The A3BU-kit is a great tool for a "proof of concept" design. But outputting data to a display has limitations. A need to log data soon arise. Also, the Mahony Observer had to be implemented on the microcontroller given in the requirement specification. Atmel also provides developments kits with

this MCU;

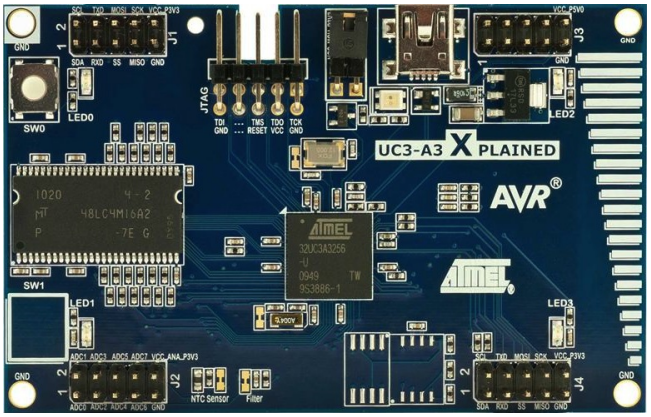

Figure 8.4: UC3-A3 Xplained 1:1 scale

- UC3-A3 Xplained development kit
  Unit price of $29 USD
- AVR® AT32UC3A3256
  High performance low power 32-bit MCU
  128KB of RAM and 256KB programmable high-speed storage
- 66MHz clock frequency
- Power consumtion < 1 Watt for entire kit
- Otherwise equal to the A3BU

The Hardware-team is also using this development platform to test the internal communication protocol of the NUTS satellite. The C-code for the nonlinear observer was ported and extended with data logging support. The UC3-A3 device connects via USB and creates a *virtual com port*. Any kind of data can then be serialized and sent to the host computer. It does however affect the performance of the microcontroller to some degree.

8.5 Creating a handheld battery powered IMU

The NUTS team often have displays and stands where they show of their work. To better provide NUTS with a more tangible ADCS system, some effort where made to create a display unit. A box with a battery compartment and plexiglass cover was modeled on the SolidWorks™ CAD platform. The box houses the A3BU Xplained kit and Inertial One sensor unit as a stand alone IMU. The box was printed with a 3D-printer in black ABS-plastic at the Department of Engineering Cybernetics Workshop. The final unit makes it very easy to explain how the system works. It also makes it easier to continue the work on the ADCS system. Estimated total cost is roughly $60,- USD plus a single AAA-battery.

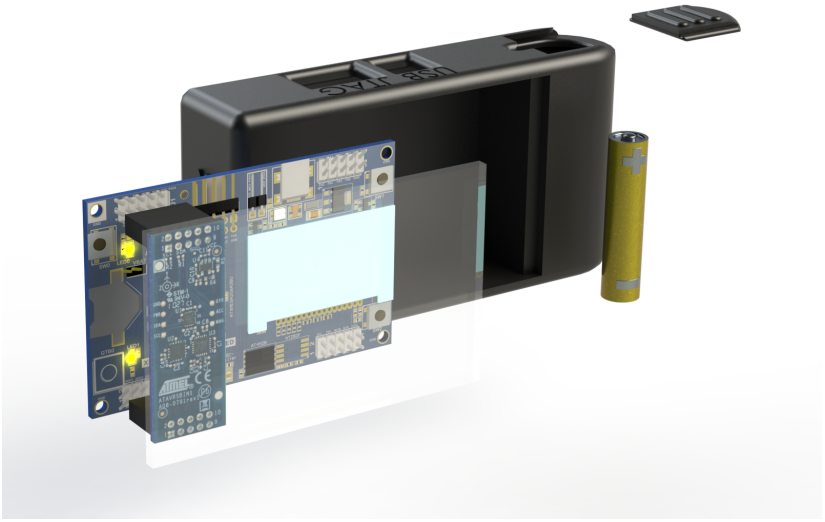

Figure 8.5: 3D-printed IMU "exploded" view

Figure 8.6: 3D-printed IMU assembled view

Chapter 9

Design Verification

The simulink simulator and the embedded system are here put through testing and verification. This is to make sure both the developed systems confines with the NUTS ADCS requirement specification.

To limit the amount of presented data, only the most important 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.

9.1 Sumlink simulator environment

Two-stage controller from chapter (5).

Detumbling;

$$\begin{aligned} \boldsymbol{V} &= -\frac{k_d}{|\boldsymbol{B}^b|}\boldsymbol{K}_{coil}^{-1}\left[\boldsymbol{B}^b \times (\boldsymbol{\omega}_{ob}^b - \hat{\boldsymbol{b}}_{gyro}^b)\right] \\ ||\boldsymbol{\omega}_{ob}^b|| &\leq 0.03 \\ |\boldsymbol{V}| &< 5\,\text{volts} \end{aligned}$$

PD-controller;

$$\begin{aligned} \boldsymbol{V} &= -\frac{\boldsymbol{K}_{coil}^{-1}}{|\boldsymbol{B}^b|}\left[-k_p(\boldsymbol{B}^b \times \boldsymbol{\epsilon}) - k_d\left(\boldsymbol{B}^b \times (\boldsymbol{\omega}_{ob}^b - \hat{\boldsymbol{b}}_{gyro}^b)\right)\right] \\ ||\boldsymbol{\omega}_{ob}^b|| &> 0.03 \\ |\boldsymbol{V}| &< 5\,\text{volts} \end{aligned}$$

Mahony Nonlinear Attitude Observer from section (6.3);

$$\begin{aligned} \boldsymbol{\omega}_{mes}^b = &\left[\frac{k_1}{2}\left(\frac{\boldsymbol{s}_{sensor}^b}{\|\boldsymbol{s}_{sensor}^b\|}\right) \times \left(\boldsymbol{R}_o^b(\hat{\boldsymbol{q}})\frac{\boldsymbol{R}_s^o\boldsymbol{s}^s}{\|\boldsymbol{R}_s^o\boldsymbol{s}^s\|}\right) + \cdots\right. \\ &\left.\cdots\ \frac{k_2}{2}\left(\frac{\boldsymbol{m}_{mag}^b}{\|\boldsymbol{m}_{mag}^b\|}\right) \times \left(\boldsymbol{R}_o^b(\hat{\boldsymbol{q}})\frac{1}{\sqrt{m_x^2+m_y^2+m_z^2}}\begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix}\right)\right] \\ \dot{\hat{\boldsymbol{q}}} &= \boldsymbol{T}_q(\hat{\boldsymbol{q}})\left[\boldsymbol{\omega}_{imu}^b - \hat{\boldsymbol{b}}_{gyro}^b + \boldsymbol{K}_p\boldsymbol{\omega}_{mes}^b\right] \\ \dot{\hat{\boldsymbol{b}}}_{gyro}^b &= -\frac{1}{2}\boldsymbol{K}_i\boldsymbol{\omega}_{mes}^b \end{aligned}$$

Table (9.1) lists the common parameters used for all simulator test cases. Table (9.2) to (9.11) is provided for each test case to differentiate the scenarios.

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.

Because of the initially high angular velocity, the dampening gain in the detumbling controller was halved compared to the PD-controller. This is to avoid both high energy consumption and aggressive behavior. The proportional gain in the PD-controller is as high as it gets, before turning unstable.

The initial gains for the observer was found in [27]. The most thrusted measurement is considered to be the sun sensor, that is why it is gained almost twice as much as the magnetometer. The injection term is relatively small, and only a small correction should be necessary, so it is not gained. The gyroscope bias was tuned to converge efficiently fast, without any overshoots or reaction changes in the angular velocity.

Test plan

1. Fully working satellite with all sensor systems, controller and estimation algorithm
2. Sun sensor loss during 40 % of orbit (shadow of Earth)
3. Orbital magnetic reference with limited resolution like a LUT
4. Disturbance torque from aerodynamic drag
5. Sudden tumbling caused by micro meteorite
6. Total loss of gyroscope sensor data
7. Loss of gyroscope with measurement noise
8. Total loss of sun sensor
9. Loss of magnetometer with measurement noise
10. Gyroscope signal freeze

Table 9.1: Common parameters used for all simulator test cases

Parameter	**Value**	**Comment**
Simulator	Simulink Simulator	See section (7.1)
Simulation time	60.000 seconds	$\approx$ 10 orbits
Solver	ode3 (Bogacki-Shampine)	Stepsize = 0.1 (no significant changes compared to 0.01, saved time)
Altitude	600 km	Low Earth orbit (LEO)
Mass	2.6 kg	Estimated
Coils	$N_x = 355$ $N_y = 800$ $N_z = 800$ $A_x = 144\,cm^2$ $A_y = 144\,cm^2$ $A_z = 64\,cm^2$	Copper windings Area
(from model in [3])	$R_x, R_y, R_z = 110\,\Omega$	Resitance
Battery	3.3 V 8800 mAh	
Initial attitude	$\boldsymbol{\Theta}_{ob} = \begin{bmatrix} 10^\circ \\ 5^\circ \\ -2^\circ \end{bmatrix}$	Roll, pitch yaw in degrees
Initial angular velocity	$\boldsymbol{\omega}^b_{ob} = \begin{bmatrix} 5.7^\circ/s \\ -11.5^\circ/s \\ 2.9^\circ/s \end{bmatrix}$	Roll, pitch, yaw in degrees/sec
Gyro bias	$\boldsymbol{b}^b_{gyro} = \begin{bmatrix} -30^\circ/s \\ 40^\circ/s \\ 25^\circ/s \end{bmatrix}$	Roll, pitch, yaw in degrees/sec
Mahony observer gains	$k_1 = 1$ $k_2 = 0.55$ $K_p = 1$ $K_i = 0.008$	Sun sensor Magnetometer Injection Gyro bias
Detumbling controller gains	$k_d = 4 \cdot 10^{-5}$	Damping
Reference controller gains	$k_p = 3 \cdot 10^{-5}$ $k_d = 8 \cdot 10^{-5}$	Proportional Damping

Test case 1 - All systems perfect

Table 9.2: Parameters for simulator in test case 1

Parameter	Value		Comment
Sun sensor availability	100 % of orbit		Resembles zero inclination angle
Magnetic reference	Continous		
Disturbances	Gravity	ON	
	Aerodynamic drag	OFF	
	Micro meteorite	OFF	
Single point failures	Gyroscope	None	
	Sun sensor	None	
	Magnetometer	None	

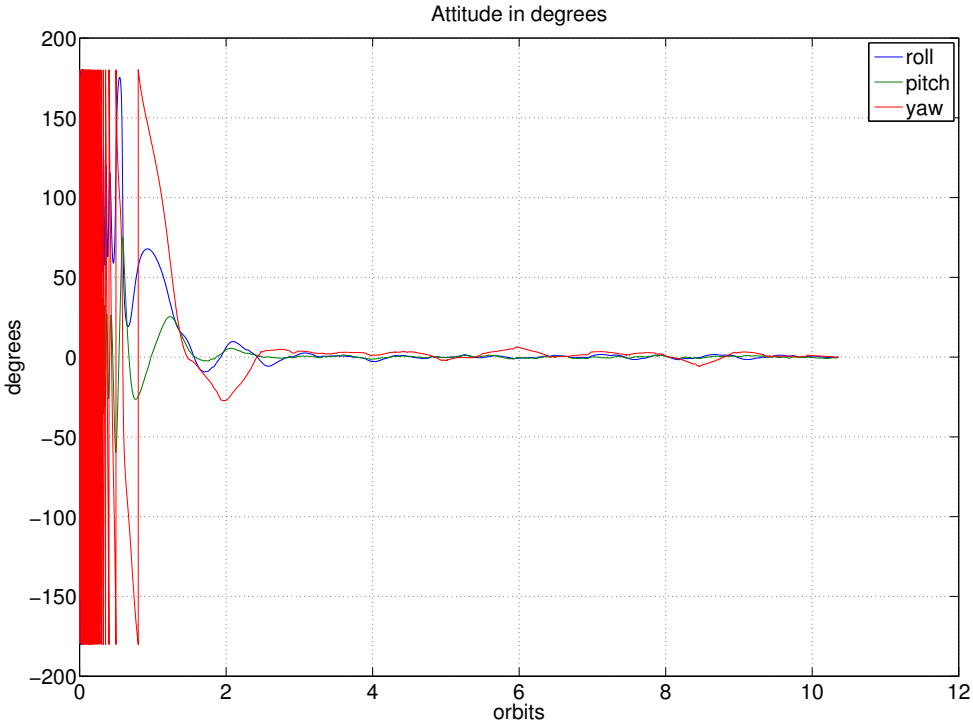

Figure 9.1: Attitude - Test case 1

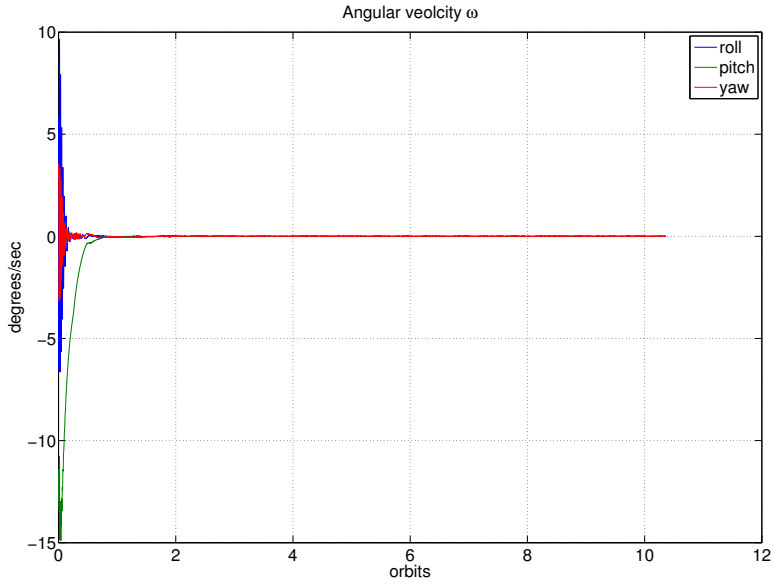

Figure 9.2: Angular velocity - Test case 1

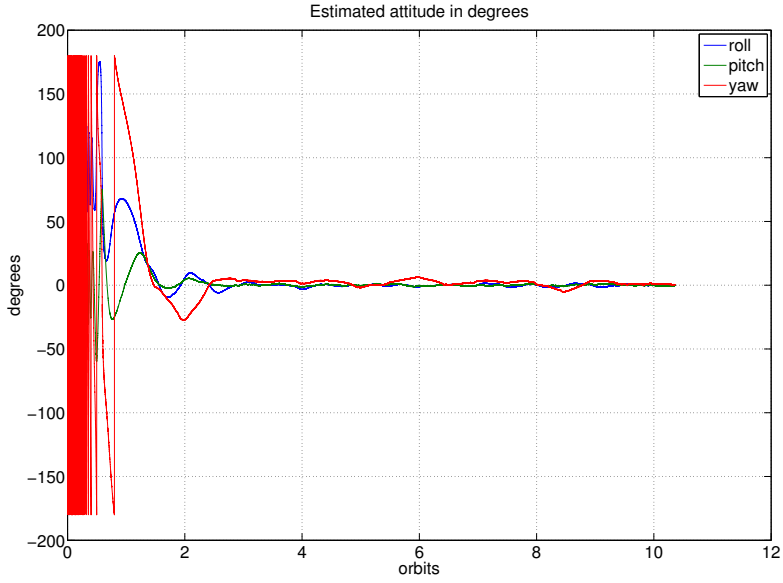

Figure 9.3: Estimated attitude - Test case 1

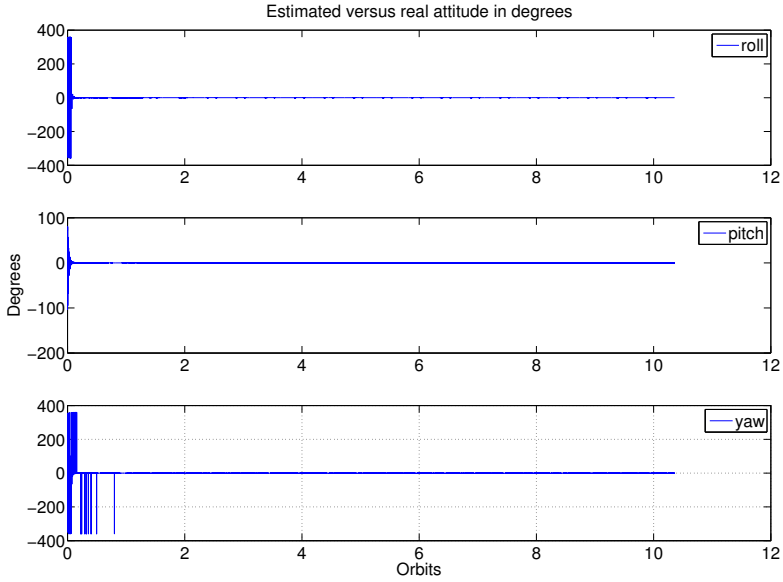

Figure 9.4: Estimated versus real attitude - Test case 1

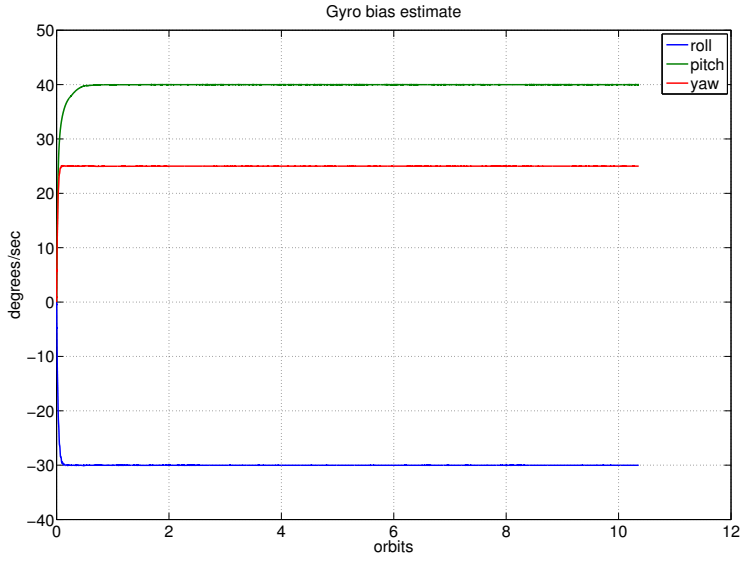

Figure 9.5: Gyroscope bias estimate - Test case 1

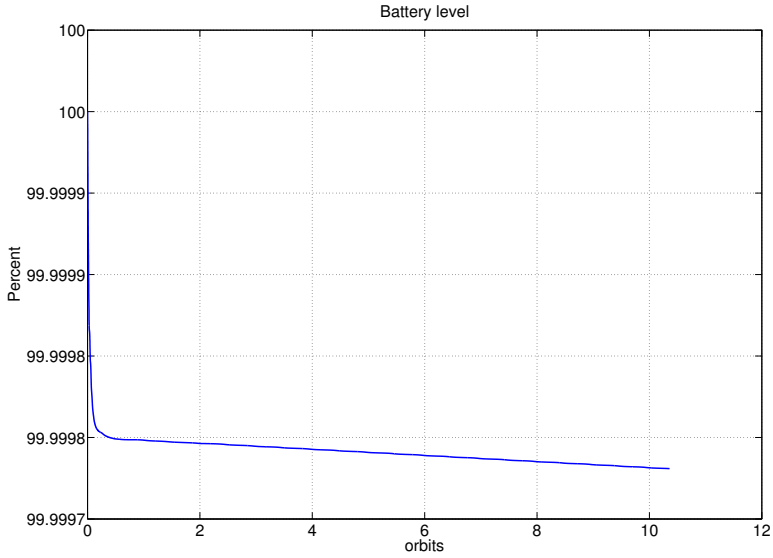

Figure 9.6: Battery level - Test case 1

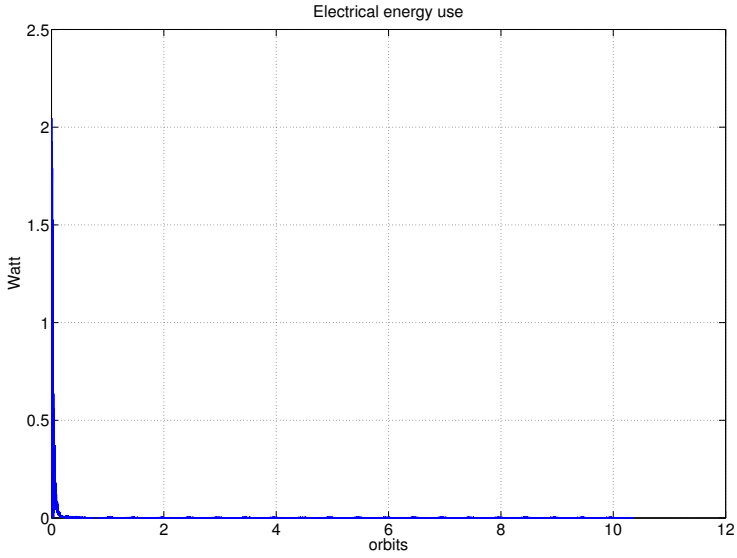

Figure 9.7: Watt use - Test case 1

Test case 2 - Sun sensor in Earth's shadow

Table 9.3: Parameters for simulator in test case 2

Parameter	Value		Comment
Sun sensor availability	60 % of orbit		Resembles normal inclination angle
Magnetic reference	Continous		
Disturbances	Gravity	ON	
	Aerodynamic drag	OFF	
	Micro meteorite	OFF	
Single point failures	Gyroscope	None	
	Sun sensor	None	
	Magnetometer	None	

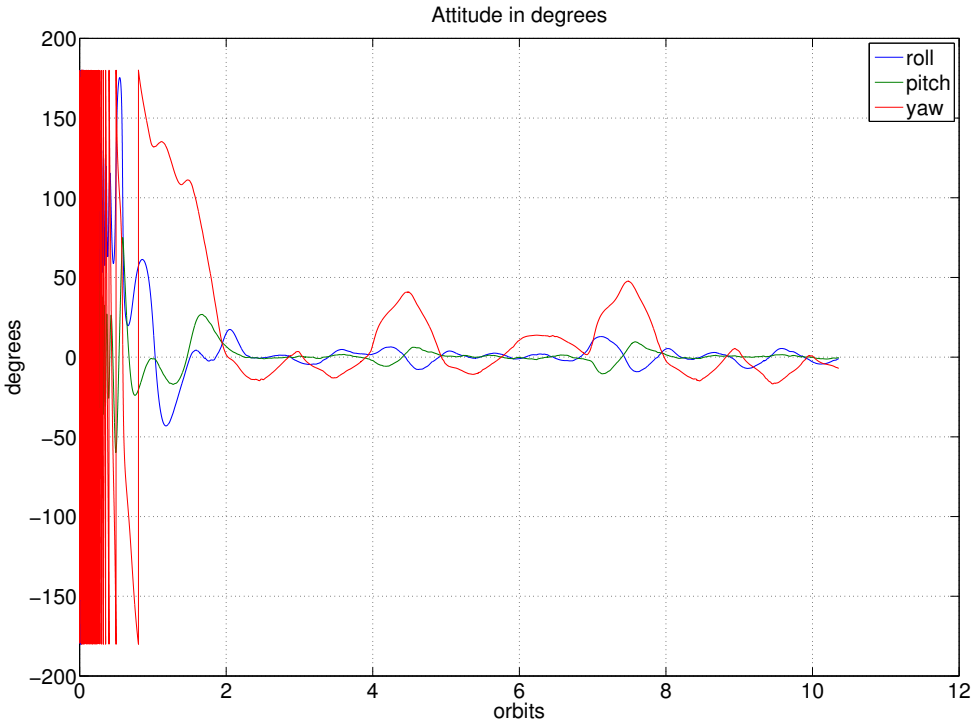

Figure 9.8: Attitude - Test case 2

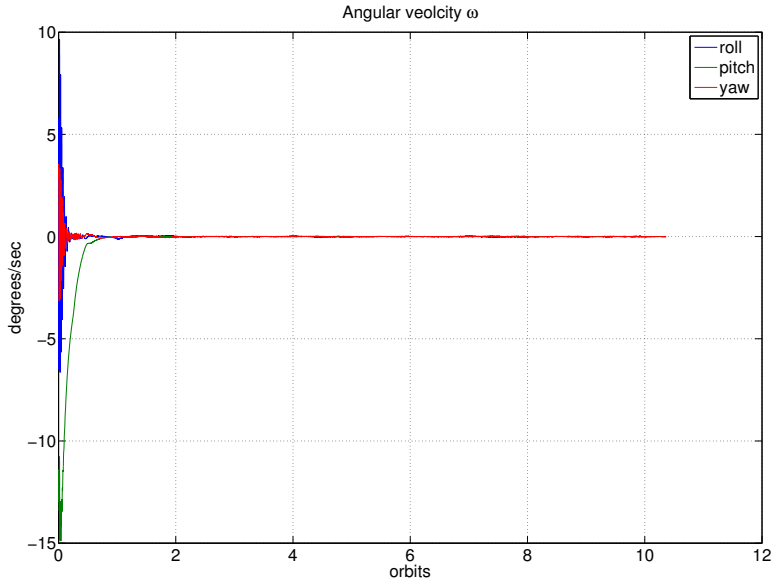

Figure 9.9: Angular velocity - Test case 2

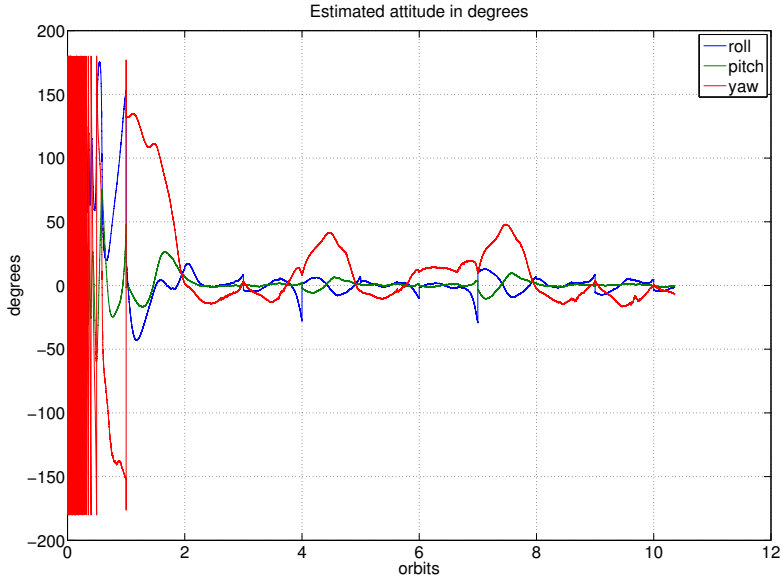

Figure 9.10: Estimated attitude - Test case 2

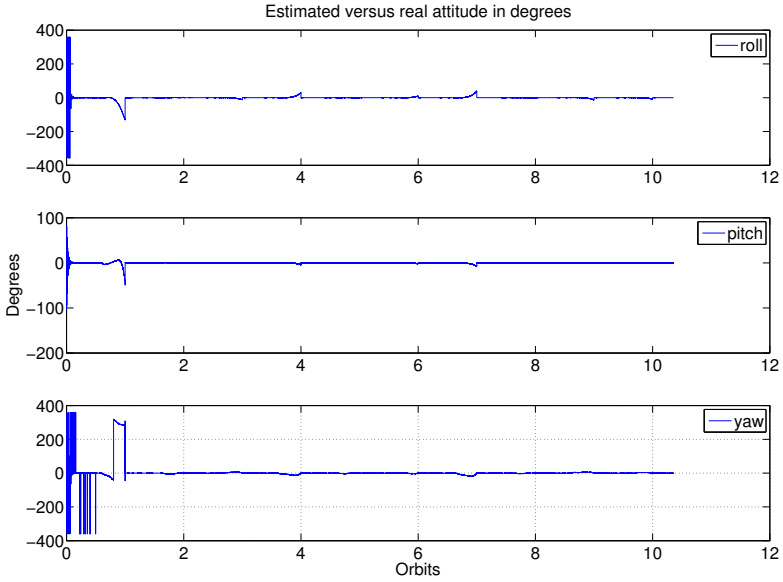

Figure 9.11: Estimated versus real attitude - Test case 2

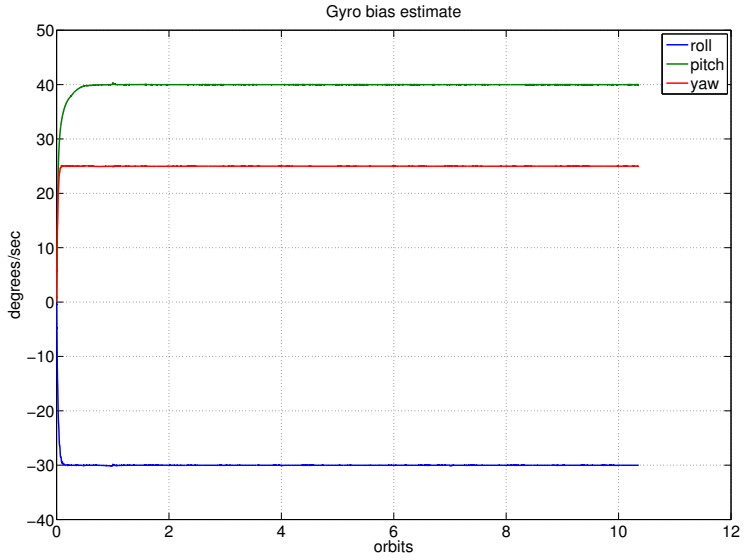

Figure 9.12: Gyroscope bias estimate - Test case 2

Test case 3 - Low resolution magnetic reference

Table 9.4: Parameters for simulator in test case 3

Parameter	Value		Comment
Sun sensor availability	60 % of orbit		Resembles normal inclination angle
Magnetic reference	Sampled every 100 second		Resembles a LUT size of 10 kilobytes
Disturbances	Gravity	ON	
	Aerodynamic drag	OFF	
	Micro meteorite	OFF	
Single point failures	Gyroscope	None	
	Sun sensor	None	
	Magnetometer	None	

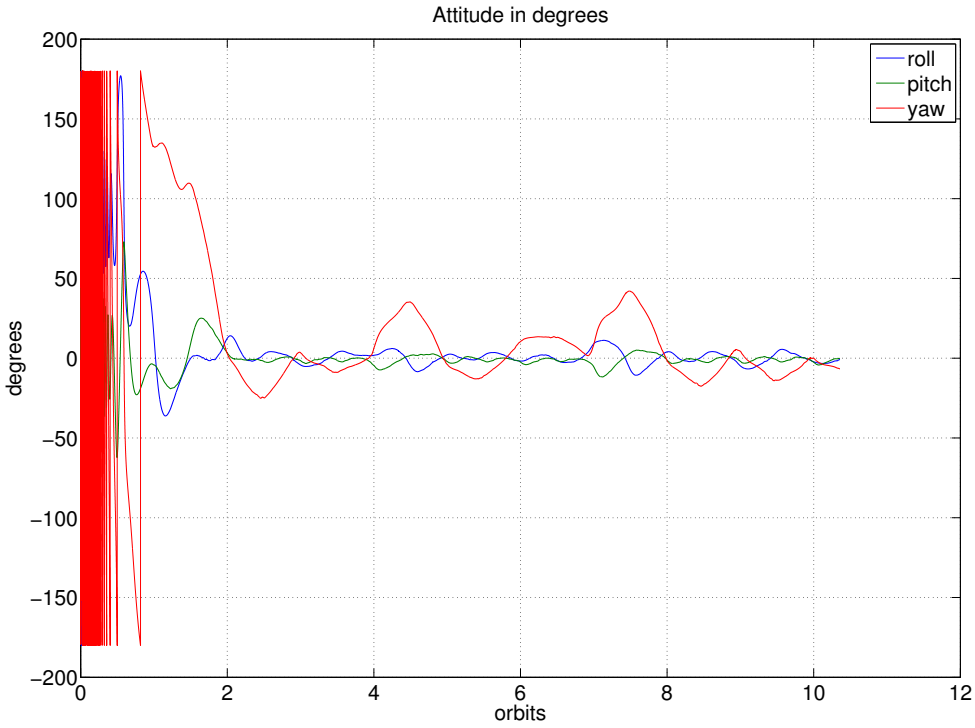

Figure 9.13: Attitude - Test case 3

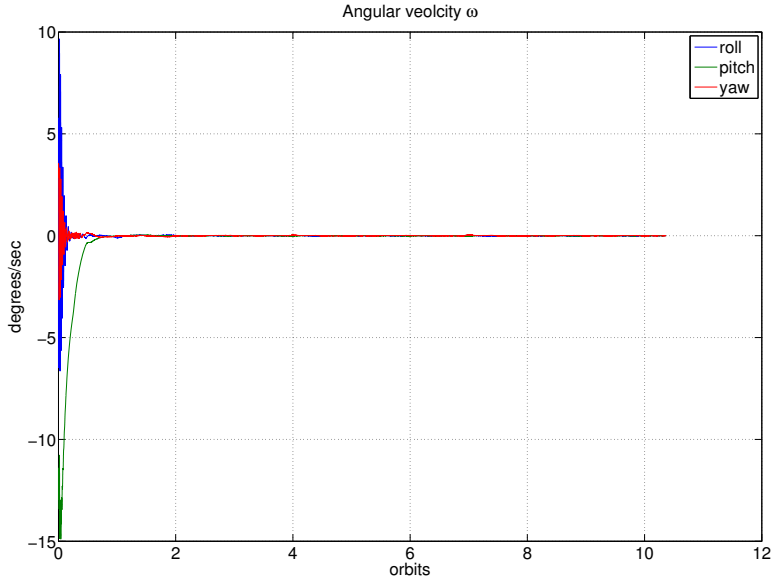

Figure 9.14: Angular velocity - Test case 3

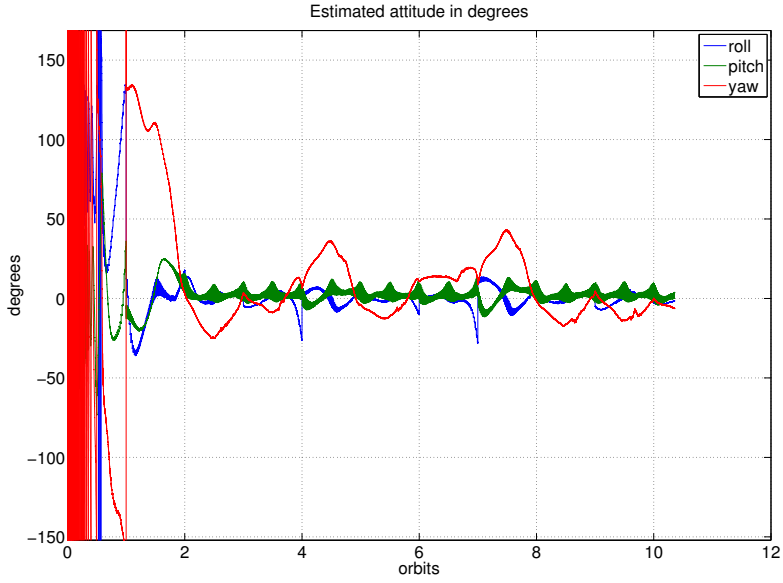

Figure 9.15: Estimated attitude - Test case 3

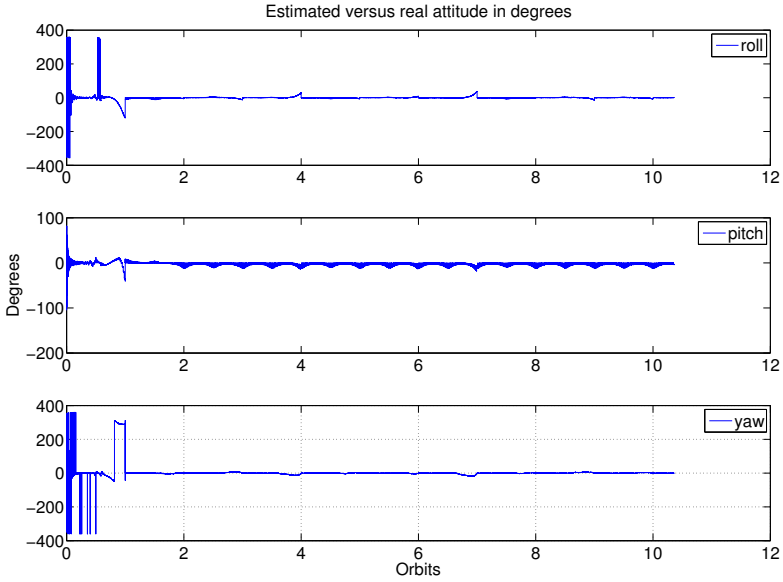

Figure 9.16: Estimated versus real attitude - Test case 3

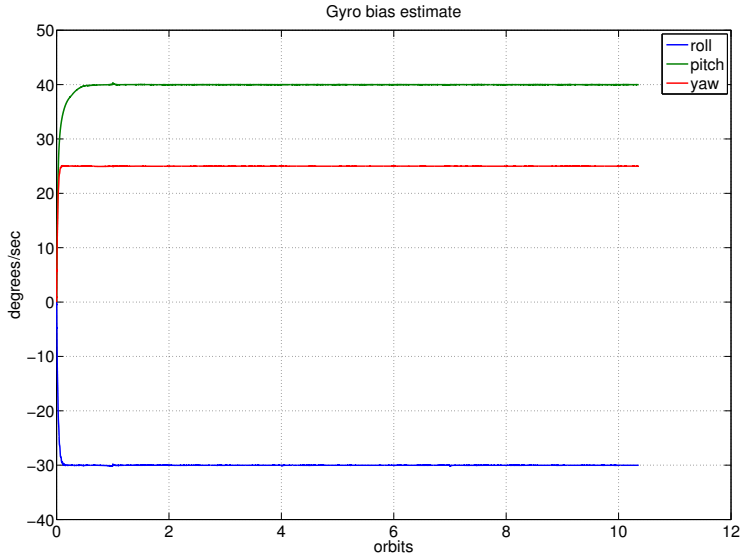

Figure 9.17: Gyroscope bias estimate - Test case 3

Test case 4 - Aerodynamic drag

Table 9.5: Parameters for simulator in test case 4

Parameter	Value		Comment
Sun sensor availability	60 % of orbit		Resembles normal inclination angle
Magnetic reference	Sampled every 100 second		Resembles a LUT size of 10 kilobytes
Disturbances	Gravity	ON	
	Aerodynamic drag	ON	
	Micro meteorite	OFF	
Single point failures	Gyroscope	None	
	Sun sensor	None	
	Magnetometer	None	

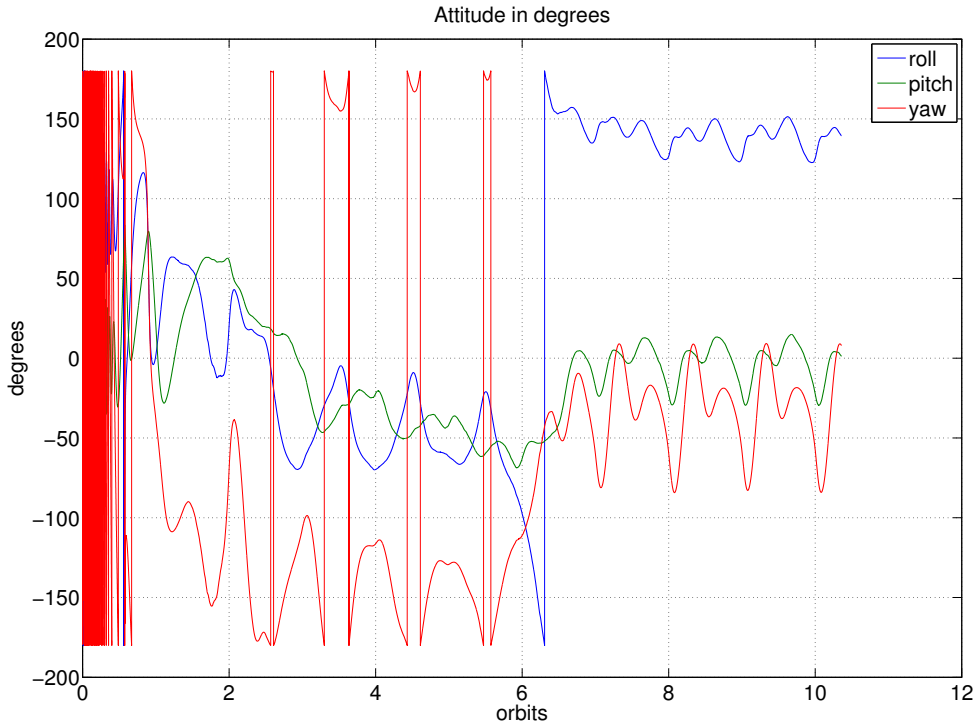

Figure 9.18: Attitude - Test case 4

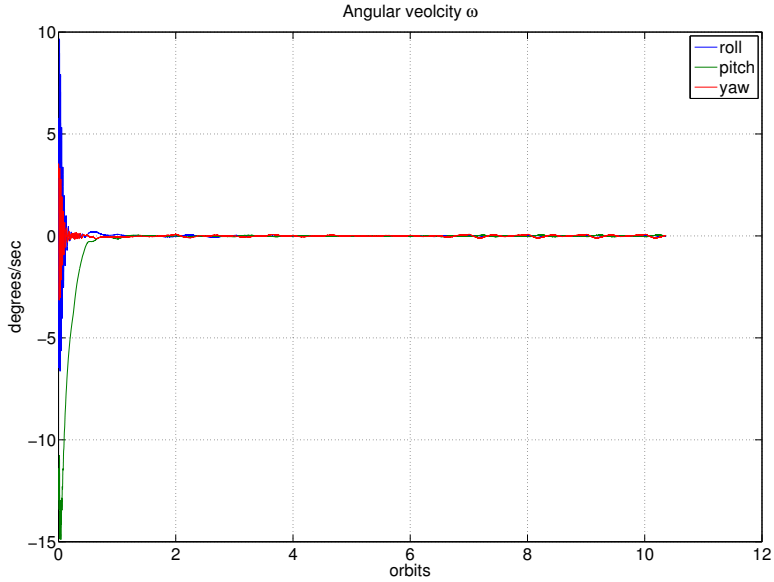

Figure 9.19: Angular velocity - Test case 4

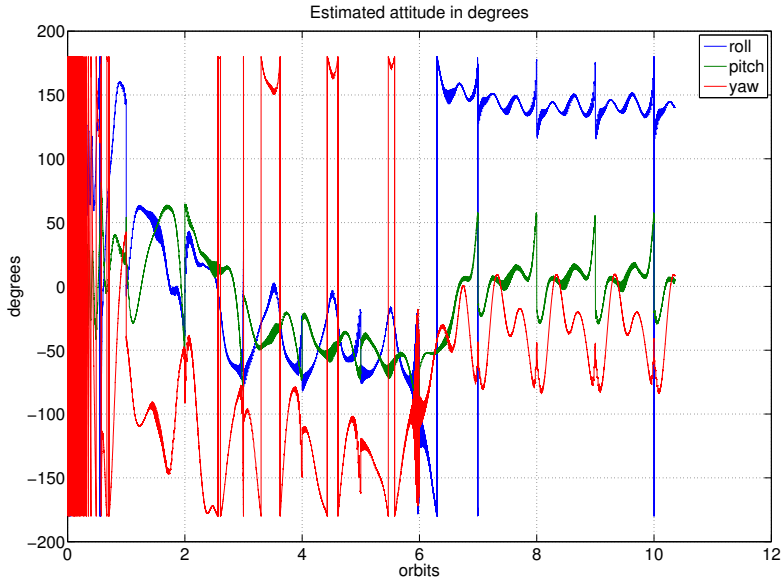

Figure 9.20: Estimated attitude - Test case 4

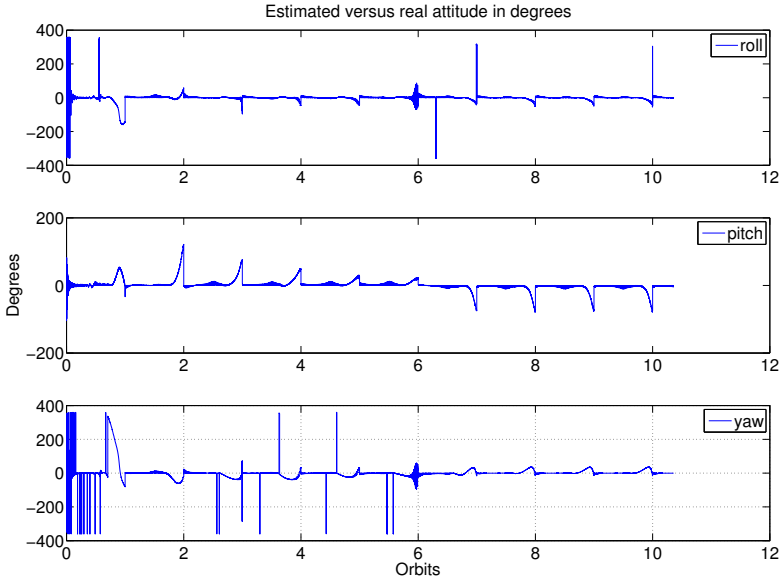

Figure 9.21: Estimated versus real attitude - Test case 4

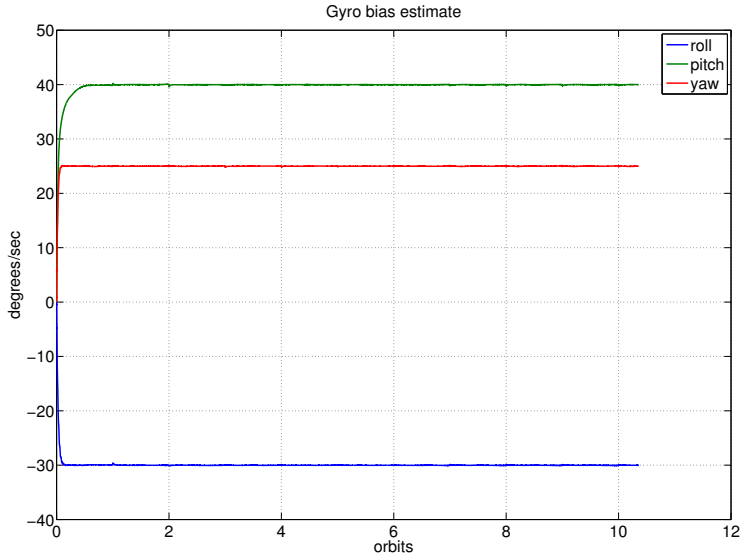

Figure 9.22: Gyroscope bias estimate - Test case 4

Test case 5 - Micro meteorite

Table 9.6: Parameters for simulator in test case 5

Parameter	Value		Comment
Sun sensor availability	60 % of orbit		Resembles normal inclination angle
Magnetic reference	Sampled every 100 second		Resembles a LUT size of 10 kilobytes
Disturbances	Gravity	ON	
	Aerodynamic drag	OFF	Meteorite strucks after 1 orbit
	Micro meteorite	ON	
Single point failures	Gyroscope	None	
	Sun sensor	None	
	Magnetometer	None	

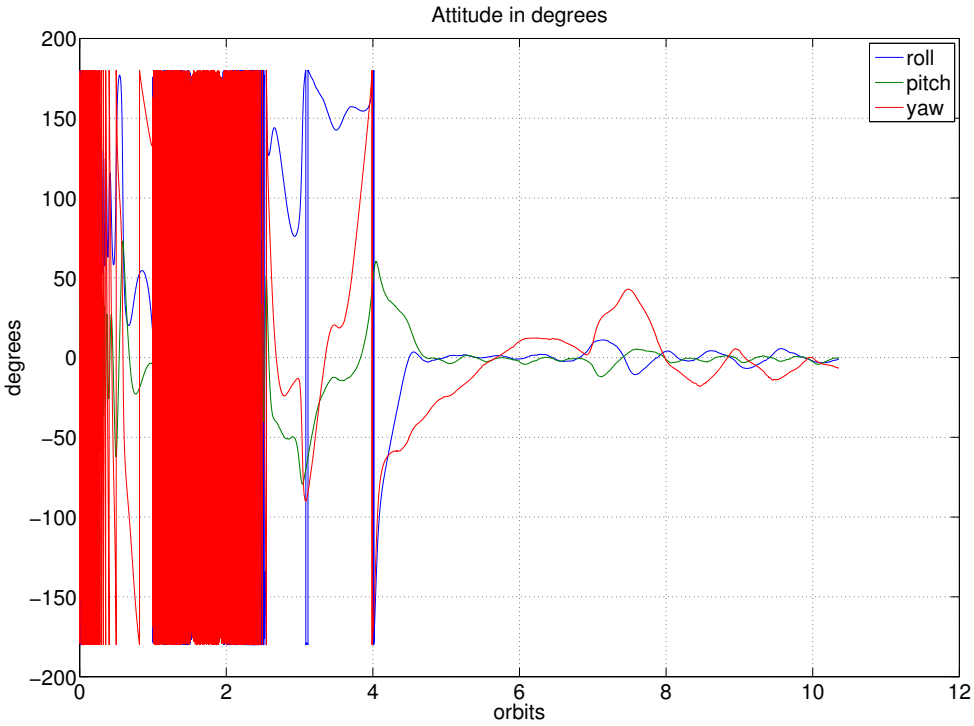

Figure 9.23: Attitude - Test case 5

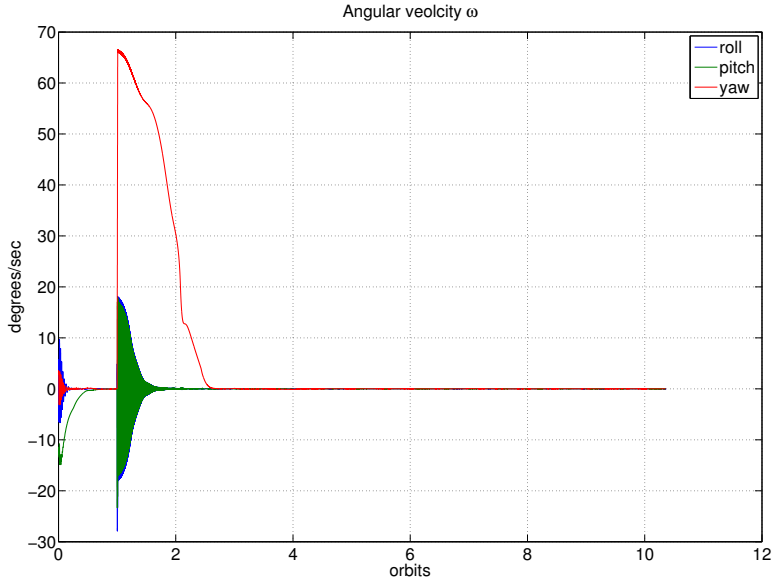

Figure 9.24: Angular velocity - Test case 5

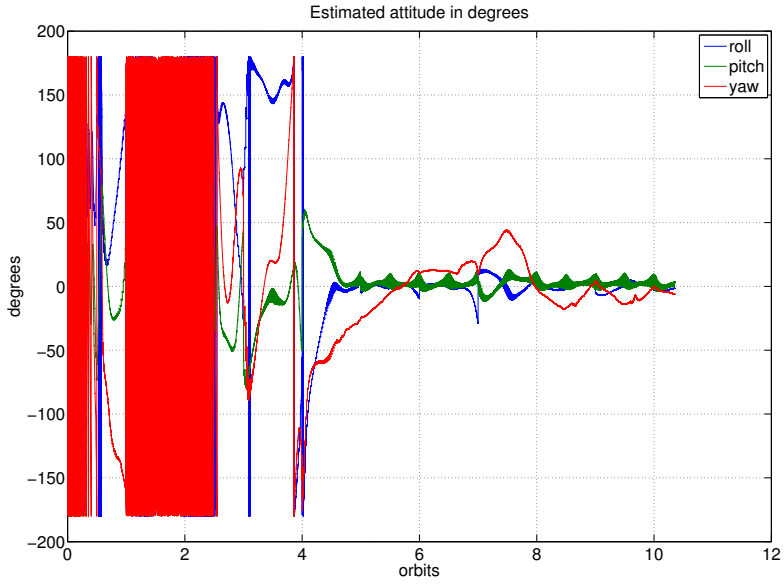

Figure 9.25: Estimated attitude - Test case 5

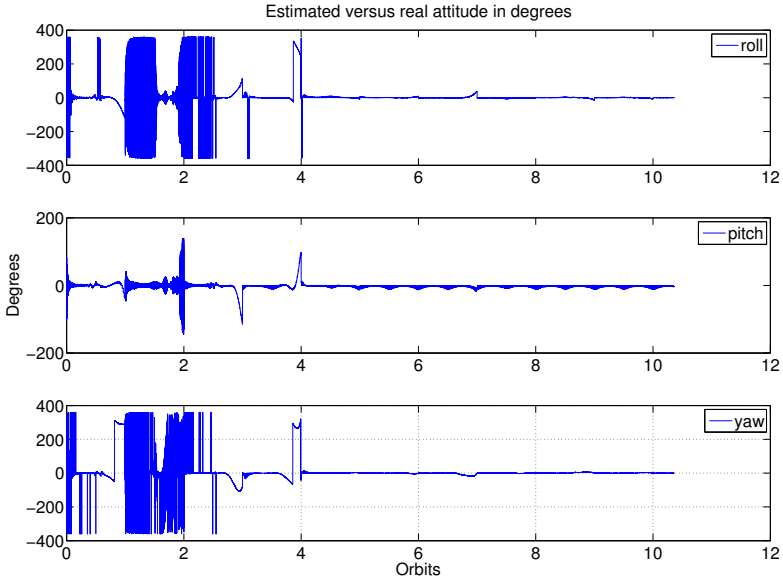

Figure 9.26: Estimated versus real attitude - Test case 5

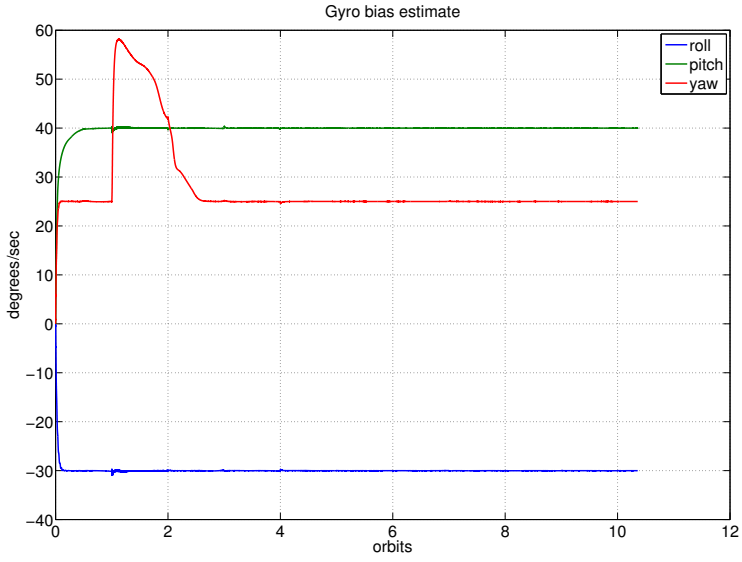

Figure 9.27: Gyroscope bias estimate - Test case 5

Test case 6 - Gyroscope loss

Table 9.7: Parameters for simulator in test case 6

Parameter	Value		Comment
Sun sensor availability	60 % of orbit		Resembles normal inclination angle
Magnetic reference	Sampled every 100 second		Resembles a LUT size of 10 kilobytes
Disturbances	Gravity	ON	
	Aerodynamic drag	OFF	
	Micro meteorite	OFF	
Single point failures	Gyroscope	Total loss	During entire simulation
	Sun sensor	None	
	Magnetometer	None	

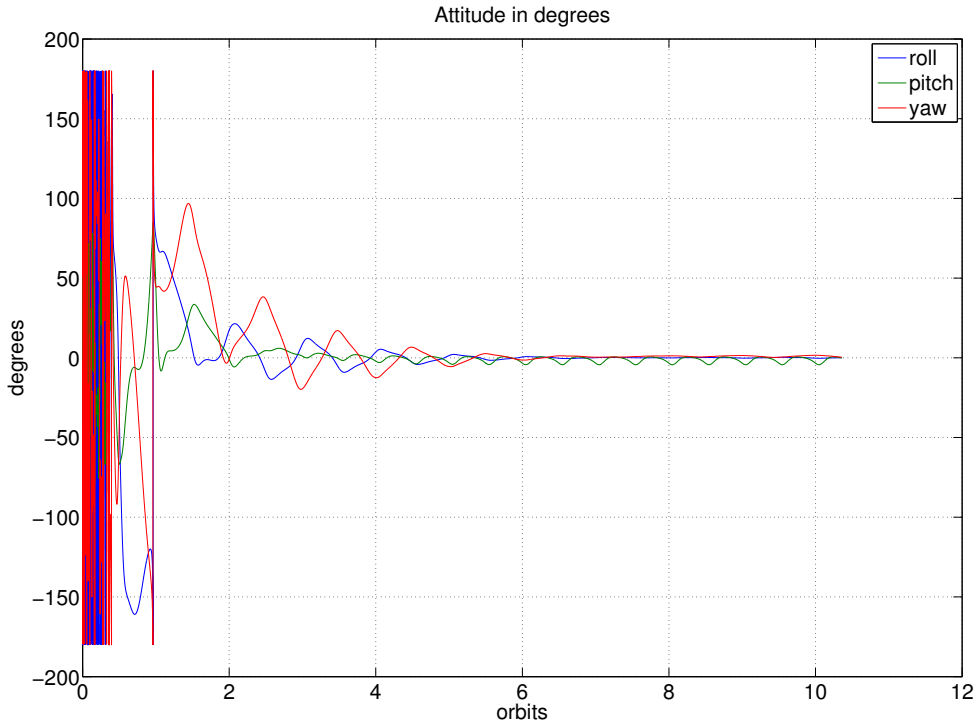

Figure 9.28: Attitude - Test case 6

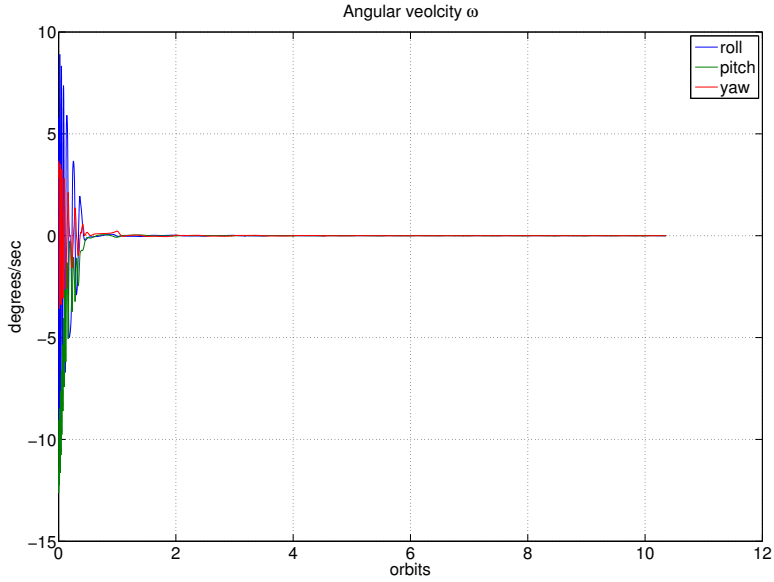

Figure 9.29: Angular velocity - Test case 6

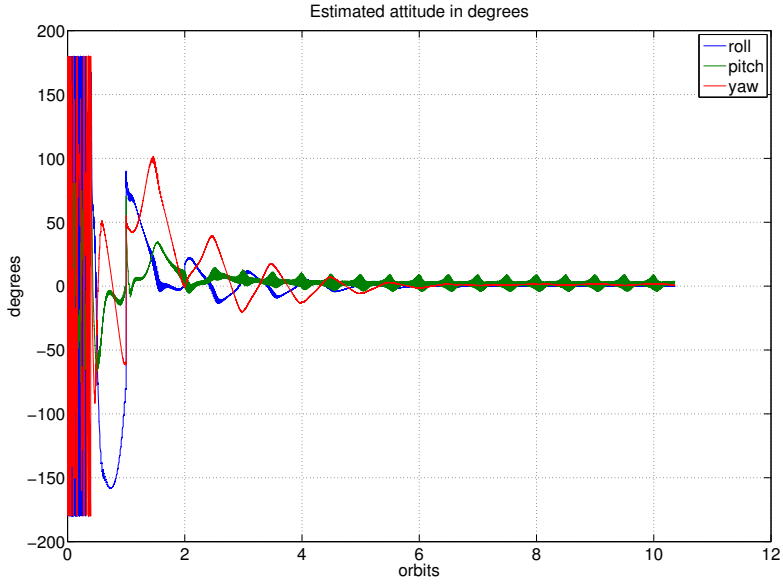

Figure 9.30: Estimated attitude - Test case 6

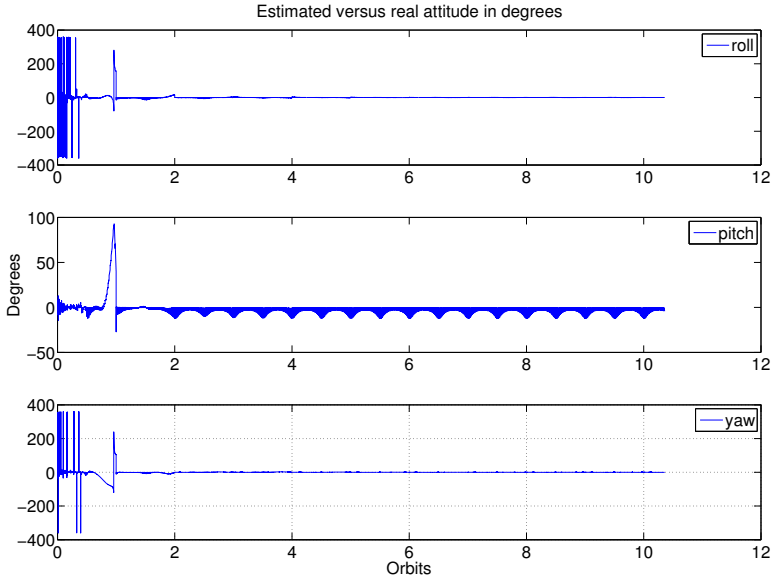

Figure 9.31: Estimated versus real attitude - Test case 6

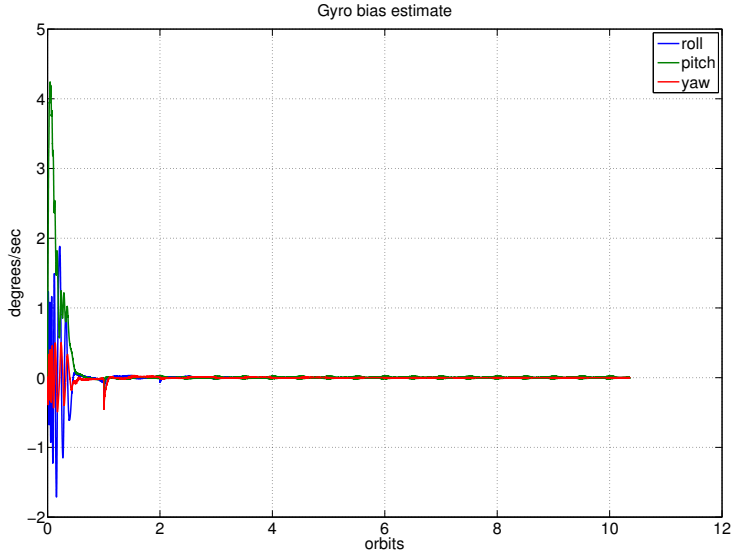

Figure 9.32: Gyroscope bias estimate - Test case 6

Test case 7 - Gyroscope loss with measurement noise

Table 9.8: Parameters for simulator in test case 7

Parameter	Value		Comment
Sun sensor availability	60 % of orbit		Resembles normal inclination angle
Magnetic reference	Sampled every 100 second		Resembles a LUT size of 10 kilobytes
Disturbances	Gravity	ON	
	Aerodynamic drag	OFF	
	Micro meteorite	OFF	
Single point failures	Gyroscope	Only noise	During entire simulation
	Sun sensor	None	
	Magnetometer	None	

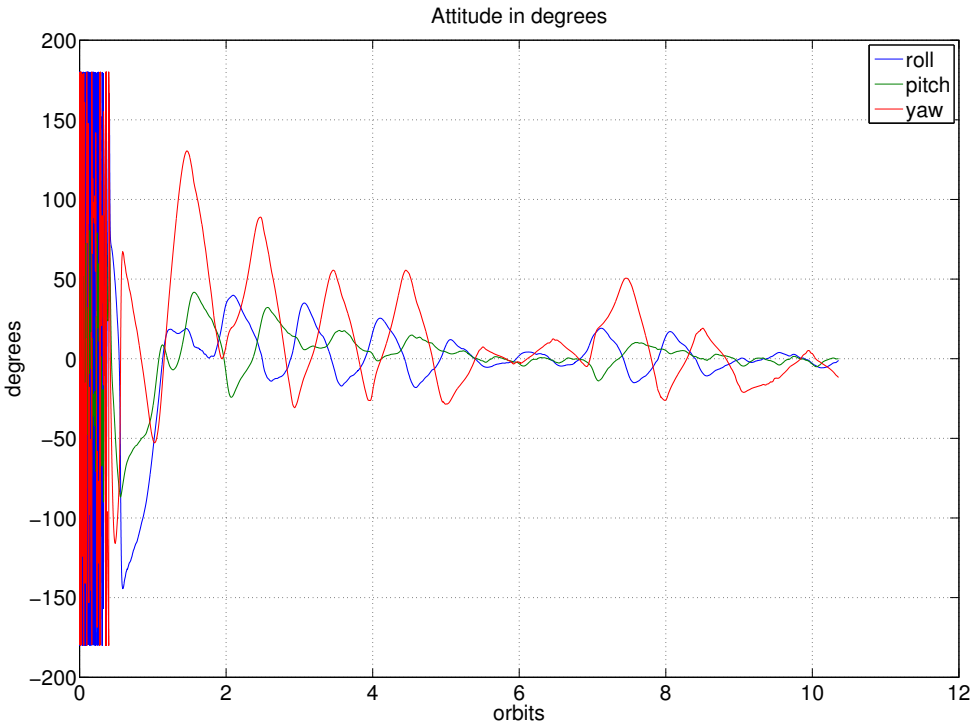

Figure 9.33: Attitude - Test case 7

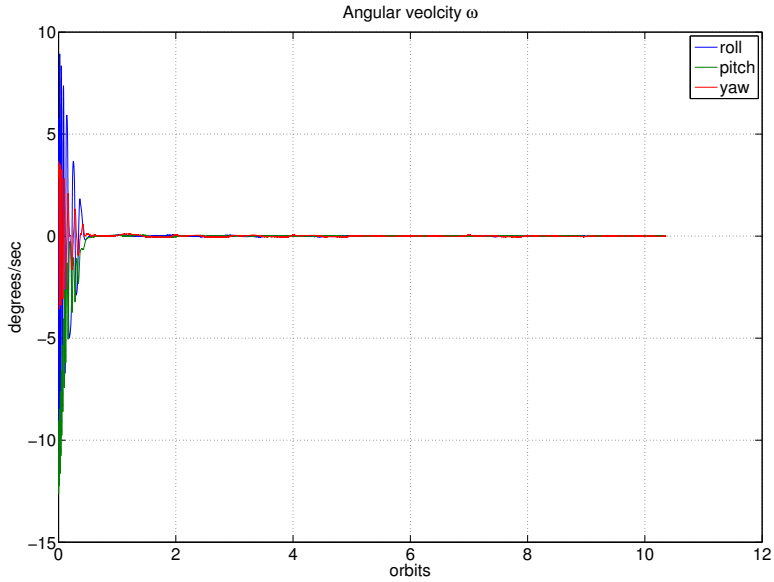

Figure 9.34: Angular velocity - Test case 7

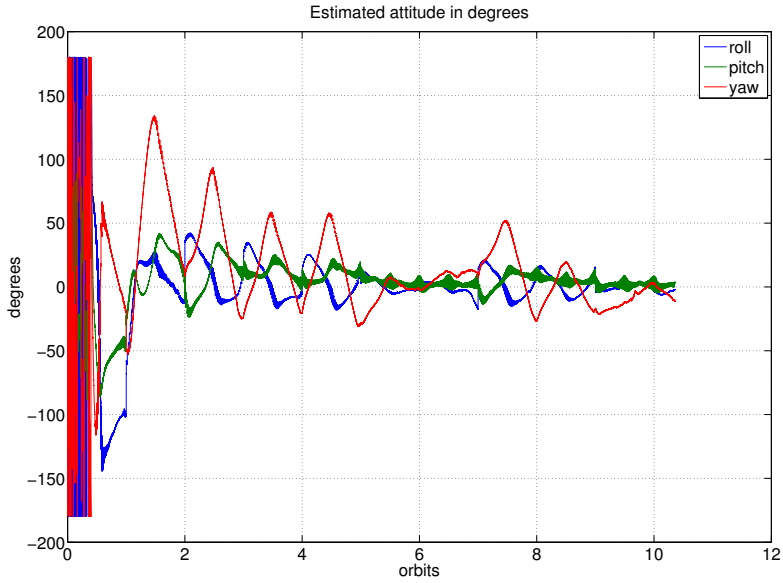

Figure 9.35: Estimated attitude - Test case 7

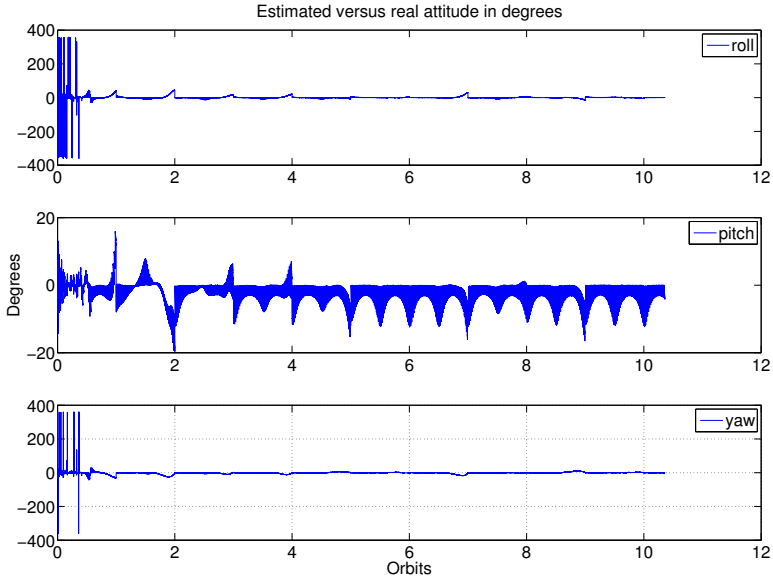

Figure 9.36: Estimated versus real attitude - Test case 7

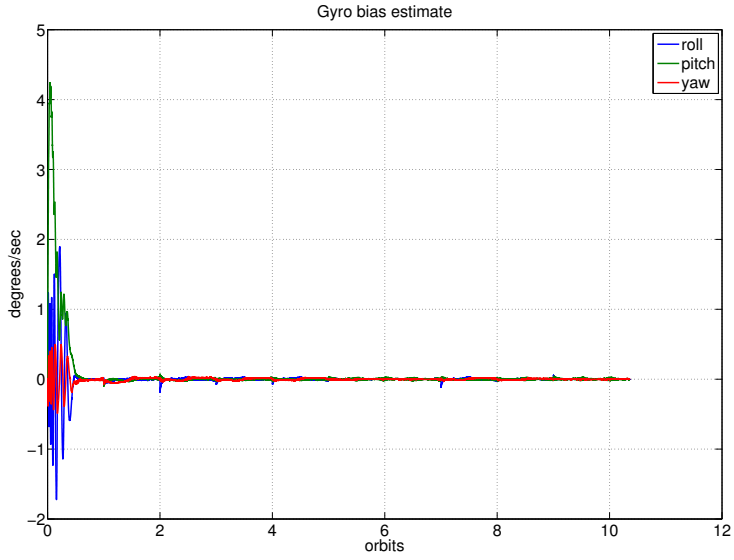

Figure 9.37: Gyroscope bias estimate - Test case 7

Test case 8 - Sun sensor loss

Table 9.9: Parameters for simulator in test case 8

Parameter	Value		Comment
Sun sensor availability	60 % of orbit		Failure
Magnetic reference	Sampled every 100 second		Resembles a LUT size of 10 kilobytes
Disturbances	Gravity	ON	
	Aerodynamic drag	OFF	
	Micro meteorite	OFF	
Single point failures	Gyroscope	None	
	Sun sensor	Total loss	During entire simulation
	Magnetometer	None	

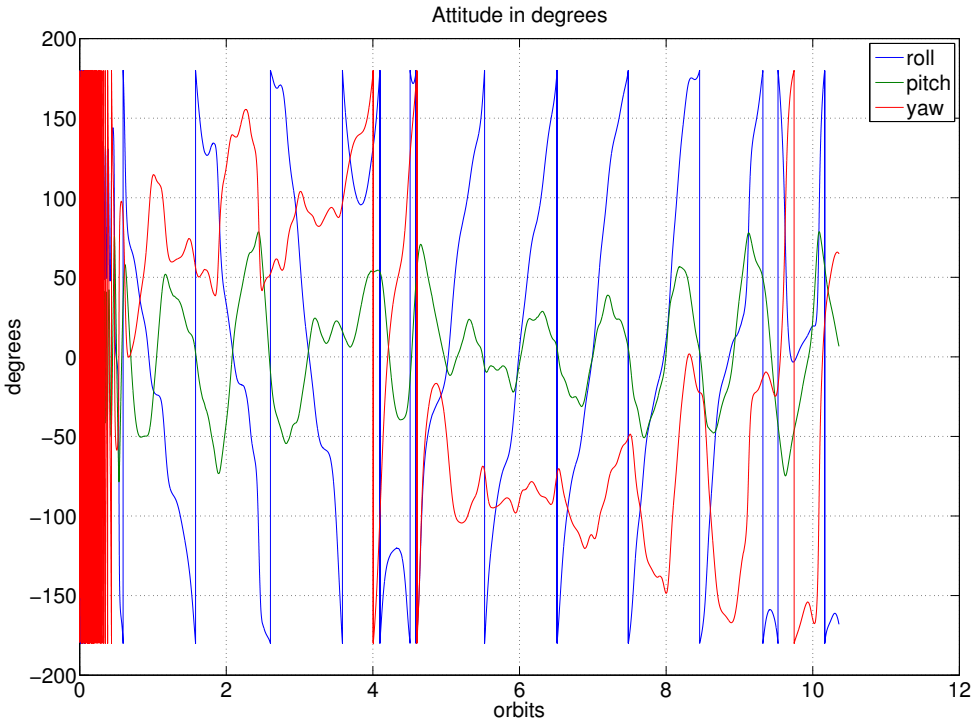

Figure 9.38: Attitude - Test case 8

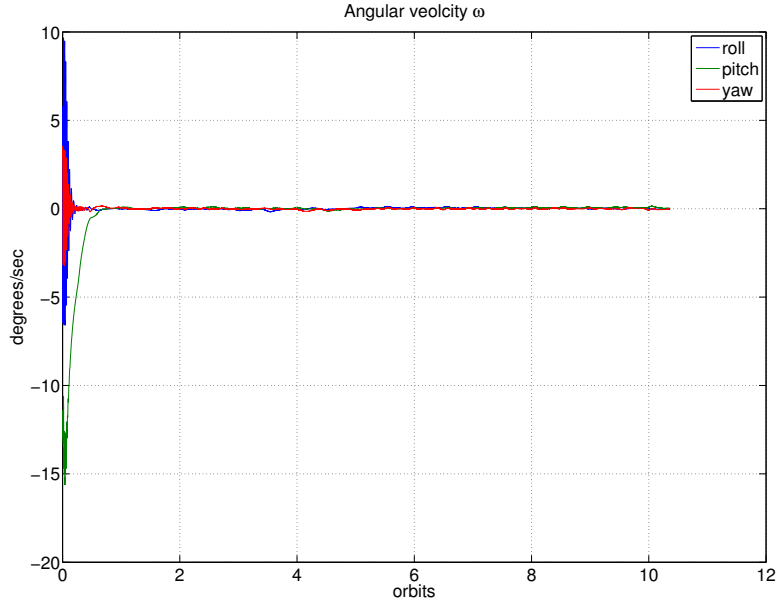

Figure 9.39: Angular velocity - Test case 8

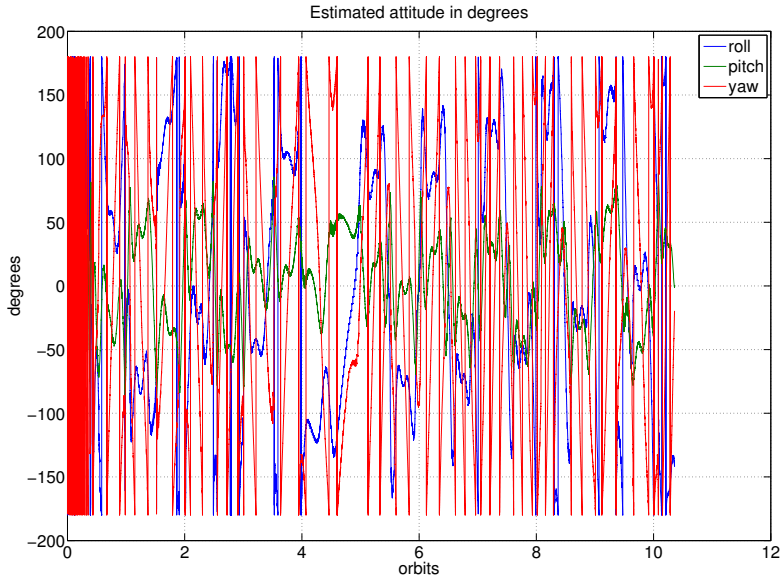

Figure 9.40: Estimated attitude - Test case 8

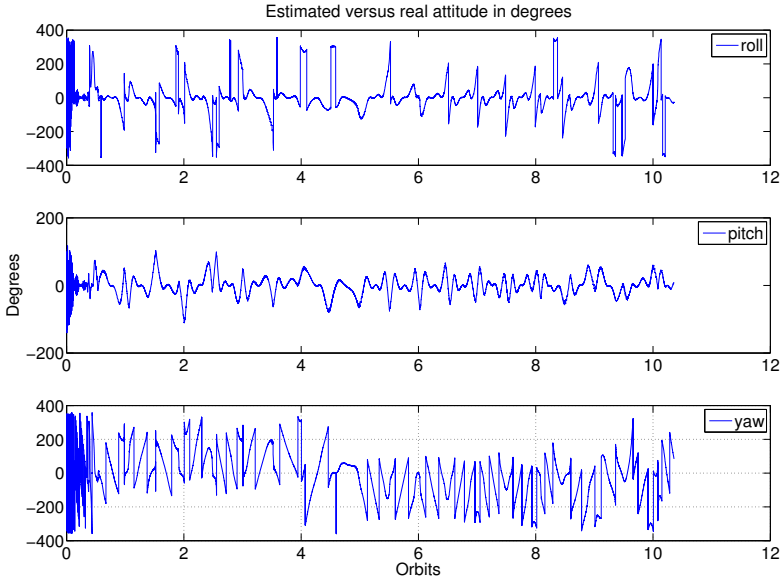

Figure 9.41: Estimated versus real attitude - Test case 8

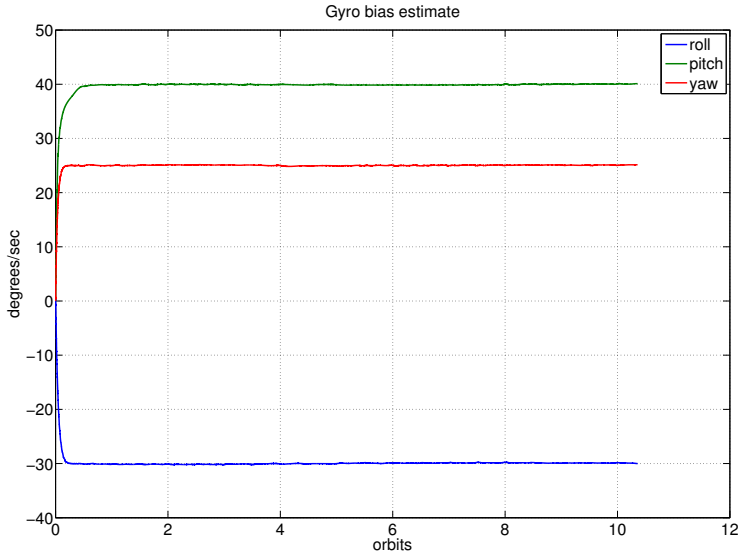

Figure 9.42: Gyroscope bias estimate - Test case 8

Test case 9 - Magnetometer loss

Table 9.10: Parameters for simulator in test case 9

Parameter	Value		Comment
Sun sensor availability	60 % of orbit		Resembles normal inclination angle
Magnetic reference	Sampled every 100 second		Resembles a LUT size of 10 kilobytes
Disturbances	Gravity	ON	
	Aerodynamic drag	OFF	
	Micro meteorite	OFF	
Single point failures	Gyroscope	None	
	Sun sensor	None	During entire simulation
	Magnetometer	Only noise	

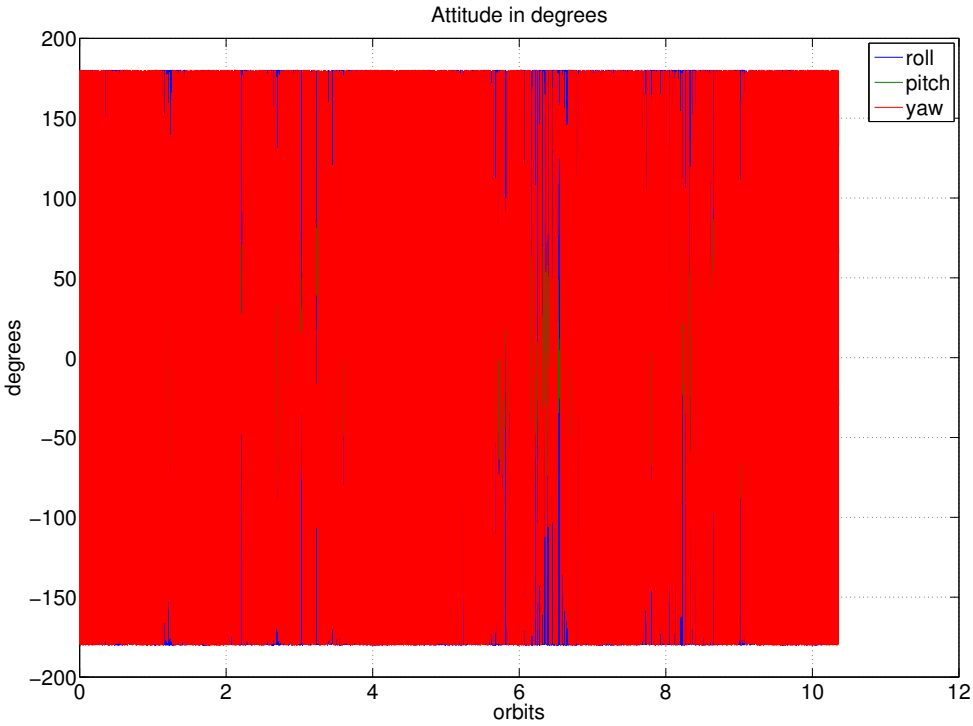

Figure 9.43: Attitude - Test case 9

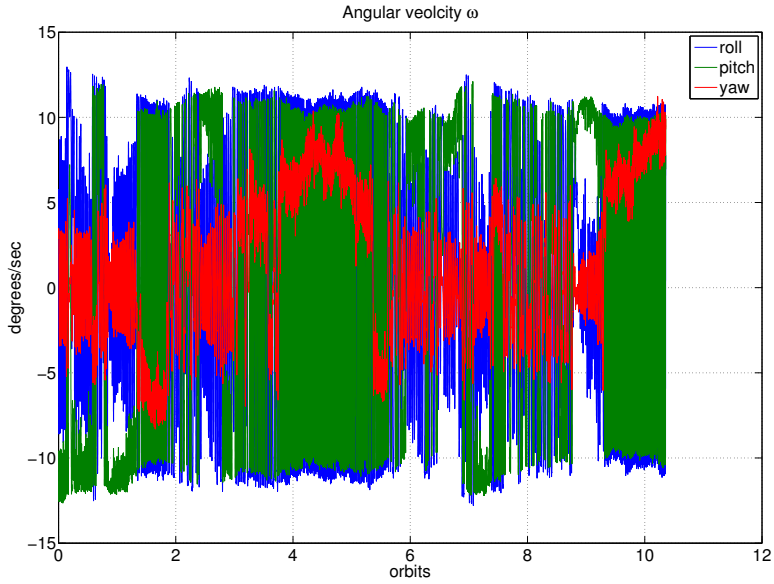

Figure 9.44: Angular velocity - Test case 9

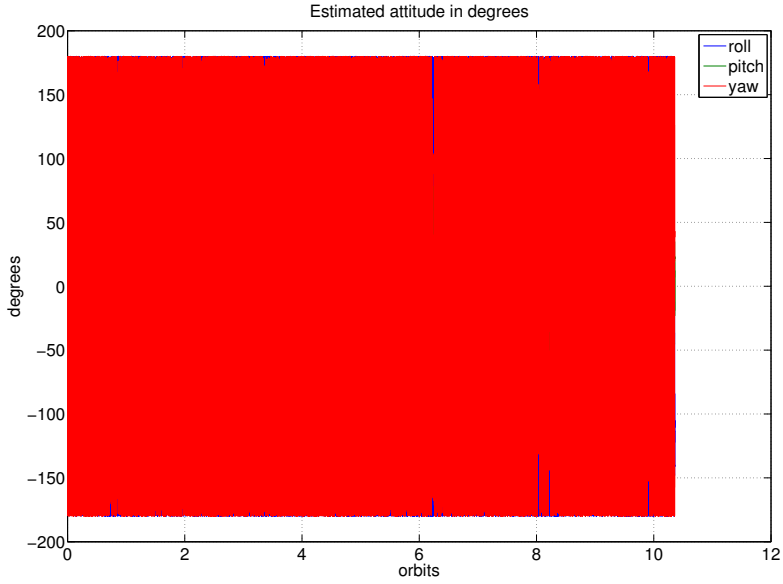

Figure 9.45: Estimated attitude - Test case 9

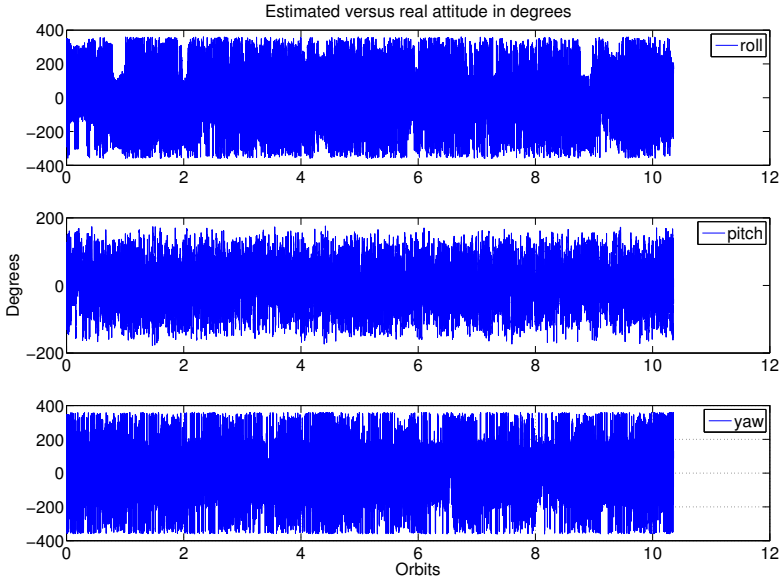

Figure 9.46: Estimated versus real attitude - Test case 9

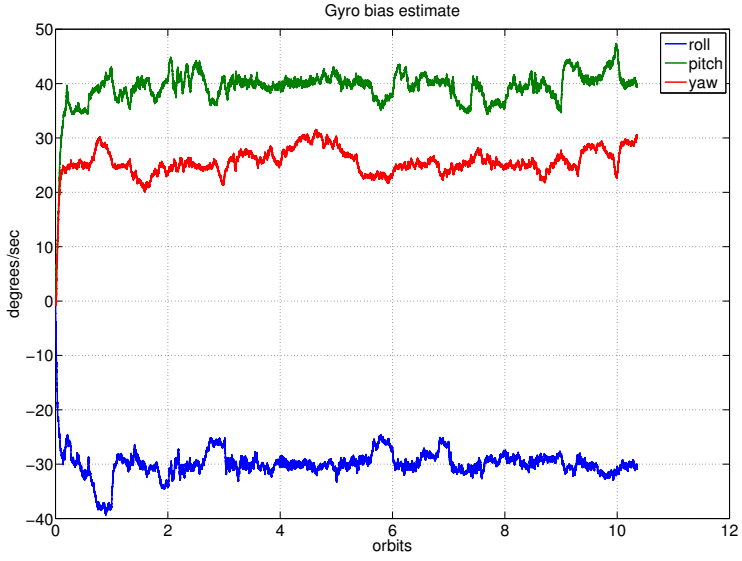

Figure 9.47: Gyroscope bias estimate - Test case 9

Test case 10 - Gyroscope freeze

Table 9.11: Parameters for simulator in test case 10

Parameter	Value		Comment
Sun sensor availability	60 % of orbit		Resembles normal inclination angle
Magnetic reference	Sampled every 100 second		Resembles a LUT size of 10 kilobytes
Disturbances	Gravity	ON	
	Aerodynamic drag	OFF	
	Micro meteorite	OFF	
Single point failures	Gyroscope	Freeze	
	Sun sensor	None	Occurs after 0.5 orbit
	Magnetometer	None	

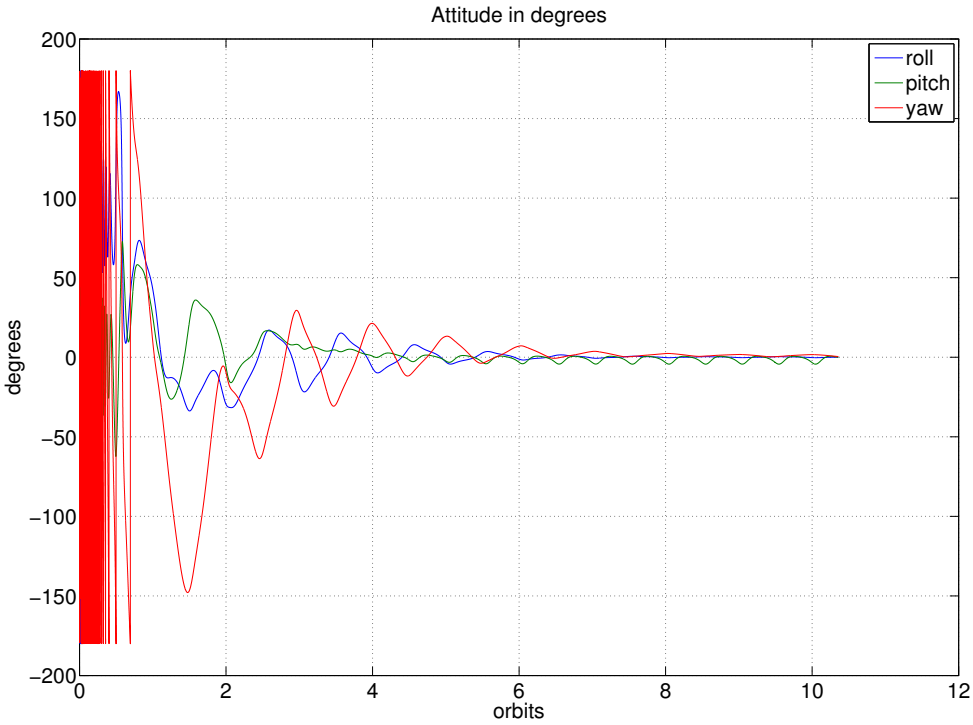

Figure 9.48: Attitude - Test case 10

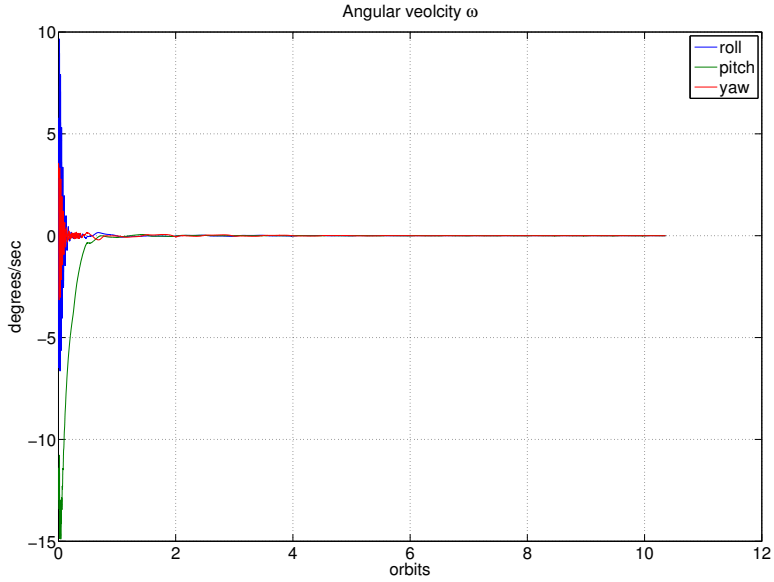

Figure 9.49: Angular velocity - Test case 10

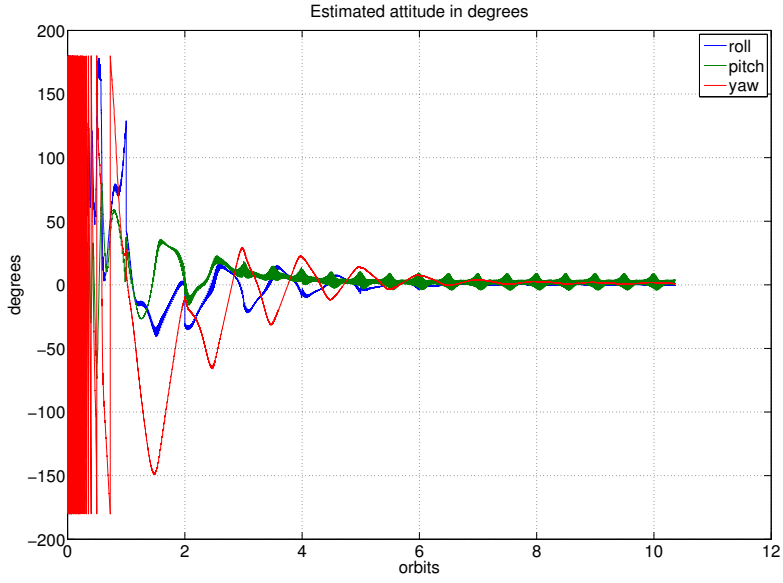

Figure 9.50: Estimated attitude - Test case 10

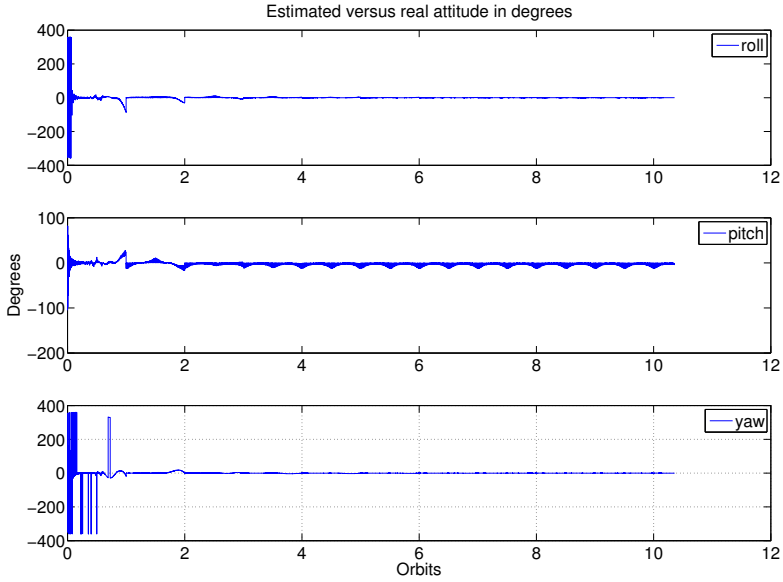

Figure 9.51: Estimated versus real attitude - Test case 10

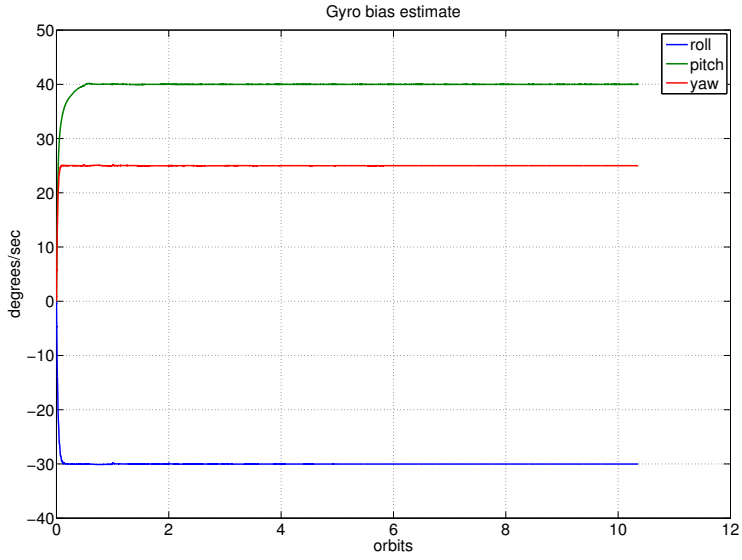

Figure 9.52: Gyroscope bias estimate - Test case 10

9.2 Embedded hardware plattform

The Nonlinear Mahony Attitude Observer from section (8.3) was implemented;

$$\begin{aligned}
\boldsymbol{\omega}^b_{mes} &= \left[\frac{k_1}{2} \left(\frac{\boldsymbol{a}^b_{acc}}{\|\boldsymbol{a}^b_{acc}\|} \right) \times \left(\boldsymbol{R}^b_o(\hat{\boldsymbol{q}}) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) + \cdots \right. \\
&\quad \left. \cdots \frac{k_2}{2} \left(\frac{\boldsymbol{m}^b_{mag}}{\|\boldsymbol{m}^b_{mag}\|} \right) \times \left(\boldsymbol{R}^b_o(\hat{\boldsymbol{q}}) \frac{1}{\sqrt{m_x^2 + m_y^2 + m_z^2}} \begin{bmatrix} m_x \\ m_y \\ m_z \end{bmatrix} \right) \right] \\
\dot{\hat{\boldsymbol{b}}}^b_{gyro} &= -\frac{1}{2} \boldsymbol{K}_i \boldsymbol{\omega}^b_{mes} \\
\dot{\hat{\boldsymbol{q}}} &= \boldsymbol{T}_q(\hat{\boldsymbol{q}}) \left[\boldsymbol{\omega}^b_{imu} - \hat{\boldsymbol{b}}^b_{gyro} + \boldsymbol{K}_p \boldsymbol{\omega}^b_{mes} \right]
\end{aligned}$$

The gains for the observer was also here found in [27]. The magnetometer is by far the most uncertain sensor, and is gained at half compared to the accelerometer. Gyro bias convergence was found to be somewhat slow, but higher gains resulted in instability and reaction to general angular velocity. The injection term was left as pass through ($K_p = 1$), since performance was sufficient.

The different parameters used for all hardware tests are given in table (9.12).

Table 9.12: Common parameters for embedded hardware tests

Parameter	**Value**	**Comment**
Mahony observer gains	$k_1 = 1$	Accelerometer
	$k_2 = 0.55$	Magnetometer
	$K_p = 1$	Injection
	$K_i = 0.008$	Gyro bias
Solver	Forward Euler	Timestep = 0.01
Data sampling/logging	Every 10th iteration of code loop	Serial data over USB
Magnetic reference (for Trondheim)	$m_x = 0.5467\,\mu T$	Aligning X with East
	$m_y = 13.5961\,\mu T$	Aligning Y with North
	$m_z = -49.9338\,\mu T$	Aligning Z with Up
Gyroscope bias	$\boldsymbol{b}^b_{gyro} \approx \begin{bmatrix} 12 \\ 1 \\ 0 \end{bmatrix} {}^\circ/_s$	Measured from gyroscope lying still

To test how fast the Mahony Observer was running on the micro controller, a simple yet effective solution was chosen. One of the GPIO-pins on the AT32 UC3-A3 Xplained kit was set to "toggle" a pin for each time the code-loop was finished. These pins can be seen in the right hand corner of figure (8.4) marked as J3. This output was then connected to a *Saleae Logic* device (figure (9.53)) and logged with their software.

Figure 9.53: Saleae Logic analyzer with probes

Test plan

1. Logged attitude estimate from UC3-A3 microcontroller
2. Logged gyroscope bias estimate on UC3-A3
3. Benchmark of frequency of Mahony Observer on UC3-A3
4. Benchmark of frequency of Mahony Observer on A3BU IMU

Test case 1 - Logged attitude estimate UC3-A3

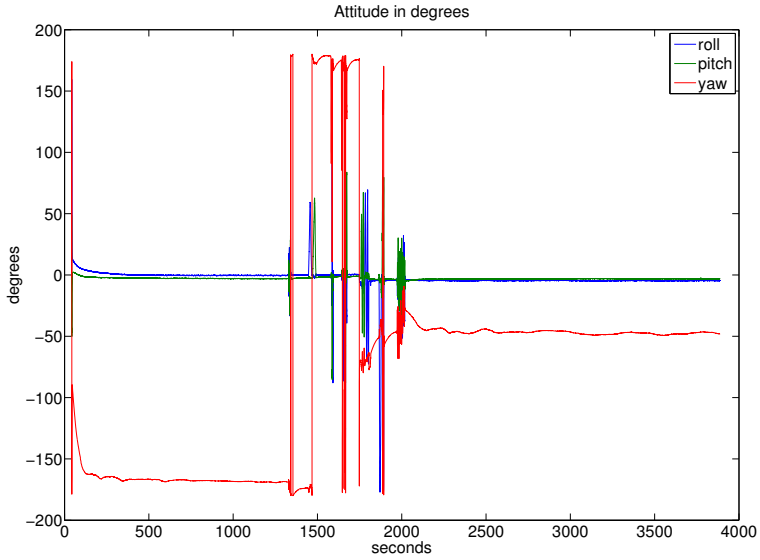

Figure 9.54: Attitude estimate UC3-A3 - Test case 1

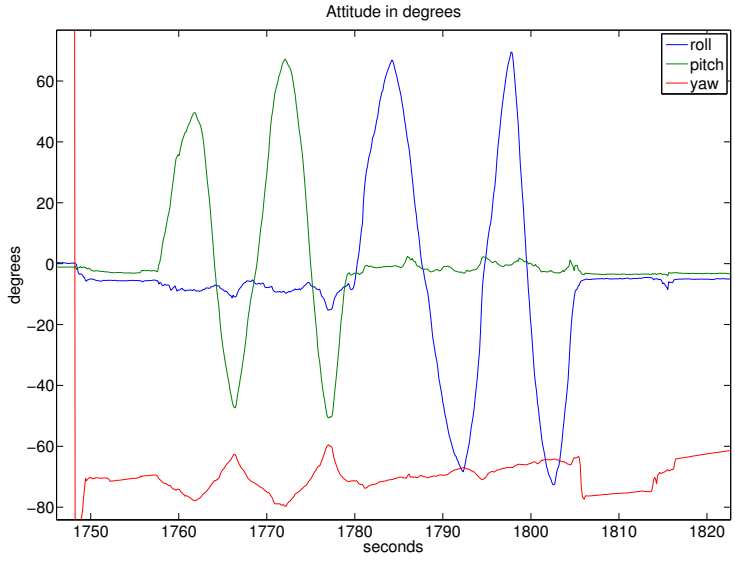

Figure 9.55: Attitude estimate UC3-A3 zoomed - Test case 1

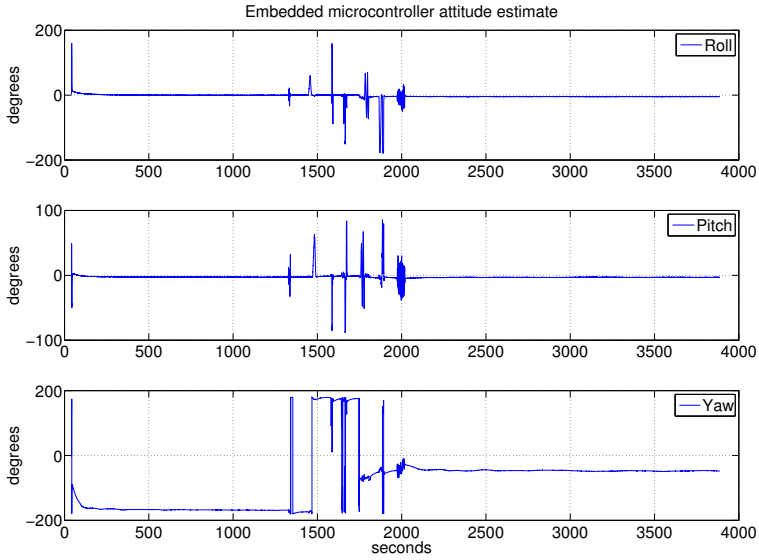

Figure 9.56: Attitude estimate UC3-A3 separate axes - Test case 1

Test case 2 - Gyroscope bias estimate on UC3-A3

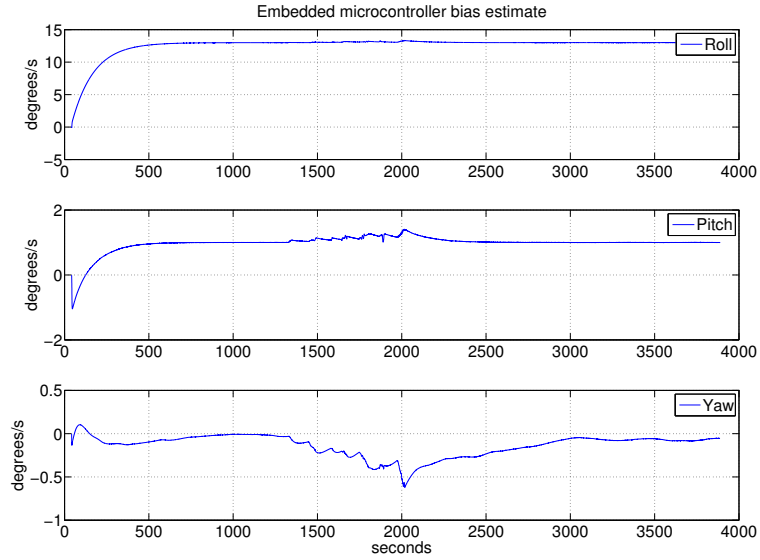

Figure 9.57: Gyro bias estimate UC3-A3 - Test case 1

Test case 3 - Benchmark UC3-A3

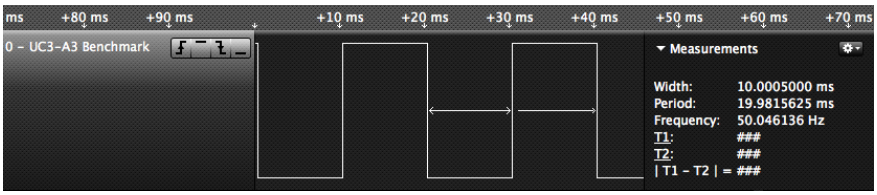

Figure 9.58: UC3-A3 pin toggle from Saleae Logic - Mahony loop

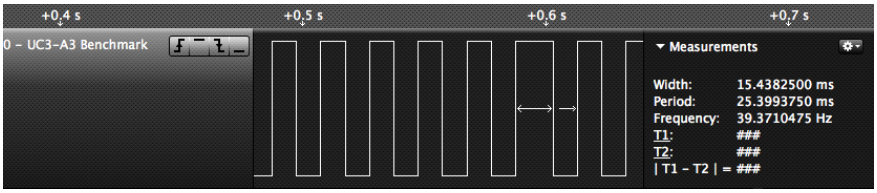

Figure 9.59: UC3-A3 pin toggle from Saleae Logic - Entire loop

Test case 4 - Benchmark A3BU

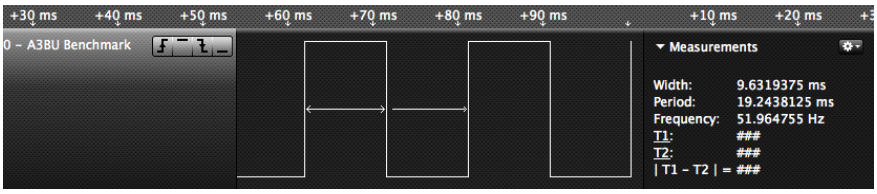

Figure 9.60: A3BU pin toggle from Saleae Logic - Mahony loop

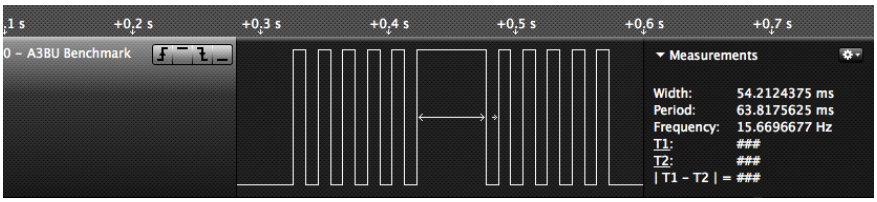

Figure 9.61: A3BU pin toggle from Saleae Logic - Entire loop

Chapter 10

Discussion

10.1 Sumlink simulator environment

Test case 1 - All systems perfect
When all systems a fully operative, the attitude system show excellent performance. The satellite is almost fully damped within one orbit, and a useful attitude is achieved after three, well within the requirement specification.

Both the attitude estimate and gyro bias estimate is nearly correct within one orbit. Note that the attitude estimate plot (9.10) is a little thicker then the real attitude plot (9.1). This is because there is noise in the system.

It is also interesting that the ADCS system consumes very little power. It peaks with a draw of 2 watt and quite rapidly falls down to 0.01 watt. One could argue that this is above the 1.3 watt the solar panels can charge. But the peak draw is only during detumbling. By looking at the battery (that is not re-charged by a solar panel in this model), it is clear that this initial

peak has almost no effect at all. This is also the result from all the other test cases. The plots are therefore left out to save space.

Test case 2 - Sun sensor in Earth's shadow
This test emphasizes the importance of the directional sun sensor. The attitude estimate slowly drifts off when the sensor is lost. Both roll and pitch are effected with some degrees, but yaw is struggling the most. However, all are within the requirement specification.

Test case 3 - Low resolution magnetic reference
This is where the condition for the ADCS system is getting realistic. The sun sensor will probably be shadowed and the magnetic reference will be part of a look-up-table. The 10kB LUT size is well within the storage limit of the microcontroller. Even though the attitude is not affected much compared to test case 2, the attitude estimate performance is decreasing. Having a limited resolution on the magnetic reference is somewhat equal to the shadowing of the sun sensor, only on a smaller scale. Between each 100th second, the attitude estimate drifts off slightly, before correcting in the next sampling. This is what causing the small "waves" in plot (9.11).

Overall the performance is still acceptable and within requirement specification.

Test case 4 - Aerodynamic drag
The ADCS system cannot handle aerodynamic torque in it's current form. The attitude estimate is sometimes up to a 100 degrees off in one axis. The satellite has low angular velocity, so the movement is not as dramatic as it might look. But the attitude is still not within requirement specification.

Because of these results, the aerodynamic torque was turned off for further testing.

Test case 5 - Micro meteorite

The micro meteorite is based solely on an estimated impact. It does however show that the system can in fact handle a sudden detumbling from an unknown cause. The overall attitude is within requirement specification only four orbits after impact.

During the tumbling state the attitude estimate is far off. The spin in the yaw direction is picked up as gyroscope bias in the observer. However the controller does manage to detumble, an as soon as the the angular velocity is calmed, so is the attitude estimate. From these results one can suspect that the dampening in the controller is not as important as initially assumed.

Test case 6 - Gyroscope loss
Without the gyroscope, the controller essentially becomes a P-controller with no damping. Amazingly the overall attitude is by far the best of all the test cases. The attitude estimate is however not that correct. The gyro bias estimate uses longer time to converge then usual, before it rests on the correct zero value. The attitude itself also has a greater induced wave motion in pitch. Compared to the real attitude, it seems to have lower amplitude, especially in the first two orbits. This attitude is the input on the controller, and that will in turn make the controller less agile. This could suggest the controller is not optimally tuned.

Removing the gyroscope also removes a lot of measurement noise from the system. The noise from from the gyroscope affects the entire feedback in the observer, resulting in less disturbance on all available signals.

Even though these results are within the requirement specification, they should be compared with test case 7, and the discussion given there.

Test case 7 - Gyroscope loss with measurement noise
A non functional gyroscope can be a non existing one, like in test case 6, or a sensor that outputs zero when measured. The latter is the basis for this

test case. Now the ADCS performance is back to test case 3 levels.

The gyroscope noise is from the data sheet provided by the manufacturer. It is implemented without any filtering. One explanation to the results might be that the noise levels are too high. In for example test case 3, when the satellite is close to attitude reference, the gyroscope won't contribute much. Since the angular velocity is so low, and might "drown" in measurement noise.

It is surprising that the observer works so well without the gyroscope. But doing attitude estimation with only directional vectors has been proven in both the TRIAD algorithm and QUEST.

In this simulation, all sensors are sampled equally fast. Providing no advantage to a usually fast gyroscope. All degrees of freedom are also connected to all sensors. In some cases it might be useful to only let the magnetometer affect yaw, since this is usually more inaccurate and sampled slower then the other sensors. Gaining more accurate roll and pitch estimates, with a gyroscope (and accelerometer) superior to the magnetometer.

The results from this test case might exploit some inaccuracy in the simulator regarding the gyroscope, and the fact that the algorithm is not utilizing the full strong points of the separate sensors. But it is again within requirement specification.

Test case 8 - Sun sensor loss
Without the sun sensor the ADCS system is struggles heavily. Compared with the angular velocity, the attitude is slowly drifting back and forth. The satellite even rotates around the roll axis. This test shows that even though the sun sensor is normally lost in 40 % of orbit, a zeroed signal will cause the attitude estimate to completely drift away. Interestingly the gyroscope bias is almost perfect, but it is not aiding the system much, since the angular velocity is close to zero after one orbit.

Test case 9 - Magnetometer loss
Loosing the magnetometer is an absolute disaster. This is because the output from the magnetometer goes directly to the controller. With only noise as input (can't be zero because of singularities in the scaling function), the controller won't be able to calculate a desired torque. It is known from test case 5 that the observer is not performing too well with very high angular velocities. Especially while simultaneously loosing a directional reference vector. Even though the gyroscope bias is somewhat close, it is oscillating with up to $10°/s$ error. If the bias hadn't initially beed so high, it would have looked a lot worse.

One can clearly see that the magnetometer is the most important sensor in the system, and without it the ADCS goes down.

Test case 10 - Gyroscope freeze
The gyroscope freeze scenario resembles test case 6. Only this time the output freezes on constant value. Consequentially the bias estimate converges to a constant value, rather then zero as i test 6. Otherwise the test cases resembles each other, and the result is within requirement specification.

10.2 Embedded hardware plattform

Test case 1 - Logged attitude estimate UC3-A3
The MCU was laying flat on a surface during the first part of the the data logging. After the bias estimate seemed stable, several movements was carried out. In figure (9.55) a small portion is highlighted. Here the MCU was first pitched, then rolled. After these movements, the MCU was laying still on the surface. Laying still is also the only verifiable reference to the attitude. The periods of no movement can clearly be observed in figure (9.54) and (9.56).

It is important to emphasize that real attitude was not logged. Neither was the sensor data ran through a separate algorithm. The data was only verified by looking at the data logging, while moving the MCU.

Despite this, the attitude estimate appears proper. It might be up to 5° degrees off in the roll and pitch axis. The yaw estimate is supposed to point magnetic north, but compared to where the test was carried out, this can't be verified. After the movement phase (from $\approx$ 1300 seconds to $\approx$ 2300 seconds) the MCU was rotated in the yaw axis to avoid the $\pm 180°$ degrees skipping.

There is some noise in the signals, but not of any significance. The "saw tooth" pattern in figure (9.55) can also have been caused by the small vibrations from the hand holding the MCU. Also, the data logging is only done for each 10th loop of the Mahony Observer. So the attitude estimate could have been even smoother.

Test case 2 - Gyroscope bias estimate on UC3-A3
The gyro bias estimate is somewhat easier to verify. Compared the readout from the non moving gyroscope, the bias estimate converges around these correct values, as seen in figure (9.57). Both roll, pitch and yaw converges around 500 seconds.

During the movement phase, the bias estimates slightly drifts off. The different axes are plotted at different scales, but overall it is closely the same. This drift did not seem to affect the attitude estimate much.

Test case 3 - Benchmark UC3-A3
Figure (9.58) shows the pulses received by the Saleae Logic. Each vertical line represents a full iteration of the Mahony Observer. The width of each pulse is $\approx$ 10 milliseconds. This means the program runs at 100 Hz[1].

[1]The frequencies listed in figure (9.58) to (9.61) is measured between wave tops and should be doubled for this type of signal.

In figure (9.59) the pulses are zoomed out. Here one can observe that the program does 10 iterations before sending the attitude estimate over USB (the widest pulse). It is actually more "expensive" for the microcontroller to send data over USB, then one iteration of the Mahony Observer, at a ratio of about 150 %. But keep in mind that the microcontroller converts the quaternions to degrees for each transmission over USB. This could be those extra 50%.

Test case 4 - Benchmark A3BU

Benchmarking the portable IMU from section (8.5) was done as a simple verification. But it does enlighten some important properties. The A3BU used in the portable IMU is supposed to be slower then the UC3-A3 used in the data logging. However, the A3BU is capable of iterating the Mahony Observer at a slightly higher frequency (≈103 Hz). The reason might be that there are other bottlenecks in the system then the MCU itself. This could be the sensors, data busses, memory or others.

The portable IMU displays the attitude on screen for every 10th iteration of the Mahony Observer. The time it takes to output to the screen can clearly be seen from the wide pulse in figure (9.61). Compared to sending data over USB, this is even more expensive, at a ratio of over 550 % compared to one iteration of the Mahony Observer.

Chapter 11

Conclusion

Designing a complete ADCS system for the NUTS satellite requires knowledge of mathematical modeling, estimation algorithms, actuator design, control theory, Matlab programming and embedded programming. Considering the challenges met in the NUTS requirement specification, and the limitations of a space environment, this thesis shows it is still possible to create a satisfying ADCS system.

Using standard theory for satellite attitude, an effective PD-controller and relatively new estimation theory, the ADCS system shows good results.

Even with a shadowed sun sensor and limited magnetic reference the attitude is within limits of the requirement specification. The simulation also shows that with aerodynamic disturbance, the system does not meet expected performance. This might be the result of a non optimized controller, or a too harsh resulting torque. Further testing of the NUTS aerodynamic properties is advised.

Compared to a torque disturbance over a limited amount of time, like a micro meteorite, the ADCS performes quite well. This is however solely based on

an estimated induced torque, and should also be further researched. Even though the importance can be discussed.

The three sensors used to estimate attitude all have their importance. Most interesting is how well the system performs without a gyroscope (in the simulator). Attitude estimation without a gyroscope is nothing new, but it is odd to see how little it contributes. The noise from the gyroscope used in simulations is believed to be the main cause of the irregular results.

The simulations does show that the sun sensor is of critical importance to the ADCS system. Also, the magnetometer is absolutely vital. This sensor is connected to both the attitude estimation and the controller. Without it the NUTS satellite has no ability to even set up a torque.

The main mission of the satellite is to photograph the Earth's atmosphere. The attitude of the satellite is sufficient for such photography. But the camera system needs the attitude data for further calculations on ground. For the most realistic test cases, the attitude estimate should be good enough, but this is yet to be determined.

Testing the attitude estimation algorithm on the NUTS embedded computer gave positive results. Even though the environment on Earth is more forgiving then space, the implementation is comparable. First of all the algorithm is not too intensive for the microcontroller. Also, using two directional sensors and a rate gyroscope, the attitude can be estimated within requirement specifications.

Chapter 12

Further work

- Implemenation of reliable clock signal for orbit propagator
- IGRF look-up-table in connection with clock signal
- Redunancy and fault tolerance

  Projection algorithm (Grip observer)

  Bias filtering or latch between actuators and magnetometer
- Sensor dynamics and specifiactions

  Availability and accuracy of sun sensor

  Realistic noise, bias and sampling frequencies gathered from an experimental setup
- Supply the future ADCS developer with a working prototype board (same type that fits in the satellite).
- Include the control law in the embedded microcontroller

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 Lindgard 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 Kolbjørn Soglo, *3-aksestyring av gravitasjonsstabilisert satellitt 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] Bjørnar Vik, *Integrated Satellite and Inertial Navigation Systems*, Department of Engineering Cybernetics, NTNU, 2012.

[14] Toril Bye Rinnan, *Development and Comparison of Estimation Methods for Attitude Determination*, Department of Engineering Cybernetics, NTNU, 2012.

[15] Kasper Vinther, Kasper Fuglsang Jensen, Jesper Abildgaard Larsen, Rafal Wisniewski, *Inexpensive CubeSat Attitude Estimation Using Quaternions and Unscented Kalman Filtering*, Aalborg University, 2011.

[16] Robert Mahony, Tarek Hamel, Jean-Michel Pflimlin, *Nonlinear Complementary Filters on the Special Orthogonal Group*, IEEE Transactions on Automatic Control VOL. 53, 2008.

[17] Grace Wahba, *A Least Squares Estimate of Satellite Attitude*, SIAM Review, Vol.7, p.409, 1965.

[18] M.D. Shuster, S.D. Oh *Three-axis attitude determination from vector observations*, AIAA, Journal of Guidance and Control, 1981.

[19] Harold D. Black *A Passive System for Determining the Attitude of a Satellite*,

AIAA Journal, 1964.

[20] F. Landis Markley and Daniele Mortari *How to estimate attitude from vector observations*,
Advances in the Astronautical Sciences, Vol. 103, 1999.

[21] Mark L. Psiaki *Attitude-Determination Filtering via Extended Quaternion Estimation*,
Journal of Guidance, Control, and Dynamics, Vol. 23, No. 2, 2000.

[22] Håvard Fjær Grip, Thor I. Fossen, Tor A. Johansen, Ali Saberi *Attitude Estimation Using Biased Gyro and Vector Measurments With Time-Varying Reference Vectors*,
IEEE Transactions on Automatic Control, Vol. 57, No. 5, 2012.

[23] João Luís Marins, Xiaoping Yun, Eric R. Bachmann, Robert B. McGhee, Michael J. Zyda *An Extended Kalman Filter for Quaternion-Based Orientation Estimation Using MARG Sensors*,
International Conference on Intelligent Robots and Systems, 2001.

[24] Tale Sundlisaeter, *Spacecraft Attitude and Orbit Estimation using GPS and Inertial Measurements*,
Department of Engineering Cybernetics, NTNU, 2012.

[25] Rudolph E Kalman, *A new approach to linear filtering and prediction problems*,
ASME

Transactions, Volume 82, Part D (Journal of Basic Engineering, 1960.

[26] Osamu Mori et al., *First Solar Power Sail Demonstration by IKAROS*,
JAXA Space Exploration Center, Sagamihara, Japan, 2009.

[27] Håvard Fjær Grip, Thor I. Fossen, Tor A. Johansen, Ali Saberi *Nonlinear Observer for Aided Inertial Navigation: Theory and Experiments*,
Control Engineering Practise, 2013 (to appear).

[28] Håvard Fjær Grip, Thor I. Fossen, Tor A. Johansen, Ali Saberi *Globally Exponentially Stable Attitude and Gyro Bias Estimation with Application to GNSS/INS Integration*,
Submitted to Automatica, 2013 (to appear).

[29] Thor I. Fossen *Low-Cost Integrated Navigation Systems for Autonomous Underwater Vehicles*,
Plenary paper presented at the XXXIII Jornadas de Automatica, 2012.

[30] J. Thienel and R. M. Sanner *A Coupled Nonlinear Spacecraft Attitude Controller and Observer With an Unknown Constant Gyro Bias and Gyro Noise*,
IEEE Transactions on Automatic Control, Vol. 48, No. 11, 2003.

[31] S. Salcudean*A Globally Convergent Angular Velocity Observer for Rigid Body Motion*,

IEEE Transactions on Automatic Control, Vol. 36, No. 12, 1991.

[32] Gaute Bråthen*Design of Attitude Control System of a Double CubeSat*,
Department of Engineering Cybernetics, NTNU, 2013.

[33] Håvard Fjær Grip, Thor I. Fossen, Tor A. Johansen, Ali Saberi *Nonlinear Observer for Aided Inertial Navigation: Theroy and Experiments*,
Control Engineering Practice, 2013 (submitted).

Prepared in LaTeX by Fredrik Alvenes