Master's Thesis

Attitude determination for the Norwegian student satellite nCube

by

Stian Søndersrød Ose

Department of Engineering Cybernetics,
Norwegian University of Science and Technology
Trondheim Norway

June 07, 2004

Preface

During the last 5 months I have worked on this project concerning attitude determination for the nCube satellite. Previously, much work has been done on the subject of attitude determination and control system. Many students have worked on different parts of this system separately and a solid theoretical foundation is available (Fauske 2002), (Kristiansen 2000), (Kyrkjebo 2000) and (Svartveit 2003). A fully integration of the different parts of the system are presented in this report.

A mathematical approach has been done using matlab and simulink to simulate the satellite and it's environment, after the initial modeling of the respective systems. The work has also consisted of implementing the estimator on a microcontroller. Many hours has been spent refreshing my skills in the programming language C and it's features.

I would like to thank my advisor Tommy Gravdahl for valuable feedback during this last semester. Also, the rest of the nCube project team has given me inspiration along the way. And finally, I want to thank Dr Ing Bjørnar Vik who has much time explaining the different aspects of the Kalman Filter.

Stian Søndersrød Ose 04.06.2004, NTNU, Trondheim.

Abstract

The nCube project is a satellite designed, built, integrated and tested by students. It aims to increase the interested and expertise the field of space technology. The mission of the satellite is to receive messages from ships and animals, and transmit these to a ground station.

To achieve better communication capabilities, the satellite needs an attitude determination and control system. The control system has previously been designed. It needs both the angular velocity and the orientation of the satellite to perform the necessary control. This report presents the attitude determination system for the satellite. A Kalman Filter has been implemented to produce the estimated states needed by the controller. It is shown that the attitude of the satellite can be found by using light sensors and a three-axes magnetometer in combination with reference models. By using the estimated states, the satellite can be controlled with the previously designed controller.

The estimator has been simplified in the interest of implementing it on board a real satellite. The microcontroller used, has limited speed, and thus a simple estimator should be developed. The proposed simplifications have proven not to fulfill the requirements of accuracy, and at the same time be simple enough to use on board the satellite.

A scheme of how to implement the estimator on a microcontroller for use on board the satellite is given. It is shown how this can be done using a low-power microcontroller. The interaction with sensors and other subsystems on board the satellite is also implemented.

Contents

1 Introduction **1**
- 1.1 Background . . . 1
  - 1.1.1 Cubesat . . . 1
  - 1.1.2 The nCube . . . 1
  - 1.1.3 Previous work . . . 2
- 1.2 This report . . . 2
- 1.3 Outline of the report . . . 3
  - 1.3.1 Chapter 2 . . . 3
  - 1.3.2 Chapter 3 . . . 3
  - 1.3.3 Chapter 4 . . . 3
  - 1.3.4 Chapter 5 . . . 3
  - 1.3.5 Chapter 6 . . . 3
  - 1.3.6 Chapter 7 . . . 3
  - 1.3.7 Chapter 8 . . . 4
  - 1.3.8 Chapter 9 . . . 4

2 Definitions and notation **5**
- 2.1 Reference frames . . . 5
  - 2.1.1 Earth-Centered Inertial (ECI) frame . . . 5
  - 2.1.2 Earth-Centered Earth Fixed (ECEF) frame . . . 5
  - 2.1.3 Earth-Centered Orbit frame . . . 6
  - 2.1.4 Orbit frame . . . 6
  - 2.1.5 Body frame . . . 6
- 2.2 Attitude Representation . . . 7
  - 2.2.1 Euler Angles . . . 7
  - 2.2.2 Unit quaternions . . . 7
- 2.3 Definitions . . . 7
- 2.4 Rotation matrix . . . 8
  - 2.4.1 Derivative . . . 9
- 2.5 Transformation between different frames . . . 9
  - 2.5.1 Transformation from Earth-Centered Orbit to ECI and ECEF frames . . . 9
  - 2.5.2 Transformation from ECEF to ECI frame . . . 9
  - 2.5.3 Transformation from ECI to Orbit frame . . . 10
  - 2.5.4 Transformation from Orbit to Body frame . . . 10

3 Reference Models **12**
3.1 Magnetic . . . 12
3.1.1 Orbit estimator . . . 12
3.1.2 IGRF . . . 16
3.2 Light . . . 17
3.2.1 Sun model . . . 18
3.2.2 Earth Albedo Model . . . 19

4 Mathematical modeling **20**
4.1 Satellite model . . . 20
4.1.1 Kinematics . . . 20
4.1.2 Dynamics . . . 20
4.2 Linearized model . . . 21
4.2.1 Linearized kinematics . . . 21
4.2.2 Linearized dynamics . . . 22

5 Hardware **24**
5.1 Sensors . . . 24
5.1.1 Light Dependent Resistors . . . 24
5.1.2 Magnetometer . . . 25
5.2 Communication . . . 26
5.2.1 Interface to Light Dependent Resistors . . . 26
5.2.2 Interface to magnetometer . . . 27
5.2.3 Interface to other subsystems . . . 27
5.3 Covariance of the sensors . . . 31
5.3.1 Light Dependent Resistors . . . 31
5.3.2 Magnetometer . . . 32
5.4 Microcontroller . . . 32

6 Kalman Filter **34**
6.1 Background . . . 34
6.1.1 Filter theory . . . 34
6.1.2 Incorporating the sensors . . . 36
6.2 Extended Kalman Filter . . . 36
6.3 Linearized Kalman Filter . . . 37
6.4 Discrete Kalman Filter . . . 39
6.4.1 Prediction . . . 39
6.4.2 Update . . . 40
6.4.3 Offline calculations . . . 41

7 Simulation **43**
7.1 Extended Kalman Filter . . . 44
7.2 Linearized Kalman Filter . . . 46
7.3 Discrete Kalman Filter . . . 49

8 Implementation **54**
8.1 Introduction . . . 54
8.2 Top view . . . 54
8.3 The Wait process . . . 58
8.4 The Initiate system process . . . 58
8.5 The Estimate state process . . . 59
8.5.1 Propagate states (3.1) . . . 59
8.5.2 Get measurements (3.2) . . . 61
8.5.3 Calculate error in measurements (3.3) . . . 61
8.5.4 Innovate processes (3.4) . . . 62
8.5.5 Perform measurement update (3.5) . . . 62
8.5.6 Estimate full state (3.6) . . . 63
8.5.7 Transmit estimate (3.7) . . . 63
8.6 The Communicate process . . . 64
8.6.1 Transmit data (4.1) . . . 64
8.6.2 Receive data (4.2) . . . 64
8.6.3 Interpret data (4.3) . . . 64
8.7 Post launch initialization . . . 67

9 Conclusion **69**
9.1 Conclusion . . . 69
9.2 Recommendations . . . 69

A Simulink Diagram **72**
A.1 Extended Kalman Filter . . . 72
A.2 Linearized Kalman Filter . . . 75
A.3 Discrete Kalman Filter . . . 78

B Parameters & CD contents **81**
B.1 Parameters for linearized model . . . 81
B.2 CD Contents . . . 82

Chapter 1

Introduction

The Cubesat concept, which the nCube satellite is based on, is briefly introduced. The work previously done on the nCube project is presented. In addition, the work done on this Master's thesis is introduced. Finally, the outline if this report is described.

1.1 Background

1.1.1 Cubesat

Professor Bob Twiggs at Stanford University developed the Cubesat concept in order to enable his students to complete a satellite project during their education, preferably in just a year. The idea was to have a small, standardized satellite as a framework, and let the students modify it. The size, a $10 \times 10 \times 10$ cm cube, and weight under one kilogram, make for small launch costs. A pod, able to launch multiple Cubesat satellites, was also designed reducing the costs even more. Both the satellite and the launcher are shown in Figure 1.1.

1.1.2 The nCube

Andøya Rocketrange, ARS, and the Norwegian Space Center, NRS, have taken the initiative to start a Norwegian student satellite program. Their goal is to increase the competence in

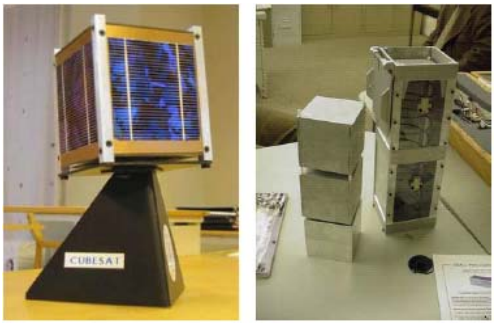

Figure 1.1: The Cubesat satellite (left) and the Cubesat-launcher (right)

space related activities. Students, and with that their respective teaching institutions, will get hands-on experience with satellite related technology and hopefully develop a higher interest in the field of space technology. The goal will be obtained through designing, building, testing and launching of a small satellite. The satellite will be based on the work done at universities and university-colleges. Depending on the success of the mission, future satellite projects may be considered. It is already planned for a second student satellite with launch 6 months after nCube.

Students at the Norwegian University of Science and Technology, NTNU, the University College of Narvik, HiN, and the Agricultural University of Norway, NLH, have already performed pre-studies. Different aspects of the satellite have been studied at the different institutions, and choices have been made which have led the way for later work.

The project has been divided into several subgroups. NTNU will study the areas of Structure, Power, Attitude Determination and Control Systems, Communications and downlink, On Board Data Handling, and technical framework for payload. HiN will study Power and Ground Segment/Uplink. UiO will perform testing and study power supply with emphasis on the solar panels, and NLH will study payload and applications.

The nCube satellite is based on the Cubesat concept. The payload is an Automatic Identification System, AIS. AIS is a mandatory system on all larger ships that transmits identification and position data messages. The satellite will redirect these messages along with messages from reindeer collars produced by NLH students. The collected data will with each pass over Norway be transmitted to a ground station.

Launch is scheduled in the fall of 2004, along with 14 other student satellites. NCube will share launch pod with Taiwan and South Korea. In addition many other projects are under development, and the potential of international exchange of information is large.

1.1.3 Previous work

This report is mostly based on the work of (Svartveit 2003). Here a magnetometer based attitude determination approach was introduced. Along with other students at the Department of Engineering Cybernetics, NTNU, he worked on the attitude determination and control system (ADCS). Control strategies have been developed and implemented but only based on idealized states. The robustness of the controllers has been tested, but only to a certain degree. Some of the work done in (Kristiansen 2000) and (Kyrkjebo 2000) has also been used in this report.

1.2 This report

In order to get high bandwidth on the downlink from the satellite to the ground station, the antenna has to point towards the Earth. The antenna needs accuracy between 10 and 20 degrees off the direct line to the ground station to achieve better communication bandwidth. This motivates the use of attitude control to get the necessary accuracy. However, the attitude control system needs information about the attitude in order to perform. This attitude is not directly measurable.

An estimation of attitude of the satellite is thus necessary. Here, a Kalman Filter is implemented to do this. It uses measurements of the magnetic field around the Earth and light received from the sun. Because of the limited computation powers, a simplified model of the

filter is developed. This simplified model is compared to the more accurate Extended Kalman Filter to get an idea of the performance. It is also integrated with the previously designed attitude controller in order to study the effect of using only estimated states in the controller instead of the measured states.

Together with the Attitude Control System, the filter will make up the Attitude Determination and Control System printed circuit board. As the there is a microcontroller able to run on low voltage with acceptable speed is necessary.

1.3 Outline of the report

1.3.1 Chapter 2

The mathematical model of the satellite and it's environment can be developed in a number of different reference frames. A short introduction to these and to the mathematical tools used throughout this report, is given.

1.3.2 Chapter 3

For determining the attitude of the satellite, two different sensors are used. Each needs a reference model to compare the measured data with. These models are presented here.

1.3.3 Chapter 4

In order to be able to simulate the satellite in it's environment, a mathematical model of both the dynamics of the satellite and it's kinematics is derived here. A linearization is also done for use in the Kalman Filter.

1.3.4 Chapter 5

The attitude estimator is implemented in a real satellite. Some specifications were outlined early in the process of designing the satellite and the mission for the Attitude Determination and Control System. Among these were what should be measured and how. The hardware components of the Attitude Determination subsystem are described here.

1.3.5 Chapter 6

A short introduction to Kalman Filter in general is given. The Kalman Filter can take many forms, and each one used in this work is described. Since the Attitude Determination System in our case has very limited computation abilities, a simplified implementation, that will be less demanding for the microcontroller to run, is preferred. Simplifications and how they influence the overall performance is analyzed. The filter is also presented in discrete time.

1.3.6 Chapter 7

The estimator was first implemented as an Extended Kalman Filter in continuous-time. It was assumed that this would give the best performance, and thus be the reference for other, less accurate approaches. The behavior of the satellite is dependent on the initial condition after

detumbling phase. As this unknown, a predefined trajectory would almost certain be incorrect. While the Linearized Kalman Filter can estimate small perturbations from the trajectory, the Extended Kalman Filter would show a better performance.

1.3.7 Chapter 8

The Discrete Kalman Filter is now implemented on a microcontroller. Processes for interaction with the other subsystems and the sensors are also implemented. As the estimator uses multi-dimensional mathematical operations, a library was made that performs these operations. This is used by the estimator when needed.

1.3.8 Chapter 9

The results and conclusion of the work presented in the previous chapters are described here. Some guidelines for possible future work are also presented.

Chapter 2

Definitions and notation

The mathematical model of the satellite and it's environment can be developed in a number of different reference frames. A short introduction to these and to the mathematical tools used throughout this report, is given.

2.1 Reference frames

The different frames used in this report are defined. The context, in which the different frames are used, is also explained. In which frame the vector is given in, is denoted v^F, where F is frame depended.

2.1.1 Earth-Centered Inertial (ECI) frame

The ECI frame is an inertial frame used for navigation. It is a non-accelerated reference frame in which Newton's laws is valid. This frame is fixed in space. The origin of the frame is located at the center of the earth with the z-axis pointing towards the North Pole. The x-axis points towards vernal equinox, the point where the plane of the Earth's orbit about the Sun, crosses the Equator going from south to north, and the y-axis completes the right hand Cartesian coordinate system. All the different motions of the satellite can be presented in this frame, but only the velocity of the Orbit frame and the motion of the sun is directly compared to this frame. The frame is denoted I.

2.1.2 Earth-Centered Earth Fixed (ECEF) frame

This frame also has its origin located in the center of the Earth but the x- and y-axes rotate with the Earth relative to the ECI frame. The rotation is about the z-axis, both of the ECI and the ECEF frame, and has a rate of $\omega_e = 7.2921 * 10^{-5}$ rad/s. The x-axis points toward the intersection between the Greenwich meridian and the Equator, which is at 0^o longitude and 0^o latitude, and the y-axis completes the right handed orthogonal system. By introducing this frame, the magnetic field around the Earth, IGRF, can be used along with an orbit estimator to create a reference model. The frame is denoted E.

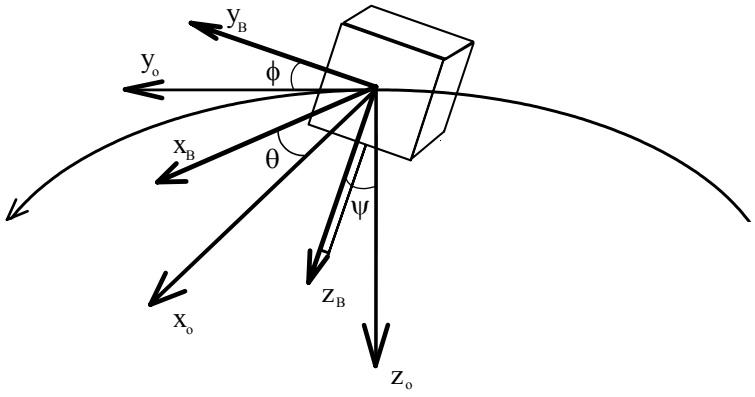

Figure 2.1: The Body and Orbit reference frames, with the angles describing the orientation.

2.1.3 Earth-Centered Orbit frame

This is the frame in which the Keplerian elements are given. The orbit estimator calculates the Keplerian elements of the satellite to use with the model of the magnetic field. The center of the frame coincides with that of the Earth, with x-axis pointing towards perigee, y-axis along the semiminor-axis and z-axis perpendicular to the orbit plane. The frame is denoted OC.

2.1.4 Orbit frame

The origin of this frame coincides with the center of mass of the satellite. It rotates relative to the ECI frame, with a rate of ω_o depending on the altitude of the orbit. The y-axis points in the direction of motion tangentially to the orbit. However, the tangent is only perpendicular to the radius vector in the case of a circular orbit. In elliptic orbits, the y-axis does not align with the velocity vector of the satellite. The z-axis of the frame points towards the center of the Earth, and the x-axis completes the right hand system, see Figure 2.1. The Orbit frame is denoted O.

2.1.5 Body frame

This frame is fixed with the satellite, and its origin is placed in the center mass of the satellite, simplifying the kinematic equations. The orientation of the satellite are described relative to the Orbit frame, while angular velocities are expressed in the Body frame. The nadir side of the satellite is in the z-axis direction, and the last two axes, x-axis and y-axis, coincides with x-axis and y-axis of the Orbit frame when the satellite has an attitude of 0^o in roll, pitch and yaw. It is these angles that describe the attitude of the satellite, see Figure 2.1. This frame is denoted B.

2.2 Attitude Representation

There are a number of ways to represent the attitude of the satellite in a reference frame. These, along with tools to convert between the frames, are described here.

2.2.1 Euler Angles

The attitude can be represented using roll, pitch and yaw angles. In the case of a satellite, these angles describe the attitude of the satellite relative to the Orbit frame. Each of the angles is a rotation about a given axis. The roll angle is a rotation θ about the x_O-axis, the pitch angle a rotation ϕ about the y_O-axis and finally, the yaw angle is a rotation ψ about the z_O-axis. These angles are only used as input and output from the filter. This to best visualize to the behavior of the satellite and performance of the filter. The main problem with Euler angles is the existence of singularities. Introducing a fourth parameter to represent the attitude can solve this.

2.2.2 Unit quaternions

To overcome the problem with singularities in the attitude representation, quaternions are used in computations. A quaternion is defined to have one real part, η, and three imaginary parts, $\boldsymbol{\epsilon}$, defined by

$$\eta = \cos\frac{\beta}{2}, \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 & \epsilon_2 & \epsilon_3 \end{bmatrix}^T = \boldsymbol{\lambda}\sin\frac{\beta}{2} \tag{2.1}$$

where β is the rotation about the unit vector $\boldsymbol{\lambda}$. The unit quaternions satisfy the constraint $\mathbf{q}^T\mathbf{q} = \mathbf{1}$, or

$$\eta^2 + \epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 = 1. \tag{2.2}$$

2.3 Definitions

Here some of the definitions that are not straightforward are explained.

Cross product operator

The vector cross product, $\times$, is defined by:

$$\boldsymbol{\lambda} \times \mathbf{a} := S(\boldsymbol{\lambda})\mathbf{a} \tag{2.3}$$

where $\boldsymbol{\lambda}$ and $\mathbf{a}$ are two arbitrary 3×1 vectors, and the skew-symmetric matrix $S(\boldsymbol{\lambda})$ is defined as

$$S(\boldsymbol{\lambda}) = -S(\boldsymbol{\lambda})^T := \begin{bmatrix} 0 & -\lambda_3 & \lambda_2 \\ \lambda_3 & 0 & -\lambda_1 \\ -\lambda_2 & \lambda_1 & 0 \end{bmatrix} \tag{2.4}$$

Quaternion product

The quaternion product, $\otimes$, is defined as

$$\mathbf{p}_1 \otimes \mathbf{p}_2 := \begin{bmatrix} \eta_1\eta_2 - \epsilon_1^T\epsilon_2 \\ \eta_1\epsilon_2 + \eta_2\epsilon_1 + S(\epsilon_1)\epsilon_2 \end{bmatrix} \tag{2.5}$$

where $\mathbf{p_1}$and $\mathbf{p}_2$ are quaternions, $\mathbf{p}_i = \begin{bmatrix} \eta_i & \epsilon_{1i} & \epsilon_{2i} & \epsilon_{3i} \end{bmatrix}^T$. This is a rotation $\mathbf{p}_1$ about a vector $\mathbf{p}_2$. The product is used in the Kalman Filter update algorithm.

2.4 Rotation matrix

The rotation matrix can be interpreted in three different ways; as a transformation of a vector represented in one coordinate frame to another frame, as a rotation of a vector within the same frame and finally as a description of mutual orientation between two frames. The rotation matrix $\mathbf{R}$ from frame a to b is denoted $\mathbf{R}_a^b$. According to (Egeland & Gravdahl 2001) this matrix is an element in $\mathbf{SO}(3)$, which is defined by

$$\mathbf{SO}(3) = \{\mathbf{R}|\mathbf{R} \in \mathbf{R}^{3\times 3}, \mathbf{R}^T\mathbf{R} = \mathbf{I} \text{ and } \det \mathbf{R} = 1\}, \tag{2.6}$$

where $\mathbf{R}^{3\times 3}$ is the set of all 3×3 matrices with real elements and $\mathbf{I}$ is the 3×3 identity matrix. In general, the rotation of a vector from one frame to another, can be written with the following notation:

$$\mathbf{v}^{to} = \mathbf{R}_{from}^{to}\mathbf{v}^{from} \tag{2.7}$$

A useful parameterization of the rotation matrix is the angle-axis parameterization, $\mathbf{R}_{\lambda,\beta}$, corresponding to a rotation β about the $\boldsymbol{\lambda}$-axis:

$$\mathbf{R}_{\lambda,\beta} = \mathbf{I} + S(\boldsymbol{\lambda})\sin\beta + (1 - \cos\beta)S^2(\boldsymbol{\lambda}) \tag{2.8}$$

where S is the skew-symmetric operator. The rotation matrix also satisfies

$$\mathbf{R}_a^b = (\mathbf{R}_b^a)^{-1} = (\mathbf{R}_b^a)^T. \tag{2.9}$$

Simple rotations using Euler angles as parameters, are defined as

$$\mathbf{R}_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{bmatrix} \tag{2.10}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.11}$$

$$\mathbf{R}_z(\phi) = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

where the subscriptions x, y and z denotes the axis the angles ψ, θ and ϕ revolves about, respectively.

2.4.1 Derivative

The rotation matrix has the orthogonality property

$$\mathbf{R}_b^a(\mathbf{R}_b^a)^T = \mathbf{I} \tag{2.13}$$

Using this, the time derivative of the matrix is (Egeland & Gravdahl 2001)

$$\dot{\mathbf{R}}_b^a = \boldsymbol{\omega}_{ab}^a \times \mathbf{R}_b^a \tag{2.14}$$

where ω_{ab}^a is the angular velocity vector of b relative to a given in frame a. The angular velocity has the property $\boldsymbol{\omega}_{ab}^a = -\boldsymbol{\omega}_{ba}^a$,resulting in

$$\dot{\mathbf{R}}_b^a = S(\boldsymbol{\omega}_{ab}^a)\mathbf{R}_b^a = \mathbf{R}_b^a S(\boldsymbol{\omega}_{ab}^b), \tag{2.15}$$

when using the definition in Equation 2.4.

2.5 Transformation between different frames

The different rotations between frames used in this report are described here.

2.5.1 Transformation from Earth-Centered Orbit to ECI and ECEF frames

The rotation between these frames can only be done using the orbit estimator, and is necessary to be able to compare measurements with their respective reference model. The rotation is done (Svartveit 2003)

$$\mathbf{R}_{OC}^I = \mathbf{R}_z(-\Omega)\mathbf{R}_x(-i)\mathbf{R}_z(-\omega) \tag{2.16}$$

$$\mathbf{R}_{OC}^E = \mathbf{R}_z(-\Omega+\theta)\mathbf{R}_x(-i)\mathbf{R}_z(\omega) \tag{2.17}$$

where Ω is the Right Ascension of Ascending Node, i is the inclination of the satellite, ω is Argument of Perigee and θ is the ascension of the zero meridian. The $\mathbf{R}_y$ and $\mathbf{R}_z$ are the different simple rotations defined by Equations 2.11 and 2.12, respectively

2.5.2 Transformation from ECEF to ECI frame

The rotation of the ECEF frame relative to the ECI frame is a rotation about the coincident z_I and z_E-axes, equal to an angle $\alpha = \omega_e t$, where ω_e is the Earth rotation rate, and t is the time passed since the ECEF and ECI frame were aligned. This rotation can be expressed as

$$\mathbf{R}_E^I = \mathbf{R}_{z_I,\alpha} = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.18}$$

2.5.3 Transformation from ECI to Orbit frame

This rotation is dependent on the satellite rotation velocity ω_O. The Orbit frame is rotated an angle β about the y_I-axis, and is expressed as $\beta = \beta_0 + \omega_O t$, where β_0 is the drop angle, or latitude position, of the satellite and t is the time since last passing of 0° latitude. This can be expressed as

$$\mathbf{R}_{y_I,\beta} = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}. \tag{2.19}$$

In addition, the Orbit frame is upside-down relative to the ECI frame. This motivates the following rotation about the x_I-axis:

$$\mathbf{R}_{x_I,\pi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\pi & -\sin\pi \\ 0 & \sin\pi & \cos\pi \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{2.20}$$

which, combined with Equation 2.19, gives the total rotation necessary to transform a vector given in ECI frame to an Orbit frame representation:

$$\begin{aligned} \mathbf{R}_I^O &= \mathbf{R}_{x_I,\pi}\mathbf{R}_{y_I,\beta} \\ &= \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & -1 & 0 \\ \sin\beta & 0 & -\cos\beta \end{bmatrix} = \begin{bmatrix} \cos\mu & 0 & \sin\mu \\ 0 & -1 & 0 \\ \sin\mu & 0 & -\cos\mu \end{bmatrix} \end{aligned} \tag{2.21}$$

since both μ and β represent the latitude position of the satellite.

2.5.4 Transformation from Orbit to Body frame

The rotation matrix used extensively in this report, is the transformation between Orbit and Body frame, represented by $\mathbf{R}_O^B$. This rotation is dependent on the attitude of the satellite, and by estimating this rotation matrix, the attitude can be determined. Using Equation 2.8 with $\boldsymbol{\lambda} = \boldsymbol{\epsilon}$ and $\beta = \eta$ we get the rotation

$$\mathbf{R}_{\boldsymbol{\epsilon},\eta} = \mathbf{1} + 2\eta S(\boldsymbol{\epsilon}) + 2S^2(\boldsymbol{\epsilon}). \tag{2.22}$$

The rotation matrix from Body to Orbit frame is thus

$$\mathbf{R}_B^O = \mathbf{1} + 2\eta S(\boldsymbol{\epsilon}) + 2S^2(\boldsymbol{\epsilon}), \tag{2.23}$$

and by using the definition Equation 2.9 on Equation 2.23, a representation of the rotation from Orbit to Body frame can be calculated:

$$\mathbf{R}_O^B = (\mathbf{R}_B^O)^T = \begin{bmatrix} -\epsilon_1^2 - \epsilon_2^2 + \epsilon_3^2 + \eta^2 & 2(\epsilon_1\epsilon_2 + \eta\epsilon_3) & 2(\epsilon_1\epsilon_3 - \eta\epsilon_2) \\ 2(\epsilon_1\epsilon_2 - \eta\epsilon_3) & -\epsilon_1^2 + \epsilon_2^2 - \epsilon_3^2 + \eta^2 & 2(\epsilon_2\epsilon_3 + \eta\epsilon_1) \\ 2(\epsilon_1\epsilon_3 + \eta\epsilon_2) & 2(\epsilon_2\epsilon_3 - \eta\epsilon_1) & -\epsilon_1^2 - \epsilon_2^2 + \epsilon_3^2 + \eta^2 \end{bmatrix} \tag{2.24}$$

This rotation matrix can also be written as

$$\mathbf{R}_O^B = \begin{bmatrix} \mathbf{c}_1^B & \mathbf{c}_2^B & \mathbf{c}_3^B \end{bmatrix}, \tag{2.25}$$

where $\mathbf{c}_i^B = \begin{bmatrix} c_{ix}^B & c_{iy}^B & c_{iz}^B \end{bmatrix}^T$ are column vectors. When $\mathbf{c}_3^B = \begin{bmatrix} 0 & 0 & \pm 1 \end{bmatrix}^T$, the z_O-axis and z_B-axis are aligned. This vector can thus be seen as a description of the deviation between z_O-axis and z_B-axis, and is an indication of the performance of attitude control system.

Chapter 3

Reference Models

For determining the attitude of the satellite, two different sensors are used. Each needs a reference model to compare the measured data with. These models are presented here.

3.1 Magnetic

One of the sensors used is a magnetometer. It measures the magnetic field in three dimensions. In order to generate a magnetic reference model in Orbit frame, an orbit estimator for the satellite orbit is needed. The magnetic reference model is based on this estimator to transform the reference model from ECEF to Orbit frame.

3.1.1 Orbit estimator

A number of estimators are considered in (Svartveit 2003). These have been compared, and a recommendation is given.

The nCube satellite has only a limited supply of power. The attitude determination and control system is only a support system for the payload and is therefore not supposed to use too much power compared to the payload. With this in mind, a low-effect processor is preferred. This again limits the speed of the computations and therefore a simple estimator should be implemented.

The estimator proposed is given in Keplerian elements. These are, see Figure 3.1:

- Orbital Inclination
- Right Ascension of Ascending Node
- Argument of Perigee
- Eccentricity
- Mean Motion
- Mean Anomaly

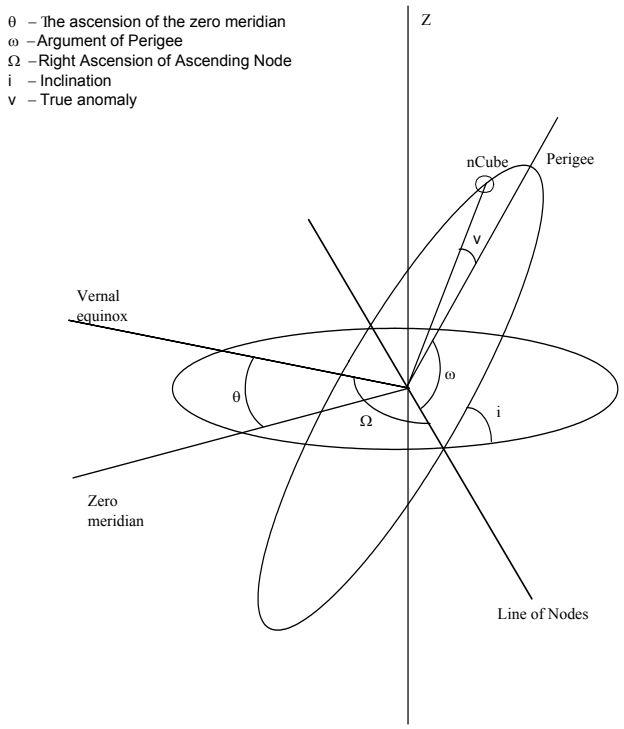

Figure 3.1: The Keplerian Elements.

These elements describe the position of the satellite at a specific time. This time must also be given, and the most widely used format is called epoch. Epoch gives the year and day of the year as a decimal number. Based on this time, the ascension of the zero meridian, θ, can also be calculated. Using Equation 2.18, with $\alpha = \theta$, the rotation between ECI and ECEF reference frame given by

$$\mathbf{R}_E^I = \mathbf{R}_z(\theta) = \mathbf{R}_z(\alpha) \tag{3.1}$$

The following four Keplerian elements specify the orientation of the orbital plane, the orientation of the orbit ellipse in the orbital plane, and the shape of the orbit ellipse.

Orbital Inclination

Denoted i. The orbit ellipse lies in the orbital plane. This always goes through the center of the Earth, but is tilted relative to the equator. The inclination is the angle between the orbital and equatorial plane. An orbit with inclination near 0° is called an equatorial orbit, and an orbit with inclination near 90° is called a polar orbit. The line of nodes is the intersection of equatorial plane and the orbital plane.

Right Ascension of Ascending Node

Denoted Ω. The satellite passes the the line of nodes in two places. The first when the satellite passes from south to north, called the ascending node, and the second when it passes from north

to south, called descending node. The angle between the ascending node and the vernal equinox is called the right ascension of ascending node. It is between 0° and 360°. This combined with the inclination defines the orbital plane in which the elliptic orbit lies.

Argument of Perigee

Denoted ω. The Earth lies in a focus point of the ellipse. The point in the ellipse closest to the Earth is called the perigee. Apogee is then the point on the ellipse farthest from the Earth. The angle between the line from perigee, through the Earth and to the apogee, and the line of nodes is the argument of perigee. This angle is defined as the angle from the ascending node, and is between 0° and 360°.

Eccentricity

Denoted e. The eccentricity is given as

$$e = \sqrt{1 - \frac{b^2}{a^2}} \tag{3.2}$$

where a is the semimajor-axis and b is the semiminor-axis. The semimajor-axis is half the distance between the apogee and the perigee, and semiminor-axis half the length between the edges perpendicular to a. For an ellipse, e is between 0 and 1, so for a perfect circle, $e = 0$.

The following Keplerian elements is time varying, and specify at what position the satellite is in the orbit described by the previous four elements.

Mean Motion

Denoted n. This is the average angular velocity, and describes the size of the ellipse. It is related to the semimajor-axis using Kepler's third law:

$$n^2 a^3 = \mu_e \tag{3.3}$$

where $\mu_e = GM_e$ is the Earth's gravitational constant. This relationship is the reason why mean motion is sometimes replaced by the semimajoraxis when describing the orbit of a satellite.

Mean Anomaly

Denoted M. This defines the position of the satellite in the ellipse. It is an angle that increases uniformly in time from 0° to 360° during one revolution. It is defined to be 0° at perigee. It is important to note that in a non-circular ellipse, this angle does not give the direction towards the satellite except at perigee and apogee. This is because satellite does not have a constant angular velocity. The different anomalies used is shown in Figure 3.2. True anomaly, v, is the direction from the Earth to the satellite. Given a circle, with center that coincides with the center of the orbital ellipse and a radius equal the semimajor axis. Then the eccentric anomaly, E, is the direction from the center of the ellipse to the point on the circle where a line, perpendicular to the semimajor axis through the satellites position, crosses the circle.

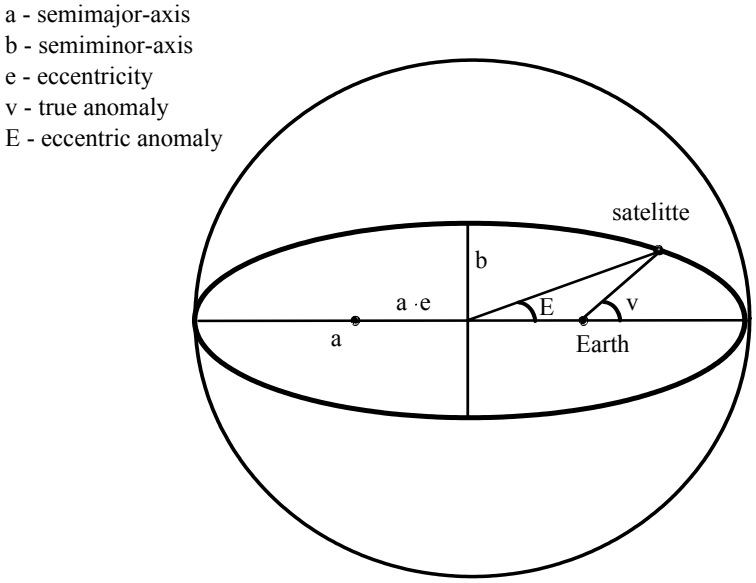

Figure 3.2: Keplerian elements

The relationship between true anomaly and eccentric anomaly is

$$\cos v = \frac{\cos E - e}{1 - e\cos E} \tag{3.4}$$

$$\sin v = \frac{\sqrt{1-e^2}\sin E}{1 - e\cos E} \tag{3.5}$$

And the relationship between mean anomaly and eccentric anomaly is

$$M = E - e\sin E(t) \tag{3.6}$$

Given the Keplerian elements for a time, t_0, a prediction of the orbit is

$$M(t_0 + t) = M(t_0) + n \cdot t \tag{3.7}$$

where t is the time passed since t_0.Equation 3.7 describes the motion of the satellite in Earth-Centered Orbit frame. To transform this to ECEF frame, a solution of

$$E(t) = M(t) + e \cdot \sin E(t) \tag{3.8}$$

must be calculated. Equation 3.8 gives the relation between the eccentric anomaly, e, and the mean anomaly, M. The vector from the center of the earth to the satellite decomposed in the Earth-Centered Orbit frame is

$$\mathbf{r}^{OC} = a \begin{bmatrix} \cos E - e \\ \sqrt{1-e^2}\sin E \\ 0 \end{bmatrix} \tag{3.9}$$

The orbit estimator can now be implemented in Earth-Centered Inertial frame and Earth-Centered Earth Fixed frame using the rotation in Equations 2.16 and 2.17:

$$\mathbf{r}^I = \mathbf{R}_z(-\Omega)\mathbf{R}_x(-i)\mathbf{R}_z(-\omega)\mathbf{r}^{OC} \tag{3.10}$$

$$\mathbf{r}^E = \mathbf{R}_z(-\Omega+\theta)\mathbf{R}_x(-i)\mathbf{R}_z(\omega)\mathbf{r}^{OC} \tag{3.11}$$

where Ω is the Right Ascension of Ascending Node, i is the inclination of the satellite, ω is Argument of Perigee and θ is the ascension of the zero meridian.

An orbit estimator based only on the Keplerian elements will degrade in accuracy over time. Effects that are not modeled when using the Keplerian elements are perturbations due to:

- The nonspherical earth
- The Sun and the Moon
- Atmospheric drag
- Solar radiation

To improve the simple orbit estimator, some of these perturbations have been considered by (Svartveit 2003) and implemented in the enhanced orbit estimator. Effects due to the nonspherical Earth, $\dot{\Omega}_{J2}$ and $\dot{\omega}_{J2}$, and effects due to the Sun and the Moon, $\dot{\Omega}_{moon}$ and $\dot{\Omega}_{sun}$, $\dot{\omega}_{moon}$ and $\dot{\omega}_{sun}$, are implemented in the estimator, given by:

$$\begin{aligned}\mathbf{r}^E = {} & \mathbf{R}_z(-\Omega_0 + (\dot{\Omega}_{J2}+\dot{\Omega}_{moon}+\dot{\Omega}_{sun})t) + \theta_0 + \omega_e)\mathbf{R}_z(-i)\cdot \\ & \mathbf{R}_z(-(\omega_0 + (\dot{\omega}_{J2}+\dot{\omega}_{moon}+\dot{\omega}_{sun})t))a \begin{bmatrix} \cos E - e \\ \sqrt{1-e^2}\sin E \\ 0 \end{bmatrix}\end{aligned} \tag{3.12}$$

This model will also degrade with time, but not as fast as the simple estimator. To keep the estimator accurate, it is possible to update it with the accurate Keplerian elements at given intervals.

3.1.2 IGRF

As seen on Figure 3.3, the magnetic field is highly varying over the Earth's surface. To use a high-resolution lookup-table, where each entry represents the magnetic field at that given position, would demand a too large memory bank. A crude simplification is to model the field as a dipole, but a more accurate model is the IGRF model.

The International Geomagnetic Reference Field, IGRF, is an attempt by the International Association of Geomagnetism and Aeronomy to provide a model acceptable to a variety of users. It gives an approximation of the Earth's magnetic field originating from the Earth's core. At any given time, the IGRF specifies the numerical coefficients of a truncated spherical harmonic series. As this report is written, the truncation is at $n = 10$, resulting in 120 coefficients. The IGRF is revised every 5 years, the last done in 2000, and is thus called IGRF 2000.

The IGRF model comprises a set of spherical harmonic coefficients called Gauss coefficients, g_n^m and h_n^m, in a truncated series expansion of a geomagnetic potential function of internal origin

$$V = a\sum_{n=1}^{N}\sum_{m=0}^{n}(\frac{a}{r})^{n+1}(g_n^m\cos m\phi + h_n^m\cos m\phi)P_n^m\cos\theta \tag{3.13}$$

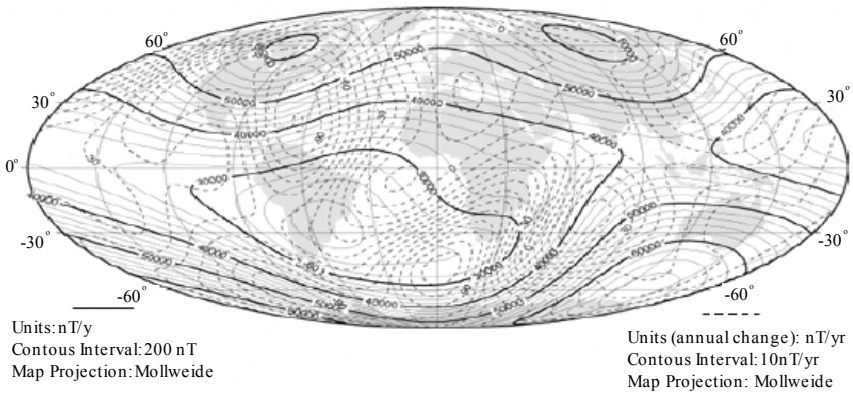

Figure 3.3: Magnitude of the Earth's magnetic field

where a is the mean radius of Earth, r is the distance from the center of Earth, ϕ and θ are the longitude and colatitude, respectively. The colatitude is 90° minus the latitude. $P_n^m(\cos\theta)$ are Schmidt quasi-normalized associated Legendre functions of degree n and order m, where $n \geq 1$ and $m \leq n$. The maximum spherical harmonic degree of the expansion is N.

The IGRF model is rotating with the Earth, and thus given in ECEF frame. Using Equation 2.9 on Equation 3.11 results in

$$\mathbf{B}^{OC} = (\mathbf{R}_z(-\Omega+\theta)\mathbf{R}_x(-i)\mathbf{R}_z(-\omega))^{-1}\mathbf{B}^{ECEF} \tag{3.14}$$

$$\implies \mathbf{B}^{OC} = \mathbf{R}_z(\omega)\mathbf{R}_x(i)\mathbf{R}_z(\Omega-\theta)\mathbf{B}^{ECEF} \tag{3.15}$$

where B^{ECEF} is the resulting vector from the IGRF model.

A final transformation is done to get the magnetic field in Orbit frame:

$$\mathbf{B}^{O} = \mathbf{R}_x(\frac{\pi}{2})\mathbf{R}_z(\nu+\frac{\pi}{2})\mathbf{B}^{OC} \tag{3.16}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} -\sin\nu & \cos\nu & 0 \\ -\cos\nu & -\sin\nu & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{B}^{OC} \tag{3.17}$$

$$= \begin{bmatrix} -\sin\nu & \cos\nu & 0 \\ 0 & 0 & 1 \\ \cos\nu & \sin\nu & 0 \end{bmatrix} \mathbf{B}^{OC} \tag{3.18}$$

where ν is the true anomaly.

3.2 Light

The second sensor is a Light Dependent Resistor, or LDR. This measures the light received on each of the six sides of the satellite and, based on this, determines the attitude of the satellite. The light received can either origin from the sun, or reflected from the Earth. The Earth Albedo model describes the latter.

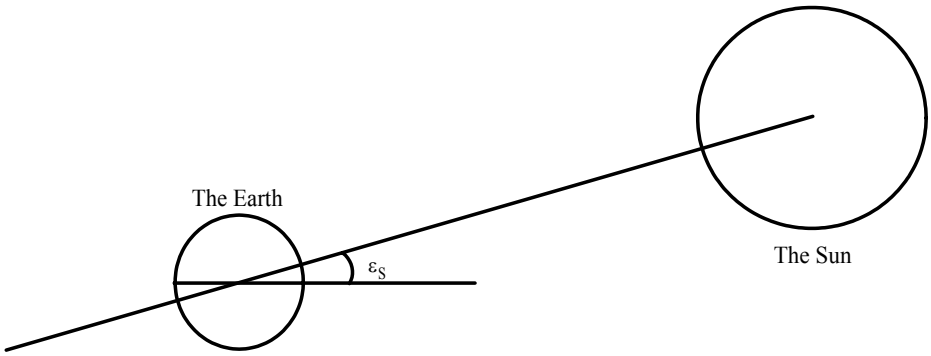

Figure 3.4: Elevation of the Sun as experienced from the Earth

3.2.1 Sun model

The motion of the Sun, as seen from Earth, is necessary to describe in order to get a reference that the measured light can be compared to. This model is based on the work of (Kristiansen 2000).

To simplify the model of the sun, it is convenient to describe the Sun-Earth relationship as seen from the Earth. This implies that the Sun evolves around the Earth with the same orbit-parameters as the Earth-orbit around the Sun has.

The elevation of the Sun, ε_s, seen on Figure 3.4, varies between 23° and -23° degrees, depending on the time of the year. The elevation has a period that begins at the first day of spring at 0°, increases to 23°, decreases to -23° and back to the origin. The period is approximately 365 days, and can be described as

$$\varepsilon_s = \frac{23\pi}{180}\sin(\frac{T_s}{365}2\pi) \tag{3.19}$$

where T_s is the time since the first day of spring.

In addition to the variation in elevation, the Sun can also be described as to orbit the Earth, instead of the other way around, with a period of 365 days. This orbit parameter is

$$\lambda_s = \frac{T_s}{365}2\pi, \tag{3.20}$$

where again T_s is the time since the the first day of spring. The position of the Sun at the day the Earth passes the vernal equinox, expressed in ECI frame, is

$$\mathbf{s}_0^I = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}. \tag{3.21}$$

Rotating this position λ_s about the z_I-axis, and ε_s about the y_I-axis, the position of the sun at a given time, t, is

$$\mathbf{s}^I = \mathbf{R}_y(\varepsilon_s)\mathbf{R}_z(\lambda_s)\mathbf{s}_0^I, \tag{3.22}$$

where $\mathbf{R}_y$ and $\mathbf{R}_z$ are defined in Equations 2.11 and 2.12, respectively. The motion of the Sun, as experienced from the Earth, can now be described as

$$\mathbf{s}^I = \begin{bmatrix} \cos\lambda_s \cos\varepsilon_s \\ \sin\lambda_s \\ \cos\lambda_s \sin\varepsilon_s \end{bmatrix} \tag{3.23}$$

This representation can now be transformed to the Orbit frame using Equation 2.21, resulting in

$$\mathbf{s}^O = \mathbf{R}_I^O \mathbf{s}^I, \tag{3.24}$$

and be compared with the measurement done in Body frame.

3.2.2 Earth Albedo Model

Work has previously been done on this model, see (Appel 2002), that indicates a deterioration of the accuracy of the LDR since much of the light detected by these is in fact light reflected from the Earth. This reflection varies heavily with the terrain on the Earth. Given the altitude of the satellite, and latitude of the Sun, this model will give an albedo vector that will describe the light reflected from the Earth. Combining this with the model of the motion of the Sun will create a good reference model for the light measurements done on board the satellite.

Chapter 4

Mathematical modeling

In order to be able to simulate the satellite in its environment, a mathematical model of both the dynamics of the satellite and it's kinematics is derived here. A linearization is also done for use in the Kalman Filter.

4.1 Satellite model

This model can be split up into different models. One describes the behavior of the satellite given external force acting on it. The other describes the relation between the Body frame and the Orbit frame i.e. the attitude of the satellite.

4.1.1 Kinematics

The orientation of the satellite is found by integrating its velocities. Quaternions are used for representation to prevent the existence of singularities. The differential equations are given by (Egeland & Gravdahl 2001)

$$\dot{\eta} = -\frac{1}{2}\boldsymbol{\epsilon}^T \omega_{OB}^B \tag{4.1}$$

$$\dot{\boldsymbol{\epsilon}} = \frac{1}{2}\eta\boldsymbol{\omega}_{OB}^B - \frac{1}{2}\boldsymbol{\omega}_{OB}^B \times \boldsymbol{\epsilon} \tag{4.2}$$

Using the skew-symmetric operator Equation 4.2 can be written as

$$\dot{\boldsymbol{\epsilon}} = \frac{1}{2}[\eta\mathbf{I} - S(\boldsymbol{\epsilon})]\boldsymbol{\omega}_{OB}^B. \tag{4.3}$$

4.1.2 Dynamics

The satellite is modeled as a rigid body. The dynamics can, from Euler's moment equation, be derived as (Kyrkjebo 2000)

$$\mathbf{I}\dot{\boldsymbol{\omega}}_{IB}^B + \boldsymbol{\omega}_{IB}^B \times \mathbf{I}\boldsymbol{\omega}_{IB}^B = \boldsymbol{\tau}^B, \tag{4.4}$$

where $\mathbf{I}$ is the identity matrix, $\boldsymbol{\omega}_{IB}^{B}$ the angular velocity of the Body frame relative to the ECI frame expressed in the Body frame and τ^{B} is the sum of all torques acting on the satellite. Using the skew-symmetric operator the dynamics can be expressed as

$$\mathbf{I}\dot{\boldsymbol{\omega}}_{IB}^{B} + S(\boldsymbol{\omega}_{IB}^{B})\mathbf{I}\boldsymbol{\omega}_{IB}^{B} = \boldsymbol{\tau}^{B}. \tag{4.5}$$

The angular velocity $\boldsymbol{\omega}_{IB}^{B}$ can be written as the sum of two angular velocities, as

$$\boldsymbol{\omega}_{IB}^{B} = \boldsymbol{\omega}_{IO}^{B} + \boldsymbol{\omega}_{OB}^{B} = \mathbf{R}_{O}^{B}\boldsymbol{\omega}_{IO}^{O} + \boldsymbol{\omega}_{OB}^{B} \tag{4.6}$$

where $\mathbf{R}_{O}^{B}$ is defined in Equation 2.24 and $\boldsymbol{\omega}_{IO}^{O} = [0 \quad -\omega_o \quad 0]$ is the known angular velocity of the Orbit frame relative to the ECI frame, expressed in Orbit frame. This velocity depends on the altitude of the orbit, and can be calculated (Kyrkjebo 2000) according to

$$\omega_o = \sqrt{\frac{GM_e}{R^3}} \tag{4.7}$$

where G is the gravitational constant of the Earth, M_e is the mass of the Earth and R is the distance from the centre of the Earth to the satellite.

4.2 Linearized model

A linearized model is needed to implement the Kalman Filter. Both the kinematics and the dynamics are linearized here.

4.2.1 Linearized kinematics

The nonlinear kinematics are given by Equations 4.1 and 4.2. To get the linearized model, the nonlinear equations for the kinematics are differentiated with respect to the state vector. The states are

$$\mathbf{x}^{T} = \begin{bmatrix} \eta & \epsilon_1 & \epsilon_2 & \epsilon_3 & \omega_{OB,1}^{B} & \omega_{OB,2}^{B} & \omega_{OB,3}^{B} \end{bmatrix}^{T}. \tag{4.8}$$

To find the linear kinematics, $\mathbf{F}_{kin}$, for the system, the nonlinear function $\mathbf{f}(\mathbf{x}, \mathbf{u})$ is differentiated according to

$$\mathbf{F}_{kin} = \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}. \tag{4.9}$$

By differentiating each of the individual equations of the kinematics, Equation 4.9 can be written as

$$\mathbf{F}_{kin} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_7} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_4}{\partial x_1} & \cdots & \frac{\partial f_4}{\partial x_7} \end{bmatrix}, \tag{4.10}$$

where x_1 through x_7 are the states given in Equation 4.8, and

$$f_1 = -\frac{1}{2}\boldsymbol{\epsilon}^T \omega_{OB}^B \tag{4.11}$$

$$\begin{bmatrix} f_2 \\ f_3 \\ f_4 \end{bmatrix} = \frac{1}{2}[\eta \mathbf{I} - S(\boldsymbol{\epsilon})]\boldsymbol{\omega}_{OB}^B. \tag{4.12}$$

The resulting matrix is (Kyrkjebo 2000)

$$\mathbf{F}_{kin} = \begin{bmatrix} \frac{1}{2}(\boldsymbol{\omega}_{OB}^B)^T & 0 & -\frac{1}{2}\boldsymbol{\epsilon}^T \\ -\frac{1}{2}S(\boldsymbol{\omega}_{OB}^B) & \frac{1}{2}\boldsymbol{\omega}_{OB}^B & \frac{1}{2}[\eta \mathbf{1} + S(\epsilon) \end{bmatrix} \tag{4.13}$$

4.2.2 Linearized dynamics

Using the same method as in the previous section, the linear dynamics can be found by using

$$\mathbf{F}_{dyn} = \begin{bmatrix} \frac{\partial f_5}{\partial x_1} & \cdots & \frac{\partial f_5}{\partial x_7} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_7}{\partial x_1} & \cdots & \frac{\partial f_7}{\partial x_7} \end{bmatrix}. \tag{4.14}$$

To make the differentiating less complex, the dynamics can be written as a sum of three parts:

$$\begin{bmatrix} f_5 \\ f_6 \\ f_7 \end{bmatrix} = \dot{\boldsymbol{\omega}}_{OB}^B = \mathbf{f}_{inert}(\mathbf{x}) + \mathbf{f}_{torq}(\mathbf{x}) + \mathbf{f}_{add}(\mathbf{x}) \tag{4.15}$$

where

$$\mathbf{f}_{inert}(\mathbf{x}) = \mathbf{I}^{-1}[-(\boldsymbol{\omega}_{OB}^B + \mathbf{R}_O^B \boldsymbol{\omega}_{IO}^O) \times (\mathbf{I}(\boldsymbol{\omega}_{OB}^B + \mathbf{R}_O^B \boldsymbol{\omega}_{IO}^O)) \tag{4.16}$$

$$\mathbf{f}_{torq}(\mathbf{x}) = \mathbf{I}^{-1}\boldsymbol{\tau}^B \tag{4.17}$$

$$\mathbf{f}_{add}(\mathbf{x}) = S(\boldsymbol{\omega}_{OB}^B)\mathbf{R}_O^B \boldsymbol{\omega}_{IO}^O. \tag{4.18}$$

The angular velocity of the Body frame relative to the Inertial frame has been substituted using Equation 4.6 . The linearization can now be done by differentiating each of the Equations 4.16, 4.17 and 4.18 with respect to the state vector, and sum up the result. This can be written as:

$$\mathbf{F}_{dyn} = \frac{\partial \mathbf{f}_{inert}}{\partial \mathbf{q}} + \frac{\partial \mathbf{f}_{torq}}{\partial \mathbf{q}} + \frac{\partial \mathbf{f}_{add}}{\partial \mathbf{q}} \frac{\partial \mathbf{f}_{inert}}{\partial \boldsymbol{\eta}} + \frac{\partial \mathbf{f}_{torq}}{\partial \boldsymbol{\eta}} + \frac{\partial \mathbf{f}_{add}}{\partial \boldsymbol{\eta}} \frac{\partial \mathbf{f}_{inert}}{\partial \boldsymbol{\omega}_{OB}^B} + \frac{\partial \mathbf{f}_{torq}}{\partial \boldsymbol{\omega}_{OB}^B} + \frac{\partial \mathbf{f}_{add}}{\partial \boldsymbol{\omega}_{OB}^B} \tag{4.19}$$

where the dependence of $\mathbf{x}$ in each function is left out for notation convenience.

The complete linearized model is (Kyrkjebo 2000)

$$\mathbf{F}_{total}(\mathbf{x}) = \begin{bmatrix} \mathbf{F}_{kin}(\mathbf{x}) \\ \mathbf{F}_{dyn}(\mathbf{x}) \end{bmatrix}, \tag{4.20}$$

where $\mathbf{F}_{kin}$ is found in Equation 4.13, and

$$\mathbf{F}_{dyn}(\mathbf{x}) = \begin{bmatrix} a_{51} & a_{52} & a_{53} & a_{54} & 0 & a_{56} & a_{57} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & 0 & a_{67} \\ a_{71} & a_{72} & a_{73} & a_{74} & a_{75} & a_{76} & 0 \end{bmatrix} \tag{4.21}$$

where each of the components a_{ij} is presented in Appendix B.1. The matrix $\mathbf{F}_{total}$ is a 7×7 matrix.

Chapter 5

Hardware

The attitude estimator is implemented in a real satellite. Some specifications were outlined early in the process of designing the satellite and the mission for the Attitude Determination and Control System. Among these were what should be measured and how. The hardware components of the Attitude Determination subsystem are described here.

5.1 Sensors

The two sensors used in the satellite use different measuring techniques. Here, the hardware of the sensors and how it works is described.

5.1.1 Light Dependent Resistors

The first sensor used for attitude determination, is a light sensor. One sensor is placed on each side of the satellite, six in total, see Figure 5.1. They all measure the amount of light received on their respective side. These measurements can be combined, and vector pointing in the direction of the Sun calculated.

The output from the sun sensor is proportional with the intensity of light that is detected by the sensor. This intensity is dependent of the angle of attack. A higher angle of attack result in less intensity of the light received at a given side of the satellite. But as the surface of the resistor is small, less than 0.5 cm^2, it is not the experienced light by one sensor that will change much with changing angle of attack. It is the experienced light of one sensor in relation to the rest. As the satellite changes attitude, some sensors will get more and more light, while others will get less. It is the internal relation that can be used to estimate in what direction the Sun is.

The combined output vector from the sensors is

$$\mathbf{S}^B = \begin{bmatrix} s_1 - s_6 \\ s_2 - s_5 \\ s_3 - s_4 \end{bmatrix} \tag{5.1}$$

where s_i is the measured voltage over sensor i, and placed on the satellite shown on Figure 5.1.

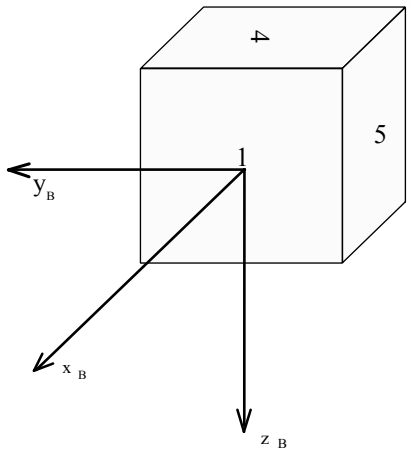

Figure 5.1: NCube with sensors 1, 4 and 5. The three remaining is placed on the opposite, making the sum of two opposite sides 7.

The measurements are done in the sensor frame. Since the sensor frame coincides with the Body frame, we only need a reference in Orbit frame to compute the attitude. This reference, the Sun model, is described in Chapter 3.2.1.

Again there is some inaccuracy with our method; first, the Sun model used here is crude. Some assumptions are done that degrade the accuracy, but they simplify the computations, and are therefore done in the interest of the overall performance of the ADCS. The motion of the Sun is described as a circular orbit around the Earth, and the satellite is assumed placed in the center of the Earth. This is not accurate, but since the distance between the Sun and the Earth is large relative to the variations in this distance, the model will suffice in our case. (Kristiansen 2000) has found this error to be:

$$\xi = \arctan\left(\frac{R_O}{R_S}\right) \approx 1.43 \cdot 10^{-7} rad \tag{5.2}$$

where R_O is the distance between the Earth and nCube, R_S is the distance between the Earth and the Sun, and ξ is the error. The error is small relative to the overall accuracy of the system and can therefore be ignored.

And second, a numerous light sources are neglected - the stars. It is assumed that the light received from the sun will be much brighter than that received from the stars, and the stars are thus ignored.

Third, and last, the Earth will reflect some of the light received from the Sun, and some of this will reach the satellite. This is effect is called the Albedo.

5.1.2 Magnetometer

The magnetometer is used to measure a magnetic field. To be able to use this kind of sensor on a satellite, it has to be in a relative low orbit. The nCube has a Low Earth Orbit, an altitude of approximately 700km, and the magnetic field in this altitude is well modeled.

Measurements are done in three dimensions, by three independent orthogonal sensors. The Earth's magnetic field is thus measured in the sensor frame. To get the best result, the sensor and the satellite axes should be parallel, or at least have a known configuration relative to another.

The measurements from the magnetometer are compared with the modeled magnetic field. As the magnetometer gives an output in Body frame, and the model is in Orbit frame, the relation between these frames can be used to calculate the attitude of the satellite.

The accuracy of the magnetometer is, according to (Bak 1999), limited mainly by two factors:

- Disturbance fields due to spacecraft electronics
- External disturbances

Disturbance fields due to spacecraft electronics

In orbit, the measured magnetic field is not only originating from the Earth. The satellite, with its electronic circuits, may also influence the magnetometer and degrade the estimation. To overcome this, one has to consider shielding when implementing the hardware. In addition, a different alignment of single components can increase or decrease the disturbance greatly.

External disturbances

The ionosphere is not a uniform field. Electrical currents within the ionosphere induce unpredictable magnetic disturbances. How this influence the magnetometer is not known.

Not only is the magnetometer inaccurate, the model, which the measurements are compared to, also has errors. These errors decrease with increased complexity of the model. A better model will decrease the errors, but demand more resources of the processor running the ADCS. Also, errors in the orbit estimator, i.e. the estimated position of the satellite, will introduce errors.

5.2 Communication

5.2.1 Interface to Light Dependent Resistors

To read the output from the sensors, an Analog-to-Digital - converter (ADC) is used. As there are six sensor, six different channels connected to the ADC are used. They all use the same ADC, but the microcontroller switches, or multiplexes, between the channels. When the conversion for one resistor is completed, the next resistor is multiplexed in and when this conversion is done, the next is multiplexed in and so on.

By measuring the voltage over these resistors, the resistance can be found using Ohm's law,

$$U = R \cdot I \tag{5.3}$$

where U is voltage over the Light Dependent Resistors, R is the resistance of the resistor and I is the current through the it. Thus by measuring the voltage over a resistor, and given a constant current, the value will reflect the resistance.

To be able to measure changes in the voltage over resistor as it changes resistance, a voltage divider is used. This is basically two resistors connected in series. One is the LDR and has a

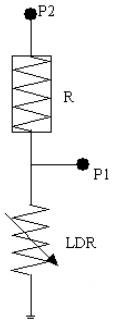

Figure 5.2: The voltage divider. Voltage is connected to P2, and measurements are done between P1 and ground.

variable resistance, and the other has a known, fixed resistance, see Figure 5.2. The voltage will divide over these two resistors as

$$U_{LDR} = \frac{R_{LDR}}{R_{LDR} + R_{known}} \cdot V_{ref}, \tag{5.4}$$

where U_{LDR} is the voltage over the LDR, R_{LDR} is the resistance in the LDR, R_{known} is the size of known resistor and V_{ref} is a known voltage source. The measured voltage is thus reflecting the current resistance in the LDR. The connection of the LDR to the microcontroller can be seen on Figure 5.3

5.2.2 Interface to magnetometer

In this sensor an interface using a serial communication bus is implemented. This is called Universal Asynchronous serial Receiver and Transmitter, or UART, and uses a standard known as RS232 which defines several aspects of the bus. The bus uses three wires, one for transmitting, one for receiving and ground. In addition, the standard states that a high level on either communication line are 12V. The magnetometer follows this, but the microcontroller operates at maximum 5V. A chip from Maxim, called MAX233, transforms the voltage level from the microcontroller to 12V and vice versa, see Figure 5.4.

By sending commands to the sensor over the bus, different modes of operation of the magnetometer can be chosen. One of these will tell the magnetometer to average the measurements, that is to low pass filter the measurements. When in operation, the continuous output mode is chosen. This will generate new, updated measurements at 20 samples per second.

5.2.3 Interface to other subsystems

The estimator must also communicate with other subsystems on the satellite.

- The estimator must receive updated Keplerian elements from the groundlink system.
- The attitude control system needs the estimated states to control the attitude of the satellite.

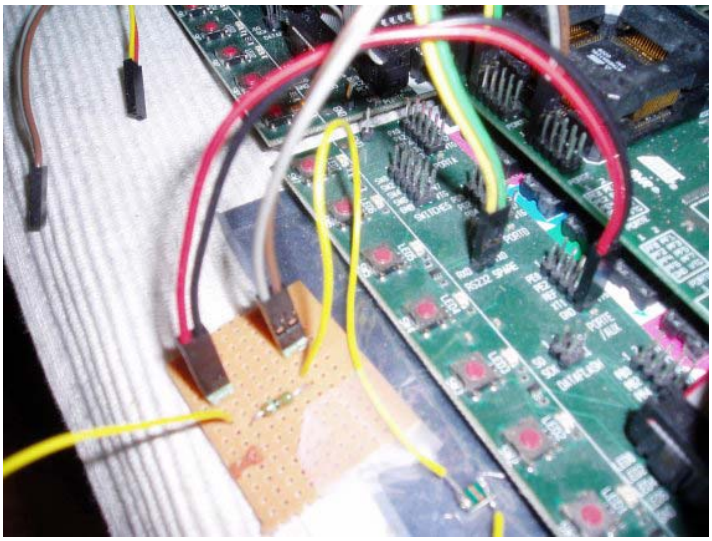

Figure 5.3: The LDR (lower-middle) is connected to the microcontroller development board (green) via the voltage divider-board (light brown).

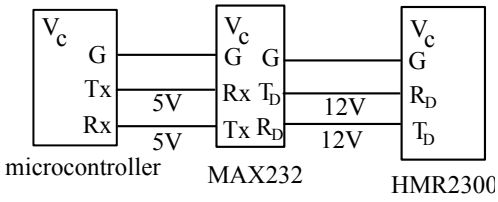

Figure 5.4: Connection of the magnetometer using MAX233. V_c is the supply voltage.

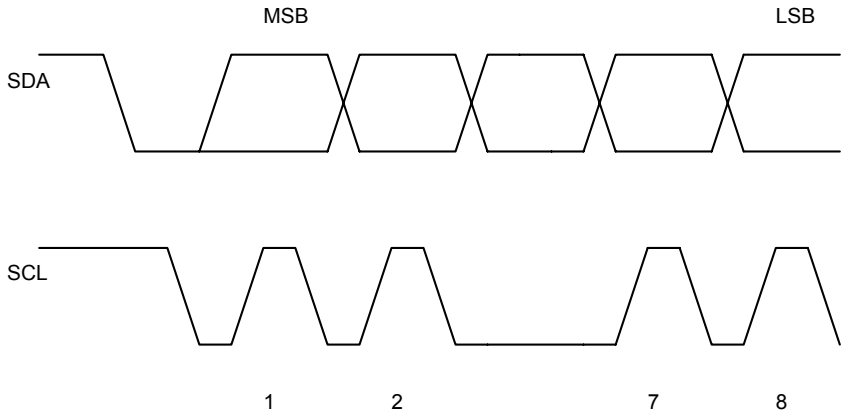

Figure 5.5: The transmission of one byte using the TWI.

- The groundlink system also needs the estimated states. To verify that the estimator is fully functional, the ground station needs the estimates at given intervals.

In addition, several other subsystems are connected to the communication bus, although they will not need any communication with the estimator. A communication bus, called Two-Wire-Interface is used for inter-node communication.

As the name suggests, the bus consists of only two wires. The first wire is the data-wire, and the second is the clock-wire. The bus uses masters and slaves to communicate. It is the master that generates the clock signal, and controls the transmission, and the transmitting node, either the slave or the master, synchronizes the data on the data-wire with this clock, Figure 5.5.

Each of the different nodes needs an unique address. To initiate communication, this address is the first thing to be sent over the bus. The owner of this address answers, and communication has been established. The initiating node decides which of the two nodes is the transmitting one, and when the data has been acknowledged as received by the receiver, the bus is released and free for other nodes to initiate communication.

It is only the masters that can initiate a transmission. Since many of the subsystem in the satellite must be able to initiate transmission, they have to be considered as masters on the TWI bus. There are several aspects of the Two-Wire-Interface that must be taken into account when using the bus with multiple masters, one of the most important be the arbitration process. Two nodes can be seen on Figure 5.6. These are the Attitude Determination System and a debug node, the latter used for debugging during development.

Two-Wire-Interface and multiple masters

Every transmission is initiated with a START condition put on the bus. If the bus is available, the master continue with the address of the node that it wants to communicate with, and at the same time indicating that the bus is occupied. A problem arises when two or more masters initiate with a START condition at the same time. If they all would be allowed to use the bus at the same, no useful data can be read from the bus. This is where arbitration is used. Every

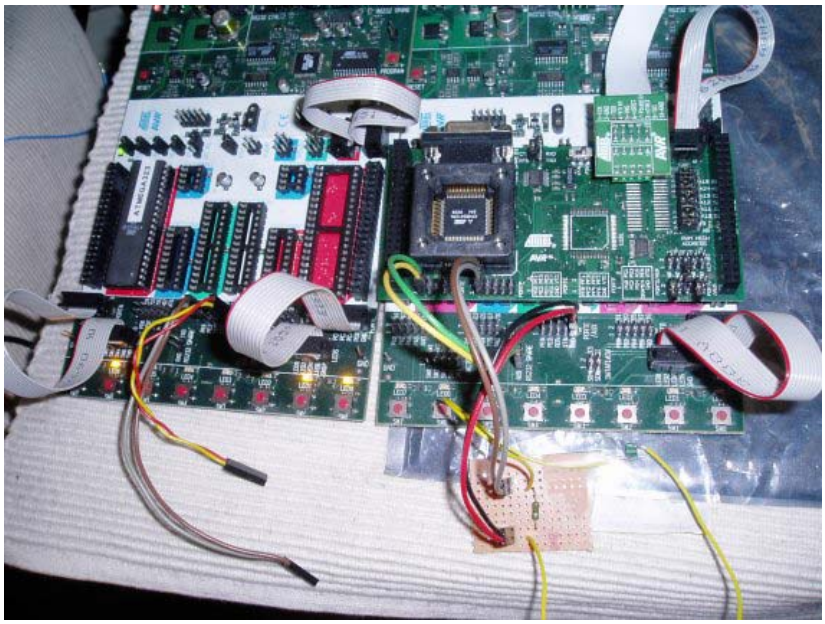

Figure 5.6: The debug node (left) and the Attitude Determination node (right), with the connection to a LDR. TWI communcation uses the grey and brown not-yet-connected wires.

master that puts data on the bus, also listen on the bus. If the current bit on the bus differ from what it transmitted, the master has lost arbitration and enters slave mode. It will then listen on the bus to check whether it is being addressed by the winning master. Arbitration will continue until only one master remains, and this may take many bits. If several masters are trying to address the same slave, arbitration will continue into the data packet.

The second thing to take into account when using multiple masters on a TWI bus, is the clock signal. A wired-ANDing of the bus-lines is used to solve this problem. The serial clocks from all masters will be wired-ANDed, yielding a combined clock with a high period equal to that of the master with the shortest high period. The low period of the combined clock is equal to the low period of the master with the longest low period. This will in effect change the rate of transmission, depending on each of the clock generated by the masters.

Transmission

When there is only one master left transmitting, the addressed node will acknowledge the masters attempt to initiate transmission. Depending on the direction of the transmission, two things can happen:

- Master Transmit - the master will enter transmitter mode, and slave will enter receiver mode. The message is transmitted, and then, to release the bus, a STOP condition is put on the bus.
- Master Receive - the master will enter receiver mode, and slave will enter transmitter mode. The master is still in control, so when it knows that the last byte is transmitted

by the slave, and this has been received, it will put a STOP condition on the bus and release the bus.

5.3 Covariance of the sensors

In the ideal case, the sensors would produce accurate measurements with no errors. As this is not the case, the error that influences the measurement has be modeled. The measurement can in general be expressed as

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{v} \tag{5.5}$$

where $\mathbf{y}$ is the output from the sensor, $\mathbf{H}$ is the configuration of sensor, $\mathbf{x}$ is the different states that are measured and finale, $\mathbf{v}$ is the error, or noise that deteriorates the measurement. In the case of Kalman Filter, the error is assumed to be white noise. That is (Farrell 1999)

$$E[\mathbf{v}(\lambda)] = 0 \tag{5.6}$$

$$var[\mathbf{v}(\lambda), \mathbf{v}(\tau)] = \mathbf{R}(\lambda, \tau). \tag{5.7}$$

where λ and τ are two random instants in time. This means that the expected magnitude of the error over time is zero, and that the covariance is $\mathbf{R}$. It is this value that are later calculated for the two sensors. In addition to Equations 5.6 and 5.7, it is convenient to assume that the measurement noise is independent of the current and previous states,

$$var[\mathbf{v}(t), \mathbf{x}(\tau)] = 0 \quad \text{for } t \geq \tau. \tag{5.8}$$

In order to find the solution of Equation 5.7, experiments with the sensors have to be performed. In both cases the respective output are logged over a relative long time to best minimize the effects of time-limited disturbances.

5.3.1 Light Dependent Resistors

One LDR is connected to the ADC on the microcontroller, where the voltage over the LDR are measured. This is then transmitted to a computer which logs the incoming data. As the result of the calculations in the microcontroller can be anything between 0 and 5 volts, it is stored as a float variable. This type of variable is in the microcontroller stored, according to (IEEE 2004), as four bytes. The transmission to the computer is done one byte at the time. The computer must manipulate these four bytes, or 32 bits, according to the standard, which is

$$(-1)^s \cdot 2^{e-127} \cdot 1.f, \tag{5.9}$$

where s the sign bit, e is the biased exponent and f is the normalized fraction. That f is normalized basically means that the radix point is placed after the first non-zero digit. Since we are operating with bits, the only non-zero digit is 1. The 32 bits that represent one floating number, are divided among s, e and f according to:

- s is the first, or the Most Significant Bit,
- e is the following 8 bits, and

- f is the last 23 bits.

The calculation described in Equation 5.9 is done on every measurement, and the covariance can then be found. Several independent sets of measurement data were created and the covariance was calculated for each one independent of each other. It is assumed that every resistor will have the same covariance, and is thus only necessary to experiment with one of them. The covariance used in the simulations and implementation can be found in Table 1, as the median of the seven independent experiments.

Experiment No.	1	2	3	4	5	6	7
Covariance	0.0020	0.0021	0.0025	0.0027	0.0038	0.0068	0.0070

Table 1. The results from experiments with the light sensor, sorted in increasing order.

5.3.2 Magnetometer

The datastream from the magnetometer differ from the datastream from the LDR as the former can be transmitted directly to the computer, and not via the microcontroller. Conversion from the four bytes to a floating number, as Equation 5.9 describes, is done prior to transmission in the magnetometer. The resulting measurement-set is a three dimensional matrix, one for each axis.

Integrated in the magnetometer is a possibility to average the measurements, using

$$x(n) = \frac{x(n)}{2} + \frac{x(n-1)}{4} + \frac{x(n-2)}{8} + \frac{x(n-3)}{16} + \dots \tag{5.10}$$

where n is the current measurement. This is in fact a low-pass filtering, and any spikes in the measurement are thus reduced. This is in normal operation turned on, and should also be turned on when determining the covariance. The calculated covariance is

$$var[\mathbf{v}_b(\lambda), \mathbf{v}_b(\tau)] = \begin{bmatrix} 4.2144 \cdot 10^{-6} & 4.4323 \cdot 10^{-6} & 4.2144 \cdot 10^{-6} \end{bmatrix}^T.$$

5.4 Microcontroller

This is the brain of the estimator. It will run the Kalman filter software developed in the programming language C. In addition to be able to do the computation, it must also incorporate different interfaces to the other subsystems of the satellite and also the sensors. What these are, were decided by the sensor specifications and the satellite management group.

The management group has decided that every subsystem should run on an Atmel microcontroller. The most powerful microcontroller developed by Atmel is the ATmega128. A low-power version of this, called ATmega128L, is used in this case. This version is able to run on a lower voltage supply and thus has a lower power consumption. The prize is a lower computational speed. Another solution would be to choose a microcontroller from another vendor. There exist microcontroller vendors that focus on low power consumption and high speed, but Atmel is chosen to have a common development platform for the whole satellite.

The Kalman Filter needs high-accuracy variables in the calculations. These are implemented as floating variables on microcontroller. These can have a range of $\pm 3.403 \cdot 10^{38}$ and an accuracy of $\pm 1.175 \cdot 10^{-38}$.

This microcontroller has the Two-Wire-Interface used to communicate with other subsystems. Much of the control of this interface is done in hardware on the microcontroller, but essential choices involving responds to different states of the interface are handled in software. This will be described in Chapter 8.6. Communication with the magnetometer are also handled in hardware, using the UART interface on the microcontroller. Ensuring correct transmission are handled in hardware, but exceptions, and received data, are left to the software to handle. Finally, the last feature required is the existence of an Analog-to-Digital converter. This is started and configurated in software, the rest is up to the hardware. The completed conversion is presented for the software, which will continue the process of estimation.

Chapter 6

Kalman Filter

A short introduction to Kalman Filter in general is given. The Kalman Filter can take many forms, and each one used in this work is described. Since the Attitude Determination System in our case has very limited computation abilities, a simplified implementation, that will be less demanding for the microcontroller to run, is preferred. Simplifications and how they influence the overall performance is analyzed. The filter is also presented in discrete time.

6.1 Background

R. E. Kalman developed the Kalman Filter in 1960. It has since then proved its usefulness in many applications. It is widely used in the field of navigation. Here, the filter is used to get information about different states of the satellite that we are unable to measure directly.

The filter is another way of formulating the least-square estimation problem (Farrell 1999). It is the states of a dynamic system that is estimated. By assuming stochastic covariance of the state variables, the Kalman Filter minimizes the covariance of the state variable to produce the optimal estimate. When both the measurements and the dynamic process are modeled with noise, the filter weights the difference between the model of the satellite and measurement to produce the estimate.

In addition, the general form of the filter uses the measurement history to calculate the gain, and not only the current measurement. This results in an algorithm that aims to find the best relative weighting between the measurement and the dynamic model of the satellite.

6.1.1 Filter theory

The linear continuous-time system is given by

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{E}\mathbf{w} \tag{6.1}$$

where $\mathbf{A}$ and $\mathbf{B}$ describes the physical model, $\mathbf{w}$ is the process noise assumed to be white, and $\mathbf{E}$ describes which states the noise influences and at what magnitude. The measurement is defined as

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{v} \tag{6.2}$$

where $\mathbf{v}$ is the measurement noise, again the noise is assumed to be white, and $\mathbf{H}$ is the measurement matrix derived from the sensor configuration. The state estimates are then found by the Kalman Filter to be, according to (Fossen 2002):

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\mathbf{u} + \mathbf{K}(\mathbf{y} - \mathbf{H}\hat{\mathbf{x}}) \tag{6.3}$$

$$\mathbf{K} = \mathbf{P}\mathbf{H}^T\mathbf{R}^{-1} \tag{6.4}$$

where $\mathbf{R}$ is the covariance matrix for the measurement noise, and $\mathbf{P}$ is covariance matrix of the states. The ($\hat{}$) denotes estimated states. The matrix $\mathbf{P}$ is found by

$$\dot{\mathbf{P}} = \mathbf{A}\mathbf{P} + \mathbf{P}^T\mathbf{A} + \mathbf{E}\mathbf{Q}\mathbf{E}^T - \mathbf{P}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\mathbf{P}, \tag{6.5}$$

where $\mathbf{Q}$ is the covariance matrix for the process noise.

The last term of Equation 6.3 is called the innovation process, denoted $\boldsymbol{\nu}$. This describes how the estimates are updated by weighting the difference between the real measurement and expected measurement. The latter is in our case found from the different reference models, i.e. the IGRF model and the model of the light that can be experienced in orbit around the Earth.

The problem of maintaining orthogonality of the filter attitude solution is part of all numerical attitude estimation algorithms due to round off. In the Kalman filter the problem is that each time a new estimate is formed, by adjoining new measurements, the constraint on the quaternion must be enforced in order to stay on the quaternion 3-sphere. In (Bar-Itzhack, Markley & Deutchmann 1991) several quaternion normalization algorithms were compared, and it was found that normalization improved filter convergence and accuracy.

The last term of Equation 6.3 describes the conventional innovation process. We have instead used a process that recognizes that the difference between actual and expected measurement is in fact a rotation. The innovation process then becomes:

$$\boldsymbol{\nu} = \mathbf{v}_{meas} \times \mathbf{v}_{ref} \tag{6.6}$$

where $\mathbf{v}_{meas}$ is the actual measurement and $\mathbf{v}_{ref}$ is the expected measurement given by a reference model.

The quaternion estimation part of Equation 6.3 can now be done according to (Psiaki, Martel & Pal 1990):

$$\hat{\mathbf{q}} = \hat{\mathbf{q}}^- \otimes \begin{bmatrix} \Delta\mathbf{q}_{ud} \\ \sqrt{1 - \|\Delta\mathbf{q}_{ud}\|^2} \end{bmatrix} \tag{6.7}$$

using quaternion product, and where $\Delta\mathbf{q}_{ud} = \mathbf{K}_q\boldsymbol{\nu}$ and $\hat{\mathbf{q}}^-$ is the estimate of the quaternion prior to the measurement update. The term $\sqrt{1 - \|\Delta\mathbf{q}_{ud}\|^2}$ to calculate the η in Equation 6.7 preserves the normalization of the quaternion. $\mathbf{K}_q$ is Kalman Filter gain calculated in Equation 6.4. Subscription q indicates that only the part corresponding to the quaternion update is used.

The satellite model, as described in Equation 4.4, gives the angular velocity of Body frame relative to the ECI frame. The angular velocity estimated by the filter will be that of the Body frame with relative to the Orbit frame. That the reference models can be decomposed in Orbit frame and that the measurements are done in Body frame indicates this. It is also this angular

velocity that are used in the controller. The angular velocity of the Body frame relative to ECI frame can be found by Equation 4.6.

The angular velocity estimation has the same innovation process as usual:

$$\hat{\boldsymbol{\omega}}_{OB}^{B} = \hat{\boldsymbol{\omega}}_{OB}^{B-} + \mathbf{K}_{\omega}\boldsymbol{\nu} \tag{6.8}$$

where $\mathbf{K}_{\omega}$ is again the Kalman Filter gain where the subscription ω indicates that only the part corresponding to angular velocity update is used. The innovation process $\boldsymbol{\nu}$ is defined in Equation 6.6.

6.1.2 Incorporating the sensors

The innovation process is on the form as described in Equation 6.6. Both measurements are decomposed in Body frame, and both reference models can be decomposed in Orbit frame. This leads to the idea to use a rotation between the two frames as an innovation process. From the two sensors, we have the following output, Equations 3.18 and 3.24, both in Orbit frame:

$$\mathbf{B}^{O} = \begin{bmatrix} -\sin\nu & \cos\nu & 0 \\ 0 & 0 & 1 \\ \cos\nu & \sin\nu & 0 \end{bmatrix} \mathbf{B}^{OC} \tag{6.9}$$

$$\mathbf{S}^{O} = \begin{bmatrix} \cos\mu & 0 & \sin\mu \\ 0 & -1 & 0 \\ \sin\mu & 0 & -\cos\mu \end{bmatrix} \mathbf{S}^{I}. \tag{6.10}$$

μ is the latitude of the satellite, ν is the true anomaly given by the orbit estimator, and $\mathbf{B}^{OC}$ and $\mathbf{S}^{I}$ are the reference models.

To be able to compare the measurements with their respective references, the reference models must be transformed from Orbit to Body frame using Equation 2.24. This results in the following relationship:

$$\hat{\mathbf{B}}_{ref}^{B} = \hat{\mathbf{R}}_{O}^{B-}\mathbf{B}_{ref}^{O} = \mathbf{R}_{O}^{B}(\hat{\mathbf{q}}^{-})\mathbf{B}_{ref}^{O} \tag{6.11}$$

$$\hat{\mathbf{S}}_{ref}^{B} = \hat{\mathbf{R}}_{O}^{B-}\mathbf{S}_{ref}^{O} = \mathbf{R}_{O}^{B}(\hat{\mathbf{q}}^{-})\mathbf{S}_{ref}^{O} \tag{6.12}$$

where $\hat{\mathbf{q}}^{-}$ is the attitude prediction using the dynamic model of the satellite.

6.2 Extended Kalman Filter

The dynamics and kinematics of the satellite are nonlinear. As are the measurements. Therefore, the Extended Kalman Filter should give the best result. When using the Extended Kalman Filter, a linearization is done around the estimate. If this estimate is far from the actual state, the linearization will be inaccurate, and the estimate may diverge rapidly. Good initial condition must therefore be present in order for Extended Kalman Filter to be of any use at all. However, the estimated states are more likely to be closer to the actual state than a predefined

trajectory. This usually allows the Extended Kalman Filter to give a good performance. The Kalman Filter equations are now described by Equations 6.3 and 6.4, reproduced here:

$$\dot{\hat{\mathbf{x}}} = \mathbf{f}(\hat{\mathbf{x}}, \mathbf{u}) \tag{6.13}$$
$$\dot{\mathbf{P}} = \mathbf{AP} + \mathbf{PA}^T + \mathbf{EQE}^T - \mathbf{PH}^T\mathbf{R}^{-1}\mathbf{HP} \tag{6.14}$$
$$\mathbf{K} = \mathbf{PH}^T\mathbf{R}^{-1} \tag{6.15}$$

where the $\mathbf{A}$ and $\mathbf{H}$ are the linearized model and measurement, respectively. This is done about the at any time best estimate of the states, and are given by:

$$\mathbf{A} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}} \tag{6.16}$$
$$\mathbf{H} = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}} \tag{6.17}$$

where $\hat{\mathbf{x}}$ is the estimate. The matrix $\mathbf{A}$ is given by Equation 4.20 and the result of Equation 6.17 is (Psiaki et al. 1990):

$$\mathbf{H} = \begin{bmatrix} 2S(\hat{\mathbf{B}}^B_{ref}) & \mathbf{0} \\ 2S(\hat{\mathbf{S}}^B_{ref}) & \mathbf{0} \end{bmatrix} \tag{6.18}$$

where $\hat{\mathbf{B}}^B_{ref}$ and $\hat{\mathbf{S}}^B_{ref}$ are the expected measurements given by the two reference models decomposed in Body frame. The resulting innovation process is now:

$$\boldsymbol{\nu}_B = \mathbf{B}_{meas} \times \hat{\mathbf{B}}^B_{ref} \tag{6.19}$$
$$\boldsymbol{\nu}_S = \mathbf{S}_{meas} \times \hat{\mathbf{S}}^B_{ref} \tag{6.20}$$

The Kalman Filter gain is a 6×6 matrix. When using the measurement update expressions in Equations 6.7 and 6.8, this matrix has to be divided up in a quaternion part and an angular velocity part. Considering the measurement matrix in Equation 6.18 and the estimation in Equation 6.3, it is seen that the gain matrix will have the following configuration:

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{q,B} & \mathbf{K}_{q,S} \\ \mathbf{K}_{w,B} & \mathbf{K}_{w,S} \end{bmatrix} \tag{6.21}$$

where each of the four submatrices of $\mathbf{K}$ are a 3×3 matrix.

6.3 Linearized Kalman Filter

The satellite has limited computation powers. A less complex version of the Kalman Filter is therefore wanted to estimate the attitude. A linear version of the Kalman Filter is developed.

A linearized model of the satellite's dynamics and kinematics was developed in Chapter 4.2. In difference from the Extended Kalman Filter where the linearization was done about the best estimate, it is now linearized about a predefined trajectory. This trajectory is given by the

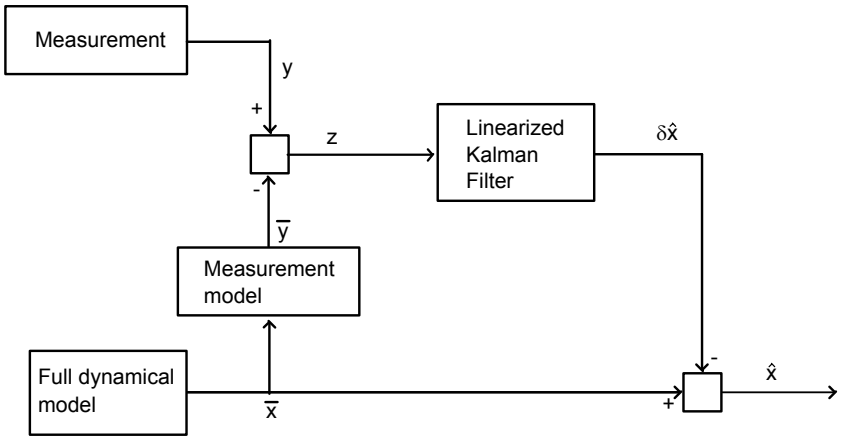

Figure 6.1: The linearized Kalman Filter.

mission of the Attitude Determination and Control System, which is to stabilize the orientation and the spin of the satellite. Thus the trajectory should reflect this and every trajectory will converge towards zero. The principle behind the Linearized Filter is illustrated in Figure 6.1.

The estimation of the perturbation is done according to (Farrell 1999)

$$\delta\hat{\mathbf{x}} = \delta\hat{\mathbf{x}}^{-} + \mathbf{K}(\mathbf{z} - \mathbf{H}\delta\hat{\mathbf{x}}^{-}) \tag{6.22}$$

where the gain matrix are computed in Equation 6.15 and the linearized measurement matrix $\mathbf{H}$ is defined in Equation 6.17, only with $\mathbf{x} = \bar{\mathbf{x}}$ instead of $\mathbf{x} = \hat{\mathbf{x}}$. The superscription $(^{-})$ indicates that the estimates are based only on the process model without use of the sensordata. Again, the update of quaternions will be done as described in Equation 6.7, while the update of angular velocities are done on standard form.

The expression in Equation 6.22 needs, in addition to the current measurements, a prediction of the perturbated states, which is found using the linearized model:

$$\dot{\delta\hat{\mathbf{x}}}^{-} = \mathbf{A}\boldsymbol{\delta}\hat{\mathbf{x}}. \tag{6.23}$$

This will be weighted against sensordata to produce an estimate of the perturbation.

In Equation 6.22, $\mathbf{z}$ is defined as the difference between the expected and real measurement, or mathematically:

$$\mathbf{z} = \mathbf{y} - \bar{\mathbf{y}} \tag{6.24}$$

where $\bar{\mathbf{y}} = h(\bar{\mathbf{x}})$, i.e. the expected measurement on the trajectory, and $\mathbf{y}$ is the actual measurement. Again, since this will be a rotation, Equation 6.24 can be described as:

$$\mathbf{z} = \mathbf{y} \times \bar{\mathbf{y}} \tag{6.25}$$

where the $(-)$ indicates the use of the predefined, expected, trajectory.

The innovation process is now

$$\boldsymbol{\nu} = \mathbf{z} \times \mathbf{H}\boldsymbol{\delta}\hat{\mathbf{x}}^{-}, \tag{6.26}$$

again by recognizing the process as a rotation.

The linearized filter is designed to estimate the perturbation of the states from the trajectory, not the total state. These total states can be found by adding the perturbation to the trajectory, as:

$$\hat{\mathbf{x}} = \bar{\mathbf{x}} + \delta\hat{\mathbf{x}}, \tag{6.27}$$

where $\bar{\mathbf{x}}$ is the trajectory and $\delta\hat{\mathbf{x}}$ the estimation of perturbation from the trajectory. Again, the use of quaternion product is necessary to add, or rotate, the perturbated attitude, with the trajectory. This will ensure normalized quaternions, and is done

$$\hat{\mathbf{q}} = \bar{\mathbf{q}} \otimes \delta\hat{\mathbf{q}}, \tag{6.28}$$

where $\delta\hat{\mathbf{q}}$ is the estimated perturbation from the trajectory $\bar{\mathbf{q}}$.

6.4 Discrete Kalman Filter

As the Kalman Filter eventually is going to be implemented on a processor, the continuous-time model has to be converted to discrete time. It is only the final, linearized version of the filter with best performance, that is discretizised. The discrete Kalman Filter has also some good qualities when used with sensors with different sample times. One sensor with low sampling rate, often with a bias, can generate the trajectory which the system is linearized about, and the second sensor, with higher sampling rate, will then estimate the perturbation from this trajectory. This is, however, not utilized in our filter because this will further complicate the filter implementation on the microcontroller.

The discrete filter will not have the same performance as the continuous filter. The results gathered from the continuous case can be used as a guideline, but further improvements and adjustments on the Kalman Filter parameters must be done to get to the best result.

The system equation is now

$$\mathbf{x}_{k+1} = \boldsymbol{\Phi}_k\mathbf{x}_k + \boldsymbol{\Delta}\mathbf{u}_k + \boldsymbol{\Gamma}\mathbf{w}_k \tag{6.29}$$

where $\boldsymbol{\Phi}$ is the state transition matrix computed from the continuous-time system matrix, $\boldsymbol{\Delta}$ is the discrete input matrix and $\boldsymbol{\Gamma}$ is the discrete noise configuration matrix.

The Discrete Kalman Filter is a two-part procedure. First a prediction of the states is done. This is based on the equations given by the dynamics and kinematics of the satellite. Filtering is then done on the measurements and combining this with the predictions, an estimate can be computed.

6.4.1 Prediction

The prediction part of the filter uses a state transition matrix to predict the states of the model:

$$\delta\hat{\mathbf{x}}_{k+1}^{-} = \boldsymbol{\Phi}_k\boldsymbol{\delta}\hat{\mathbf{x}}_k \tag{6.30}$$

Predictions are done a priori, that is before any measurements are done at that specific timestep. To indicate that an estimate is only a prediction, the superscription $(-)$ is used. The covariance matrix $\mathbf{P}_k$ is in the discrete case (Farrell 1999):

$$\mathbf{P}_{k+1}^{-} = \mathbf{\Phi}_k \mathbf{P}_k \mathbf{\Phi}_k^T + \mathbf{Q}_k \tag{6.31}$$

where $\mathbf{Q}_k$ is the discrete version of the covariance matrix for the process noise, $\mathbf{Q}$. This can be found using an algorithm outlined in (Brown 1997) The algorithm will also give the discrete state transition matrix $\mathbf{\Phi}_k$.

1. First, form a $2n \times 2n$ matrix that we call $\mathbf{A}_d$, where n is the dimension of $\mathbf{x}$.

$$\mathbf{A}_d = \begin{bmatrix} -\mathbf{A} & \mathbf{G}\mathbf{Q}\mathbf{G}^T \\ \mathbf{0} & \mathbf{A}^T \end{bmatrix} \Delta t \tag{6.32}$$

2. Form e^{A_d}, and call it $\mathbf{B}_d$.

$$\mathbf{B}_d = \exp(\mathbf{A}_d) = \begin{bmatrix} \cdots & \mathbf{\Phi}_k^{-1}\mathbf{Q}_k \\ \mathbf{0} & \mathbf{\Phi}_k^T \end{bmatrix} \tag{6.33}$$

   (upper-left partition of $\mathbf{B}_d$ is of no concern here)

3. Transpose the lower-right partition of $\mathbf{B}_d$ to get $\mathbf{\Phi}_k$.

$$\mathbf{\Phi}_k = \text{transpose of lower-right part of } \mathbf{B}_d \tag{6.34}$$

4. And finally, $\mathbf{Q}_k$ is obtained from the upper-right partition of $\mathbf{B}_d$ as follows:

$$\mathbf{Q}_k = \mathbf{\Phi}_k \cdot (\text{the upper-right part of } \mathbf{B}_d) \tag{6.35}$$

6.4.2 Update

Now that we have all the estimates a priori we can incorporate these with the measurements. Both the covariance matrix and the state estimations are updated. The state estimations are updated using

$$\delta\hat{\mathbf{x}}_k = \delta\hat{\mathbf{x}}_k^- + \mathbf{K}(\mathbf{z}_k - \mathbf{H}_k\delta\hat{\mathbf{x}}_k^-) \tag{6.36}$$

where $\mathbf{z}_k$ is defined as

$$\mathbf{z}_k = \mathbf{y}_k \times \bar{\mathbf{y}}_k, \tag{6.37}$$

and describes the error between the the expected measurement and the actual measurement. In the case of the quaternion update, the measurement update should be done as

$$\delta\hat{\mathbf{q}}_k = \delta\hat{\mathbf{q}}_k^- \otimes \begin{bmatrix} \Delta\mathbf{q}_{ud} \\ \sqrt{1 - \|\Delta\mathbf{q}_{ud}\|^2} \end{bmatrix}, \tag{6.38}$$

where $\Delta\mathbf{q}_{ud} = \mathbf{K}_q\boldsymbol{\nu}$. In the case of angular velocities, the standard form is used:

$$\delta\boldsymbol{\omega}_{OB}^B = \delta\boldsymbol{\omega}_{OB}^{B-} + \mathbf{K}(\mathbf{z}_k - \mathbf{H}_k\delta\boldsymbol{\omega}_{OB}^{B-}) \tag{6.39}$$

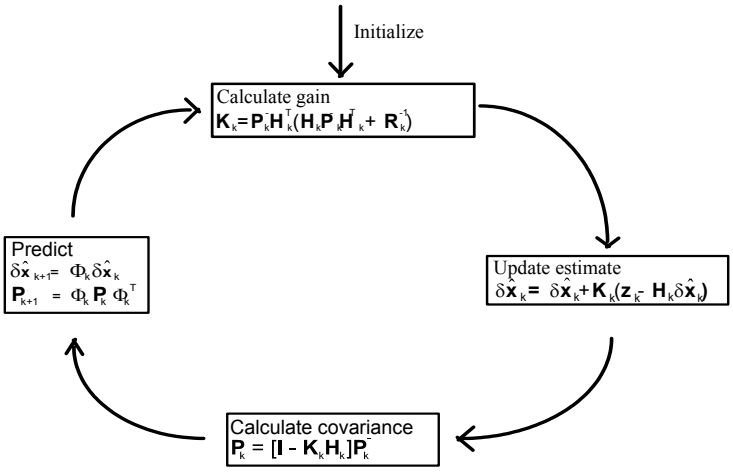

Figure 6.2: The discrete Kalman Filter. After each update of the estimates, these are combinded with a predefined trajectory to form the full state estimates.

The updating of the covariance matrix can be done in a numerous ways (Farrell 1999), one of them being

$$\mathbf{P}_k = [\mathbf{I} - \mathbf{K}_k\mathbf{H}_k]\mathbf{P}_k^- \tag{6.40}$$

The Kalman gain can now be computed according to

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T[\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}]^{-1} \tag{6.41}$$

where $\mathbf{R}$ is the covariance matrix for the measurement noise.

In addition, the filter needs initialization. Both the state estimate and the covariance matrix need initial values defined by:

$$\delta\hat{\mathbf{x}}_0^- = \bar{\mathbf{x}}(0) \tag{6.42}$$
$$\mathbf{P}_0^- = var[\delta\hat{\mathbf{x}}_0^-] \tag{6.43}$$

where both values are calculated offline and fed to the Kalman Filter at startup.

The discrete Kalman Filter can be summarized in Figure 6.2.

6.4.3 Offline calculations

Given the trajectory, some of the calculations needed for the Kalman Filter can be done off-line. This will reduce the load on the processor even more. These calculations can either be stored as a look-up table or be implemented with a curvefit algorithm. The trajectory of the states must be stored, using the same method. The calculations that can be done prior to launch, the covariance matrix $\mathbf{P}$, and the gain matrix $\mathbf{K}$, are:

$$\mathbf{P}_{k+1}^{-} = \mathbf{\Phi}_k \mathbf{P}_k \mathbf{\Phi}_k^T + \mathbf{Q}_k \tag{6.44}$$

$$\mathbf{P}_k = [\mathbf{I} - \mathbf{K}_k \mathbf{H}_k] \mathbf{P}_k^{-} \tag{6.45}$$

$$\mathbf{K}_k = \mathbf{P}_k^{-} \mathbf{H}_k^T [\mathbf{H}_k \mathbf{P}_k^{-} \mathbf{H}_k^T + \mathbf{R}]^{-1}, \tag{6.46}$$

where $\mathbf{A}$ and $\mathbf{H}$ are calculated using Equations 6.16 and 6.17, respectively, only with $\mathbf{x} = \bar{\mathbf{x}}$ instead of $\mathbf{x} = \hat{\mathbf{x}}$. Calculations in Equations 6.44 and 6.46 are presented in continuous-time, but this is only used for analysis purposes. When implementing the filter in the final application, all calculations are done in discrete-time.

The linearized discrete Kalman Filter uses several predefined trajectories. These are stored in the final application as curvefitted trajectories. In addition to the trajectories for the states and the elements in the gain matrix, the two measurement trajectories, each consisting of measurements in three dimensions, are also stored as curvefitted trajectories. As the state transition matrix and measurement matrix vary with their respective trajectories, these are calculated online. Only the expression for each matrix are stored. These are:

$$\mathbf{\Phi}_k = e^{\mathbf{A} \cdot \Delta t} \approx \mathbf{I} + \mathbf{A} \cdot \Delta t \tag{6.47}$$

$$\mathbf{H}_k = \begin{bmatrix} 2S(\bar{\mathbf{B}}_k) & \mathbf{0} \\ 2S(\bar{\mathbf{S}}_k) & \mathbf{0} \end{bmatrix} \tag{6.48}$$

where $\mathbf{A}$ is the linearized model of the satellite and Δt the sampling time, and where $\bar{\mathbf{S}}_k$ and $\bar{\mathbf{B}}_k$ are the expected measurements at timestep k. Both measurement-trajectories are dependent on the initial position of the satellite in the Earth-Centered Earth-Fixed frame. In addition, the trajectory of the magnetic field depends on what orbit the satellite will take, i.e. the Keplerian Elements. These are, except for an approximate value for inclination, unknown prior to launch. Thus it will be necessary to upload these trajectories post launch.

The dynamics of the satellite is very slow. This is because the limited magnetic field the actuators can create. The stabilizing phase of the satellite will in effect take up to 5000 seconds before accomplish it's mission, which is to reach near-zero angular velocities and small angles in orientation. If the actuators were more efficient, the trajectories of the states would reflect this by having a constant value of zero angular velocities and zero angles.

But as this is not the case, simulations, creating an expected trajectory of the states, have to be performed. With this, the measurement trajectories, both for the magnetic field and the light, are also created. The mission of the filter will then be to estimate the perturbation from this trajectory.

Chapter 7

Simulation

The estimator was first implemented as an Extended Kalman Filter in continuous-time. It was assumed that this would give the best performance, and thus be the reference for other, less accurate approaches. The behavior of the satellite is dependent on the initial condition after detumbling phase. As this unknown, a predefined trajectory would almost certain be incorrect. While the Linearized Kalman Filter can estimate small perturbations from the trajectory, the Extended Kalman Filter would show a better performance.

The model is implemented with noise on measurement as discussed in Chapter 6.1.1. To simulate the sensors, each of the reference models is the rotated from Orbit to Body frame using the orientation of the satellite model, see Figure 7.1. Noise is added with a magnitude corresponding to the covariance of the respective sensor. These simulated measurements are then used in the Kalman Filter.

When the Kalman Filter is initiated, the satellite will have small angular velocities and small angles in orientation. They will be unknown, and only a maximum absolute value is assumed achieved after the detumbling phase is completed. It is therefore necessary for the filter to perform well even without good initial estimates. The covariance matrix for the states will also need initial values. These can be calculated using (Farrell 1999):

$$\mathbf{P}_0 = var[\mathbf{x}_0] \tag{7.1}$$

where $\mathbf{x}_0$ is the different initial values the states of the satellite can have.

In every simulation, the initial position of the satellite is assumed to be in 0° in both

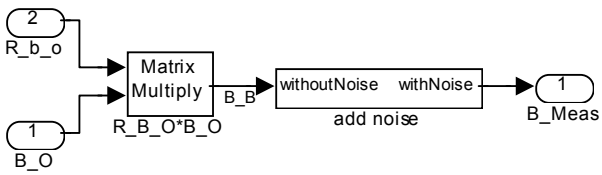

Figure 7.1: Simulation of the measurement of the magnetic field.

longitude and latitude. The performance of the different estimators are independent of this initial position in Earth-Centered Earth-Fixed frame, but in the finale implementation, these angles must reflect the real initial position, as the reference models depend on this. The angular velocity of the Orbit frame is assumed the be $\omega_0 = 0.0011$, as the orbit will have an altitude of approximately 700km.

The initial values of the satellite, both in orientation and the angular velocity are assumed to be small because of the detumbling phase. The detumbling algorithm uses information of the change in magnetic field to dissipate the kinetic energy of the satellite (Fauske 2002):

$$\mathbf{m}^b = -k\dot{\mathbf{B}}^b - \mathbf{m}_c, \tag{7.2}$$

where $\mathbf{m}^b$ is the is the magnetic dipole moment generated by the coils, $\dot{\mathbf{B}}^b$ is the change in magnetic field and $\mathbf{m}_c = \begin{bmatrix} 0 & 0 & m_c \end{bmatrix}^T$. The magnetic field is approximately constant near the poles (Fauske 2002),

$$\begin{aligned} \dot{\mathbf{B}}^B &\approx \mathbf{B}^B \times \boldsymbol{\omega}_{IB}^B, & (7.3)\\ &= S(\mathbf{B}^B)\boldsymbol{\omega}_{IB}^B & (7.4)\\ \boldsymbol{\omega}_{IB}^B &= \dot{\mathbf{B}}^B S(\mathbf{B}^B)^{-1} & (7.5) \end{aligned}$$

where $\mathbf{B}^B$ is the measured magnetic field, and $\dot{\mathbf{B}}^B$ can be found by measuring the field over a short timeperiod and use the definition of differentiation,

$$\dot{f} = \frac{f(t + \Delta t) - f(t)}{\Delta t}, \tag{7.6}$$

on these measurements. This can thus give an approximately value of the angular velocity near the poles, and be used as initial value of the Kalman Filter. This introduces some restriction on when the estimation can be started, but as the satellite uses about 90 minutes on one orbit, it is not too strict.

7.1 Extended Kalman Filter

There is no need for information calculated prior to launch when implementing this approach - everything is calculated online. This is also the reason why it is seen as to be too computational demanding to use on board the nCube. It does, however, prove to be a useful reference when the performance of the linearized filter, and later the discretizised filter, is discussed.

The filter is initialized with 0° in roll, pitch and yaw. By using Equation 7.5 online in the satellite, the filter can be initialized with angular velocities close to the actual ones, which is reflected in the simulations by assuming that the angular velocities are known with a up to 10 percent error.

The performance of the Extended Kalman Filter can be studied in Figure 7.2. The estimated attitude follows the actual attitude well, only with small deviations. It is seen that the estimation of the angular velocities is good. Figure 7.3 also shows the estimated attitude, represented using euler angles to get a better idea of the attitude. The jump in roll around 1300 seconds can be explained by the fact that euler angles are defined as to be between ±180°.

Figure 7.4 shows the estimation error in attitude and angular velocity. The error in angular velocities are acceptable. When considering the attitude, the error in estimation is varying

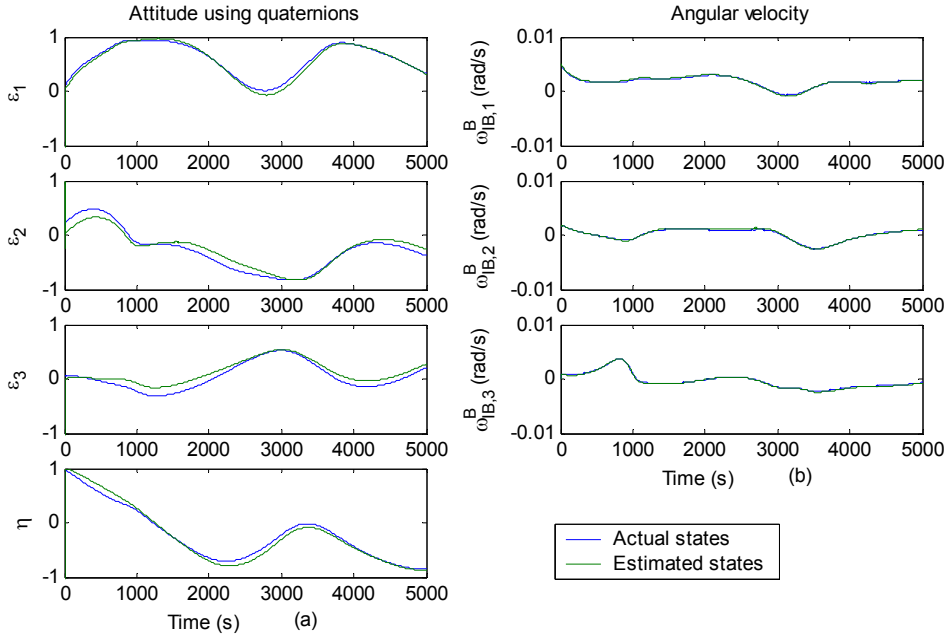

Figure 7.2: Estimation (a) of attitude represented using quaternions, with initial value of $\mathbf{q}_0 = [-0.0704 \;\; 0.1798 \;\; 0.0704 \;\; 0.9786]^T$. And estimation (b) of angular velocities, with initial value of $\boldsymbol{\omega}^B_{IB,0} = [0.0048 \;\; 0.0019 \;\; 0.0001]^T$.

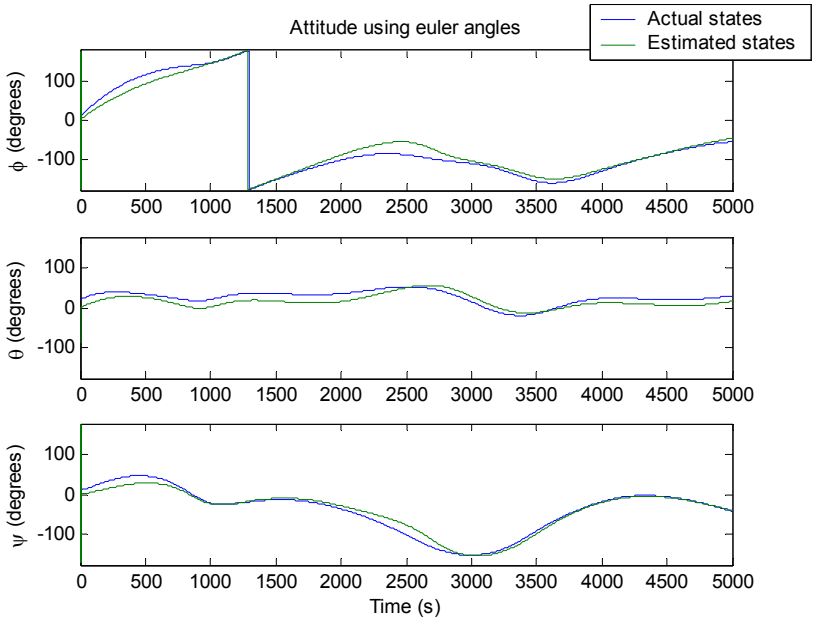

Figure 7.3: Estimation of attitude represented using euler angles. Initial orientation of the satellite is 10°, 20° and 10°.

somewhat, but overall, the error is acceptable. The pike in roll error just before 1300 seconds can again be explained by the definition of euler angles.

The estimated states can now be used by the controller to stabilize the satellite, see Figure 7.5. The controller has a good performance also when using estimated states. The plot shows the third direction cosine, as defined in Equation 2.25, and indicates that z_O and z_B-axes are aligned when the value is ± 1.

The original implementation of the filter showed some symptoms of bad behavior. The prediction would in some cases be too far of the actual state. When filtering the measurements, this process would try to compensate and result in a value of $\|\Delta\mathbf{q}_{ud}\|^2$ in Equation 6.7 greater than one, and thus the process would return a complex number. To overcome this problem the following measurement update algorithm were implemented (Psiaki et al. 1990):

$$\hat{\mathbf{q}} = \frac{\hat{\mathbf{q}}^- \otimes \begin{bmatrix} \Delta\mathbf{q}_{ud} \\ 1 \end{bmatrix}}{\sqrt{1 + \|\Delta\mathbf{q}_{ud}\|^2}}, \tag{7.7}$$

where $\Delta\mathbf{q}_{ud} = \mathbf{K}_q\boldsymbol{\nu}$.

7.2 Linearized Kalman Filter

The nonlinear system dynamics and kinematics are linearized as described in Chapter 4.2. The linearized filter estimates a perturbation from a predefined trajectory. This trajectory is found

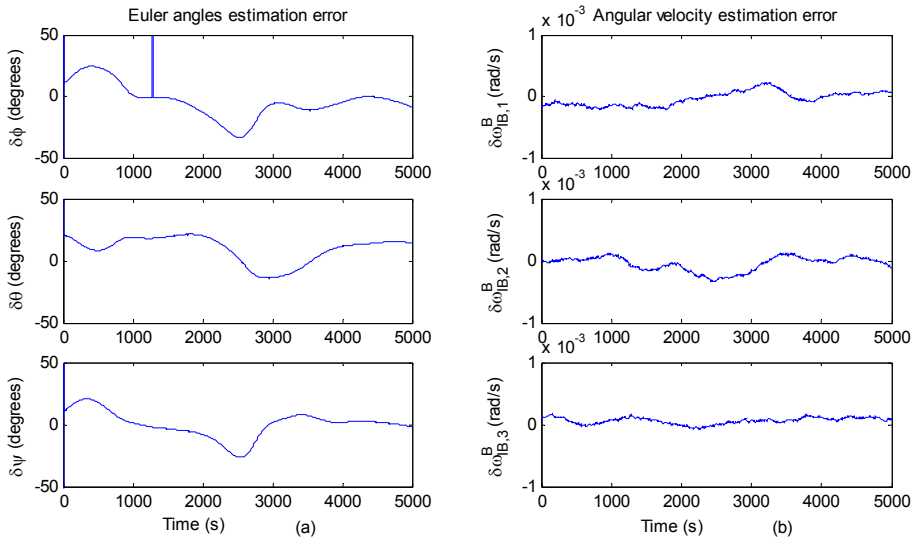

Figure 7.4: Estimation error of (a) the attitude, and (b) angular velocities.

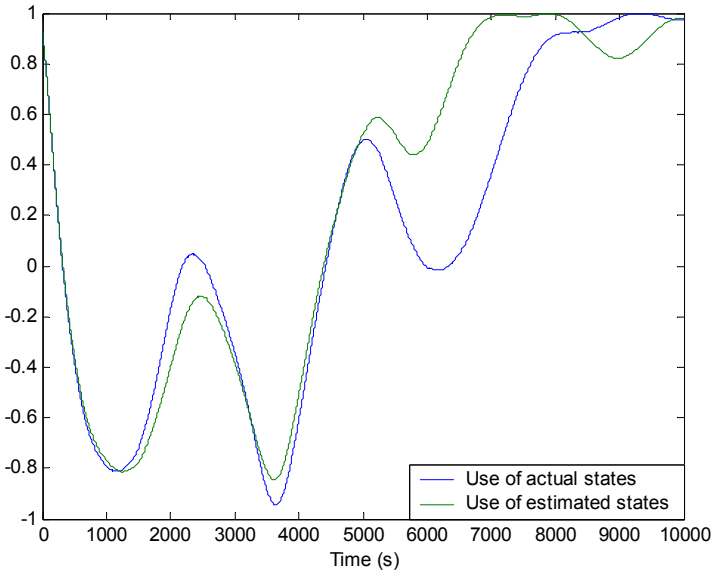

Figure 7.5: The initial orientation is 10°, 20° and 10° in roll, pitch and roll, respectively.

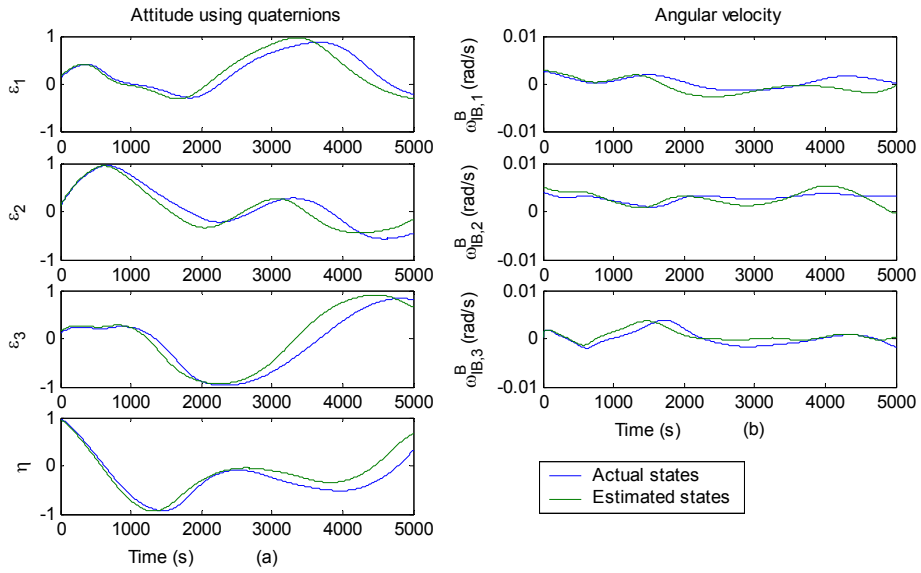

Figure 7.6: Estimation (a) of attitude represented using quaternions, with initial value of $\mathbf{q}_0 = [0.1176 \quad 0.1026 \quad 0.1176 \quad 0.9807]^T$. And estimation (b) of angular velocities, with inital value of $\boldsymbol{\omega}^B_{IB,0} = [0.0026 \quad 0.0040 \quad 0.0003]^T$.

by simulating the system offline and store the resulting trajectory in memory. A second set of the dynamics and kinematics of the satellite are initialized with different values than the first set to simulate the trajectory. The gain matrix is also calculated offline, and thus measurement data will have no effect on this calculation.

A trajectory is calculated with initial orientation of 20°, 10° and 20° in roll, pitch and yaw, respectively, and initial angular velocity: $\boldsymbol{\omega}^B_{OB,0} = [0.003 \quad 0.005 \quad 0.001]^T$ rad/s. The satellite is simulated with small initial perturbations from this trajectory, and the filter assumes zero perturbation in orientation when initiated. The initial perturbation of the angular velocities $\boldsymbol{\omega}^B_{IB,0}$ can be found to by using the difference between the result from Equation 7.5, the calculation of $\dot{\mathbf{B}}$, and the initial value of the trajectory. As the trajectory gives the velocity of Body frame relative to Orbit frame, a transformation is needed:

$$\delta\boldsymbol{\omega}^B_{OB,0} = \boldsymbol{\omega}^B_{IB,0} - \bar{\boldsymbol{\omega}}^B_{IB,0} - \mathbf{R}^B_O\boldsymbol{\omega}^O_{IO}, \tag{7.8}$$

where $\mathbf{R}^B_O$ is found using the initial value of the trajectory of the orientation.

As seen on Figure 7.6, the estimated orientation lies ahead in time of the actual orientation. The estimated states follows the actual states up to 2000 seconds, but then shows symptoms of diverging. The estimated angular velocity is also good up to 2000 seconds, and then shows the same symptoms as in the case of the orientation.

Figure 7.7 shows the performance of the filter represented using Euler Angles, and Figure 7.8 shows the estimation error. Both in roll and pitch the filter shows acceptable performance, but in yaw, that is the rotation about the z-axis, the estimation is not good. The error in angular velocity begins with a relative good performance, but as time passes, the estimation is

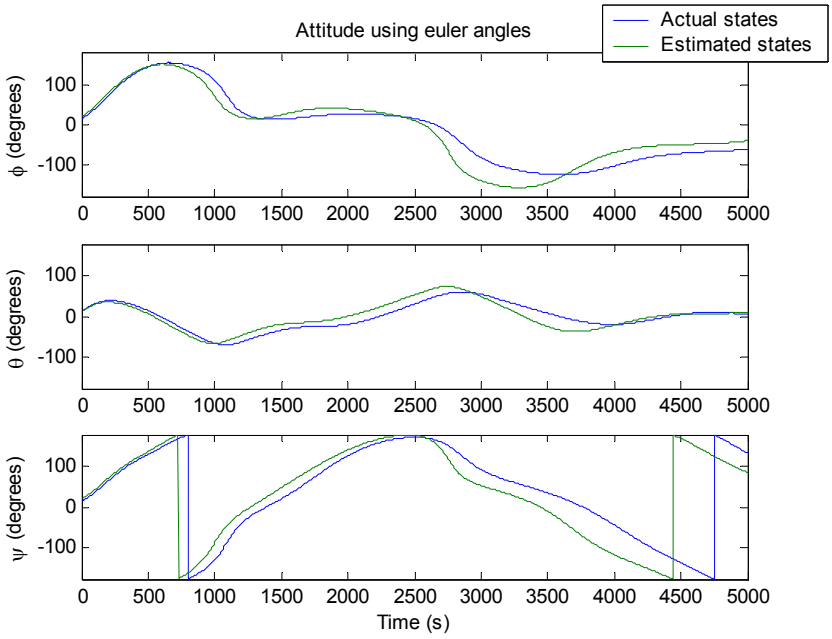

Figure 7.7: Estimation of attitude represented using euler angles. Initial orientation of the satellite is 15°, 10° and 15°.

seen to show symptoms of diverging.

Two elements in the gain matrix can be seen on Figure 7.9. In the case of the quaternion update, the gain is varying some, but the value is always less than 0.015. This means that the measurements will have little influence on the final estimation. When the predicted perturbation also is very small, the calculation of $\mathbf{z}_k - \mathbf{H}_k \delta\hat{\mathbf{x}}_k^-$ will be small, and the result will be that the full estimated state will be close to the trajectory, even when this is different from the actual state. The element in the gain matrix involved in the angular velocity update, give the innovation process some initial influence on the estimates. After some time, about 150 seconds, the element reaches a value close to zero, and the innovation process will no longer have any influence on the estimation of the angular velocities. A predicted perturbation in angular velocities has no effect on $\mathbf{z}_k - \mathbf{H}_k \delta\hat{\mathbf{x}}_k^-$, because corresponding elements in the measurement matrix are zero. The measurement will thus no effect in updating the angular velocities, if only the angular velocities differ from the trajectory.

7.3 Discrete Kalman Filter

The linearized filter is now simulated using discrete timesteps. The covariance matrix of the states is implemented differently than that of the continous-time, as is the system model. The two sensors are assumed to have the same sampling rate. The filter, and the sensors, are simulated with a sampling rate of 0.25 seconds. As with the linearized filter, the discrete filter uses trajectories of the states of the satellite and the measurements, and is implemented with

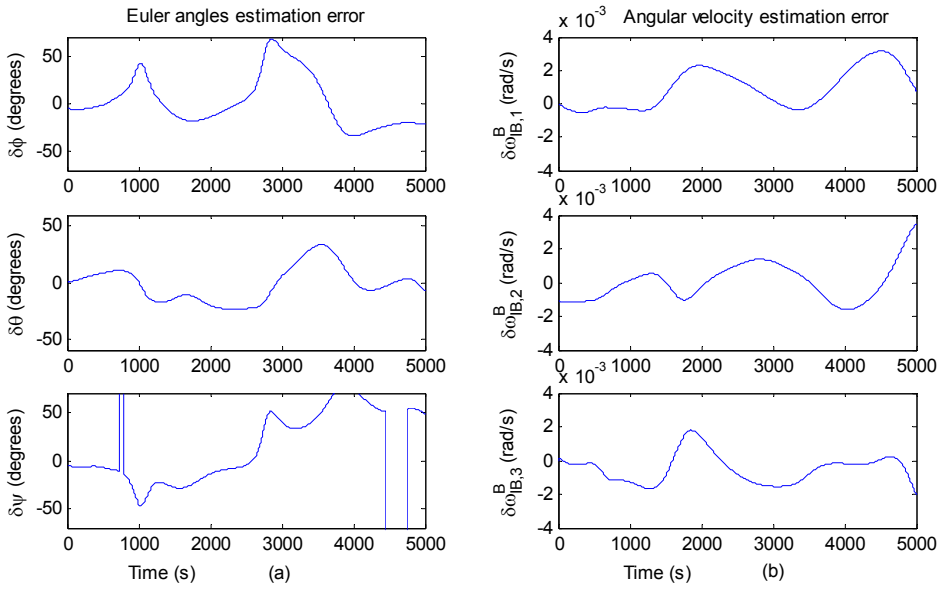

Figure 7.8: Estimation error of (a) the attitude, and (b) angular velocities.

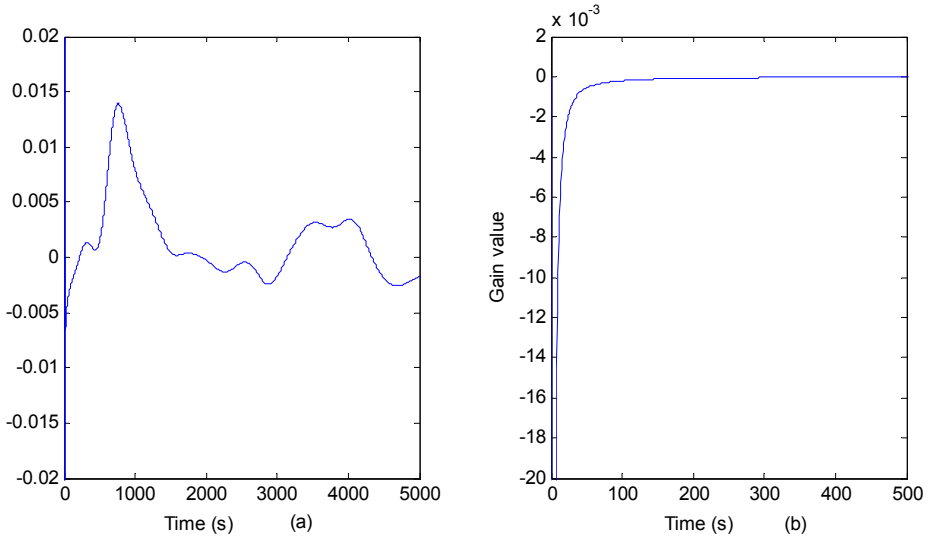

Figure 7.9: Two elements of the gain matrix, (a) is element (1,1) and (b) is element (4,4). Notice range on the x-axis in the latter plot.

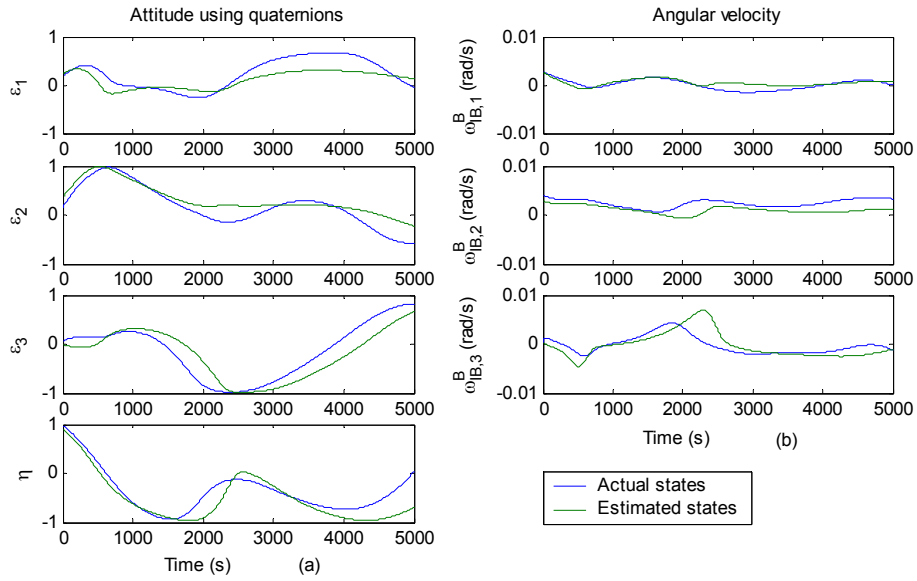

Figure 7.10: Estimation (a) of attitude represented using quaternions, with initial value of $\mathbf{q}_0 = [0.0625 \quad 0.1431 \quad 0.1603 \quad 0.9746]^T$. And estimation (b) of angular velocities, with inital value of $\boldsymbol{\omega}^B_{IB,0} = [0.0028 \quad 0.0040 \quad 0.0003]^T$.

a second set of the satellite model, having another initial state. The trajectory has an initial attitude of 30°, 40° and 10° in roll, pitch and yaw, respectively. The angular velocity of the Body frame relative to Orbit frame is set to be $\boldsymbol{\omega}^B_{OB,0} = [0.003 \quad 0.005 \quad 0.001]^T$, and Equation 7.8 is used to initiate the filter with a good initial guess of the angular velocity. The initial perturbation form the attitude-trajectory is in the filter assumed to be 0° in roll, pitch and yaw.

In Figure 7.10, it can be seen that the performance of the discrete filter is much like that of the linearized filter. The estimation begins well with little error, but as time passes, the estimation starts to drift off compared to the actual state. This is true both for the estimated attitude and angular velocities.

When considering Figure 7.11, it is seen that both roll and pitch are estimated with an acceptable accuracy. Yaw, however, is not acceptable estimated. Even when ignoring the jump from 180° to −180°, the estimation in yaw is not good. It is seen that the performance is of such nature, that the results should not be used in by a controller.

By altering the expected initial orientation, that of the trajectory, to 15°, 10° and 15°, it is assumed that the performance would be better. The results are shown in Figures 7.12 through 7.14. Still, the estimations drift away after some time. The error can be explained as it was with the linearized filter. The gain in the discrete case is the same as that of the continuos case. That is, the gain is close to zero, and thus errors between expected and real measurements are not correcting the prediction. Only when large perturbation are predicted by the filter will the innovation process correct the predication to some extent.

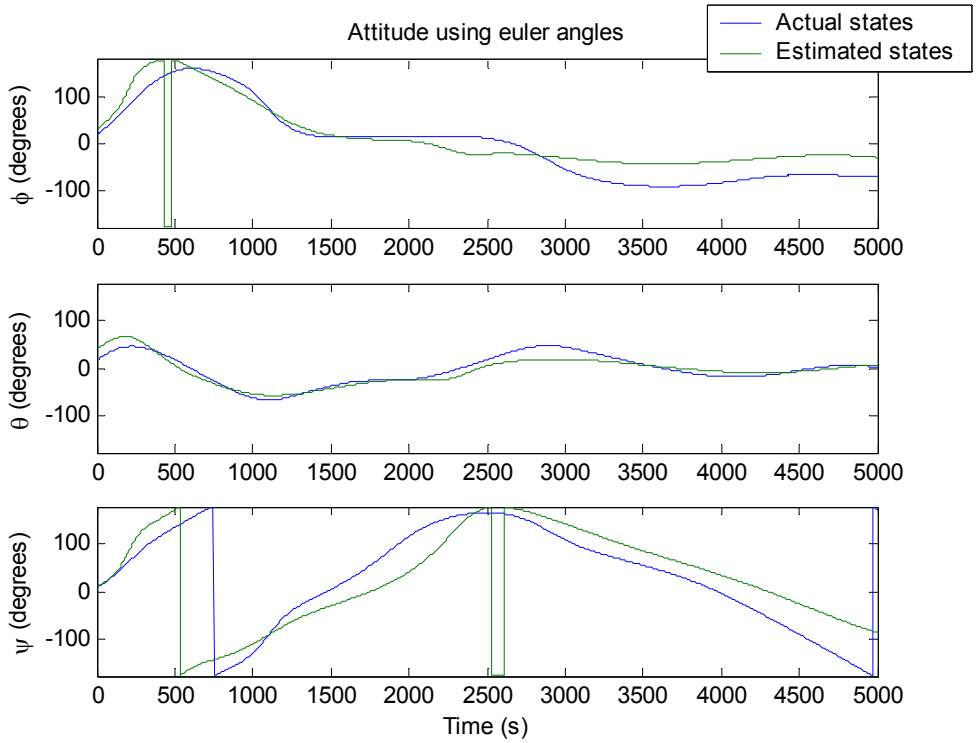

Figure 7.11: Estimation of attitude represented using euler angles. Initial orientation of the satellite is 20°, 15° and 10°.

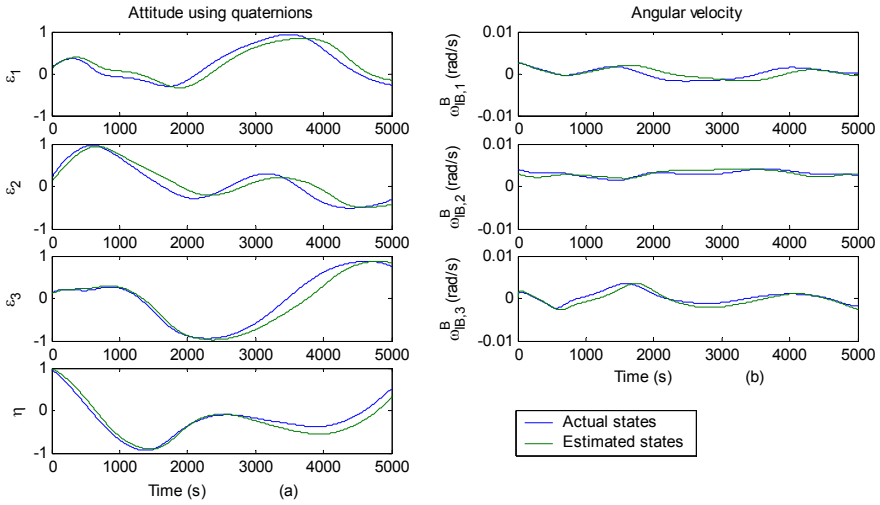

Figure 7.12: Estimation (a) of attitude represented using quaternions, with initial value of $\mathbf{q}_0 = [0.1387 \quad 0.1981 \quad 0.1387 \quad 0.9603]^T$. And estimation (b) of angular velocities, with inital value of $\boldsymbol{\omega}^B_{IB,0} = [0.0027 \quad 0.0040 \quad 0.0002]^T$.

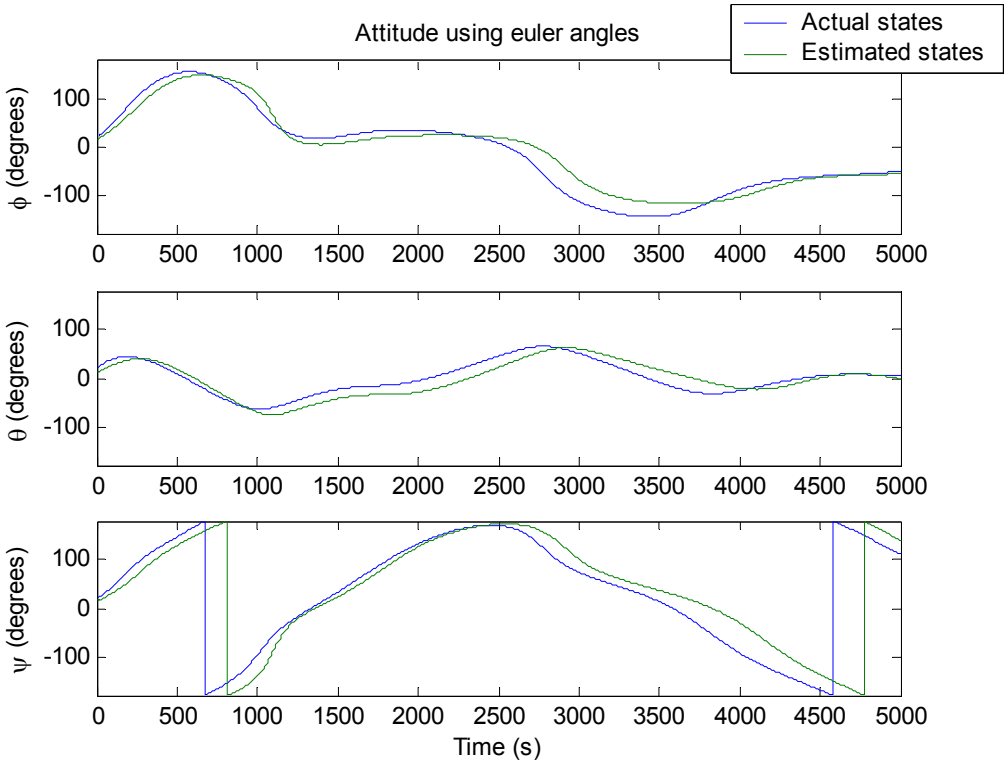

Figure 7.13: Estimation of attitude represented using euler angles. Initial orientation of the satellite is 20°, 15° and 10°.

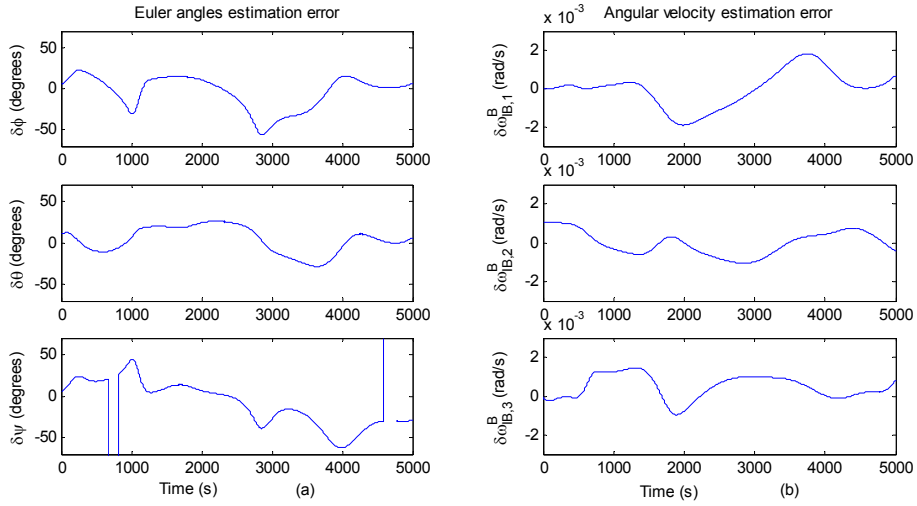

Figure 7.14: Estimation error of (a) the attitude, and (b) angular velocities.

Chapter 8

Implementation

The Discrete Kalman Filter is now implemented on a microcontroller. Processes for interaction with the other subsystems and the sensors are also implemented. As the estimator uses multi-dimensional mathematical operations, a library was made that performs these operations. This is used by the estimator when needed.

8.1 Introduction

The software is developed in the programming language C, using IAR Embedded Workbench. This compiler is designed to use with the Atmel microcontroller, and has some features that ease the implementation on these microcontrollers.

Dataflow charts, and some state transition diagrams, are used to describe the behavior of the estimator. The notation used to most efficient document this behavior is described in Figure 8.1.

In addition, the description of the different processes implemented on the microcontroller is built up according to:

- The process name in the dataflow chart, followed by a number to indicate the parent process, if any.
- The name the actual process name as coded in C; processName(the parameters)
- The parameters, if any.
- A description of the behavior of the process.

8.2 Top view

The system, and the external interactors, can be seen in Figure 8.2. The magnetometer receives and transmits messages using RS232, the microcontroller uses the built-in Analog-to-Digital Converter to interact with the light dependent resistors and the interface between the Attitude Determination System and the other subsystems are taken care of by the Two-Wire-Interface, TWI. In the "estimated states"-message from the Attitude Determination System are also the

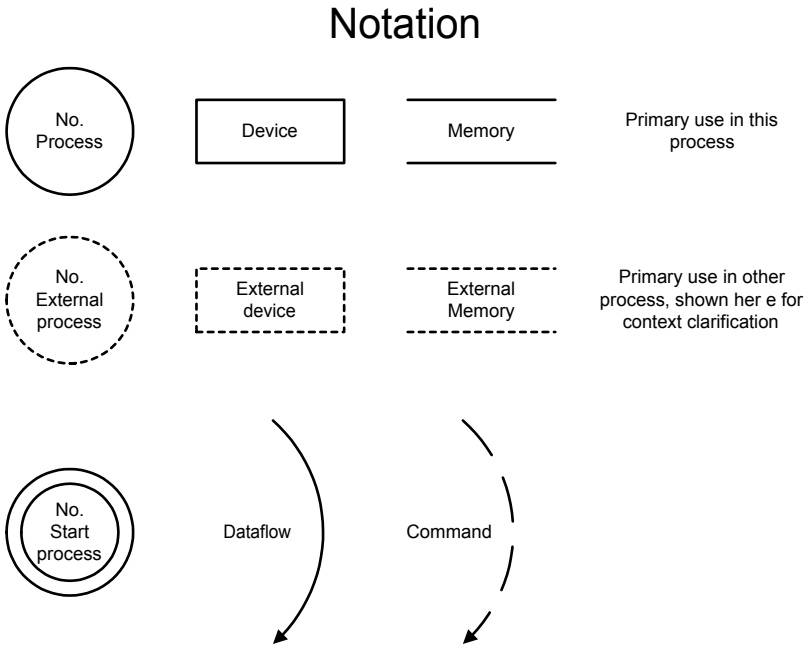

Figure 8.1: The notation used in the following dataflow charts.

TWI-address for the recipient included, in addition to the states. This address can either be that of the Ground Communication System or the Attitude Control System.The estimator, with it's ability to interface with peripheral devices, consists of four main processes, see Figure 8.3. These are:

- Waiting. This process controls the entire system. During the detumbling phase, the estimator is idle, and waiting for the initiating message. The task starts the estimation at given intervals.
- Initiate system. The first thing that has to be done is to set up the different devices connected to the microcontroller and also the microcontroller itself. Parameters for setting up the devices are stored in memory.
- Estimate state. The Kalman Filter is implemented in this process. It gets it's data from the to external sensors, and the different trajectories from memory. This process will also initiate transmission of the estimated states to the control system.
- Communicate. Control of the Two-Wire-Interface. Messages are prepared for transmission and incoming messages are interpreted.

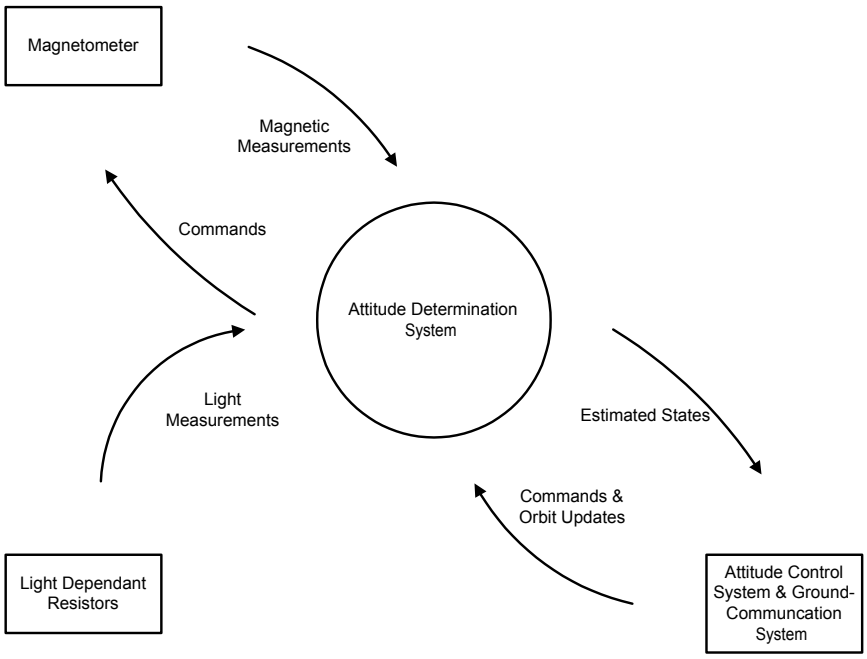

Figure 8.2: The estimator with sensors and other subsystems of the satellite.

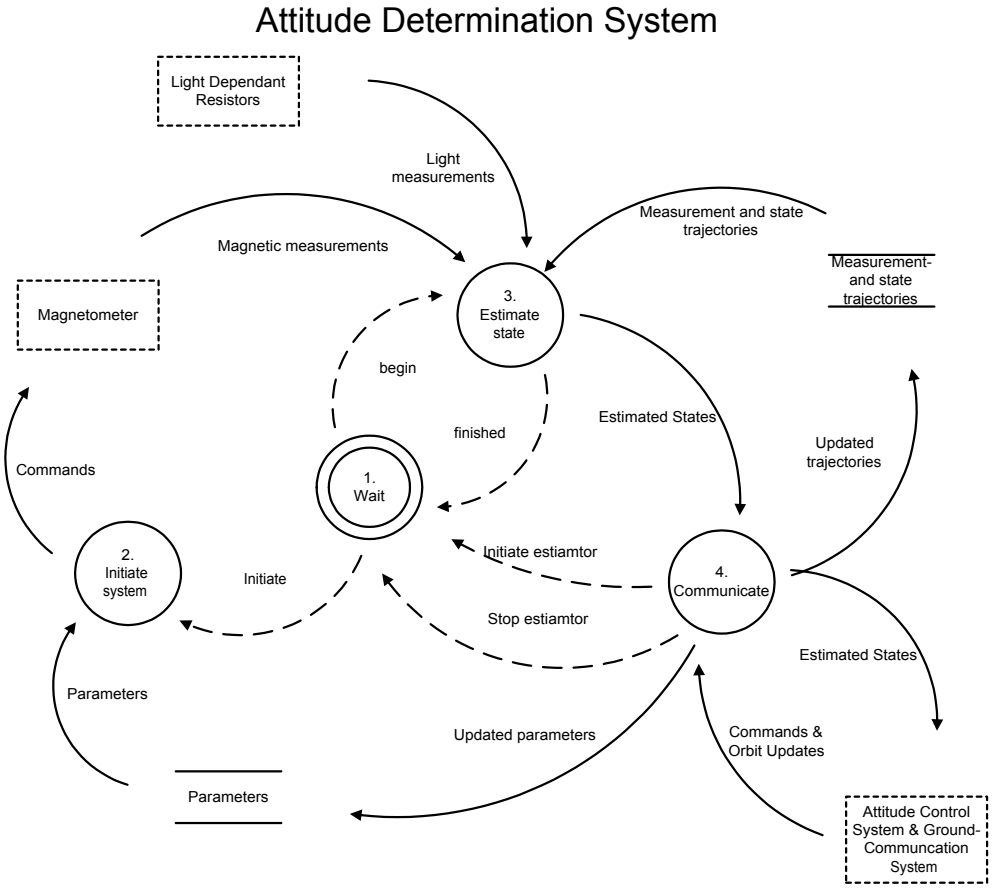

Figure 8.3: Top level view of the estiamtor with the different support processes.

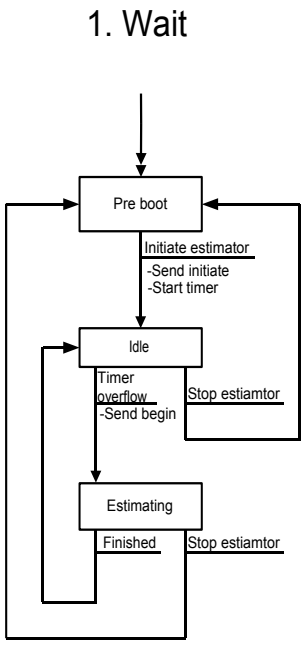

Figure 8.4: The wait process, with the different signals that can be sent.

8.3 The Wait process

This process is the first process that is run after bootup. It will wait for commands from the ground station, see Figure 8.4. Depending on the current state, this process will either initiate configuration of the estimator or start the estimation.

8.4 The Initiate system process

When called, this process will initiate four subroutines, each in turn will configure a device. These four subroutines are, see Figure 8.5:

- Load Kalman Filter constants (2.1); initKalmanF(). Load the Kalman Filter constants so they can be used by every process on the microcontroller.
- Startup magnetometer (2.2): initMag(). Configures the interface with the magnetometer and ensures that the magnetometer are running in wanted mode. That is, with averaging turned on and with continuos output.
- Set up communication - LDR (2.3): initADC(). Initiates the Analog-to-Digital converter. Registers are set according to the wanted behavior of the converter.
- Set up communication - TWI (2.4): initTWI(). Initiates the TWI. Setting speed of communication and which TWI-address to respond to.

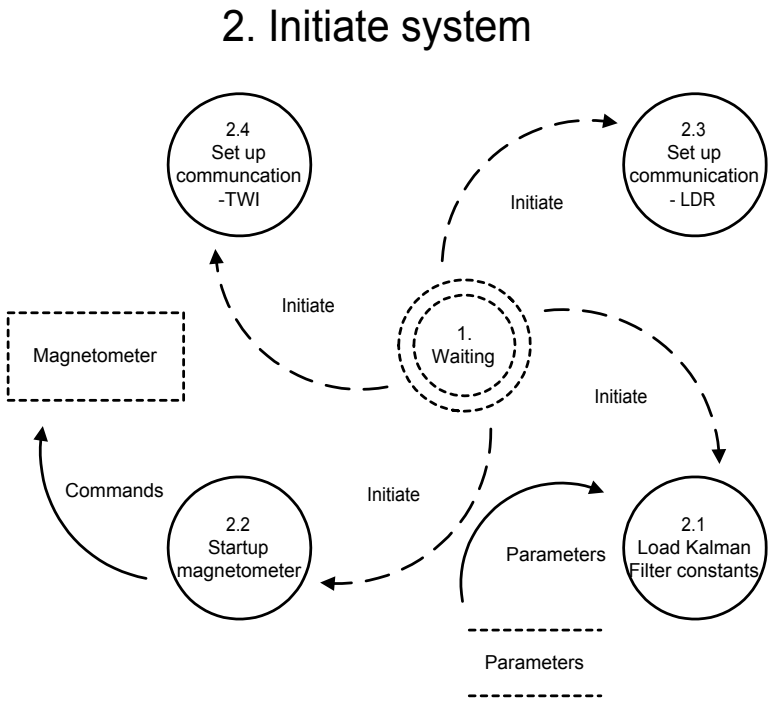

Figure 8.5: The process that initiates the external devices of the Attitude Determination system.

8.5 The Estimate state process

This process performs the actual estimation. It controls the sensors, through their respective interface, and initiate transmission of the estimated states. The estimation is split up in several subroutines. Each of these performs a calculation using the standard mathematical library or the custom-made multidimensional library.

Pointers are widely used in this process. Pointers to both float-variables, to represent vectors, and pointers to pointers, to represent matrices. This is common to the programming language C. Instead of passing on the variable itself between processes, the address of the location in memory of this variable is passed along to the next process. This approach is more efficient in addition to being easier to implement. The complete, full state estimation, is a five step task, in addition to one task getting the measurements and one task initiating the transmission, see Figure 8.6.

8.5.1 Propagate states (3.1)

Implementation in C-code:

propagateStates(qPertPred, wPertPred, qPertEst, wPertEst),

where qPertPred and wPertPred are the prediction of the perturbation at the current timestep, and wPertEst and wPertEst are the estimated perturbation from last timestep. The corre-

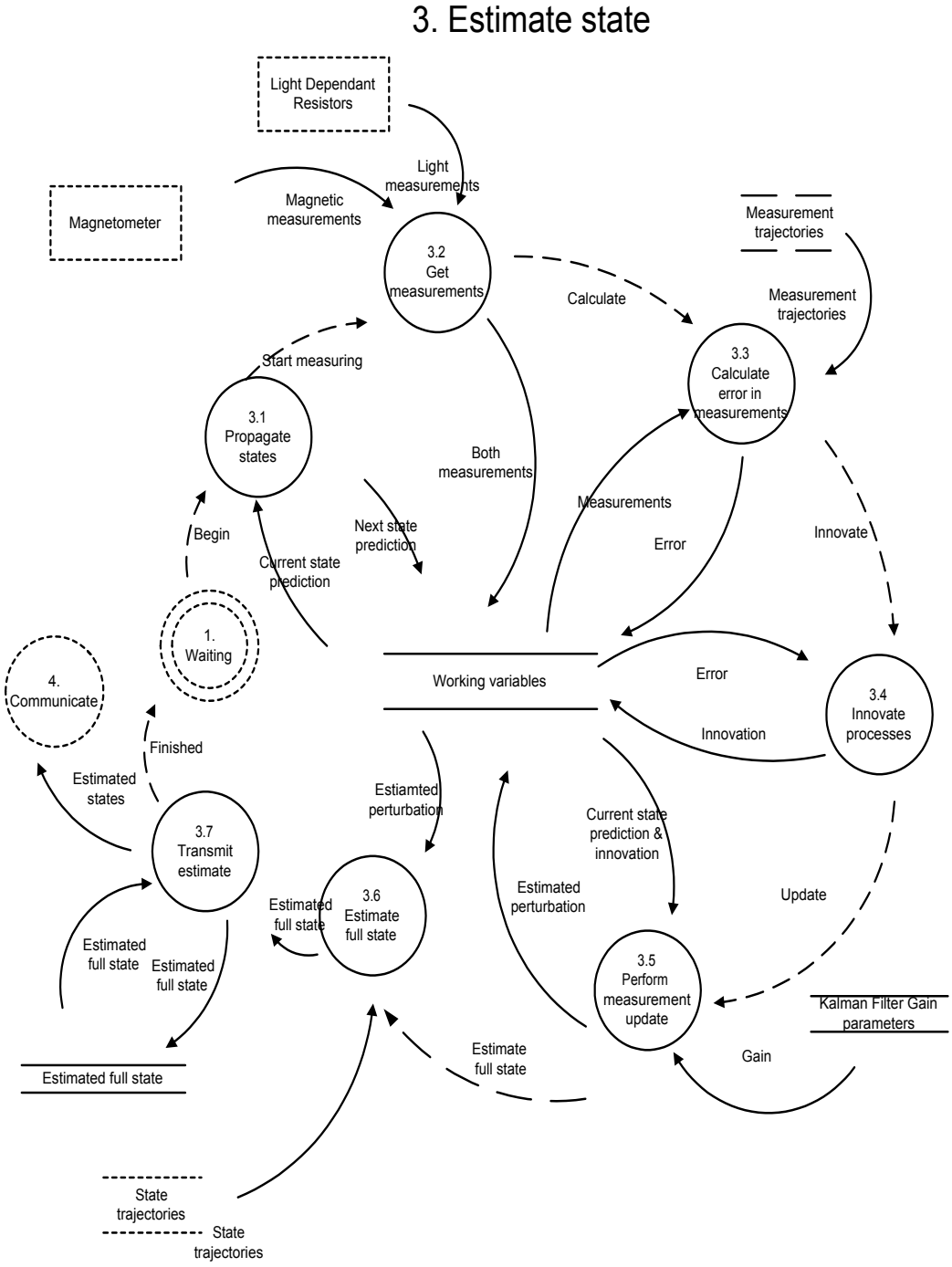

Figure 8.6: The process that performs the actual estimation.

sponding mathematical expression is stated in Equation 6.30 and reproduced here:

$$\boldsymbol{\delta}\hat{\mathbf{x}}_{k+1}^{-} = \boldsymbol{\Phi}\boldsymbol{\delta}\hat{\mathbf{x}}_k \tag{8.1}$$

The state transition matrix is stored as an expression, see Equation 6.47, in the microcontroller and is calculated using the state-trajectory at that given timestep. The matrixVectorMultiply()-routine in the multidimensional library are used for the matrix multiplication. To perform the entire multiplication of the 6×6 matrix with the 6×1 vector, the routine is called 4 times. Each call will multiply a 3×3 submatrix with the correct 3×1 part of the vector. After all four operations are completed, the result are added to form the correct 6×1 vector. This is stored in memory.

8.5.2 Get measurements (3.2)

Implementation in C-code:

```
startConversion(FALSE) & startMagnetometer(),
```

where FALSE implies that only one conversion is to be done, instead of a continuously-output mode. The measuring are done sequentially. First, the voltage over six light dependent resistors are converted. These are then combined to get the three dimensional light measurement vector, as done in Equation 5.1. When the light measuring is done, the microcontroller reads the data sent from the magnetometer, and stores this, along with the light measurements, in memory.

8.5.3 Calculate error in measurements (3.3)

Implementation in C-code:

```
errorCalc(errorB, measurementB, 'B') &
errorCalc(errorS, measurementS, 'S'),
```

where errorB and errorS are the resulting vectors after the calculations in this process, measurementB and measurementS are the respective measurements of the magnetic field and the light. The 'B' or the 'S' only indicate which measurement is currently used. Mathematically, the process performs Equation 6.37, reproduced here:

$$\mathbf{z}_{k,B} = \mathbf{B}_k \times \bar{\mathbf{B}}_k \tag{8.2}$$

$$\mathbf{z}_{k,S} = \mathbf{S}_k \times \bar{\mathbf{S}}_k, \tag{8.3}$$

where $\mathbf{B}_k$ and $\mathbf{S}_k$ are the actual measurements, $\bar{\mathbf{B}}_k$ and $\bar{\mathbf{S}}_k$ the expected measurements, and where the vector multiplication are done using the vectorVectorMultiply()-routine in the multidimensional library. Both the expected measurements are stored as curvefitted trajectories is memory. Both errors are then stored in memory for use with the next step in the estimation.

8.5.4 Innovate processes (3.4)

Implementation in C-code:

```
innovateProcess(vb, errorB, qPertPred, 'B') &
innovateProcess(vs, errorS, qPertPred, 'S'),
```

where vb and vs are the result of the innovation process, for the magnetic field and received light, respectively. ErrorB and errorS are the result from the previous routine, and 'B' and 'S' is used so that the routine will relate the error with the correct trajectory. The calculation done here is

$$\boldsymbol{\nu}_B = \mathbf{z}_{k,B} \times \mathbf{H}_B \boldsymbol{\delta}\hat{\mathbf{x}}_k^- \tag{8.4}$$

$$\boldsymbol{\nu}_S = \mathbf{z}_{k,S} \times \mathbf{H}_S \boldsymbol{\delta}\hat{\mathbf{x}}_k^- \tag{8.5}$$

where $\mathbf{H}_B$ and $\mathbf{H}_S$ are the skew-symmetric matrices of the expected trajectories of the magnetic field and experienced light, respectively. Both measurement matrices are stored as curvefitted trajectories, using the internal timer to calculate the expected measurement vectors at each call. Calculations of both $\boldsymbol{\nu}_B$ and $\boldsymbol{\nu}_S$ are done in a two-step procedure. First $\mathbf{H}_B \boldsymbol{\delta}\hat{\mathbf{x}}_k^-$ is calculated. The routine recognizes this to be a vector-vector multiplication, as $\mathbf{H}_B$ only has three different elements, and uses the vectorVectorMultiply()-routine to perform the operation. This is again multiplied with $\mathbf{z}_{k,B}$ using the same routine again. And then the whole process is repeated for $\boldsymbol{\nu}_S$. These results are then stored in memory for later use.

8.5.5 Perform measurement update (3.5)

Implementation in C-code:

```
measurementUpdate(qPertEst, wPertEst, vs, vb, qPertPred, wPertPred),
```

where qPertEst and wPertEst are the finale estimation of the perturbation. In this process, the measurements and the prediction of the perturbation from the predefined trajectory are combined. Because of the nature of the quaternions they are treated differently than the angular velocities, as described in Chapter 6.4.2. Equation 6.39 uses the conventional Kalman Filter update technique. This is implemented with a slight modification of notation, splitting up the gain that influences the angular velocities, according to

$$\boldsymbol{\delta\omega}_{OB}^B = \boldsymbol{\delta\omega}_{OB}^{B-} + \mathbf{K}_{\omega,B}\boldsymbol{\nu}_B + \mathbf{K}_{\omega,S}\boldsymbol{\nu}_S \tag{8.6}$$

where $\mathbf{K}_{\omega,B}$ and $\mathbf{K}_{\omega,S}$ are defined in Equation 6.21. Again the use of the matrixVectorMultiply()-routine is used for the multiplication, and then summed up.

In the case of the quaternions, the modified update technique in Equation 7.7 is used. This must also be done in two steps; first filter the magnetic measurements and then filter the light measurements:

$$\hat{\mathbf{q}}_B = \frac{\hat{\mathbf{q}}^- \otimes \begin{bmatrix} \Delta\mathbf{q}_{ud} \\ 1 \end{bmatrix}}{\sqrt{1 + \|\Delta\mathbf{q}_{ud}\|^2}} \tag{8.7}$$

$$\hat{\mathbf{q}}_{tot} = \frac{\hat{\mathbf{q}}_B \otimes \begin{bmatrix} \Delta\mathbf{q}_S \\ 1 \end{bmatrix}}{\sqrt{1 + \|\Delta\mathbf{q}_S\|^2}} \tag{8.8}$$

where $\hat{\mathbf{q}}^-$ is the prediction of the states, $\Delta\mathbf{q}_{ud} = \mathbf{K}_{q,B}\boldsymbol{\nu}_B$ and $\Delta\mathbf{q}_S = \mathbf{K}_{q,S}\boldsymbol{\nu}_S$, with $\mathbf{K}_{q,B}$ and $\mathbf{K}_{q,S}$ defined in Equation 6.21.

The Kalman Filter gain matrix is also in memory as curvefitted trajectories. As this is a 6×6, a total of 36 trajectories must be calculated. However, since many of these trajectories have the same form, they can be calculated using the same expression. Especially the elements involving calculation of angular velocity have the same form.

8.5.6 Estimate full state (3.6)

Implementation in C-code:

fullState(qEst, wEst, qPertEst, wPertEst),

where qEst and wEst are the finale estimation of the full state of the quaternions and angular velocities, respectively.

Again, the states must be divided into two groups. One, consisting of the angular velocities, using the standard form, and the other, the quaternions, using quaternion product:

$$\hat{\boldsymbol{\omega}} = \bar{\boldsymbol{\omega}} + \delta\hat{\boldsymbol{\omega}} \tag{8.9}$$

$$\hat{\mathbf{q}} = \bar{\mathbf{q}} \otimes \delta\hat{\mathbf{q}} \tag{8.10}$$

where the $\hat{\mathbf{q}}$ and $\hat{\boldsymbol{\omega}}$ are the estimated full state, $\bar{\mathbf{q}}$ and $\bar{\boldsymbol{\omega}}$ are the trajectories the state of the satellite is expected to follow, and $\delta\hat{\mathbf{q}}$ and $\delta\hat{\boldsymbol{\omega}}$ are the estimated perturbation from these trajectories.

The trajectories are stored as curvefitted expressions on the microcontroller. As the controller only uses the imaginary parts of the quaternion to control the satellite, only a total of six state-trajectories are necessary. A timer will be used to get the correct states from the trajectories.

8.5.7 Transmit estimate (3.7)

Implementation in C-code:

transmitEstimation(qEstF, wEstF),

where qEst and wEst are the six elements to be transmitted to the Attitude Control System, using the Two-Wire-Interface. Every result from the estimation process will be transmitted to the control system, and it will be up this system to determine the use.

The six float-elements must be converted to bytes, or characters, prior to transmission. Each element will then be represented by four characters, making up for a total of 24 characters for the 6 elements. The receiver must reassembled these characters to get the state of the satellite, and use these with the desired control algorithm.

8.6 The Communicate process

All communication with other subsystems are done using the Two-Wire-Interface. One process controls this interface on each subsystems. In this process, the different rules of communication are defined. It is important that the different subsystems have a control process that has a predictable behavior. Only this can ensure the right response to the events that can happen on the interface.

The process is divided into three subroutines. The first routine will receive data and pass it on to the second routine that interprets the incoming data, and the third routine will transmit data when called upon, see Figure 8.7. The communication is interrupt driven. That is, when anything happens on the bus that involves this node, an interrupt signal is sent by the hardware to the software, and it is up to the software to handle the event after given rules.

8.6.1 Transmit data (4.1)

This process will build the message that is about to be transmitted, according to given rules. The address for the different subsystems, which this subsystem will communicate with, are stored in memory. When data is being prepared for transmission, there must also be an indication of which subsystem is the recipient. The address of this subsystem is fetched from memory, and included in the message that are about to be transmitted, see Figure 8.8. A command is sent to the process controlling the TWI, requesting transmission, and communication is initiated.

8.6.2 Receive data (4.2)

An interrupt will be sent to the software indicating an event on the bus. The type of event can be interpreted by reading the TWI status register. This event can be that the Attitude Determination node is being addressed by another node. It will then enter Slave Receiver mode and receive every byte sent on the bus until a STOP condition is sent, ending the transmission. The bytes will then be sent to a process to interpret the newly received message.

The reception and transmission of data are controlled by state machine. After every action on the bus, the status register will reflect the response of the other node and the bus itself. By reading this register the next appropriate action can be taken.

8.6.3 Interpret data (4.3)

Implementation in C-code:

```
receiveUpdate(dataPointer)
```

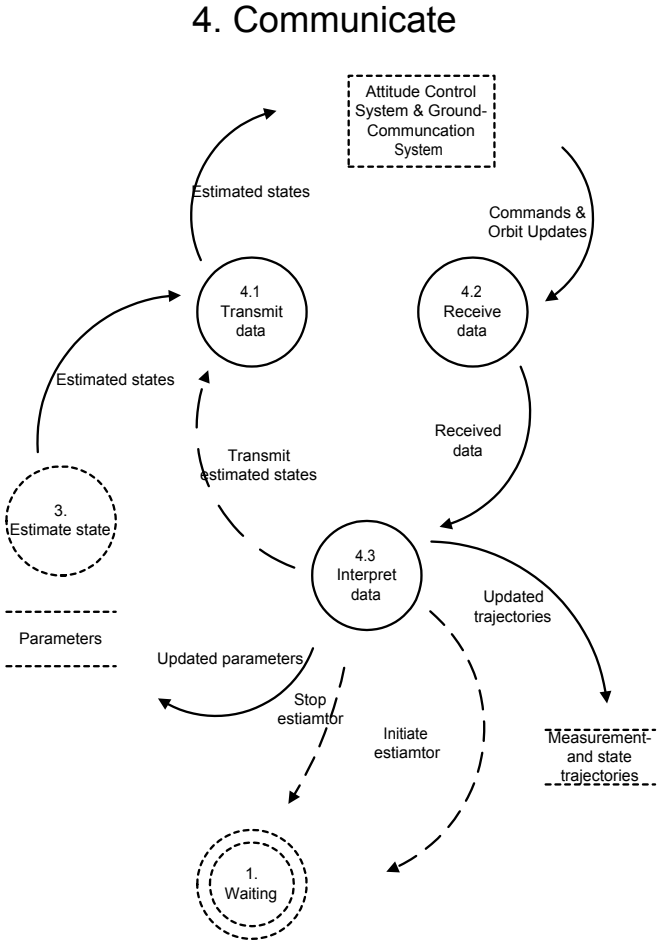

Figure 8.7: The communcation process.

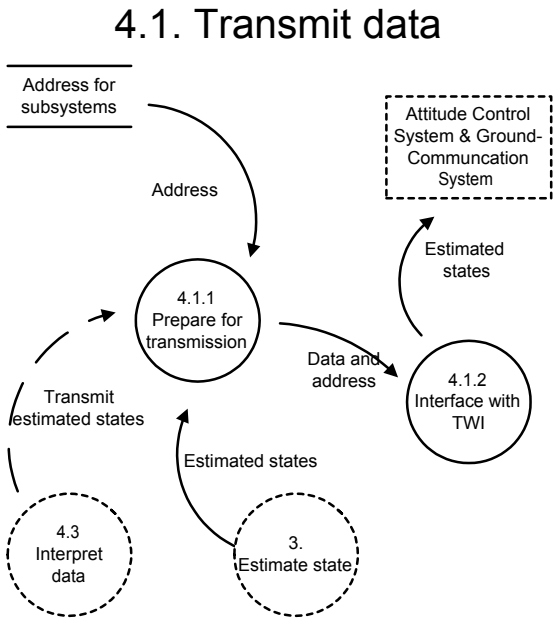

Figure 8.8: The transmit data process.

where dataPointer is the address in memory of the first byte in the received message. The first byte of this message indicates what type of data has been received and the appropriate action will be taken accordingly, see Figure 8.9.

The different types messages that can be received are:

- Updated parameters for measurement trajectories. Recognized by a leading B or S character, for magnet field and light trajectory, respectively. 10 parameters are necessary for each of the six trajectories to produce accurate enough curvefitted trajectories.
- Updated parameters for state trajectory. Recognized by a leading Q character. Both the trajectories for the angular velocity and orientation are updated. Also here 10 parameters are necessary for each of the six state-trajectories to produce accurate enough curvefitted trajectories.
- Stop command. Recognized by a leading C, for command, and then the word 'STOP' to stop the estimator.
- Start command. Recognized by a leading C, and then the word 'START' to start the estimator.
- Request for transmission. Again the leading C and then a T will tell the estimator to initiate transmission of the last estimated states. Transmission will commence as soon as the TWI is free.

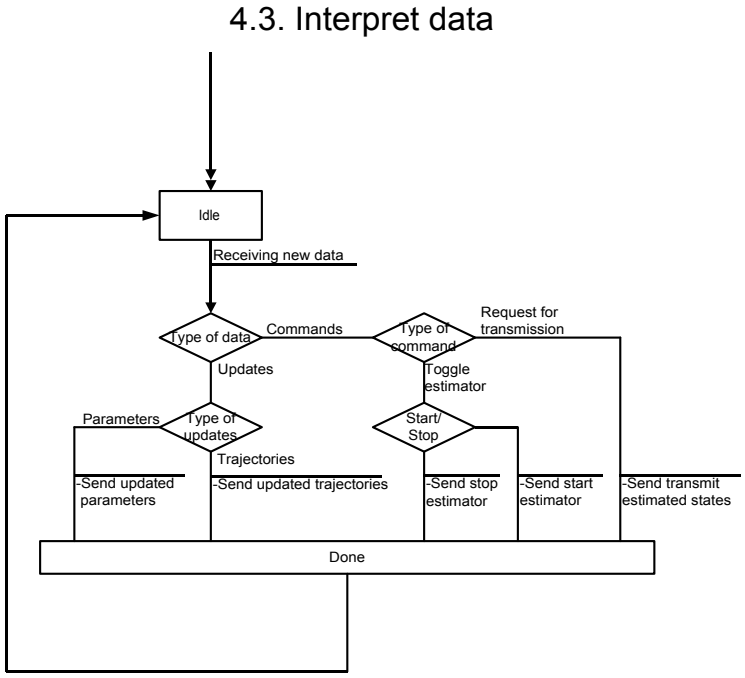

Figure 8.9: The interpret data process. Illustrates how received data is interpreted.

8.7 Post launch initialization

Some work must be done after launch to get a fully functional estimator. The Keplerian elements of the Orbit must be determined, as well as the position of the satellite in ECEF frame. The first is needed to transform the magnetic field model from Earth-Centered Orbit frame to Earth-Centered Earth-Fixed frame, and the latter is needed to transform vectors between ECEF and ECI. By initiating the different models with their correct initial values, the trajectory of the measurements and the states can be calculated.

These must then be curvefitted. On board the satellite, the expressions of the different trajectories are implemented using either Fourier or polynomial method, depending on the trajectory. The magnetic field and light trajectories, and the orientation trajectory are implemented using the Fourier method:

$$f = a_0 + a_1\cos(x \cdot w) + b_1\sin(x \cdot w) + a_2\cos(2x \cdot w) + b_2\sin(2x \cdot w) + a_3\cos(3x \cdot w) \quad (8.11)$$
$$+b_3\sin(3x \cdot w) + a_4\cos(4x \cdot w) + b_4\sin(4x \cdot w) \quad (8.12)$$

where $a_0 \cdots a_4$, $b_1 \cdots b_4$ and w are the parameters that must be upload prior to initiation, and x is the free variable to indicate the current timestep. No good curvefitted were found for the angular velocities using the Fourier method, and the use of the polynomial method proved to give better results:

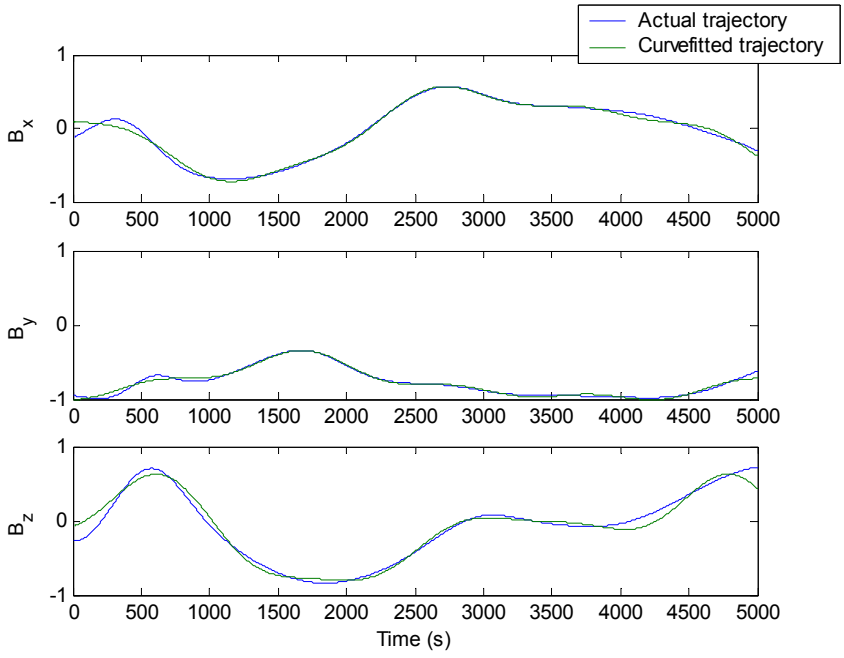

Figure 8.10: The magnetic field is cruvefitted using the Fourier method.

$$f = p_1 \cdot x^9 + p_2 \cdot x^8 + p_3 \cdot x^7 + p_4 \cdot x^6 + p_5 \cdot x^5 + p_6 \cdot x^4 + p_7 \cdot x^3 \quad (8.13)$$

$$+p_8 \cdot x^2 + p_9 \cdot x + p_{10} \quad (8.14)$$

where $p_1 \cdots p_{10}$ are the parameters that must be upload prior to initiation, and x is the free variable to indicate the current timestep. On Figure 8.10 the measurement trajectory of the magnetic field is plotted. It is seen that the curvefitting can be done very accurate.

Chapter 9

Conclusion

The results and conclusion of the work presented in the previous chapters are described here. Some guidelines for possible future work are also presented.

9.1 Conclusion

An Extended Kalman Filter is implemented with a satellite model. The performance of this filter is proven to be good. It is robust and, given an initial detumbling phase of the satellite, the filter will converge and produce acceptable results. In practice this filter will prove to produce the best performance. The estimated states of the Kalman Filter are used by the controller to stabilize the satellite with good performance. However, the Extended Kalman Filter is too demanding to implement on a standard microcontroller, thus other options have been studied.

When using a linearized filter, much of the calculations can be done either prior to launch or on the ground station and uplink the result. The performance of this filter, both in continuous and discrete time, was, however, not acceptable. It was too dependent on the initial state of the satellite and the deviation from a predefined trajectory, that it should not be used as it is. Small perturbation from the trajectory produced good estimates, but the robustness of the filter must be further improved.

Experiments have been done with the sensor that is used on board the nCube to determine the accuracy of these. The results from these experiments are used both in the simulation and implementation of the filter to best reflect the actual hardware used on board the satellite.

The filter has been implemented on a microcontroller. Both reading from the sensors and communicating with other subsystems are fully operational. The estimator with all the calculations involved, is also implemented, and the different process have been tested separately with good results.

9.2 Recommendations

To get a fully functional estimator on board either nCube 1 or nCube 2, some work must be done both on simulations in matlab and om implementation on the microcontroller.

A further study of the robustness of the simplified estimator should performed. The performance of the current estimator is dependent on the initial condition of the satellite and should

thus not be used in a real application. The presented work is based upon a linearized filter where some of the calculations are done offline, and includes the use trajectories. The initial performance is good, but the filter drifts away.

The current controller is set to stabilize the orientation and angular velocities. A perhaps better approach would be for the controller to minimize the difference between the trajectory and the actual state. Another approach could be to study the use of a nonlinear observer, instead of a Kalman Filter. This might also prove to be a good choice when implementing on a final satellite.

The microcontroller used is not optimized for both speed and low power consumptions. A different vendor might have better microcontroller able to run a more complex model. In addition, the final touch on the software must be completed. Amongst the things that must be completed are error handling and increasing the general fault tolerance. Also, fully integration with the other subsystems must be completed. This to ensure that the different subsystems can predict the response of the every other subsystem as the TWI changes state.

Bibliography

Appel, P (2002), Satellite attitude estimation using magnetometer and albedo model enhanced coarse sun sensor measurements, Master's thesis, Center of Applied Space Technology an Microgravity (ZARM) at the University of Bremen, Germany.

Bak, T (1999), Spacecraft Attitude Determination - a Magnetometer Approach, PhD thesis, Department of Control Engineering, Aalborg University.

Bar-Itzhack, I. Y., F. L. Markley & J. Deutchmann (1991), Quaternion normalization in additive EKF for spacecraft attitude determination, *in* 'Flight Mechanics/Estimation Symposium'.

Brown, R. G. & Hwang, P. Y. C (1997), *Introduction to Random Signals and Applied Kalman Filtering 3rd Edition*, John Wiley & Sons, New York.

Egeland, O. & J. T. Gravdahl (2001), *Modeling and Simulation for Control*, Department of Engineering Cybernetics, NTNU.

Farrell, J. A. & Barth, M (1999), *The Global Positioning System & Inertial Navigation*, McGraw-Hill, New York.

Fauske, K. M. (2002), NCUBE attitude control, Master's thesis, Department of Engineering Cybernetics, NTNU.

Fossen, T. I. (2002), *Marine Control Systems - Guidance, Navigation and Control of Ships, Rigs and Underwater Vehicles*, Marine Cybernetics, Trondheim, Norway.

IEEE (2004), 'Floating point standard. last accessed: 28.05.04', Website: www.math.byu.edu/ schow/work/IEEEFloatingPoint.htm.

Kristiansen, R. (2000), Attitude control of mini satellite, Master's thesis, NTNU.

Kyrkjebo, E (2000), Three-axis attitude determination using magnetometers and a startracker, Master's thesis, NTNU.

Psiaki, M. L., F. Martel & P. K. Pal (1990), 'Three-axis attitude determination via kalman filtering of magnetometer data', *Journal of Guidance, Control and Dynamics* **13**(3), 506–514.

Svartveit, K (2003), Attitude determination of the NCUBE satellite, Master's thesis, Department of Engineering Cybernetics, NTNU.

Appendix A

.

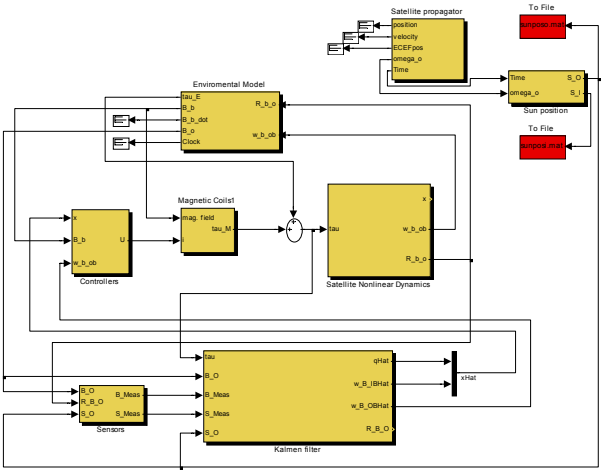

Figure A.1. The overall system. Every environment model, and the satellite with the estiamtor and controller can be seen here.

A.1 Simulink diagram of Extended Kalman Filter

.

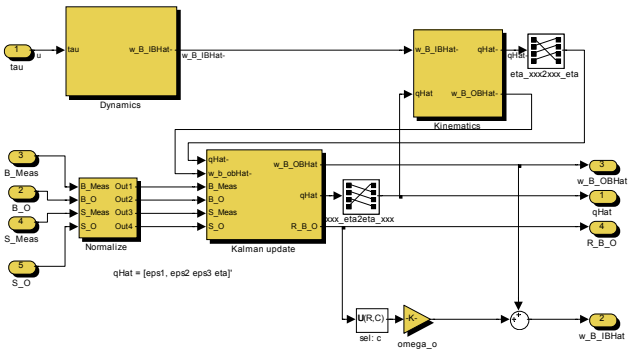

Figure A.2. The Kalman Filter block

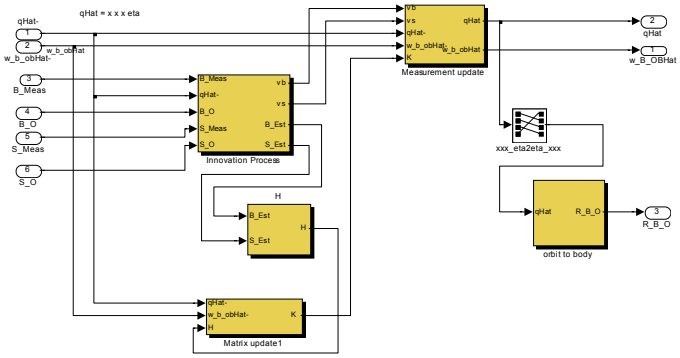

Figure A.3. The Kalman update block.

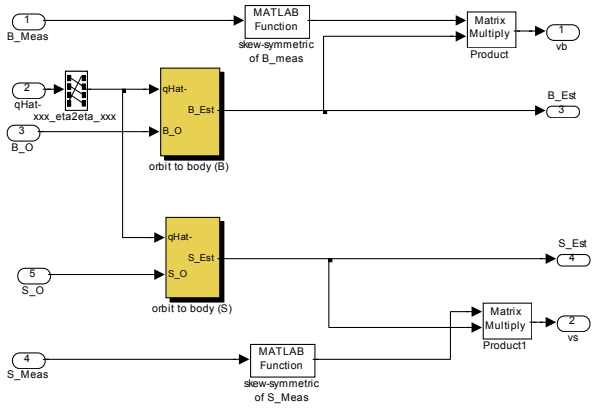

Figure A.4. The innovation process block.

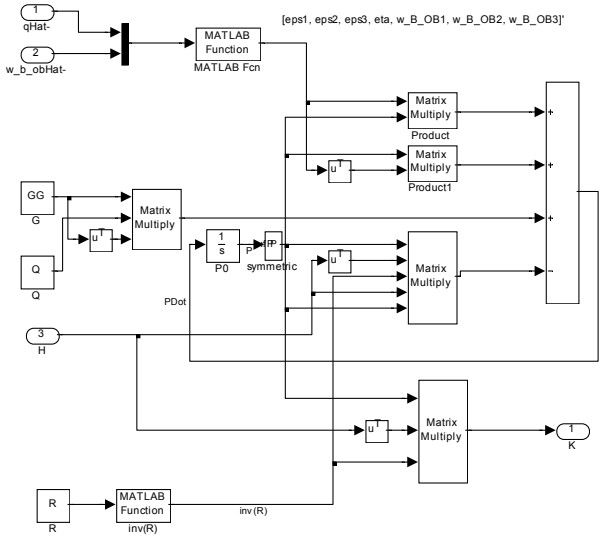

Figure A.5. The gain and state covariance matrices are calculated here.

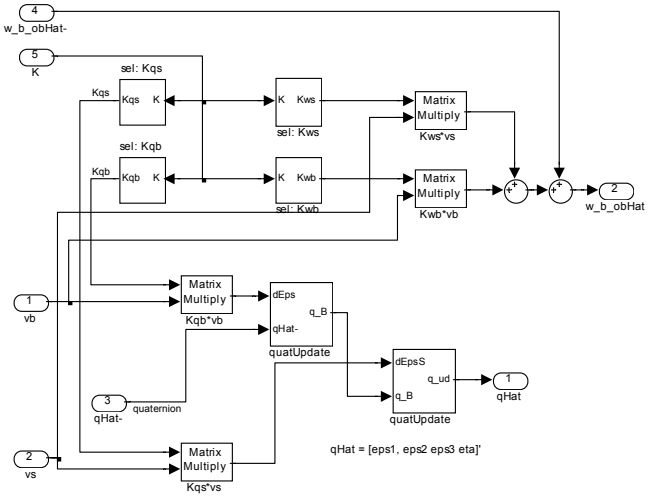

Figure A.6. The measurement update block.

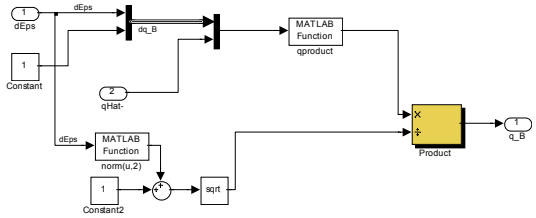

Figure A.7. The quaternion update block.

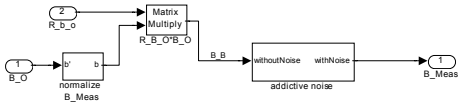

Figure A.8. The simulation of a sensor are done in this block. One for light sensor is also used.

A.2 Simulink diagram of Linearized Kalman Filter

.

The simulink blocks that differ from those used in the Extended Kalman Filter is shown here.

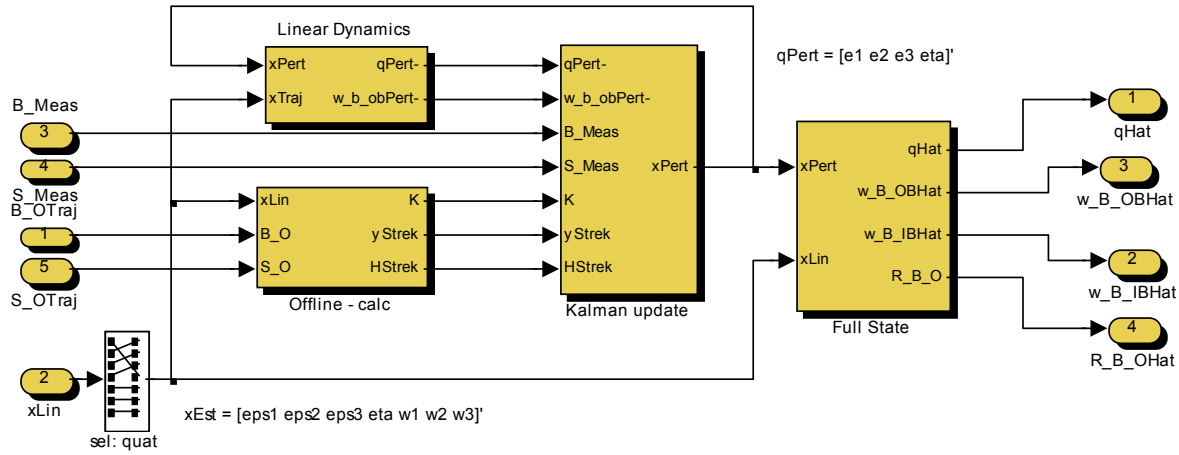

Figure A.9. The Linearized Kalman Filter.

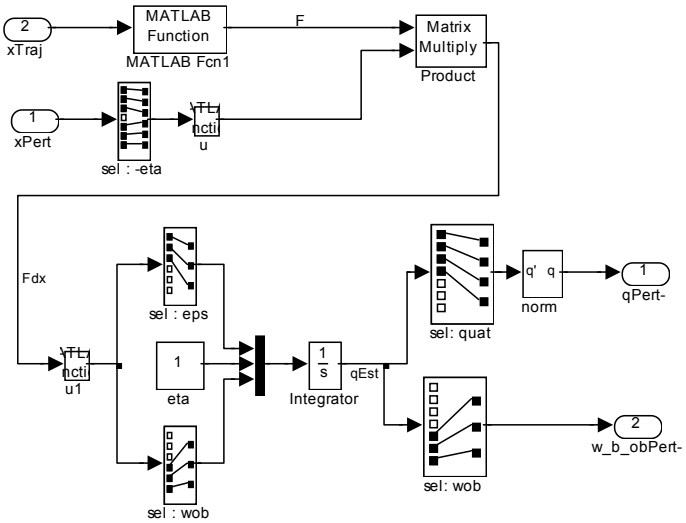

Figure A.10. The linearized dynamics and kinematics block.

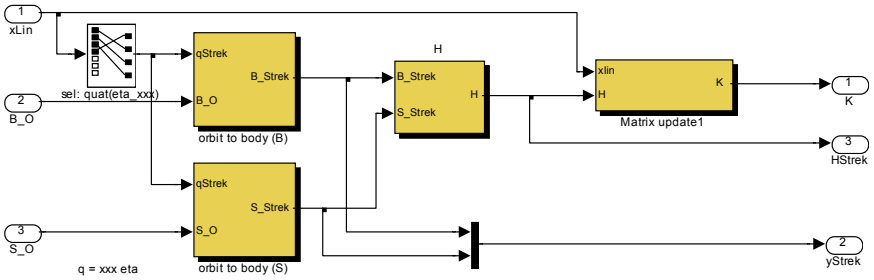

Figure A.11. The offline calculations are simulated using this block.

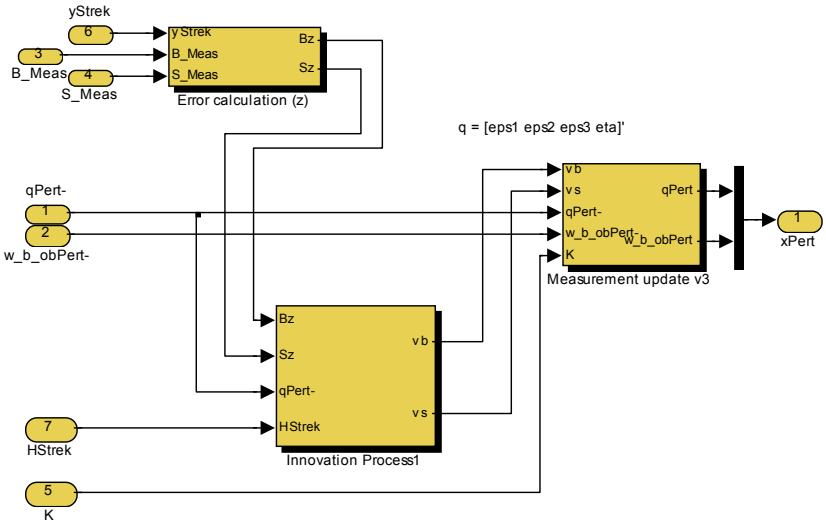

Figure A.12. The Kalman Filter update block.

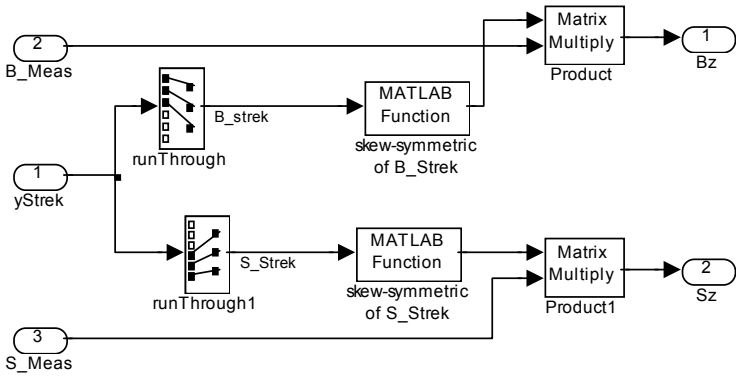

Figure A.13. The calculation of error between expected and actual measurements.

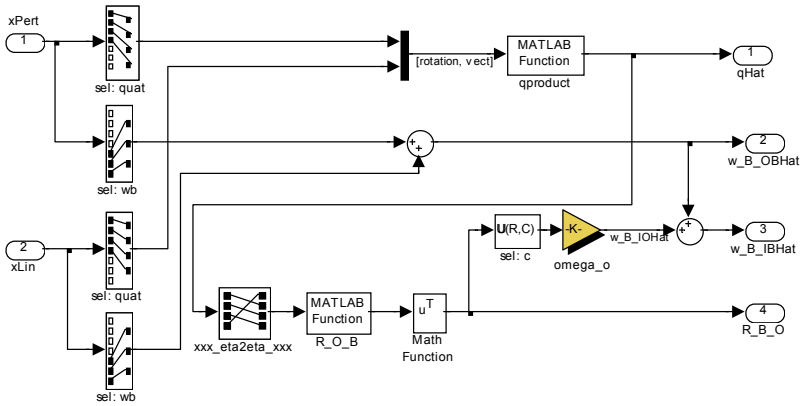

Figure A.14. To full state estiamtion.

A.3 Simulink diagram of Discrete Kalman Filter

.

The blocks that differ from the previously presented are shown here.

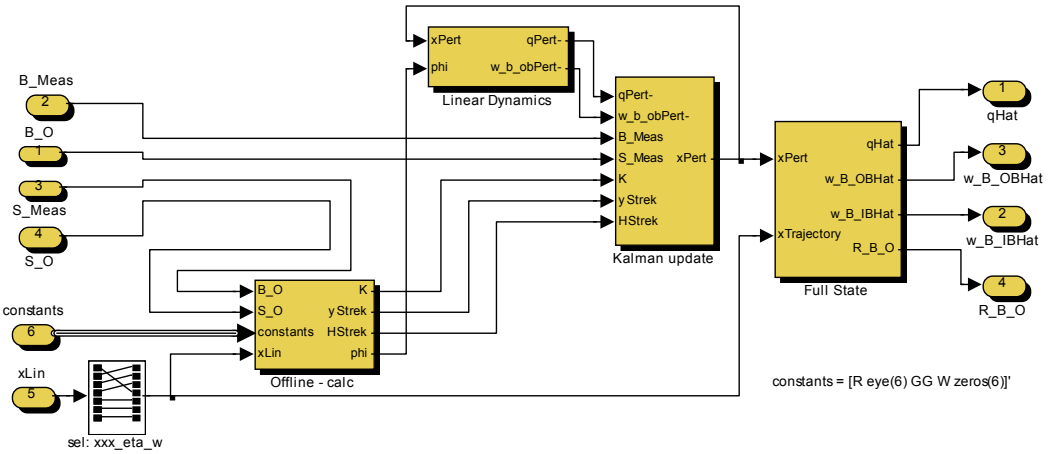

Figure A.15. The discrete Kalman Filter block.

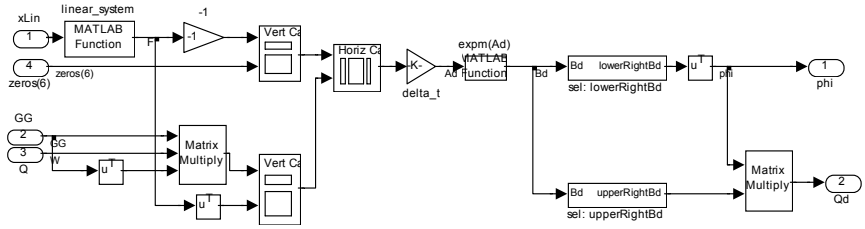

Figure A.16. Calculation of state transition matrix and discrete covariance matrix for the process noise are done here.

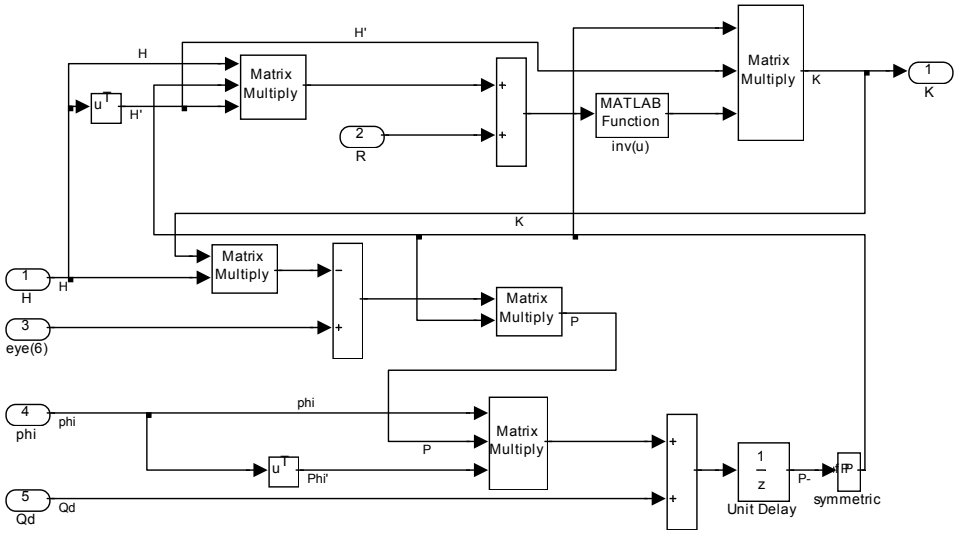

Figure A.17. The covariance matrix and gain matrix are calculated here.

Appendix B

.

B.1 Parameters for the linearized model

.

The linearized model is written as

$$\mathbf{F}_{total}(\mathbf{x}) = \begin{bmatrix} \mathbf{F}_{kin}(\mathbf{x}) \\ \mathbf{F}_{dyn}(\mathbf{x}) \end{bmatrix}, \tag{1}$$

where $\mathbf{F}_{kin}$ is found in Equation 4.13 and

$$\mathbf{F}_{dyn}(\mathbf{x}) = \begin{bmatrix} a_{51} & a_{52} & a_{53} & a_{54} & 0 & a_{56} & a_{57} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & 0 & a_{67} \\ a_{71} & a_{72} & a_{73} & a_{74} & a_{75} & a_{76} & 0 \end{bmatrix}, \tag{2}$$

where

$$\begin{array}{lll}
a_{51} = 2k_x\omega_o(\eta\omega_y + \varepsilon_1\omega_z) & +6k_x\omega_o^2(\varepsilon_1 c_{23} - \eta c_{33}) & +2(\eta\omega_2 - \varepsilon_1\omega_3)\omega_o \\
a_{52} = -2k_x\omega_o(\varepsilon_3\omega_y + \varepsilon_2\omega_z) & +6k_x\omega_o^2(\varepsilon_2 c_{23} - \varepsilon_3 c_{33}) & +2(\varepsilon_2\omega_3 - \varepsilon_3\omega_2)\omega_o \\
a_{53} = 2k_x\omega_o(\varepsilon_3\omega_z - \varepsilon_2\omega_y) & -6k_x\omega_o^2(\varepsilon_2 c_{33} + \varepsilon_3 c_{23}) & -2(\varepsilon_3\omega_3 + \varepsilon_2\omega_2)\omega_o \\
a_{54} = 2k_x\omega_o(\varepsilon_1\omega_y - \eta\omega_z) & -6k_x\omega_o^2(\varepsilon_1 c_{33} + \eta c_{23}) & +2(\eta\omega_3 + \varepsilon_1\omega_2)\omega_o \\
a_{56} = k_x\omega_z - c_{32}\omega_o & & \\
a_{57} = k_x\omega_y + c_{22}\omega_o & &
\end{array}$$

$$\begin{array}{lll}
a_{61} = 2k_y\omega_o(\varepsilon_2\omega_z - \eta\omega_x) & +6k_y\omega_o^2(\varepsilon_3 c_{33} - \varepsilon_1 c_{13}) & -2(\eta\omega_1 + \varepsilon_2\omega_3)\omega_o \\
a_{62} = 2k_y\omega_o(\varepsilon_1\omega_z + \varepsilon_3\omega_x) & -6k_y\omega_o^2(\eta c_{33} + \varepsilon_2 c_{13}) & +2(\varepsilon_3\omega_1 - \varepsilon_1\omega_3)\omega_o \\
a_{63} = 2k_y\omega_o(\eta\omega_z + \varepsilon_2\omega_x) & +6k_y\omega_o^2(\varepsilon_1 c_{33} + \varepsilon_3 c_{13}) & +2(\varepsilon_2\omega_1 - \eta\omega_3)\omega_o \\
a_{64} = 2k_y\omega_o(\varepsilon_3\omega_z - \varepsilon_1\omega_x) & +6k_y\omega_o^2(\eta c_{13} - \varepsilon_2 c_{33}) & -2(\varepsilon_1\omega_1 + \varepsilon_3\omega_3)\omega_o \\
a_{65} = -k_y\omega_z + c_{32}\omega_o & & \\
a_{67} = -k_y\omega_x - c_{12}\omega_o & &
\end{array}$$

$$\begin{array}{lll}
a_{71} = 2k_z\omega_o(\varepsilon_2\omega_y - \varepsilon_1\omega_x) & +6k_z\omega_o^2(\varepsilon_3 c_{23} + \eta c_{13}) & +2(\varepsilon_1\omega_1 + \varepsilon_2\omega_2)\omega_o \\
a_{72} = 2k_z\omega_o(\varepsilon_1\omega_y + \varepsilon_2\omega_x) & +6k_z\omega_o^2(\varepsilon_3 c_{13} - \eta c_{23}) & +2(\varepsilon_1\omega_2 - \varepsilon_2\omega_1)\omega_o \\
a_{73} = 2k_z\omega_o(\eta\omega_y - \varepsilon_3\omega_x) & +6k_z\omega_o^2(\varepsilon_1 c_{23} + \varepsilon_2 c_{13}) & +2(\varepsilon_3\omega_1 + \eta\omega_2)\omega_o \\
a_{74} = 2k_z\omega_o(\varepsilon_3\omega_y + \eta\omega_x) & +6k_z\omega_o^2(\varepsilon_1 c_{13} - \varepsilon_2 c_{23}) & +2(\varepsilon_3\omega_2 - \eta\omega_1)\omega_o \\
a_{75} = -k_z\omega_y - c_{22}\omega_o & & \\
a_{76} = -k_z\omega_x + c_{12}\omega_o & &
\end{array}$$

The direction cosines are given in Equation 2.24 as

$$\mathbf{R}_O^B = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} = \begin{bmatrix} -\epsilon_1^2 - \epsilon_2^2 + \epsilon_3^2 + \eta^2 & 2(\epsilon_1\epsilon_2 + \eta\epsilon_3) & 2(\epsilon_1\epsilon_3 - \eta\epsilon_2) \\ 2(\epsilon_1\epsilon_2 - \eta\epsilon_3) & -\epsilon_1^2 + \epsilon_2^2 - \epsilon_3^2 + \eta^2 & 2(\epsilon_2\epsilon_3 + \eta\epsilon_1) \\ 2(\epsilon_1\epsilon_3 + \eta\epsilon_2) & 2(\epsilon_2\epsilon_3 - \eta\epsilon_1) & -\epsilon_1^2 - \epsilon_2^2 + \epsilon_3^2 + \eta^2 \end{bmatrix}, \tag{3}$$

the ω_x, ω_y and ω_z are the three elements of $\boldsymbol{\omega}_{IB}^B$, and

$$k_x = \frac{I_{yy} - I_{zz}}{I_{xx}}, \quad k_y = \frac{I_{xx} - I_{zz}}{I_{yy}} \quad \text{and} \quad k_z = \frac{I_{yy-}I_{xx}}{I_{zz}} \tag{4}$$

B.2 CD contents

.

The special files used in matlab are described here

- hex2float.m - Converts the hexadecimal numbers received from the LDR, via the ADC, to determine the covariance of this sensor.
- normalString.m - Used in relation with hex2float to get the correct conversion.
- readMeasurement.m - The file to execute to calculate the covariance. Uses the two scripts mentioned above.
- selSubMatrix.m - Returns a 3 by 3 submatrix from a 6 by 6 matrix
- selSubMatrix6.m - Returns a 6 by 6 submatrix form a 12 by 12 matrix
- maketable.mdl - Creates the magnetic field
- Discrete KF.mdl - Discrete KF in simulink
- Linearized KF.mdl - Linearized KF in simulink
- Extended KF.mdl - Extended KF in simulink

The files used to implement the estimator on a microcontroller are described here

- com.c - Interface with the UART used for debugging by communicating with a computer, and to communicate with magnetometer
- highLayerConversion.c - Performs conversion from the six LDR sensors to a 3×1 measurement vector
- highLayerTrans.c - Conversion between float and char variable types to communicate using the TWI.
- init.c - Initiates the necessary variables and features in the microcontroller
- kalmanF.c - Does the actual estimation.

- LDRinterface.c - Interface with ADC.
- multiDimMath.c - The multidimensional operations library.
- orbit.c - Used to propagate the trajectories.
- twi.c - Controls the TWI bus.
- twiSlave.c - Controls the TWI bus on a debug node.

The datasheet used in the work

- ATmega128.pdf - Microcontroller.
- ATmega323.pdf - Microcontroller.
- JTAG.pdf - Interface between computer and development board, and programmer of MCU.
- STK-500.pdf - Development board.
- STK-501.pdf - Development board.
- MAX220-MAX249.pdf - RS232 chip.