NTNU
Innovation and Creativity

Kalman filter for attitude determination of student satellite

Jan Rohde

Master of Science in Engineering Cybernetics
Submission date: July 2007
Supervisor: Jan Tommy Gravdahl, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

NTNU is planing to build a student satellite made exclusively by students attending the university. The satellite is going to be outfitted with an estimator for attitude determination. The task is to create, implement and if possible test a Kalman filter based estimator that is going to be used for attitude determination on board the student satellite.

Assignment given: 15. January 2007
Supervisor: Jan Tommy Gravdahl, ITK

Kalman filter for attitude determination of student satellite

Master's Thesis

by

Jan Rohde

Norwegian University of Science and Technology
Department of Engineering Cybernetics
Trondheim

July 2007

Abstract

In the autumn of 2006 a satellite project was started at NTNU. The goal of the project is two-folded, first it seeks to create more interest and expertise around the field of space technology, secondly to create a satellite platform which can be modified and equipped with different payloads to perform selected tasks in a Low Earth Orbit.

For a satellite to be able to complete missions involving sensory and imaging, an attitude determination and control system is needed to give the satellite a stable attitude. In order to create a good attitude control system, a Gauss-Newton improved extended Kalman filter is used together with reference models to supply the controller with estimates of both satellite angular velocity and orientation.

This report focuses on the Attitude Determination System, ADS, realized by implementing the improved extended Kalman filter on a microcontroller. The challenge is to create an estimator that will provide the control system with adequate estimates without requiring to much computational power, as this is a limiting factor on board a micro satellite. The need for good computational power comes from the multidimensional matrix mathematical operations performed on float numbers. Based on previous work, an improved Extended Kalman filter has been developed and implemented on a microcontroller for further testing. A new filter, the Unscented Kalman Filter has also been explored but not implemented.

Preface

This master thesis is part of an ongoing project that started in the autumn of 2006 at the Norwegian University of Science and Technology with the goal of creating the University's first student satellite. The work began with the announcement of a competition by NAROM, National Center for Space-related Education, where a satellite proposal should be handed in. The student satellite is based upon the Cubesat framework which means that available power is a limiting factor. A second goal of the satellite project is to increase the interest and competence within space related fields.

This master thesis focuses on the attitude determination system that is going to be implemented on the satellite. It was the intention that the Attitude Determination System, ADS, should be based on previous work done at the Department of Cybernetics Engineering at NTNU but also that new approaches should be investigated and refinements be done to existing solutions if needed. The attitude determination system is implemented on a microcontroller and it is important to design it in such a way that computational requirements are kept low. In addition, a new type of Kalman filter has been explored called the Unscented Kalman Filter.

I would like to thank my Advisors, Jan Tommy Gravdahl and Jo Arve Alfredsen for their valuable feedback through this project as well as the people in office D444 for creating a good and inspiring environment to work in.

Trondheim, 05.06.2007

Jan Rohde

Contents

Abbreviations

Abbreviations used in this report

- Digitally controlled oscillator - DCO
- Extended Kalman Filter - EKF
- Kalman Filter - KF
- Light to Frequency converter - LF-converter
- Low Earth Orbit - LEO
- Multiplexer - MUX
- Serial Clock - SCL (TWI transmission)
- Serial Data - SDA (TWI transmission)
- Two-wire interface - TWI
- Unscented Kalman Filter - UKF
- Universal Serial Synchronous/Asynchronous communication interface - USART
- Unscented Transformation - UT

X

Chapter 1

Introduction

Background

The Cubesat concept was developed at Stanford University by professor Bob Twiggs. His motivation for doing this was to allow his students to complete a satellite project during their education. The basic idea was to create satellite framework which should be small and standardized and let students modify the interior to create a satellite. To keep the launch costs down, the size of the standard framework was chosen to be 10 x 10 x 10 cm, weighing up to one kilogram. To further reduce costs of launch, a pod was created to fit with commercial rockets. The pod also allowed several cubes to be launched so that the cost would be spread out. Figure 1.1 shows a Cubesat and the

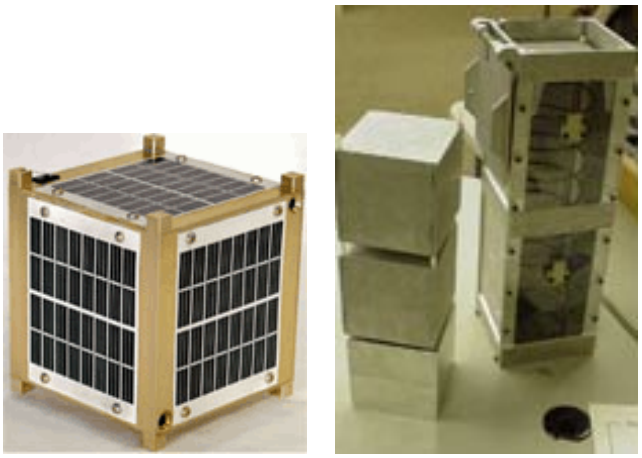

Figure 1.1: The Cubesat and Cubesat launcher

launcher

Norwegian educational institutions have in resent years attempted to design and launch a student satellite into orbit. Two satellites were built, nCube1 and nCube2, both based on the Cubesat framework. The work was a joint effort between several universities in Norway. Both the projects ended in failure with one satellite ending up dead in orbit and the other went down with the delivery rocket, only two minutes after being launched from Baikonur Cosmodrome, Kazakhstan.

In the autumn of 2006 plans were made for making a new student satellite at NTNU. The idea sprung to life with NAROM's, National Center for Space-related Education, announcement of a competition where students were encouraged to create a proposal for a student satellite. The proposals would be judged by a jury at NAROM and the winning team would be awarded a free launch. In December of 2006 a proposal for a student satellite created at NTNU was handed in to NAROM. The proposal can be found in appendix C. In order to reduce the number of people involved it was decided that the satellite is to be designed and developed only by students attending NTNU. This should also serve to make the project more manageable.

The proposal, Narverud et al. [Nov, 2006], suggested the use of a double Cubesat structure. Several reasons was laid to ground for this, among others increased available power through bigger solar panels and the possibility to add more payload if needed. This also conforms with the idea to create a standardized fully functional satellite where different payload can be added and removed without the need to change or reprogram the satellite.

As part of easing the project management work, a satellite database was been developed where all the work done on the project is stored and explained. This also eases the work for new students who come into the project and continue on previous work.

Previous work

There has not been done a lot of prior work with respect to attitude determination on this project. Even so a huge amount of work has been done on the subject at NTNU previous years. This work is the basis for the attitude determination system that will be developed for this new satellite. The Kalman filter that will be used has been developed and tested in Sunde

[2004]. That work is further based on Svartveit [2003] and Ose [2004].

This report

The goal of this report is to start the work on developing a prototype unit of the Attitude Determination System, ADS. This prototype is going to be used for testing to uncover possible errors in the complete Attitude Determination and Control System, ADCS. The suggested Kalman filter must be implemented on a target microcontroller to uncover if it is sufficiently simplified to meet the computational power limitations. A new filter, the Unscented Kalman filter is also presented in this report.

Report outline

Chapter 2 gives an introduction to attitude determination and the underlying theories. This involves different sensor models and how they are used to calculate the satellite attitude. In chapter 3 the Kalman filter technology is presented, along with the implemented Extended Kalman filter and the Unscented filter. A simulink model of the filter is presented in chapter 4 and chapter 5 introduces the hardware components that are used in the prototype. Some comparisons are made between different hardware. The actual implementation is presented in the next chapter, chapter 6 and ideas on how to perform tests on the prototype ADS is given in chapter 7.

Chapter 2

Attitude Representation

In this chapter an introduction on how to describe the attitude of a satellite is given. Different reference models will be introduced as well as a model of the satellite.

2.1 Parameter representation

In this section we introduce the basics of vectors and rotation matrices which are widely used in attitude representation.

2.1.1 Vectors

A vector $\vec{u}$ can be described by its magnitude $|\vec{u}|$ and its direction. This description does not rely on the definition of any coordinate frame and is said to be coordinate-free. A vector can also be represented in the Cartesian coordinate frame a defined by the orthogonal unit vectors $\vec{a}_1$, $\vec{a}_2$ and $\vec{a}_3$ along the x_1, x_2 and x_3 axes. Here the vector $\vec{u}$ can be expressed as a linear combination of the orthogonal unit vectors in the following way

$$\vec{u} = u_1\vec{a}_1 + u_2\vec{a}_2 + u_3\vec{a}_3 \tag{2.1}$$

where

$$u_i = \vec{u} \cdot \vec{a}_i, \qquad i \,\epsilon\, \{1, 2, 3\} \tag{2.2}$$

are the unique components or coordinates of $\vec{u}$ in a. A related description of the vector is the coordinate vector form where the coordinates of the vector

are written as a column vector

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} \tag{2.3}$$

The scalar product

The scalar product can be written in three different alternative forms

$$\vec{u} \cdot \vec{v} = \sum_{i=1}^{3} u_i v_i = \mathbf{u^T v} \tag{2.4}$$

The vector cross product

In coordinate-free form the vector cross product is given by

$$\vec{u} \times \vec{v} = \vec{n}|\vec{u}||\vec{v}|sin\theta \tag{2.5}$$

where $0 \leq \theta \leq \pi$ and $\vec{n}$ is a unit vector that is orthogonal to both $\vec{u}$ and $\vec{v}$ and defined such that $(\vec{u}, \vec{v}, \vec{n})$ forms a right-hand system as figure 2.1. In a

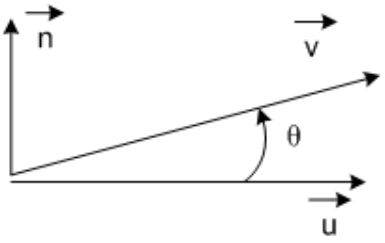

Figure 2.1:

Cartesian frame the vector cross product can be evaluated from

$$\vec{w} = \vec{u} \times \vec{v} = \begin{vmatrix} \vec{a_1} & \vec{a_2} & \vec{a_3} \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{vmatrix} \tag{2.6}$$

In coordinate vector notation we can define a vector in a skew-symmetric form as

$$u^{\times} := \begin{pmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{pmatrix} \tag{2.7}$$

This allows us to write the vector cross product in coordinate form as

$$\mathbf{w} = \mathbf{u}^{\times}\mathbf{v} = \begin{pmatrix} u_2v_3 - u_3v_2 \\ u_3v_1 - u_1v_3 \\ u_1v_2 - u_2v_1 \end{pmatrix} \tag{2.8}$$

2.1.2 Rotation matrices

A vector can be represented in different Cartesian frames. Given two coordinate frames a and b with orthogonal unit vectors $\vec{a_1}$, $\vec{a_2}$, $\vec{a_3}$ and $\vec{b_1}$, $\vec{b_2}$, $\vec{b_3}$, a vector $\vec{v}$ can be represented with respect to any of the systems a and b. This is written as

$$\vec{v} = \sum_{i=1}^{3} v_i^a \vec{a}_i \qquad \text{and} \qquad \vec{v} = \sum_{i=1}^{3} v_i^b \vec{b}_i \tag{2.9}$$

where

$$\begin{aligned} v_i^a &= \vec{v} \cdot \vec{a}_i \\ v_i^b &= \vec{v} \cdot \vec{b}_i \end{aligned} \tag{2.10}$$

are the coordinates of $\vec{v}$ in a and b respectively. The relation between the two vectors $\mathbf{v}^a$ and $\mathbf{v}^b$ in frames a and b is given by the following calculation

$$\begin{aligned} v_i^a &= \vec{v} \cdot \vec{a}_i = (v_1^b\vec{b}_1 + v_2^b\vec{b}_2 + v_3^b\vec{b}_3) \cdot \vec{a}_i \\ &= \sum_{j=1}^{3} v_j^b(\vec{a}_i \cdot \vec{b}_j) \end{aligned} \tag{2.11}$$

The coordinate transformation from frame b to frame a is given by

$$\mathbf{v}^a = \mathbf{R}_b^a \mathbf{v}^b \tag{2.12}$$

where

$$\mathbf{R}_b^a = \{\vec{a}_i \cdot \vec{b}_j\} \tag{2.13}$$

is the rotation matrix from a to b. The elements of the rotation matrix are called the direction cosines.

Properties of the rotation matrix

In this section a few properties of the rotation matrix will be listed. For more details around these properties refer to Egeland and Gravdahl [2002]. The rotation matrix from b to a can be found in the same way as from a to b by interchanging a and b in the expressions which gives

$$\mathbf{R}_a^b = \{\vec{b}_i \cdot \vec{a}_j\} \tag{2.14}$$

The rotation matrix is orthogonal and satisfies

$$\mathbf{R}_a^b = (\mathbf{R}_b^a)^{-1} = (\mathbf{R}_b^a)^T \tag{2.15}$$

Furthermore, the set of all matrices that are orthogonal and with a determinant equal to unity is denoted by $SO(3)$, which is

$$SO(3) = \{\mathbf{R} | \mathbf{R} \epsilon R^{3\times 3},\ \mathbf{R^T R} = \mathbf{I} \text{ and } \det \mathbf{R} = 1\} \tag{2.16}$$

where $R^{3\times 3}$ is the set of all 3×3 matrices with real elements.

Simple Rotations

Simple rotations, which are rotation around a fixed axis, using Euler angles as parameters are defined in the following way

$$\begin{aligned}
\mathbf{R}_x(\phi) &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix} \\
\mathbf{R}_y(\theta) &= \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \\
\mathbf{R}_z(\psi) &= \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned} \tag{2.17}$$

where x, y and z are the axes' which the angles ϕ, θ and ψ revolves around. Refer to chapter 2.2.1.

2.2 Attitude

2.2.1 Euler angles

The Euler angles, $[\psi\ \theta\ \phi]$, are a widely used set of parameters for the rotation matrix. A typical set used for describing the motion of spacecrafts are the roll-pitch-yaw angles. Using these parameters, a rotation matrix describing a rotation from a to b is given as

$$\mathbf{R}_b^a = \mathbf{R}_{x,y,z}(\psi,\theta,\phi) = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) \tag{2.18}$$

where $\mathbf{R}_z(\psi)$, $\mathbf{R}_y(\theta)$ and $\mathbf{R}_x(\phi)$ are the simple rotation matrices from equation (2.17). This yields

$$\mathbf{R}_{x,y,z}(\psi,\theta,\phi) = \begin{pmatrix} c\theta c\psi & s\theta s\phi c\psi - c\phi s\psi & s\theta c\phi s\phi + s\phi s\psi \\ c\theta s\psi & s\theta s\phi s\psi + c\phi c\psi & s\theta c\phi s\phi - s\phi c\psi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{pmatrix} \tag{2.19}$$

where $cos = c$ and $sin = s$ is used for shortening. The rotation matrix is singular for $\theta = \pm 90\,\text{deg}$.

2.2.2 Unit Quaternions

A unit quaternion is a quaternion with unit length and is written as

$$\mathbf{p} = \begin{pmatrix} \eta \\ \epsilon \end{pmatrix} \tag{2.20}$$

and satisfies

$$\mathbf{p}^T\mathbf{p} = \eta^2 + \epsilon^T\epsilon = 1 \tag{2.21}$$

where ϵ is a vector $\epsilon = [\epsilon_1\ \epsilon_2\ \epsilon_3]^T$. Quaternions are used to overcome the problems with singularities in the attitude representation. By using quaternions, rotations can be expressed by the quaternion product.

The quaternion product

It is shown in Egeland and Gravdahl [2002] that the quaternion product of two unit quaternions $\mathbf{p_1}$ and $\mathbf{p_2}$ is a unit quaternion

$$\mathbf{p} := \mathbf{p_1} \otimes \mathbf{p_2} = \begin{pmatrix} \eta_1\eta_2 - \epsilon_1^T\epsilon_2 \\ \eta_1\epsilon_\mathbf{2} + \eta_2\epsilon_1 + \epsilon_1^\times\epsilon_2 \end{pmatrix} \tag{2.22}$$

where $\mathbf{p_1}$ and $\mathbf{p_2}$ are quaternions.

2.3 Reference frames

This section describes the various reference frames relevant to a satellite.

2.3.1 Earth-centered inertial (ECI) frame

The ECI reference frame, denoted i, is non-accelerating where Newton's laws of motion apply. The center is denoted x_i y_i z_i and is located in the center of the Earth with the z-axis pointing toward the North Pole. The x-axis points toward vernal equinox and the y-axis completes the Cartesian coordinate system. The frame is fixed in space.

2.3.2 Earth-centered Earth-fixed (ECEF) reference frame

The ECEF reference frame, denoted e and given by x_e y_e z_e, has its origin fixed in the center of the Earth and the axes rotate relative to the inertial frame ECI. The frequency of the rotation is approximately $\omega_e = 7.2921 \cdot 10^{-5}$ rad/s. The z-axis points toward the north pole and the x-axis points toward the intersection between the Greenwich meridian and the equator.

2.3.3 Earth-Centered Orbit frame

The frame is denoted *oc* and has its center in the Earths center, the x-axis points toward perigee, the y-axis points along the semiminor-axis and z-axis is perpendicular to the orbital plane so that it completes the right hand Cartesian coordinate system. The perigee and semiminor axis are further explained in chapter 2.5.1.

2.3.4 Orbit frame

The orbit frame has its center in the satellites center of mass. The origin rotate relative to the ECI frame with an angular velocity ω_o. The z-axis points toward the center of the Earth, the x-axis is the normal direction of the orbital plane and it is important to notice that the tangent is only perpendicular to the radius vector in the case of a circular orbit. For elliptic orbits the x-axis does not align with the satellites velocity vector, figure 2.2 shows this. The frame is denoted o.

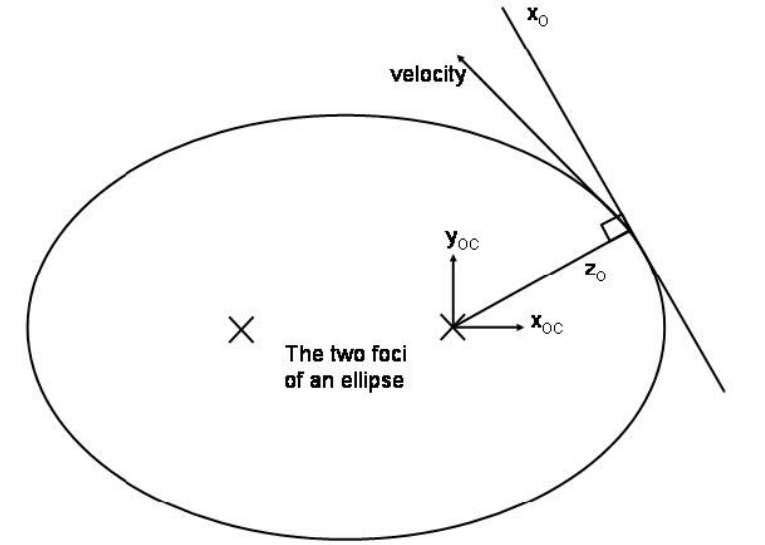

Figure 2.2: Orbit frame

2.3.5 Body frame

The body-fixed frame, denoted b and given by x_b y_b z_b, is a moving coordinate frame fixed to the satellite. The x_b-axis points forward, the z_b-axis normally points through the nadir-side of the satellite and y_b-axis completes the Cartesian coordinate system. The position and orientation of the satellite are described relative to the ECI frame.

2.4 Transformation between frames

In this section the methods used to rotate between different frames is presented.

Earth-Centered Orbit to ECEF and ECI

This transformation require knowledge of the objects orbit. The notation used for defining such an orbit will be shown in the next chapter, under the magnetic reference model, chapter 2.5.1. From Svartveit [2003] we have that the rotation matrices are

$$\begin{aligned} \mathbf{R}_{oc}^{e} &= \mathbf{R}_z(-\Omega + \theta)\mathbf{R}_x(-i)\mathbf{R}_z(-\omega) \\ \mathbf{R}_{oc}^{i} &= \mathbf{R}_z(-\Omega)\mathbf{R}_x(-i)\mathbf{R}_z(-\omega) \end{aligned} \tag{2.23}$$

where Ω is the right ascension of ascending node, i is the inclination of the satellite, ω is the argument of perigee and θ is the ascension of the zero meridian. $\mathbf{R}_x$ and $\mathbf{R}_z$ are the simple rotation matrices defined in equation (2.17). Ω, i, ω and θ are Keplerian elements and are defined in figure 2.3.

ECEF to ECI

Rotation between ECEF and ECI is a rotation about the coincident z_i and z_e axes. This can thus be described as a simple rotation from equation (2.17) with $\alpha = \omega_{ie}t$. ω_{ie} is the Earths rotation given in equation (2.3.2) and t is the time since the ECEF and ECI frames were aligned. The rotation α is negative right handed and given the fact that $\cos(-\alpha) = \cos(\alpha)$ and $\sin(-\alpha) = -\sin(\alpha)$ we get the following rotation matrix

$$\mathbf{R}_{e}^{i} = \mathbf{R}_{zi}(-\alpha) = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.24}$$

ECI to Orbit frame

The rotation between ECI and Orbit frame is dependent on the satellites rotation velocity ω_o. The orbit frame is rotated about the y_i-axes by an angle $\beta = \beta_0 + \omega_o t$, where β_0 is the latitude position of the satellite and t is the time since it last passed the 0° latitude. The rotation matrix is thus

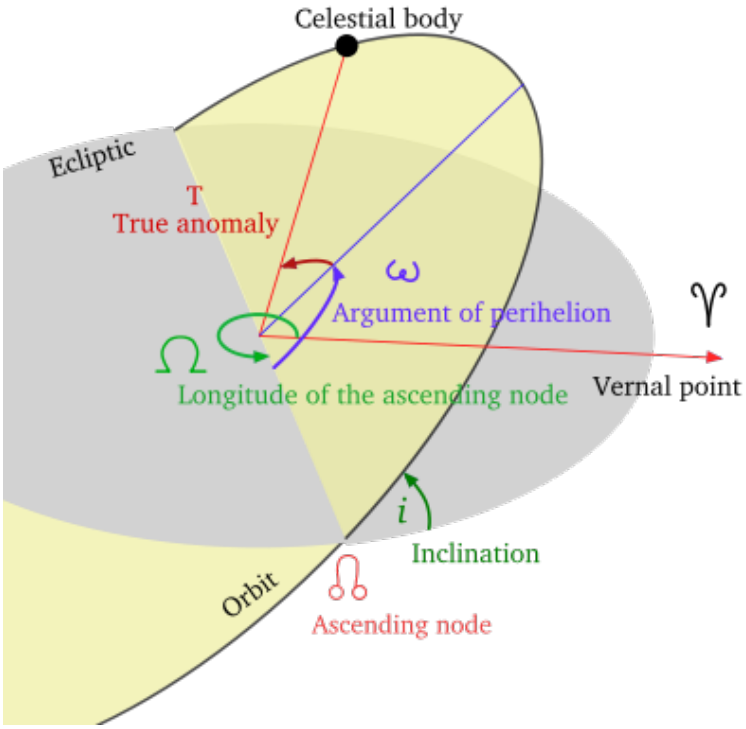

Figure 2.3: Keplerian elements. Wikipedia [2007*b*]

given as

$$\mathbf{R}_i^o = \mathbf{R}_{xi,\pi}\mathbf{R}_{yi,\beta} = \begin{pmatrix} \cos\mu & 0 & \sin\mu \\ 0 & -1 & 0 \\ \sin\mu & 0 & -\cos\mu \end{pmatrix} \tag{2.25}$$

where $\mathbf{R}_{xi,\pi}$ is introduced because the orbit frame is upside-down relative to the ECI frame and $\mu = \beta_0 + \omega_o t$.

Orbit frame to Body frame

The rotation between orbit and body frame is used to determine the Euler parameters of the satellite. To derive this rotation matrix we start describing

the rotation matrix $\mathbf{R}_b^o$ using Euler parameters.

$$\mathbf{R}_b^o = \mathbf{R}_e(\eta, \epsilon) = \begin{pmatrix} \eta^2 + \epsilon_1^2 - \epsilon_2^2 - \epsilon_3^2 & 2(\epsilon_1\epsilon_2 - \eta\epsilon_3) & 2(\epsilon_1\epsilon_3 + \eta\epsilon_2) \\ 2(\epsilon_1\epsilon_2 + \eta\epsilon_3) & \eta^2 - \epsilon_1^2 + \epsilon_2^2 - \epsilon_3^2 & 2(\epsilon_2\epsilon_3 - \eta\epsilon_1) \\ 2(\epsilon_1\epsilon_3 - \eta\epsilon_2) & 2(\epsilon_2\epsilon_3 + \eta\epsilon_1) & \eta^2 - \epsilon_1^2 - \epsilon_2^2 + \epsilon_3^2 \end{pmatrix} \tag{2.26}$$

The rotation from orbit to body frame can now be found by using equation 2.15 which yields

$$\mathbf{R}_o^b = (\mathbf{R}_b^o)^T = \begin{pmatrix} \eta^2 + \epsilon_1^2 - \epsilon_2^2 - \epsilon_3^2 & 2(\epsilon_1\epsilon_2 + \eta\epsilon_3) & 2(\epsilon_1\epsilon_3 - \eta\epsilon_2) \\ 2(\epsilon_1\epsilon_2 - \eta\epsilon_3) & \eta^2 - \epsilon_1^2 + \epsilon_2^2 - \epsilon_3^2 & 2(\epsilon_2\epsilon_3 + \eta\epsilon_1) \\ 2(\epsilon_1\epsilon_3 + \eta\epsilon_2) & 2(\epsilon_2\epsilon_3 - \eta\epsilon_1) & \eta^2 - \epsilon_1^2 - \epsilon_2^2 + \epsilon_3^2 \end{pmatrix} \tag{2.27}$$

This rotation matrix can also be written as

$$\mathbf{R}_o^b = \begin{bmatrix} \mathbf{c}_1^b & \mathbf{c}_2^b & \mathbf{c}_3^b \end{bmatrix} \tag{2.28}$$

where $\mathbf{c}_i^b = [\mathbf{c}_{ix}^b \ \mathbf{c}_{iy}^b \ \mathbf{c}_{iz}^b]^T$. When z_o and z_b are aligned, $\mathbf{c}_3^b = [0\ 0\ \pm 1]$ which gives us a quantity of the deviation between the two frames Ose [2004].

2.5 Reference models

The Attitude determination system has two different sensor systems, one for measuring the magnetic field around the satellite and one for measuring the light level on each side of the satellite. In order to use the measured values a reference model for each sensor system is needed for comparison. In this chapter both models are presented.

2.5.1 Magnetic

A magnetometer is used to measure the local magnetic field of the Earth. It measures the magnetic field in three axes in the sensor frame. By aligning the magnetometer with the satellites body frame no extra transformations are needed. The senor data is compared to a reference model of Earths magnetic field. The most commonly used reference model is the International Geometric Reference Field model, IGRF. This model is fixed in the ECEF

frame and to in order to use it we need to transform it to the orbit frame. For this an orbit estimator is needed. When describing an orbit, Keplerian elements are used. They are shown in figure 2.3 and a short description of each element follows.

Orbital Inclination, i, is the angle between the orbital and equatorial plane. If the inclination is 0° the orbit is called equatorial and if the angle is 90° the orbit is said to be polar.

Right Ascension of Ascending Node, Ω, is the angle between the ascending node and vernal equinox. When the satellite passes from south to north we get an ascending node and when it passes from north to south we get a descending node.

Argument of Perigee, ω, is the angle between the line from perigee through the Earth to the apogee and the line of nodes. The line of nodes is the intersection of the equatorial plane and the orbital plane.

Eccentricity, e, is given as

$$e = \sqrt{1 - \frac{b^2}{a^2}} \tag{2.29}$$

where a is the semimajor-axis and b is the semiminor-axis, both shown in figure 2.4.
The satellites position in orbit is described by the following time varying Keplerian elements, which use the previous four elements describing the orientation.

Mean Motion, n, is the average angular velocity and describes the size of the ellipse. It is related to the semimajor axis through Keplers third law

$$n^2 a^3 = \mu_e \tag{2.30}$$

where $\mu_e = GM_e$ is the Earths gravitational constant. This relationship is the reason why mean motion is sometimes replaced by the semimajor axis when describing a satellite orbit.

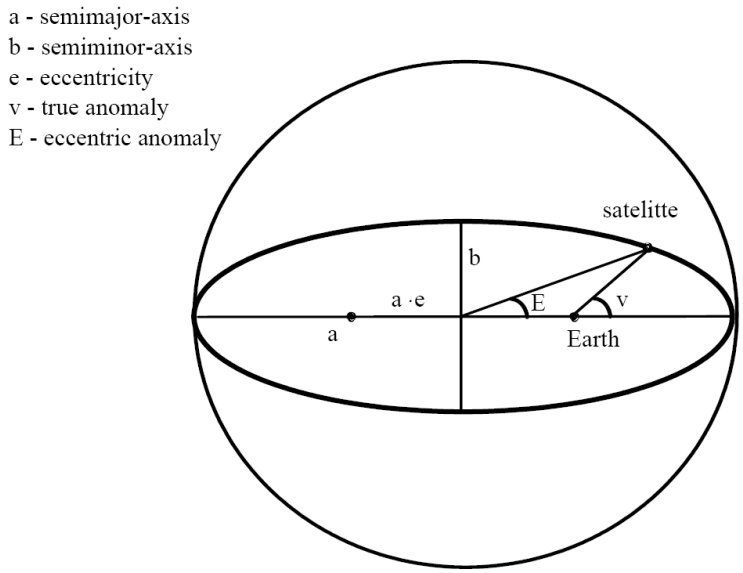

Figure 2.4: Keplerian elements.

Mean Anomaly, M, is used to describe the position of the satellite in the ellipse. The mean anomaly is an angle defined from perigee and increases uniformly from 0° to 360° during one revolution. When the orbit is a non-circular ellipse this does not point directly toward the satellite except at perigee and apogee.

Figure 2.4 shows different anomalies. True anomaly, v, is the direction from the earth to the satellite. 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. The circles center coincides with the center of the orbital ellipse and has a radius equal to the semimajor axis.

Orbit estimator

In Svartveit [2003] several possible orbit estimators were studied and a recommendation given. The recommendation was based on the available power of a small satellite.To create the orbit estimator a vector from the center of the Earth to the satellite is needed. This vector is decomposed in the

Earth-Centered Orbit frame and is

$$\mathbf{r}^{oc} = a \begin{bmatrix} \frac{\cos E - e}{\sqrt{1-e^2 \sin E}} \\ 0 \end{bmatrix} \tag{2.31}$$

With this vector, equations (2.23) can be used to implement the orbit estimator in ECI and ECEF frames. This yields

$$\begin{aligned} \mathbf{r}^i &= \mathbf{R}_z(-\Omega)\mathbf{R}_x(-i)\mathbf{R}_z(-\omega)\mathbf{r}^{oc} \\ \mathbf{r}^e &= \mathbf{R}_z(-\Omega+\theta)\mathbf{R}_x(-i)\mathbf{R}_z(-\omega)\mathbf{r}^{oc} \end{aligned} \tag{2.32}$$

where Ω, i, ω and θ are the Keplerian elements described above. Svartveit [2003] described how the accuracy of the estimator will degrade over time due to the non spheric Earth, $\dot{\Omega}_{J2}$ and $\dot{\omega}_{J2}$, the Sun, $\dot{\Omega}_{sun}$ and $\dot{\omega}_{sun}$ and the Moon, $\dot{\Omega}_{moon}$ and $\dot{\omega}_{moon}$. An improved Orbit estimator can be obtained by implementing these perturbations, but will result in a slightly more complicated model. The estimator will now be

$$\begin{aligned} \mathbf{R}^e_{oc} = \mathbf{R}_z(&-\Omega_0 + (\dot{\Omega}_{J2} + \dot{\Omega}_{sun} + \dot{\Omega}_{moon})t + \theta_0 + \omega_{ei})\mathbf{R}_x(-i) \\ &\mathbf{R}_z(-(\omega_O + (\dot{\omega}_{J2} + \dot{\omega}_{sun} + \dot{\omega}_{moon})t))a \begin{bmatrix} \frac{\cos E - e}{\sqrt{1-e^2 \sin E}} \\ 0 \end{bmatrix} \end{aligned} \tag{2.33}$$

IGRF

The Earths magnetic Field is shown in figure 2.5. The field is highly varying and a simplified model is needed. The International Association of Geomagnetism and Aeronomy offers The International Geomagnetic Reference Field model, IGRF, a model which is acceptable for a wide variety of users and revised every fifth year. The IGRF model is rotating with the Earth and thus given in the ECEF frame. More information on IGRF can be found on the IGRF homepage, *http://www.ngdc.noaa.gov/IAGA/vmod/igrf.html*. To transform the model to orbit frame we first need to transform it to the ECO frame. This is done using the property of equation (2.15) on $\mathbf{r}_E$ from equation (2.32) which yields

$$\begin{aligned} & \mathbf{B}^{oc} = (\mathbf{R}_z(-\Omega+\theta)\mathbf{R}_x(-i)\mathbf{R}_z(-\omega))^{-1}\mathbf{B}^e \\ \Rightarrow \quad & \mathbf{B}^{oc} = \mathbf{R}_z(\omega)\mathbf{R}_x(i)\mathbf{R}_z(\Omega-\theta)\mathbf{B}^e \end{aligned} \tag{2.34}$$

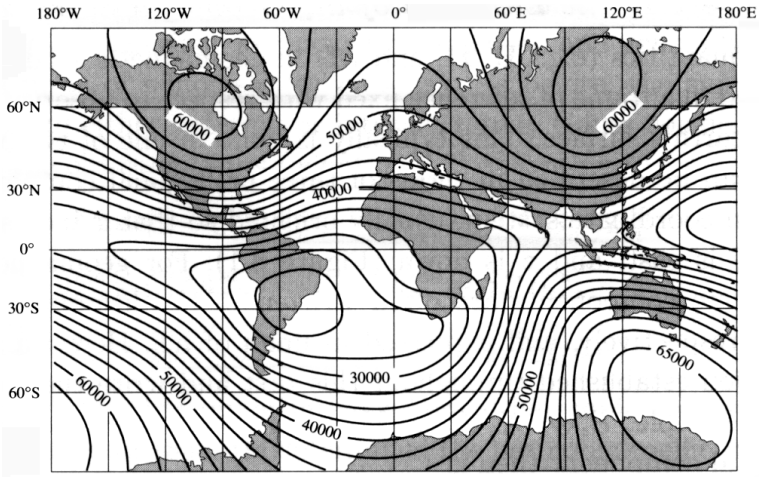

Figure 2.5: Magnitude of Earth's magnetic field

In these equations $\mathbf{B}^e$ is the vector from the IGRF model. The model in orbit frame is found from

$$\mathbf{B}^o = (\mathbf{R}_x(\frac{\pi}{2})\mathbf{R}_z(\nu + \frac{\pi}{2})\mathbf{B}^{oc} \tag{2.35}$$

where ν is the true anomaly.

2.5.2 Light

The second sensor are six Light to Frequency converters which measures the light on each side of the satellite. These measurements are used to describes the suns position relative to the satellites body frame which in turn is used to determine the attitude of the satellite. In order to use such a sensor, knowledge of the suns position relative to the satellites orbital position is needed for comparison. Also, when implementing a sun sensor, the effects of the Earth Albedo error must be taken into account. This is explained later in this section.

Sun model

To utilize the measured body frame sun vector, the sun vector in orbit frame must be known and a rotation between the two could be estimated. For the computation of the sun vector the Earth's orbit is assumed to be circular with an orbit time of 365 days. Furthermore the satellite is assumed to be

positioned in the center of the Earth. This will introduce an error to the model, but the magnitude of the error is not larger then $e = \arctan \frac{Ra}{Re} \approx 4.65e - 5\ rad$, where R_a is the radius of the satellite orbit and R_e is the Earth orbit radius. The model presented here is based on work done by Ose [2004]. The model is presented by describing the motion of the Sun as seen from the Earth and is shown in figure 2.6. The elevation of the Sun, ϵ_s,

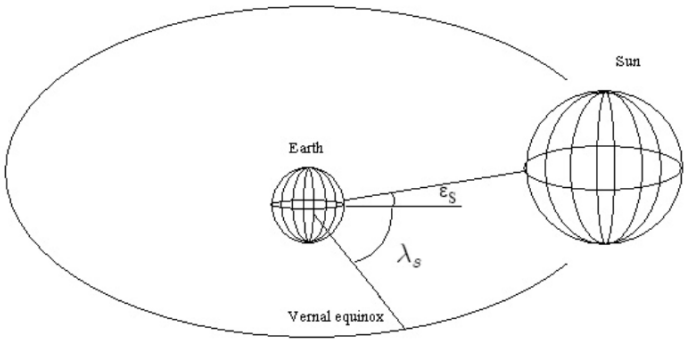

Figure 2.6: The Suns position relative to the Earth

varies between 23° and −23° depending on the time of the year. The period is given by

$$\epsilon_s = \frac{23\pi}{180} \sin(\frac{T_s}{365} 2\pi) \tag{2.36}$$

where T_s is the time since first day of spring, which also gives the starting point of the period at 0°. The Suns orbit around the Earth is given by

$$\lambda_s = \frac{T_s}{365} 2\pi \tag{2.37}$$

where λ_s is called the Suns orbit parameter. The Suns position when the Earth passes vernal equinox is

$$\mathbf{s}_0^i = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T \tag{2.38}$$

and with this vector we can express the Suns position at a given time as

$$\mathbf{s}^i = \mathbf{R}_y(\epsilon_s)\mathbf{R}_z(\lambda_s)\mathbf{s}_0^i \tag{2.39}$$

where $\mathbf{R}_y$ and $\mathbf{R}_z$ are the simple rotations given in equation (2.17). The Suns position relative to the satellite can be found by transforming equation (2.39) from the ECI frame to the Orbit frame using equation (2.25). This results in

$$\mathbf{s}^o = \mathbf{R}_i^o \mathbf{s}^i \tag{2.40}$$

Earth Albedo error

From the satellites position there are two major light sources from which the Light-to-Frequency converters will pick up light from. One is the Sun and the second is reflected light from the Earth, known as Earth Albedo light. If the Sun measurements are to be used in attitude determination, a correction system is needed. The reflected light varies due to the different reflective capabilities of the Earth's surface. The model of the Albedo light is a polynomial function which has proved to give good results in attitude determination, Appel [2004].

2.6 Satellite model

To simulate the satellite in its environment a mathematical model is needed. The model is also the basis for the Kalman Filter presented in chapter 3.

Kinematics

The attitude motion of the satellite is found by integrating the velocities. As mentioned earlier, quaternions are used to prevent the existence of singularities. Egeland and Gravdahl [2002] gives the differential equations used to describe the kinematics

$$\begin{aligned} \dot{\eta} &= -\frac{1}{2}\epsilon^T \omega_{ob}^b \\ \dot{\epsilon} &= \frac{1}{2}[\eta \mathbf{I} - S(\epsilon)]\omega_{ob}^b \end{aligned} \tag{2.41}$$

Dynamics

The satellite dynamics can be derived from elementary mechanics. By following the derivation made in Kyrkjebø [2000] we find that the dynamics,

using Euler's moment equation, can be expressed as

$$\mathbf{I}^b\dot{\omega}^b_{ib} + \omega^b_{ib} \times \mathbf{I}^b\omega^b_{ib} = \tau^b \tag{2.42}$$

where $\mathbf{I}^b$ is the inertia of the satellite given in body frame, ω^b_{ib} is the angular velocity of the body frame relative to the ECI frame expressed in body frame and τ^b is the sum of all torques acting on the satellite. By applying the skew symmetric operator the dynamics can be expressed as

$$\mathbf{I}^b\dot{\omega}^b_{ib} + S(\omega^b_{ib})\mathbf{I}^b\omega^b_{ib} = \tau^b \tag{2.43}$$

The angular velocity of the satellite relative to the inertial frame can be expressed in the body frame as

$$\omega^b_{ib} = \omega^b_{io} + \omega^b_{ob} = \mathbf{R}^b_o\omega^o_{io} + \omega^b_{ob} \tag{2.44}$$

where $\mathbf{R}^b_o$ is given in equation (2.27) and

$$\omega^o_{io} = \begin{bmatrix} 0 & -w_o & 0 \end{bmatrix}^T \tag{2.45}$$

since the orbit frame revolves relative to the inertial frame with the angular velocity of ω_o. This in turn gives the angular velocity in the body frame relative to the inertial frame as

$$\omega^b_{ib} = \omega^b_{ob} - \omega_o\mathbf{c}_2 \tag{2.46}$$

where $\mathbf{c}_2$ is the direction cosine from $\mathbf{R}^b_o$ given in equation (2.28). ω_o is derived using Newton's laws.

There are two forces acting on the satellite in orbit, the centripetal force and the gravitational force. For the satellite to maintain a stable orbit these two forces must be equal and ω_o can be found by representing these equations as scalars in a orbit with radius R, and with M and m as the mass of the Earth and the satellite respectively.

$$m\frac{v_o^2}{R} = G\frac{Mm}{R^2} \rightarrow v_o^2 = \frac{GM}{R} \tag{2.47}$$

where G is the gravitational constant of the Earth. The orbit velocity $v_o =$

$R\omega_o$, which gives us the angular velocity of the orbit frame

$$\omega_o = \sqrt{\frac{GM}{R^3}} \tag{2.48}$$

Gravitational torque

A satellite is subject to gravitational forces from the gravitational field of planets, the Moon and the Sun. At an altitude of 600 km the forces on the satellite other then the ones from the Earth is negligible and this force can be found through Newton's law of gravitation. Kyrkjebø [2000] modeled the gravitation torque on the satellite as

$$\mathbf{g}^b = 3\omega_o^2 \mathbf{c}_3 \times \mathbf{I}^b \mathbf{c}_3 \tag{2.49}$$

where $\mathbf{c}_3$ is the direct cosine from $\mathbf{R}_o^b$ defined in (2.27). The gravitational torque is used to represent the exact motion of the satellite, but will not be used in attitude determination.

Nonlinear state space model

The behavior of the satellite is described by the state space model. This model is described using the dynamical model from equation (2.42) and the attitude motion described in equations (2.41). We need a state space model which gives us the satellite attitude relative to orbit frame and thus we need to transform the angular velocity in the dynamical model to body frame. The state vector is chosen to be

$$\mathbf{x} = [\mathbf{q}\ \omega_{ob}^b]^T = [\eta\ \epsilon_1\ \epsilon_2\ \epsilon_3\ \omega_{ob,x}^b\ \omega_{ob,y}^b\ \omega_{ob,z}^b]^T \tag{2.50}$$

The dynamical equation

$$\mathbf{I}^b \dot{\omega}_{ib}^b + \omega_{ib}^b \times \mathbf{I}^b \omega_{ib}^b = \tau^b \tag{2.51}$$

can be rewritten in terms of angular velocity relative to orbit frame by using equation (2.44). By rearranging the equation and differentiating the expression with respect to time, keeping in mind that ω_{io}^o is constant, we get

$$\dot{\omega}_{ib}^b = \dot{\omega}_{ob}^b - S(\omega_{ob}^b)\mathbf{R}_o^b \omega_{io}^o \tag{2.52}$$

where the relation $\omega_{bo}^{b} = -\omega_{ob}^{b}$ is used. The dynamic equation for the satellite is now given as

$$\dot{\omega}_{ob}^{b} = (\mathbf{I}^{b})^{-1}[-(\omega_{o}^{b}b + \mathbf{R}_{o}^{b}\omega_{io}^{o}) \times (\mathbf{I}^{b} \cdot (\omega_{o}^{b}b + \mathbf{R}_{o}^{b}\omega_{io}^{o})) + \tau^{b}] + S(\omega_{ob}^{b})\mathbf{R}_{o}^{b}\omega_{io}^{o} \quad (2.53)$$

With the new dynamical equation the nonlinear state space model becomes

$$\dot{x} = \begin{bmatrix} \dot{\eta} \\ \dot{\epsilon} \\ \dot{\omega}_{ob}^{b} \end{bmatrix} = \mathbf{f}(\mathbf{x}, \tau, t) \quad (2.54)$$

where $\mathbf{f}(\mathbf{x}, \tau, t)$ is defined to be

$$\mathbf{f}(\mathbf{x}, \tau, t) = \begin{bmatrix} -\frac{1}{2}\epsilon^{T}\omega_{ob}^{b} \\ \frac{1}{2}[\eta\mathbf{I} - S(\epsilon)]\omega_{ob}^{b} \\ (\mathbf{I}^{b})^{-1}[-(\omega_{o}^{b}b + \mathbf{R}_{o}^{b}\omega_{io}^{o}) \times (\mathbf{I}^{b} \cdot (\omega_{o}^{b}b + \mathbf{R}_{o}^{b}\omega_{io}^{o})) + \tau^{b}] + S(\omega_{ob}^{o})\mathbf{R}_{o}^{b}\omega_{io}^{o} \end{bmatrix} \quad (2.55)$$

Linearized model

A linearized model of the satellite attitude is found by differentiating the nonlinear model with respect to the state vector. A linear system matrix can be found from

$$\mathbf{F} = \frac{\delta \mathbf{f}(\mathbf{x}, \tau, t)}{\delta \mathbf{x}} \quad (2.56)$$

F is computed by performing a partial differentiation on the nonlinear model with respect to the state vector $\mathbf{x}$ as shown

$$\mathbf{F} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_7} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_7}{\partial x_1} & \cdots & \frac{\partial f_7}{\partial x_7} \end{bmatrix} \quad (2.57)$$

It is practical to divide the linearization process into two parts; one attitude part consisting of the Euler parameters and one angular velocity part. This means that the derived linear matrix $\mathbf{F}$ can be separated in the same way, resulting in

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_{att} \\ \mathbf{F}_{vel} \end{bmatrix} = \begin{bmatrix} \frac{\partial \dot{\mathbf{q}}}{\partial x_1} & \cdots & \frac{\partial \dot{q}}{\partial x_7} \\ \frac{\partial \dot{\omega}_{ob}^{b}}{\partial x_1} & \cdots & \frac{\partial \dot{\omega}_{ob}^{b}}{\partial x_7} \end{bmatrix} \quad (2.58)$$

The complete differentiation results in the two matrices

$$\mathbf{F}_{att} = \begin{bmatrix} 0 & -\frac{1}{2}\omega_{ob}^{b} & -\frac{1}{2}\epsilon^{T} \\ \frac{1}{2}\omega_{ob}^{b} & -\frac{1}{2}\mathbf{S}(\omega_{ob}^{b}) & \frac{1}{2}[\eta\mathbf{I} + \mathbf{S}(\epsilon)] \end{bmatrix} \tag{2.59}$$

and

$$\mathbf{F}_{vel} = \begin{bmatrix} a51 & a52 & a53 & a54 & 0 & a56 & a57 \\ a61 & a62 & a63 & a64 & a65 & 0 & a67 \\ a71 & a72 & a73 & a74 & a75 & a76 & 0 \end{bmatrix} \tag{2.60}$$

where the elements are calculated in appendix A.

Chapter 3

Kalman Filter

A lot of work has previously been done to find the type of Kalman Filter (KF) that will yield the best results based on the computation time available in the microcontroller. This section will present the extented Kalman filter and introduce a new method called the Unscented Kalman Filter.

The Kalman filter was developed by Rudolf Kalman and is a recursive filter that estimates the state of a dynamic system from a series of measurements which can be noisy. They are based on a linear dynamical systems which has been discretized in the time domain. The Kalman filter is also known as an estimator because it is used to estimate the current state in a system based on the previous time step and a new set of measurements. The Kalman filter has two phases, one predict phase and one update phase. In the predict phase, the state estimate from the previous time step is used to produce a estimate of the current state or current time step. In the update phase, new measurements are used to improve the current prediction and create a more accurate state estimate for the current time step. In reality few systems are linear and there is thus a need for a Kalman filter which can be used on non-linear systems, such as a satellite in orbit.

3.1 Extended Kalman Filter

This section is partly based on the work done in Sunde [2004]. In an Extended Kalman Filter, (EKF), the state transition and observations do not need to be a linear function. The EKF is designed using the model derived

in (2.54), consisting of a nonlinear state-space model that includes the measurement and process noise. Since the Kalman filter is being implemented on a microcontroller we use a discrete model. The discrete model, based on continuous system discretization theory, is given by

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{x}_k) + \mathbf{w}_k \\ \mathbf{y}_k &= \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \end{aligned} \tag{3.1}$$

where k indicates the sample step, $\mathbf{f}(\mathbf{x}_k)$ is the nonlinear function (2.55) with τ^B set to zero, $\mathbf{w}_k$ and $\mathbf{v}_k$ are process and measurement noise respectively. τ^B is set to zero because the magnetic coils used to control the attitude generates a magnetic field which corrupts the magnetometer measurements and is thus turned off during estimation. The extended Kalman filter algorithm used for the system defined in equation (3.1) is

$$\mathbf{K}_k = \bar{\mathbf{P}}_k \mathbf{H}_k^T [\mathbf{H}_k \bar{\mathbf{P}}_k \mathbf{H}_k^T + \mathbf{R}]^{-1} \tag{3.2}$$

$$\hat{\mathbf{x}}_k = \bar{\mathbf{x}}_k + \mathbf{K}_k [\mathbf{y}_{m,k}^b - \mathbf{H}_k \bar{\mathbf{x}}_k] \tag{3.3}$$

$$\mathbf{P}_k = [\mathbf{I} - \mathbf{K}_k \mathbf{H}_k] \bar{\mathbf{P}}_k [\mathbf{I} - \mathbf{K}_k \mathbf{H}_k]^T + \mathbf{K}_k \mathbf{R} \mathbf{K}_k^T \tag{3.4}$$

$$\bar{\mathbf{x}}_{k+1} = \mathbf{\Phi}_k \hat{\mathbf{x}}_k \tag{3.5}$$

$$\mathbf{P}_{k+1}^- = \mathbf{\Phi}_k \mathbf{P}_k \mathbf{\Phi}_k^T + \mathbf{Q} \tag{3.6}$$

where (3.2) is the Kalman filter gain matrix, (3.3) is the update of the state estimate, (3.4) is the update of the error covariance matrix, (3.5) is the state estimation propagation and (3.6) is the error covariance propagation. $\mathbf{R}$ and $\mathbf{Q}$ are the expected covariance of the measurement noise $\mathbf{v}$ and the process noise $\mathbf{w}$. $\mathbf{y}_{m,k}^b$ is the measurement from the sensors. $\mathbf{\Phi}_k$, used in the propagation estimates, is derived using forward Euler integration defined as

$$\mathbf{\Phi}_k = \mathbf{I} + \frac{\partial \mathbf{f}(\mathbf{x}_k)}{\partial \mathbf{x}_k} \, |_{\mathbf{x}_k = \hat{\mathbf{x}}_k} \tag{3.7}$$

The state estimation update given in equation (3.3) can be divided in two, one for the quaternion estimate part and one for the angular velocity estimate. The quaternion update can be interpreted as a rotation and the quaternion product given by

$$\hat{\mathbf{q}}_k = \bar{\mathbf{q}}_k \otimes \mathbf{K}_{q,k} \nu_k \tag{3.8}$$

is used in the update. The angular velocity estimate is updated using equation

$$\hat{\omega}^b_{ob,k} = \bar{\omega}^b_{ob,k} + \mathbf{K}_{\omega,k}\nu_k \tag{3.9}$$

ν_k is the result of the innovation process given by

$$\nu_k = \mathbf{y}^b_{m,k} - \mathbf{H}_k\bar{\mathbf{x}}_k \tag{3.10}$$

which is the difference between the real and the predicted measurement.

3.2 Euler parameters

The use of Euler parameters in the filter is not a straight forward process and care must be taken to avoid errors. It is important that the constraint on the quaternion norm is maintained when used and a quaternion normalization algorithm can be used for this purpose. The quaternion in the state estimation update and the state estimation propagation equations must be normalized to maintain the physical content of the unit quaternion. The use of normalization in this part makes it difficult to maintain a singular covariance matrix $\mathbf{P}_k$ due to numerical roundoff. To overcome this problem the dimension of $\mathbf{P}$ is reduced by one, removing η from the state vector. This leads to a reduced model

$$\begin{aligned} \mathbf{x}_{r,k+1} &= \begin{bmatrix} \epsilon_{k+1} \\ \omega^b_{ob,k+1} \end{bmatrix} = \mathbf{x}_{r,k} + \mathbf{f}_r(\mathbf{x}_{r,k}) + \mathbf{w}_{r,k} \\ \mathbf{y}_{r,k} &= \mathbf{H}_{r,k} + \mathbf{v}_{r,k} \end{aligned} \tag{3.11}$$

where r denotes that it is a reduced model. Equation (3.11) is used when calculating the Kalman gain matrix, the error covariance update and the error covariance propagation. Equation (2.54) is however still used to calculate the state propagation in equation (3.5). Using equation (3.11), the filter equations can be rewritten as

$$\mathbf{K}_{r,k} = \bar{\mathbf{P}}_{r,k}\mathbf{H}^T_{r,k}[\mathbf{H}_{r,k}\bar{\mathbf{P}}_{r,k}\mathbf{H}^T_{r,k} + \mathbf{R}_r]^{-1} \tag{3.12}$$

$$\mathbf{P}_{r,k} = [\mathbf{I} - \mathbf{K}_{r,k}\mathbf{H}_{r,k}]\bar{\mathbf{P}}_{r,k}[\mathbf{I} - \mathbf{K}_{r,k}\mathbf{H}_{r,k}]^T + \mathbf{K}_{r,k}\mathbf{R}\mathbf{K}^T_{r,k} \tag{3.13}$$

$$\bar{\mathbf{P}}_{k+1} = \mathbf{\Phi}_{r,k}\mathbf{P}_{r,k}\mathbf{\Phi}^T_{r,k} + \mathbf{Q}_r \tag{3.14}$$

and

$$\mathbf{\Phi}_{r,k} = \mathbf{I} + \frac{\partial \mathbf{f}_r(\mathbf{x}_{r,k})}{\partial \mathbf{x}_{r,k}} \mid_{\mathbf{x}_{r,k}=\hat{\mathbf{x}}_{r,k}} \tag{3.15}$$

These new reduced matrices are the ones that is used in the implementation of the filter.

3.3 Gauss-Newton

There are two independent sensor systems on board the satellite, both of which are used to measure its attitude. This makes the satellite a multiple sensor system and the two sets of sensor data raises the question on how to use them. The problem faced is that there is no quaternion that converts the measurements taken in body frame exactly to the known reference values which are given in orbit frame. Sources of error are mentioned in Sunde [2004] and repeated here for convenience

- inherent sensor errors due to our crude sensor types
- variations in the magnetic and gravitational field
- sensor misalignment

This means that after the conversion the error needs to be minimized. There are two algorithms that can be used to for this, one is the Newton algorithm and the other is the Gauss-Newton algorithm. As Sunde [2004] concluded, due to our limitations with respect to computational power, the Gauss-Newton algorithm suites our problem best.

3.3.1 Gauss-Newton Algorithm

The Gauss-Newton method is a simplification of Newton's method used for nonlinear least-square problems. The algorithm is a numerical optimization algorithm that uses line search to minimize the squared error function, found in Nocedal and Wright [1999], given by

$$\mathbf{Q}^o = \epsilon^T \epsilon = (\mathbf{y}_r^o - \mathbf{M}\mathbf{y}_m^b)^T (\mathbf{y}_r^o - \mathbf{M}\mathbf{y}_m^b) \tag{3.16}$$

where $\mathbf{y}_r^o$ and $\mathbf{y}_m^b$ are the reference values in orbit frame and measurement values in body frame respectively and defined as

$$\begin{aligned} \mathbf{y}_r^o &= [\mathbf{B}_r^o \quad \mathbf{s}_r^o]^T \\ \mathbf{y}_m^b &= [\mathbf{B}_m^b \quad \mathbf{s}_m^b]^T \end{aligned} \tag{3.17}$$

$$\mathbf{M} = \begin{bmatrix} \mathbf{R}_b^o & 0 \\ 0 & \mathbf{R}_b^o \end{bmatrix} \tag{3.18}$$

where $\mathbf{R}_b^o$ is the rotation matrix from body to orbit frame. $\mathbf{R}_b^o$ is found by applying the properties in equation (2.15) and the fact that, Egeland and Gravdahl [2002]

$$\mathbf{R}_e(\eta, \epsilon)^T = \mathbf{R}_e(\eta, -\epsilon) \tag{3.19}$$

to $\mathbf{R}_o^b$ found in equation (2.27).

The Gauss-Newton method for the unit quaternion is given as

$$\hat{\mathbf{q}}_{g,k+1} = \hat{\mathbf{q}}_{g,k} - [\mathbf{J}^T(\hat{\mathbf{q}}_{g,k})\mathbf{J}(\hat{\mathbf{q}}_{g,k})]^{-1}\mathbf{J}^T(\hat{\mathbf{q}}_{g,k})\epsilon^o(\hat{\mathbf{q}}_{g,k}) \tag{3.20}$$

where $\mathbf{J}$ is the Jacobian matrix defined to be

$$\mathbf{J} = -\begin{bmatrix} (\frac{\partial \mathbf{M}}{\partial \eta_{g,k}}\mathbf{y}_m^b) & (\frac{\partial \mathbf{M}}{\partial \epsilon_{g,1,k}}\mathbf{y}_m^b) & (\frac{\partial \mathbf{M}}{\partial \epsilon_{g,2,k}}\mathbf{y}_m^b) & (\frac{\partial \mathbf{M}}{\partial \epsilon_{g,3,k}}\mathbf{y}_m^b) \end{bmatrix} \tag{3.21}$$

In Marins et al. [2000] we can read that this method has undergone extensive simulations and testing to show that the iteration algorithm converges in 3 to 4 steps.

3.4 Extended Kalman Filter with the Gauss-Newton method

The problem of implementing an EKF on a microcontroller is that the computations needed are to huge for the available computational power. This is why Sunde [2004] suggested the use of the Gauss-Newton method together with the EKF. The point is to reduce the size of the matrices that are used frequently in the EKF algorithm and from the reduced system equation (3.12), we can see that the measurement matrix $\mathbf{H}_{r,k}$ is used extensively in the computation of the Kalman gain matrix. Without the Gauss-Newton

method the measurements from the magnetic and sun sensors, $\mathbf{y}^b_{m,k}$, would be used directly in the EFK algorithm. This requires the computation of a 6 x 6 nonlinear $\mathbf{H}_{r,k}$-matrix at each sample step. The Kalman gain matrix in this case will also be a 6 x 6 matrix. By introducing the Gauss-Newton algorithm, (3.20), the measurement vector $\mathbf{y}^b_{m,k}$ is replaced with $\hat{\mathbf{q}}_{g,k}$ for the state estimate update. This change results in the measurement matrix $\mathbf{H}_{g,k}$ being reduced to a 3 x 6 constant matrix which in turn results in a 6 x 3 Kalman gain matrix. The computation required to produce $\mathbf{K}_k$ and $\mathbf{P}_k$ is reduced from 14000 to approximately 5500 matrix operations, Sunde [2004]. This reduction in the measurement matrix requires the computation of $\mathbf{q}_{g,k}$, a computation of 2500 operations.

The Gauss-Newton algorithm results in a sensor fusion and the measurements from the two independent sensor systems are represented in the form of a quaternion. This measurement is directly comparable with the estimated quaternion and the quaternion estimation error, $\Delta\mathbf{q}_k$, can be treated as a rotation. The innovation process is now found through the quaternion product

$$\Delta\mathbf{q}_k = \hat{\mathbf{q}}_{g,k} \otimes \bar{\mathbf{q}}_k^{-1} \tag{3.22}$$

where $\bar{\mathbf{q}}_k^{-1}$ is the conjugate of $\bar{\mathbf{q}}_k$. Since the system has been reduced and the Kalman gain does not contain η only $\Delta\epsilon$ of $\Delta q = [\Delta\eta_k \quad \Delta\epsilon_k]$ can be used for updating the estimated states. η is updated using the unit quaternion property, (2.21), resulting in $\Delta\eta = \sqrt{1 - \|\mathbf{K}_{\epsilon,k}\Delta\epsilon_k\|^2}$. The total quaternion estimate update becomes

$$\hat{\mathbf{q}}_k = \bar{\mathbf{q}}_{g,k} \otimes \begin{bmatrix} \sqrt{1 - \|\mathbf{K}_{\epsilon,k}\Delta\epsilon_k\|^2} \\ \Delta\epsilon_k \end{bmatrix} \tag{3.23}$$

where $\mathbf{K}_{\epsilon,k}$ is the top half of the Kalman gain matrix $\mathbf{K}_k$. The unit quaternion is maintained in the calculations for the error and update values due to the use of the quaternion product.

The angular rates, ω^b_{ob}, estimate is updated using the ordinary method and is given by

$$\hat{\omega}^b_{ob,k} = \bar{\omega}^b_{ob,k} + \mathbf{K}_{\omega,k}\bar{\Delta}\epsilon_k \tag{3.24}$$

where $\bar{\Delta}\epsilon$ is the parameter estimation error in the ϵ part of the quaternion given by

$$\Delta\epsilon_k = \hat{\epsilon}_{g,k} - \bar{\epsilon}_k \tag{3.25}$$

3.4.1 The complete filter

With the modifications made to the filter in the above sections the Extended Kalman filter algorithm with Gauss-Newton becomes

$$\mathbf{J} = -\begin{bmatrix} (\frac{\partial \mathbf{M}}{\partial \eta_{g,k}}\mathbf{y}_m^b) & (\frac{\partial \mathbf{M}}{\partial \epsilon_{g,1,k}}\mathbf{y}_m^b) & (\frac{\partial \mathbf{M}}{\partial \epsilon_{g,2,k}}\mathbf{y}_m^b) & (\frac{\partial \mathbf{M}}{\partial \epsilon_{g,3,k}}\mathbf{y}_m^b) \end{bmatrix} \tag{3.26}$$

$$\hat{\mathbf{q}}_{g,k+1} = \hat{\mathbf{q}}_{g,k} - [\mathbf{J}^T(\hat{\mathbf{q}}_{g,k})\mathbf{J}(\hat{\mathbf{q}}_{g,k})]^{-1}\mathbf{J}^T(\hat{\mathbf{q}}_{g,k})\epsilon^o(\hat{\mathbf{q}}_{g,k}) \tag{3.27}$$

$$\mathbf{K}_{r,k} = \bar{\mathbf{P}}_{r,k}\mathbf{H}_{r,k}^T[\mathbf{H}_{r,k}\bar{\mathbf{P}}_{r,k}\mathbf{H}_{r,k}^T + \mathbf{R}_r]^{-1} \tag{3.28}$$

$$\Delta\mathbf{q}_k = \hat{\mathbf{q}}_{g,k} \otimes \bar{\mathbf{q}}_k^{-1} \tag{3.29}$$

$$\hat{\mathbf{q}}_k = \bar{\mathbf{q}}_{g,k} \otimes \begin{bmatrix} \sqrt{1 - \|\mathbf{K}_{\epsilon,k}\Delta\epsilon_k\|^2} \\ \Delta\epsilon_k \end{bmatrix} \tag{3.30}$$

$$\Delta\epsilon_k = \hat{\epsilon}_{g,k} - \bar{\epsilon}_k \tag{3.31}$$

$$\hat{\omega}_{ob,k}^b = \bar{\omega}_{ob,k}^b + \mathbf{K}_{\omega,k}\bar{\Delta}\epsilon_k \tag{3.32}$$

$$\bar{\mathbf{x}}_{k+1} = \mathbf{\Phi}_k\hat{\mathbf{x}}_k \tag{3.33}$$

$$\bar{\mathbf{q}}_{k+1} = \frac{\bar{\mathbf{q}}_{k+1}}{\|\bar{\mathbf{q}}_{k+1}\|} \tag{3.34}$$

$$\mathbf{P}_{r,k} = [\mathbf{I} - \mathbf{K}_{r,k}\mathbf{H}_k]\bar{\mathbf{P}}_{r,k}[\mathbf{I} - \mathbf{K}_{r,k}\mathbf{H}_k]^T + \mathbf{K}_{r,k}\mathbf{R}\mathbf{K}_{r,k}^T \tag{3.35}$$

$$\bar{\mathbf{P}}_{k+1} = \mathbf{\Phi}_{r,k}\mathbf{P}_{r,k}\mathbf{\Phi}_{r,k}^T + \mathbf{Q}_r \tag{3.36}$$

Equations (3.26) to (3.36) are the filter equations implemented on the microcontroller.

3.5 Unscented Kalman Filter

In the resent years a new linear estimator has emerged, called an unscented Kalman Filter (UKF). The Unscented Kalman Filter is a different way of using a Kalman Filter on a nonlinear system and is developed to overcome the shortcomings of the EKF. Where EKF requires a linearization around its working point which can introduce errors into the filter, the Unscented Kalman Filter avoids this issue by not requiring any form of linearization.

3.5.1 Unscented Transformation

At the heart of the UKF lies the Unscented Transformation (UT) which is a method for calculating the statistics of a random variable that undergoes a nonlinear transformation. A set of points called sigma points are chosen such that the sample mean and sample covariance equals $\bar{\mathbf{x}}$ and $\mathbf{P}_{xx}$. These points are sent through the nonlinear function to yield the transformed points where $\bar{\mathbf{y}}$ and $\mathbf{P}_{yy}$ are the statistics of this new set of points. The number of sigma points needed is $2n+1$, where n is the dimension of the state vector. Figure 3.1 shows the principle of the unscented transformation. The samples

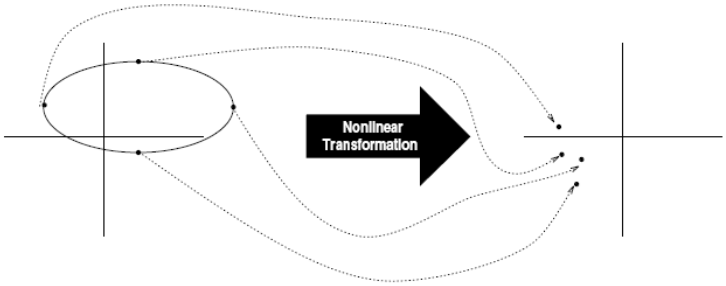

Figure 3.1: The Unscented Transform

in the transformation are not drawn at random, but according to a specific deterministic algorithm. A random variable of dimension n with mean $\bar{\mathbf{x}}$ and covariance $\mathbf{P}_{xx}$ is approximated by $2n+1$ weighted points which are given as

$$\begin{aligned} \mathcal{X}_0 = &\quad \bar{\mathbf{x}} & \mathbf{W}_0 &= \kappa/(n+\kappa) \\ \mathcal{X}_i = &\quad \bar{\mathbf{x}} + (\sqrt{(n+\kappa)\mathbf{P}_{xx}})_i & \mathbf{W}_i &= \frac{1}{2}/(n+\kappa) \\ \mathcal{X}_{i+n} = &\quad \bar{\mathbf{x}} - (\sqrt{(n+\kappa)\mathbf{P}_{xx}})_i & \mathbf{W}_{i+n} &= \frac{1}{2}/(n+\kappa) \end{aligned} \tag{3.37}$$

where κ can be any real number, $\kappa \in \Re$ and $\mathbf{W}_i$ is the weight associated with the ith point.

Transformation procedure

The transformation procedure for the sigma points in $\mathcal{X}_i$ contains three steps, first is obtaining the transformed points by sending them through a nonlinear

function **f**, followed by the computation of the mean and covariance for the transformed points. The procedure can be written as

$$
\begin{aligned}
1: \quad & \mathcal{Y}_i = \mathbf{f}[\mathcal{X}_i] \\
2: \quad & \bar{\mathbf{y}} = \sum_{i=0}^{2n} \mathbf{W}_i \mathcal{Y}_i \\
3: \quad & \mathbf{P}_{yy} = \sum_{i=0}^{2n} \mathbf{W}_i \{\mathcal{Y}_i - \bar{\mathbf{y}}\}\{\mathcal{Y}_i - \bar{\mathbf{y}}\}^T
\end{aligned}
$$

where it is seen that the mean is the weighted average of the transformed points $\mathcal{Y}_i$ and the covariance $\mathbf{P}_{yy}$ is a weighted outer product of the transformed points.

Julier and Uhlmann [1997] presents a summary of the most important properties of the unscented algorithm.

- The mean and covariance of y are correct up to the second order since the mean and covariance of the random variable, x, are captured precisely up to the second order. This result means that the mean is calculated to a higher order of accuracy then for the EKF and the covariance is calculated to the same order of accuracy.
- The sigma points capture the same mean and covariance unaffected by the choice of matrix square root used, $(\sqrt{(n+\kappa)\mathbf{P}_{xx}})_i$ in equation (3.37).
- The most promising property with respect to our design and area of application is the fact that the mean and covariance are calculated using standard vector and matrix operations, making implementation easy and rapid. This is due to the fact that there is no need to evaluate the Jacobians which are needed for the EKF.
- κ provides an extra degree of freedom that can be used to "fine tune" the higher order moments of the approximation and can be used to reduce the overall prediction error.

The benefits to the performance when using the unscented transform can be seen in figure 3.2 which shows the mean and 1σ contours from the different methods. True mean lies at $\times$ with covariance contour shown as dotted,

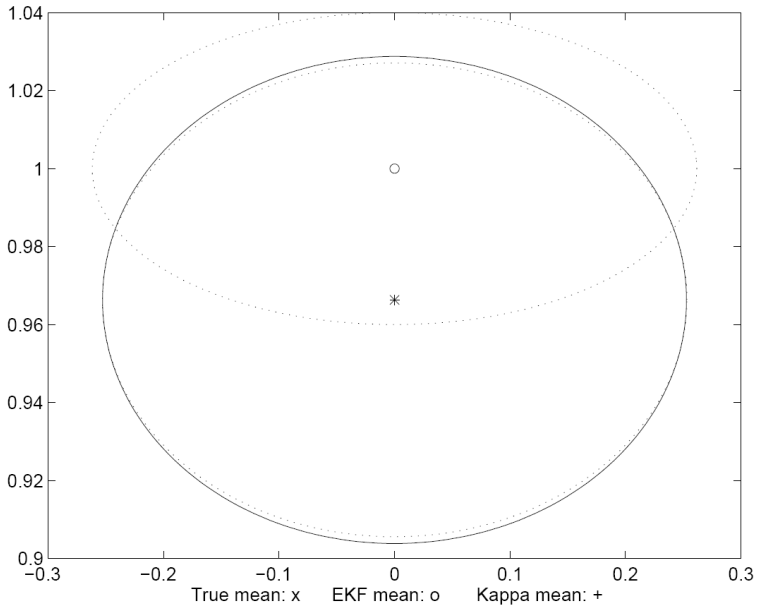

Figure 3.2: Unscented transformation applied to a measurement example Julier and Uhlmann [1997]

linearized mean lies at ∘ with a dashed covariance contour and the unscented mean lies at ★with a solid contour. As can be seen the unscented and true mean values and covariance are almost identical.

3.5.2 The Unscented Filter

In this section a general Unscented filter will be presented.

The nonlinear dynamic system in discrete time is given as

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{x}_k, \mathbf{w}_k) \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k, \mathbf{v}_k) \end{aligned} \tag{3.38}$$

where $\mathbf{x}_k$ is the state vector, $\mathbf{y}_k$ is the measurement signal and $\mathbf{w}_k$ and $\mathbf{v}_k$ are system noise and measurement noise respectively.

$$\mathbf{x}^a = \begin{pmatrix} \mathbf{x}^T & \mathbf{w}^T & \mathbf{v}^T \end{pmatrix}^T, \quad \mathcal{X}^a = \begin{pmatrix} (\mathcal{X}^x)^T & (\mathcal{X}^w)^T & (\mathcal{X}^v)^T \end{pmatrix}^T \tag{3.39}$$

are the augmented state vector and sigma points. The sigma points can be found using the method described in the previous chapter. The number

of sigma points in the augmented system is $n^a = (n^x + n^w + n^v)$. The augmented state and covariance is initialized as

$$\hat{\mathbf{x}}_0^a = E\left[\mathbf{x}_0^a\right] = \begin{bmatrix} \mathbf{x}_0^a \\ 0 \\ 0 \end{bmatrix} \tag{3.40}$$

$$\mathbf{P}_0^a = E\left[(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a)(\mathbf{x}_0^a - \hat{\mathbf{x}}_0^a)^T\right] = diag(\mathbf{P}_0, \mathbf{P}_w, \mathbf{P}_v) \tag{3.41}$$

First the state estimator equations are presented, followed by the parameter estimation equations.

State estimation

The predicted state mean and covariance are computed using the unscented transform

$$\mathcal{X}_{k|k-1}^x = \mathbf{f}(\mathcal{X}_{k-1}^x, \mathcal{X}_{k-1}^w) \tag{3.42}$$

$$\mathbf{x}_k^- = \sum_{i=0}^{2n^a} \mathbf{W}_i^{(m)} \mathcal{X}_{i,k|k-1}^x \tag{3.43}$$

$$\mathbf{P}_k^- = \sum_{i=0}^{2n^a} \mathbf{W}_i^{(c)} (\mathcal{X}_{i,k|k-1}^x - \hat{\mathbf{x}}_k^-)(\mathcal{X}_{i,k|k-1}^x - \hat{\mathbf{x}}_k^-)^T \tag{3.44}$$

$\mathbf{W}_i^{(m)}$ and $\mathbf{W}_i^{(c)}$ are defined as mean weight and covariance weight respectively.

The mean and covariance observations are computed from the following equations

$$\mathcal{Y}_{k|k-1} = \mathbf{h}(\mathcal{X}_{k|k-1}^x, \mathcal{X}_{k|k-1}^v) \tag{3.45}$$

$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2n^a} \mathbf{W}_i^{(m)} \mathcal{Y}_{i,k|k-1} \tag{3.46}$$

$$\mathbf{P}_{\tilde{y}_k \tilde{y}_k} = \sum_{i=0}^{2n^a} \mathbf{W}_i^{(c)} (\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-)(\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-)^T \tag{3.47}$$

Furthermore the cross correlation covariance becomes

$$\mathbf{P}_{x_k y_k} = \sum_{i=0}^{2n^a} \mathbf{W}_i^{(c)} (\mathcal{X}_{i,k|k-1}^x - \hat{\mathbf{x}}_k^-)(\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-)^T \tag{3.48}$$

The Kalman gain matrix can now be computed, followed by the measurement update.

$$\mathcal{K} = \mathbf{P}_{x_k y_k} \mathbf{P}^{-1}_{\tilde{y}_k \tilde{y}_k} \tag{3.49}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathcal{K}(\mathbf{y}_k - \hat{\mathbf{y}}_k^-) \tag{3.50}$$

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathcal{K}\mathbf{P}_{\tilde{y}_k \tilde{y}_k}\mathcal{K}^{-1} \tag{3.51}$$

Parameter estimation

For parameter estimation, consider the state-space representation

$$\begin{aligned} \mathbf{w}_{k+1} &= \mathbf{w}_k + r_k \\ \mathbf{d}_k &= \mathbf{G}(\mathbf{x}_k, \mathbf{w}_k) + e_k \end{aligned} \tag{3.52}$$

which is used to show a general process for parameter estimation using the unscented kalman filter. The amount of sigma points needed is n^b = dimension of w_k. Initialization of the filter is normally given as

$$\mathbf{w}_0 = E\left[\mathbf{w}\right], \quad \mathbf{P}_{w_0} = E\left[(\mathbf{w} - \bar{\mathbf{w}}_0)(\mathbf{w} - \hat{\mathbf{w}}_0)^T\right] \tag{3.53}$$

Given that $\mathbf{R}_k^r$ is the process noise covariance and $\mathcal{W}_k$ are the sigma points, the time update is given by

$$\hat{\mathbf{w}}_k^- = \hat{\mathbf{w}}_{k-1}, \quad \mathbf{P}_{w_{k-1}} = \mathbf{P}_{w_{k-1}} + \mathbf{R}_{k-1}^r \tag{3.54}$$

$$\mathcal{D}_{k|k-1} = G(\mathbf{x}_k, \mathcal{W}_{k|k-1}) \tag{3.55}$$

The mean measurement calculation is based on the statistics of the expected measurements and can be written

$$\hat{\mathbf{d}}_k = \sum_{i=0}^{2n^b} \mathbf{W}^{(m)}{}_i \mathcal{D}_{i,k|k-1} \tag{3.56}$$

With the equations above, the measurement update algorithm becomes

$$\mathbf{P}_{d_k d_k} = \sum_{i=0}^{2n^b} \mathbf{W}^{(c)}{}_i (\mathcal{D}_{i,k|k-1} - \hat{\mathbf{d}}_k)(\mathcal{D}_{i,k|k-1} - \hat{\mathbf{d}}_k)^T + \mathbf{R}_k^e \tag{3.57}$$

$$\mathbf{P}_{w_k d_k} = \sum_{i=0}^{2n^b} \mathbf{W}^{(c)}{}_i (\mathcal{W}_{i,k|k-1} - \hat{\mathbf{w}}_k^-)(\mathcal{D}_{i,k|k-1} - \hat{\mathbf{d}}_k)^T + \mathbf{R}_k^e \tag{3.58}$$

$$\mathcal{K} = \mathbf{P}_{w_k d_k} \mathbf{P}_{d_k d_k}^{-1} \tag{3.59}$$

$$\hat{\mathbf{w}}_k = \hat{\mathbf{w}}_k^- + \mathcal{K}_k(\mathbf{d}_k - \hat{\mathbf{d}}_k) \tag{3.60}$$

$$\mathbf{P}_{w_k} = \mathbf{P}_{w_k}^- - \mathcal{K}_k \mathbf{P}_{d_k d_k} \mathcal{K}_k^T \tag{3.61}$$

where $\mathbf{R}_k^e$ is the measurement noise covariance and $\mathbf{W}_i^{(m)}$ and $\mathbf{W}_i^{(c)}$ are weights for the mean and covariance respectively.

The article, Ma and Jiang [2005], which this general UKF is based on presents another way of determining the sigma points which reduces the computational costs by reducing the required number of sigma points from $2n + 1$ to $n + 1$. This procedure requires the use of a certain spherical simplex UT algorithm to determine the sigma points. For implementation on a microcontroller for the satellite, this method could prove to relieve some of the computational stress on the microcontroller. For more information on this procedure refer to Ma and Jiang [2005] and other references therein.

Based on articles and the work produced here, an implementation of the unscented Kalman filter might require a higher number of operations to produce an estimate. This increase can however give an increased accuracy in the estimates. Before any conclusions can be made, proper testing is needed.

Chapter 4

Simulink Model

In order to get an overview of the Kalman filter and to make implementation into c-code easier, a simulation model for the filter was created. The model was created using Matlab and simulink. These simulation tools were selected because of availability and because previous work and simulations has been made using these tools. As the extended Kalman filter proposed has been simulated and tested in previous work, Sunde [2004], this was not done here. The simulink model is partly made up from building blocks in previous work and makes use of function calls to m-files for certain operations, among others the Kalman filter algorithm. All the m-files are included in appendix C.

The model was also created so that it could be used for testing of the prototype unit. This method of testing is browsed upon in chapter 7. The satellite model is continuous whereas the extended Kalman filter equations are discrete and means that for testing and simulation, a fixed time step simulation method must be used, for example ode3.

The simulink model consists of the complete ADCS system for the satellite and its environment. The satellite orbit is propagated using the orbit estimator from section 2.5.1. As for the satellite attitude, the gravitational torque model from section 2.6 is used with the nonlinear model. Measurements are simulated by rotating the reference models from orbit to body. White noise is added with magnitude corresponding to the covariance of the respective sensors. Figure 4.1 shows the process of simulating the measurements. This is the same for both the magnetic sensor and the Sun sensor. For the magnetometer the noise magnitude is $4.3312e-6$, whereas for the Sun sensor

tests must be performed to acquire reasonable values for the covariance. The simulated measurements along with the reference model values are inputs to the Kalman filter. For simulation purposes, measurement noise and process

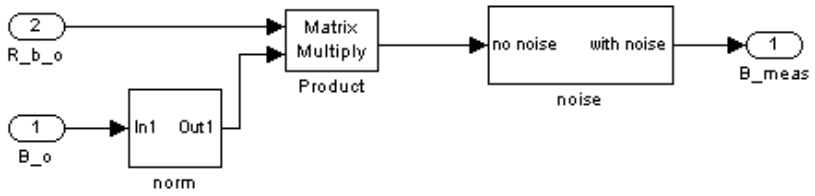

Figure 4.1: Simulated measurements for use with the Kalman filter

noise must be given. The initial measurement disturbance is guessed to be an average of both the sensors. An expected range is $2.0e-6$. Process disturbance consist of several components as presented by Kyrkjebø [2000] and can be modeled as $8.5e-9$ for $\{\epsilon_1\ \epsilon_2\ \epsilon_3\}$ and $8.5e-12$ for $\{\omega^b_{ob,x}\ \omega^b_{ob,y}\ \omega^b_{ob,z}\}$.

When the Kalman filter is initiated, the satellite will have a small angular velocity as well as angles with respect to orientation. These are unknown and the estimator have to be designed to perform well without good initial values. The values can however assumed to be small as the satellite performs detumbling prior to activating the estimator. Initial values for the covariance matrix for the states can be calculated as $\mathbf{P}_0 = var[x_o]$ where x_0 are the initial values of the different states Farrell and Barth [1999].

The top layer of the simulink model is presented in appendix B and the complete model can be found in appendix C.

Chapter 5

Hardware

The Extended Kalman filter described in chapter 3 was designed to be implemented in hardware. The hardware is supposed to function as a prototype for testing and further development with respect to the attitude determination system. Because the primary concern when building a micro satellite is power consumption, our components have been chosen mostly on this basis. In the initial report for the satellite, Narverud et al. [Nov, 2006], about 0.35W has been allocated to the ADCS system. In order to avoid strain on the bus, the attitude determination system and attitude control system are being implemented on the same microcontroller. This puts extra demands on the speed of the microcontroller and at the same time sufficient programming space is needed. In this chapter the hardware for the attitude determination system is presented, including the sensor system as well as communication with sensors and On Board Data Handling, OBDH. The chapter will also go deeper into the realization of the sensor system.

5.1 Microcontroller

As mentioned above, the attitude determination and control system is meant to be implemented on the same microcontroller. The controller is the brain of the ADCS system and is executing the Kalman filter algorithm to produce an estimate of the satellite attitude. It has to communicate on a databus, I^2C, as well as communicating with the different sensor systems. A common type of microcontroller for the entire satellite has not been decided upon, and as such there are several possible choices available.

The most obvious choice would be Atmel's flagship within 8-bit RISC controllers, the ATmega128. This is a commonly used low power controller with a lot of easily available support soft- and hardware. The controller itself features, among other:

- Low power architecture
- 128KB reprogrammable flash memory
- 6 different sleep modes
- Dual Universal serial synchronous/asynchronous communication interfaces (USART)
- Two-Wire Interface
- JTAG interface

A different choice from Atmel are their new 32-bits chip, the AP7000. Although this chip has superior computational capabilities, it's power consumption makes it unfit for our purposes. This is also a new chip as mentioned and there is the possibility that unexpected errors could arise.

Texas Instruments, TI, has a series of 16-bits microcontrollers called MSP430. One of these is the MSP430f169 which has several features that makes it a good choice for our needs.

- Ultralow-power architecture with 5 low-power modes
- 16-bit RISC CPU - applications require a fraction of the code size
- 16-bit registers
- 60KB+256B Flash memory and 2KB RAM
- Digitally controlled oscillator (DCO) - which allows wake-up from low-power modes to active mode in less then 6μs
- two Universal serial synchronous/asynchronous communication interfaces (USART)
- Two-Wire communication

The different low-power modes differ from each other by which clocks are still active. In low-power mode 0, (LPM0), the CPU is disabled along with the main clock. In low-power mode 4, (LPM4), all clocks are disabled including the crystal oscillator. This means the microcontroller can enter a mode where it hardly draws any power from the system and with the DCO it can be back if full active mode within 6μs. For more information on the microcontroller, see appendix C for the MSP430 datasheet and User Guide.

The choice of controller landed on the MSP430 chip from Texas Instruments, with its low power consumption and 16bit architecture it seems to be the best choice.

The Kalman filter require high-accuracy variables for the calculations and these are implemented as floating point variables on the microcontroller. The C-compiler used in CrossWorks, chapter 5.4, uses the 32-bits IEEE floating point which conform to the IEEE 754 standard. Following this standard, a floating point is created with a base number, an exponent and a sign bit. Figure 5.1 shows the fields of a 32-bit floating point The smallest positive

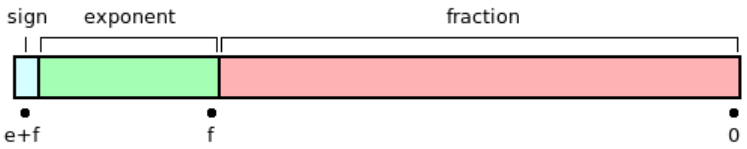

Figure 5.1: The fields in an IEEE 754 float Wikipedia [2007*a*]

number is given by the formula Kahan [1997]

$$2^{2-2^K} = 2^{2-2^7} = 1.1755e-38 \tag{5.1}$$

where K is the number of bits for the exponent part of the float, 7 in the case of the mentioned IEEE standard. This value can also been seen on as the step size for floating points and thus a measure of accuracy. This is well within our needs for implementing the Kalman filter equations. The largest number for a float is given by

$$(1-0.5^N)\cdot 2^{2^K} = (1-0.5^24)\cdot 2^{2^7} = 3.4028e38 \tag{5.2}$$

where k is the same as above and N is the number of significant bits. For a

float this number is 24 following the standard above.

In addition to running the Kalman filter, the microcontroller needs to communicate with several peripheral units using different communication protocols. The Two-Wire Interface is used to communicate on the bus and is implemented both in hardware and software. This will be further explained later in this chapter. Other communication include communicating with the sensor system to retrieve sensor date for use in the Kalman filter and for storing. The sun sensors are connected directly to Timer_B on the microcontroller. Communication with the magnetometer is done in hardware and software using the USART interface. Transmission correctness is done in hardware but data handling is done in software and will be described further in section 5.3.2.

For prototype purposes a header board was used, delivered by Olimex. The header board contain the MSP430f169 chip with a 32.768 kHz watch crystal, socket for a high frequency crystal and a JTAG connector. Figure 5.2 shows the board. In order to make prototype testing and development easier a

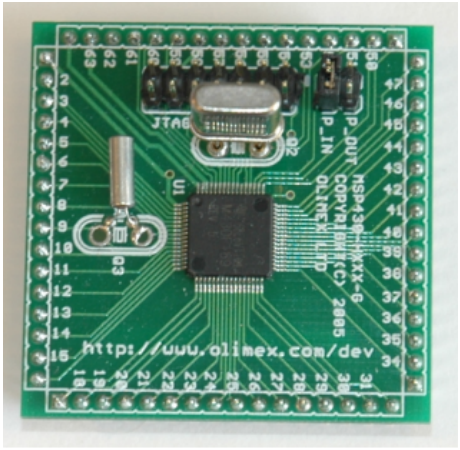

Figure 5.2: Header board for the MSP430f169

simple development board was created, with individual header connections for the different ports and pins on the microcontroller. The development board is shown in figure 5.3 The board PCB layout is included in appendix C.

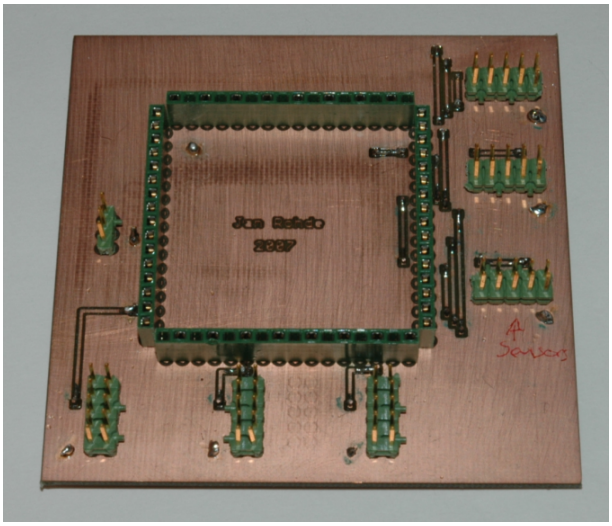

Figure 5.3: Development board for the MSP430 header board

5.2 Sensors

The satellite uses two different sensors for attitude determination. One is a sun sensor system and the second is a magnetometer to measure the local magnetic field around the satellite. In the following sections the sensor hardware is described, as well as how they work.

5.2.1 Light-to-Frequency converter

One of the sensor system on the satellite is a light sensor that measures the inbound light on each side of the satellite. This is realized by placing one light sensor on each side of the satellite. The measurements are combined and used in calculating a vector which points in the direction of the Sun.

The Sun sensors are realized using Light-to-Frequency converters, LF-converters, which output a frequency depending on the irradiance of light. The LF-converters used are TSL235R and are delivered from Texas Advanced Optoelectronic Solutions, TAOS. From simple tests performed it is seen that the frequency output is dependent on the angle of attack. From other tests, the output frequency seemed to go no higher then 804kHz. This was tested by applying direct focused light onto the diode. In a dark environment the output frequency was approximately 1.4kHz. The datasheet suggest that an output frequency above 500kHz might be exposed to saturation, see appendix

C for more information from the datasheet.

The Sun sensor works in the way that whichever side is facing the sun will register the highest light level and thus output the highest frequency compared to the other LF-converters. As the satellite changes attitude some sensors will receive more light and others less light and the internal relation between the sensors are used to estimate the direction of the Sun. The output vector from the sensors is written as

$$\mathbf{S}^b = \begin{bmatrix} s_1 - s_6 \\ s_2 - s_5 \\ s_3 - s_4 \end{bmatrix} \tag{5.3}$$

where s_i is the measured output from sensor i. The sensors are placed on the satellite as shown in figure 5.4.

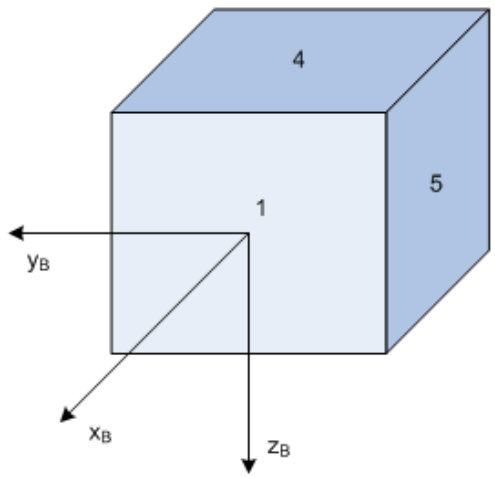

Figure 5.4: Placement of sun sensors on the satellite. The three remaining sensors are placed on the opposite sides. Sensors 6 is placed on the x_b axis, sensor 2 on the y_b axis and sensor 3 on the z_b axis

Sun sensor accuracy

The Sun measurement is given in the sensor frame which in this case coincides with the body frame. The Sun sensor is crude and contain some assumptions that degrades the accuracy. The motion of the Sun is described as a circular orbit around the Earth's center with the satellite placed in this center. The

inaccuracy from this assumption is however not very large as was proved by Kristiansen [2000] using the following formula to calculate the error

$$\xi = \arctan\left(\frac{R_o}{R_s}\right) \approx 1.43e^{-7} rad \tag{5.4}$$

where R_o and R_s is distance between the Earth and the satellite and the Earth and the Sun respectively. This error is relatively small compared to the overall accuracy of the system and is ignored. It is however important to know that such an error exists in our Sun sensor. The second error source is the light reflected back from the Earth, called the Earth Albedo error. This is explained in chapter 2.5.2.

Method of operation

The use of the LF-converters as Sun sensors open up to two different ways of performing measurements. One way is to measure the frequency of the converters by sending the signal to the Timer_B port on the microcontroller. Since there is only one Timer_B input, the measurements must be done sequentially and a multiplexer is used to send all the Sun sensor data to the input pin. The watchdog timer, connected to the watch crystal is used as a time reference and can be set to trigger after a specific time has elapsed. In order to get a good measurement of the frequency the reference time needs to be long. This will increase the time it takes to get one set of measurements and can lead to inaccuracy.

The second method is to measure the period of the signal the sensors emit. The lowest period corresponds to the highest frequency and in turn the side exposed to highest light levels. This method uses the capture/compare registers of Timer_B and does not require the use of a multiplexer since there are 7 such registers in Timer_B.

The second option is chosen for several reasons. There is no need for an external components, such as a MUX, and the measurements can be performed in parallel, which reduces the time it takes to get one measurement set. This is important because the satellite might be spinning around one of its axes' and sensor data will be highly inaccurate if a set of measurements require to much time to collect.

5.2.2 Magnetometer

The second sensor system on the satellite is a magnetometer which measures the local magnetic field around the satellite. These measurements are compared to a reference model of the Earths magnetic field and is then used to estimate the attitude of the satellite. In order to be able to use a magnetometer for attitude determination, the satellite must be in a Low Earth Orbit, LEO, where known reference models of the Earth magnetic field exists.

The magnetometer chosen is the HMR2300 - Smart Digital Magnetometer from Honeywell, see appendix C for datasheet. This magnetometer was chosen for several reasons, the most important being its low power consumption. Other features that makes implementation easier are

- Three Axis Digital outputs (16-bits)
- RS232 interface
- High accuracy over $\pm$1gauss
- Range of $\pm$2gauss, $<70\mu$guass Resolution

The magnetometer is built up with three individual orthogonal magnetoresistive sensors, measuring the X, Y and Z vector in the local magnetic field. In addition the magnetometer comes with a development kit that includes a demo program.

The measurements are taken in three dimensions in the sensor frame and best results are obtained by placing the magnetometer in such a way that the sensor frame coincide with the body frame or have a known and simple configuration relative to the body frame. Due to the internal architecture of the cube the last option seems to be the only possibility. The measurements from the body frame are compared to the model of the magnetic field, which for the IGRF model, is in orbit frame. The relation between these two frames are used to calculate the attitude of the satellite.

Magnetic sensor accuracy

The accuracy of the magnetometer is not as good as star and horizon references due to disturbances such as Bak [1999]

- Disturbance fields generated due to satellite electronics

- Model errors in the reference model
- External disturbances such as ionospheric currents

The electronics on board the satellite can influence and degrade magnetometer measurements unless shielding is considered when implementing the hardware. Another source of error is the actuating system itself with its magnetic coils. When measurements are performed, the actuate system needs to shut down the magnetic coils so they do not produce a magnetic field around the satellite.

The most commonly used reference model, IGRF, is an empirical representation of the Earth's magnetic field. It represent the main field without any external sources and is a weighted mean of models developed by a number of agencies.

The ionosphere is a non-uniform field with electrical currents inducing unpredictable magnetic disturbances. This implies that we can not predict how this disturbance will affect the magnetometer measurements.

5.3 Communication

5.3.1 Sun sensor interface

The LF-converters are connected directly to the microcontroller when used in the way described above. Each LF-converter is connected to the CCIxB-pins on the microcontroller, sensor 1 to CCI1B, sensor 2 to CCI2B and so on. CCI0B is left unused. See appendix C for datasheet and information on the input pins. The communication is one-way, from the LF-converters to the microcontroller and is always active as long as power is connected. For power consumption purposes the LF-converters should be powered down whenever measurements are not gathered.

5.3.2 Magnetometer interface

The HMR2300 magnetometer has a RS232 interface which is a serial communication bus called USART. The bus normally uses 3 wires, one for transmitting, one for receiving and one ground. The bus can also be used to power modules and the magnetometer is powered through this bus. A logic high level on the RS232 bus is 12V, whereas the microcontroller uses 5V

for logic high level. An external chip is used to convert the signal between the two modules and is called MAX232, delivered by Maxim. Figure 5.5 shows how the microcontroller and the magnetometer is connected through the MAX232 chip. Datasheet for the MAX232 can be found in appendix C. Unlike the Sun sensor, the magnetometer requires commands to be sent

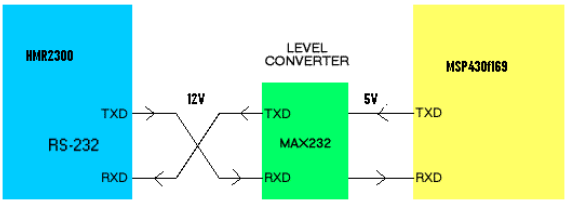

Figure 5.5: microcontroller connected to the magnetometer via the MAX232 chip

from the microcontroller. These commands will be browsed upon in chapter 6 when the implementation is described.

5.3.3 Communication bus

The estimator is a small part of the overall satellite system. In addition to communicating with the sensors the estimator must be connected to other satellite subsystems, most importantly the OBDH system which handles overall control of the satellite. It is also planned that the ADCS system should be able to receive direct commands from the antenna up-link to ensure that faults in the OBDH does not become a bottleneck point. The internal communication between the different subsystems is implemented with a two-wire communication bus, TWI.

This communication protocol uses, as the name suggest, two wires for communication, one is the Serial Data wire, referred to as SDA and the other one is the Serial Clock wire, referred to as SCL. The communication bus is built up with master and slave nodes, where a master node can control the clock line and a slave node can not. Since the master controls the clock it also controls the transmission, and the transmitting node, be it a slave or the master, synchronizes the data wire with the clock wire.
All nodes on the communication bus have a unique address of seven bits, making a single bus capable of having up to 127 nodes connected. Communication is initialized by sending out a START condition on the bus. After

the START condition has been given, the address is sent transmitted on the SDA wire. The node which is addressed replies and communication between the two is established. Figure 5.6 shows a picture of a normal transmission using TWI. The master node that initiated the communication decides which

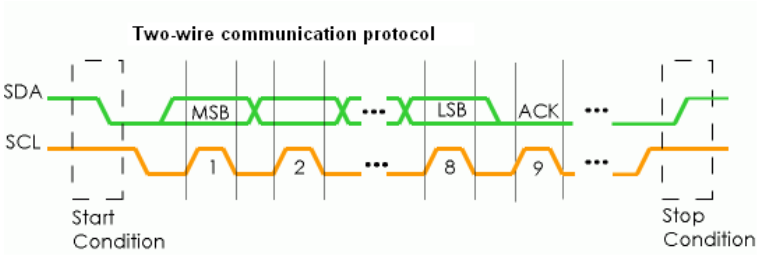

Figure 5.6: Transmission using TWI

node is the transmitting one and which is the receiving one. When the last bit has been received, a STOP condition is sent out on the bus, freeing it up for other nodes to initiate communication.

The protocol allows multiple masters on a single bus. This is necessary in our satellite because there are several subsystems that may wish to establish a communication link with each other. With only one master, it would have to periodically ask each subsystem if it wanted to transmit, a setup which is inefficient and exposed to communications errors. But the use of multiple masters introduces the possibility of timing errors, for example if two masters try to send at the same time. Such problems are handled using arbitration. This is done in the way that all nodes who transmit on the bus also listens to the bus. If a transmitting node sends out a "1", but reads a "0" it means that somebody else is also attempting to transmit. If a "0" and a "1" is written to the bus, a "0" will be read, resulting in the node who wrote a "1" to stop transmitting and enter slave mode to see if it is the addressing node. The address is the first byte to be sent on the bus when a transmission is initialized and is thus what is being arbitrated on. Transmission is executed by sending the most significant bit, MSB, first which allows the use of prioritized nodes. A low address will be given higher priority during arbitration then a node with a higher address because a high address will have to write a "1" on the bus sooner then a lower address.

When multiple masters are used, it is not always the case that they run on the same clock frequency. This means that different masters may send out

a different clock signal on the SCL wire resulting in synchronization errors. This problem is solved by making a wired-AND on all the different serial clocks, yielding a combined clock with a high period equal to that of the master that has the shortest period. The low period is equal to the master with the longest low period.

5.4 Development tools

Different development tools, both hardware and software has been used and they are browsed upon in this section

5.4.1 CrossWorks

CrossWorks is a development software for the MSP430f169, containing a development environment and a c-compiler to create and download programs to a target chip. CrossWorks is developed by Rowley Associates and can also be used on different targets like; ARM, AVR and MAXQ. More information on CrossWorks can be found on the Rowley Associates webpage, http://www.rowley.co.uk/

5.4.2 JTAG

Rowley Associates delives JTAG adapters for MSP430 that can be used for programming the target chip as well as debugging the target. The JTAG used in this work is the MSP430 CrossConnect JTAG Emulator.

5.4.3 Eagle - PCB layout editor

Eagle layout editor is delivered by CadSoft Online and from their homepage, http://www.cadsoftusa.com/, a freeware version of the software can be downloaded. Eagle provides a simple tool for creating your own PCB layouts.

Chapter 6

Implementation

In this chapter the implementation on the microcontroller is described. The extended Kalman filter algorithm from equations (3.26) to (3.36) have been implemented, but the communication between other subsystems and the sensors has not been completed. The intended communication setup is presented instead. The estimator requires several multidimensional mathematical operations and a library of matrix operations has been created.

6.1 Introduction

Because the complete system is not fully implemented yet, the intentions for the non-implemented parts are presented. The program is written using the C programming language and the complier used is CrossWorks from Rowley Associates. The compiler has been developed to work with the MSP430 series of microcontrollers from Texas Instruments. For more information, see chapter 5.4.

To describe the behavior of the estimator, dataflow charts will be used for simplification. The notation is the standard notation from Microsoft Visio and is shown in figure 6.1. First in this chapter a section on programming Real-Time systems is presented.

6.2 Programming Real-Time systems

A lot of consideration must be taken when designing a Real-Time system. There are many definitions of a Real-Time system, one of these are

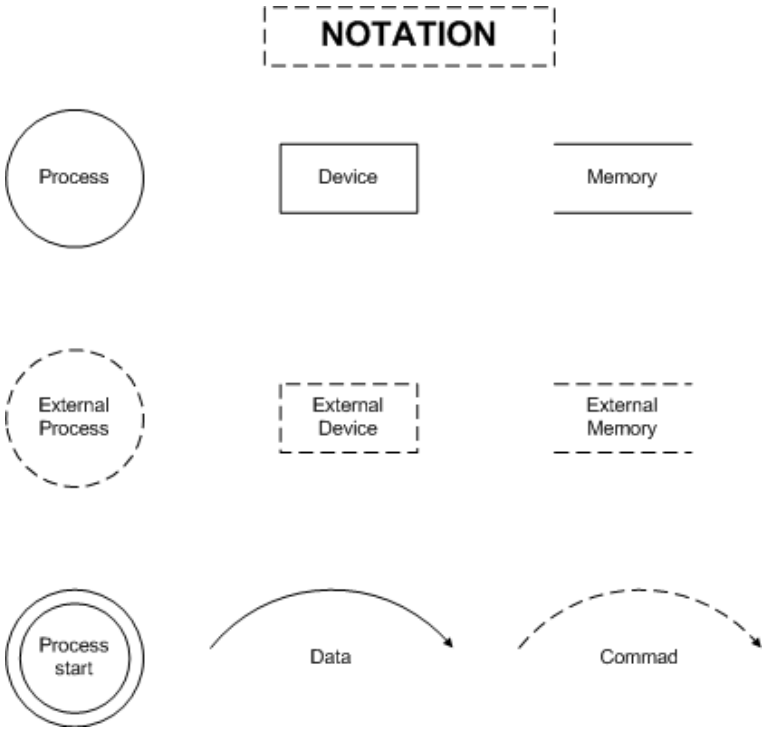

Figure 6.1: Notation used in dataflow charts

A system that reacts to an external input signal and produce an output signal within a defined time period.

Real-Time systems often control sensitive systems where an error can have serious consequences with regards to equipment and in worst case human lives. In the case of a satellite traveling in a Low-Earth orbit, service is unavailable and as such the priority is to make the software failsafe. Many errors are caught by the compiler and can be corrected before any damage is done. Run-time errors and logical errors however are not detected by the compiler which means that they have to be kept in mind when the system is developed.

For the extended Kalman filter, multidimensional mathematical matrix operations are performed on float numbers which requires many filter matrices and temporary matrices to be created. A float matrix of size 6 x 6 will for example require 6*6*4 = 144 Bytes. A microcontroller has only a limited amount of available memory and this must be taken into consideration when

the EKF is implemented. The use of memory allocation, **malloc**, and deallocation is a method for allocating memory to variables, etc where the memory is freed or deallocated at the end of its use. The major problem with memory allocation is that the programmer have now way of determining where in memory the variables are created. The use of **malloc** can easily lead to run-time errors with respect to available memory because the controller does not know if there is enough free memory at the time of creation. Two main reasons for Run-time errors are listed below and explained

- Available memory
- Memory fragmentation

At compile-time, the compiler have no way of knowing how many variables will be created using **malloc** or the size of these. It cannot test to see if enough memory will be available at the point of creation. If this happens, a run-time error will occur and the estimator will enter fail-mode due to lack of memory.

Memory fragmentation is also common when using **malloc**, which means that even though there is enough available memory, not enough continuous memory is available and the **malloc** routine will return an error saying not enough available memory. Run-time errors like these can only be rectified by rebooting the system and resulting in the loss of estimated attitude.

There are ways in which **malloc** can be made safer, for example, the routines that allocate memory can be rewritten. This gives the programmer more control over the heap, the dynamic memory, where dynamic variables are created. Using this approach, the **malloc** routine can be programmed to only create variables of a certain size in predefined areas of the memory, avoiding fragmentation when freeing up used memory. This however is tedious work and can require a lot of reprogramming which in turn can lead to other errors.

The use of malloc() in the estimator

Memory allocation, **malloc**, has been used excessively in the estimator so far. The reason for using **malloc** was that it enabled the use of software from Numerical Recipes, Press et al. [2002], mainly a function for creating the inverse of a matrix of size (n x m). The matrices has to have a certain

structure if they are to be used in this algorithm. The matrix creation function available from Numerical Recipes has been used and this is the function that makes use of **malloc()**. The multidimensional matrix mathematical operations were created to conform with the above matrix standard. The use of **malloc** is further discussed in chapter 8.2.

6.3 Overall view

The Attitude Determination System, ADS, interact with three different external devices. The Sun sensor interact directly with the Timer_B capture/compare pins, the magnetometer receives and transmit messages and measurements using RS232 and the TWI interface is used to communicate with the other subsystems. The ADS has four different modes of operation:

- **Idle**. This is the initial state in the ADS. In this state the system is dormant, waiting for the startup command from the OBDH. After ejection from the delivery system, ADS enters the idle state during the detumbling phase.
- **System Initialization**. The EKF algorithm is initialized and initial values and matrices are loaded into memory. Interface to sensors are also initialized.
- **Estimate**. The Kalman filter algorithm is executed in this process. Measurements are brought in from the sensors and reference model data is loaded from memory. The new estimates will be sent to the control system.
- **Communicate**. Messages to and from the ADS are handled here. Outgoing messages are formated and sent whereas received messages are interpreted.

The next section will explain the different states in more detail.

6.4 Idle process

This is the first state the microcontroller enters after it is powered up. Before entering the "IDLE" state, a microcontroller initialization routine is executed, *uc_init();*. Here the most important parts of the microcontroller

is initiated, such as the external crystal, watchdog timer and interrupts. This also allows the use of Low-Power modes. In this state it awaits a command for initialization from the control system. The microcontroller enters low-power mode to conserve power.

6.5 System initialization process

At receiving a "start command" from the control system, the initialization process is executed. There are four subroutines here and figure 6.2 shows a dataflow chart of the initialization process.

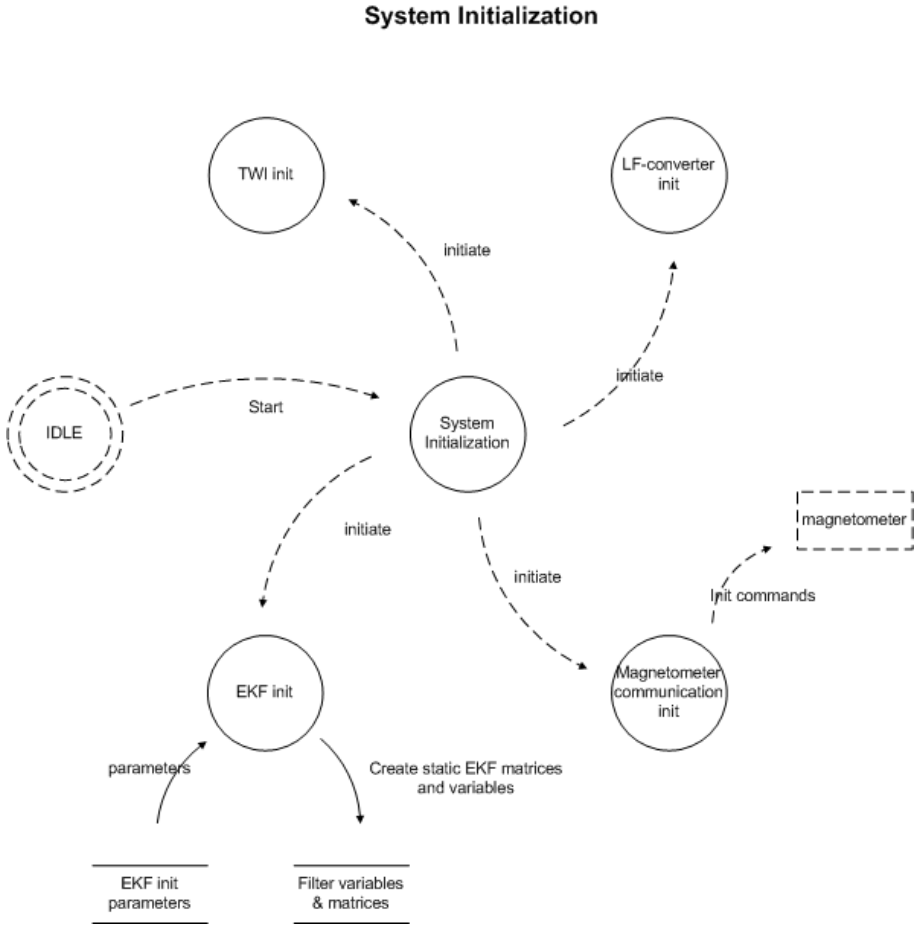

Figure 6.2: Dataflow chart of the system initialization process

- TWI init: *twi_communication_init();* - Sets up the TWI communication protocol, bus speed and address register.

- LF-converter init: *sun_sensor_init();* - Enables interface to LF-converters, setting up Timer_B for capture mode.
- Magnetometer init: *magnetometer_communication_init();* - Enables interface to the magnetometer. Setting up UART interface and initializes the magnetometer.
- EKF init: *InitEKF();* - Creates the Kalman filter matrices and variables and loads preprogrammed variables into memory.

6.6 Estimation process

In this process the estimation of new states is performed. The Kalman filter algorithm can be seen as a collection of several subroutines called in a specific order. The process also controls the sensors through their respective interfaces and transmit new estimation data to the control system. This process makes extensively use of the matrix operations created.

Before looking at the different phases of the estimation process, the available matrix operations will be listed. They can also be found in the file "matrix_operations.c", located in appendix C.

- copy_matrix
- matrix_addition
- matrix_subtraction
- matrix_multiplication
- matrix_transpose
- matrix_zeros
- matrix_ident
- matrix_diag
- matrix_const_multiplication
- skew_matrix_eps
- skew_matrix

- quat_prod
- matrix_quaternion_multiplication
- quaternion_vector_subtraction

To make the programming more intuitive, a new type Quaternion, was created and is defined as

$$
\begin{aligned}
&typedef\ struct\ quat\ \{\\
&\quad float\ eta;\\
&\quad float\ eps[3];\\
&\}\ Quaternion
\end{aligned}
$$

The use of pointers in the estimation process is substantial, both pointers to float variables and quaternion structures and double pointers to matrices. Whereas single pointers are normally used to represent a vector and a double pointer used to represent a matrix, double pointers are also used to represent vectors in this program. This enables us to use the matrix operations on the vectors. Instead of sending the matrices between different processes, the pointers are passed on, creating a more efficient program.

The estimation process has 7 steps, listed below and shown in figure 6.3.

- Gauss-Newton algorithm
- Kalman Gain update
- Estimation error update
- State estimation update
- Error Covariance update
- State transition
- Propagation

In addition there is one step for getting a new set of measurements and one for transmitting the new estimates to the control system. The source code for all the functions can be found in the file "ekf.c", see appendix C. In the

Figure 6.3: The estimate process

following sections the estimation process will be explained. References to the filter equations will be presented when the c-functions are discussed. In the file kalman.m in appendix C the corresponding implementation for Matlab is found.

6.6.1 Get measurements

Source C-code:

$$GetMeasurements(y\_meas, y\_ref);$$

where y_meas and y_ref are the pointers to the float vectors containing the new set of measurements and reference values. The y_meas vector is built up as

$$y\_meas = \begin{bmatrix} B^b_{m,x} & B^b_{m,y} & B^b_{m,z} & S^b_{m,x} & S^b_{m,y} & S^b_{m,z} \end{bmatrix}^T$$

y_ref is built up in the same way, containing the corresponding reference values. The measurements are taken sequentially, first the period length of the pulse train delivered by the LF-converters. The measurements are combined to form the light measurement vector in equation (5.3). Next, data from the magnetometer is read and both measurement vectors are saved to memory.

6.6.2 Gauss-Newton Algorithm

Source C-code:

$$GenerateJacobian(vPointers, vQuaternions);$$
$$CreateM(vPointers, vQuaternions);$$
$$Calculate\_q\_hat(vPointers, vQuaternions);$$

where vPointers and vQuaternions are two vectors of pointers to the different Kalman filter matrices and quaternions. Mathematical operations are performed on the different matrices and quaternions, and then saved to its corresponding matrix. *GenerateJacobian* is the function used to produce the Jacobian matrix of equation (3.26), reproduced here:

$$\mathbf{J} = -\begin{bmatrix} (\frac{\partial \mathbf{M}}{\partial \eta_{g,k}}\mathbf{y}^b_m) & (\frac{\partial \mathbf{M}}{\partial \epsilon_{g,1,k}}\mathbf{y}^b_m) & (\frac{\partial \mathbf{M}}{\partial \epsilon_{g,2,k}}\mathbf{y}^b_m) & (\frac{\partial \mathbf{M}}{\partial \epsilon_{g,3,k}}\mathbf{y}^b_m) \end{bmatrix} \tag{6.1}$$

The variables sent to this function are the measurement values y_meas the reference values y_ref and the previous quaternion estimate q_hat. $CreateM$ uses the previous quaternion estimate q_hat to generate the rotation matrix from equation (3.18). $Calculate\_q\_hat$ updates the new Gauss-Newton unit quaternion estimate with new measurements according to equation (3.27), reproduced here for simplicity.

$$\hat{\mathbf{q}}_{g,k+1} = \hat{\mathbf{q}}_{g,k} - [\mathbf{J}^T(\hat{\mathbf{q}}_{g,k})\mathbf{J}(\hat{\mathbf{q}}_{g,k})]^{-1}\mathbf{J}^T(\hat{\mathbf{q}}_{g,k})\epsilon^o(\hat{\mathbf{q}}_{g,k}) \tag{6.2}$$

This equation contains the innovation process as can be seen in the last element of the implemented c-code.

6.6.3 Kalman gain update

Source C-code:

$$CreateK(vPointers, vQuaternions);$$

In this function the updated Kalman Gain matrix is computed. The matrix corresponds to the reduced gain matrix of equation (3.28), reproduced here:

$$\mathbf{K}_{r,k} = \bar{\mathbf{P}}_{r,k}\mathbf{H}_{r,k}^T[\mathbf{H}_{r,k}\bar{\mathbf{P}}_{r,k}\mathbf{H}_{r,k}^T + \mathbf{R}_r]^{-1} \tag{6.3}$$

The gain matrix is computed using, among others, the reduced measurement matrix $\mathbf{H}_{r,k}$. As found in chapter 3 this matrix has not only a reduced size, it is also a constant matrix, reducing the amount of computation required to calculate $\mathbf{K}_{r,k}$. The Kalman gain matrix is further divided into two matrices, $\mathbf{K}_\epsilon$ and $\mathbf{K}_\omega$, used in updating the quaternion estimate and angular velocity estimate respectively.

6.6.4 Estimation error update

Source C-code:

$$UpdateEstimationError(vPointers, vQuaternions);$$

This function is used to find the estimation error in the quaternion estimation from the Gauss-Newton method. The calculation done here refers to equation (3.29), shown below.

$$\Delta\mathbf{q}_k = \hat{\mathbf{q}}_{g,k} \otimes \bar{\mathbf{q}}_k^{-1} \tag{6.4}$$

Since the error can be calculated using the quaternion product, the $quat\_prod()$ function is used. The epsilon-part of the resulting estimation error quaternion is used in updating the quaternion estimate part, whereas the estimation error for the angular velocity is computed according to equation (3.31).

6.6.5 State estimation update

Source C-code:

$$UpdateStateEstimation(vPointers, vQuaternions);$$

refers to two calculations, one for the quaternion update and one for the angular velocity. For the quaternion update an estimation error for η must be found before the new updated estimate can be found. The quaternion estimate update is given in equation (3.30) and reproduced here:

$$\hat{\mathbf{q}}_k = \bar{\mathbf{q}}_{g,k} \otimes \begin{bmatrix} \sqrt{1 - \|\mathbf{K}_{\epsilon,k}\Delta\epsilon_k\|^2} \\ \Delta\epsilon_k \end{bmatrix} \tag{6.5}$$

Calculations for the angular velocity estimation update is given by equation (3.32) and reproduced here for simplicity:

$$\hat{\omega}^b_{ob,k} = \bar{\omega}^b_{ob,k} + \mathbf{K}_{\omega,k}\bar{\Delta}\epsilon_k \tag{6.6}$$

At the end of these calculations both the updates are stored in a vector, x_est, and saved to memory.

6.6.6 Error covariance update

Source C-code:

$$UpdateErrorCovariance(vPointers, vQuaternions);$$

will update the error covariance matrix $\mathbf{P}$ from equation (3.35) in the filter algorithm.

$$\mathbf{P}_{r,k} = [\mathbf{I} - \mathbf{K}_{r,k}\mathbf{H}_{r,k}]\bar{\mathbf{P}}_{r,k}[\mathbf{I} - \mathbf{K}_{r,k}\mathbf{H}_{r,k}]^T + \mathbf{K}_{r,k}\mathbf{R}\mathbf{K}^T_{r,k} \tag{6.7}$$

This, as with the Kalman gain matrix, is done using the reduced measurement matrix as well as the reduced Kalman gain matrix.

6.6.7 State transition

Source C-code:

$$UpdateStateTransitionMatrices(vPointers, vQuaternions);$$

is used to compute the state transition matrix Φ, used in the propagation of states and error covariance. Φ is found using forward Euler integration. Both a linearized propagation matrix for the reduced system as well as the nonlinear propagation matrix are produced here and saved to memory. The reduced matrix is used for error covariance propagation whereas the other is used for state propagation.

6.6.8 Propagation

Source C-code:

$$StatePropagation(vPointers, vQuaternions);$$

uses the state transition matrices to propagate the states x and error covariance $\mathbf{P}$. The corresponding equations from the Kalman filter algorithm are

$$\bar{\mathbf{x}}_{k+1} = \mathbf{\Phi}_k \hat{\mathbf{x}}_k \tag{6.8}$$

$$\bar{\mathbf{P}}_{k+1} = \mathbf{\Phi}_{r,k} \mathbf{P}_{r,k} \mathbf{\Phi}_{r,k}^T + \mathbf{Q}_r \tag{6.9}$$

6.6.9 Transmit estimate

Source C-code:

$$TransmitEstimates(X_{hat});$$

will transmit the new estimates to the On-board Data Handling, OBDH. Every time a new estimate is calculated it will be transmitted to the OBDH using the TWI. Because the estimates are of type **float** a conversion to type **char** is performed. This conversion is performed in the communication process.

6.7 Communication process

The communication process handles communication with the other subsystems. Key functions are sending new estimated states, receiving commands from subsystems and interpreting them. The communication process should consist of three subprocess and are shown in figure 6.4:

- Receive data - By using the interrupts available, activity on the bus is detected and the ADS can enter slave mode and listen on the bus for incoming messages. All messages received are interpreted by a different process and correct action is then taken.
- Transmit data - The only data to be sent from the ADS are new estimated states. Before transmission the data has to be converted. The address of the recipient is loaded from memory and added to the message.
- Interpret data - This is the function used to interpret incoming messages. In this function actions are taken based on the received message. This can involve starting and stopping the magnetometer, updating parameter values and request to transmit the latest estimates.

Data conversion

As stated in the previous section, communication using Two-Wire Interface requires a data conversion. The TWI sends data one byte at the time so a conversion from **float** of four bytes to **char** is necessary. The conversion process and sequence of sent bytes must be equal and known for all subsystems who receive or transmit on the bus.

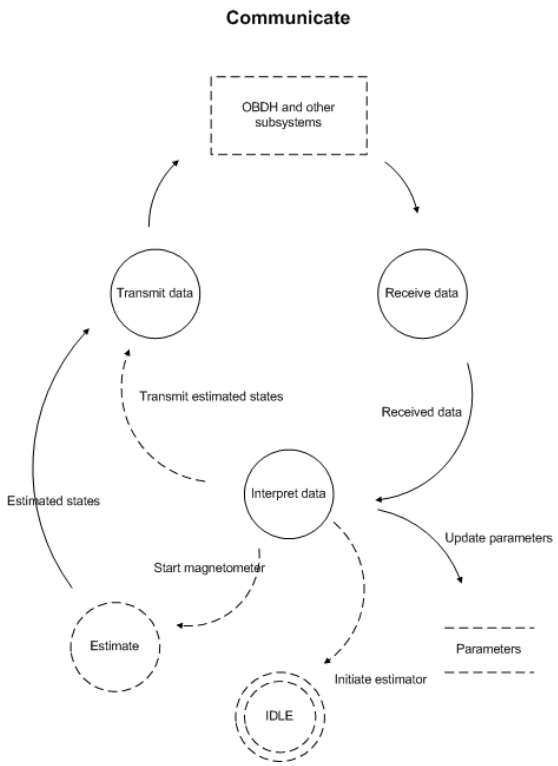

Figure 6.4: Dataflow chart for the communicate state

Chapter 7

Prototype Testing

Due to lack of time prototype testing of the extended Kalman filter has not been performed. Time and thought has however been put into ways that tests may be performed. In this section a few test schemes will be presented.

7.1 Matlab S-function

Because it can be difficult to create suitable test environments the use of Matlab S-functions can be used. This is a way to incorporate the c-code implemented Kalman filter algorithm into a simulink model to see how it performs. This method does of course have its limitations with respect to useful information that can be gathered. These limitations are, among others:

- Target capability - microcontroller speed
- Memory overflow
- Interactions with external elements; magnetometer and Sun sensors
- Communication - TWI

The Kalman filter can however be compared to the simulink implemented filter to check performance and reveal possible logical errors in the Kalman filter c-code. In order to do this, the c-files have to be slightly modified to make it compatible with Matlab. How to do this is explained in detail in the Matlab help files under "S-function".

7.2 Programmed test environment

To test the filter and overall system on the target, a testshell environment can be created. This can include a secondary target programed to act as a subsystem on the satellite. This allows for other tests to be performed on the implemented system, such as

- Sensor communication
- TWI communication
- Run-time error tests

A test environment can be programmed and executed on a desktop computer using RS232 to interface with the target. By using a terminal program, results from tests can be printed to the terminal to conclude on correctness of communication. Since the sensors will not provide any useful measurements, the Kalman filter algorithm can not be properly tested with this setup. But the sensors will provide measurements that can be used in the filter algorithm. This way timing tests and memory overflow tests can be performed by online debugging through the JTAG interface.

The two tests together will give results on the performance of the Kalman algorithm with respect to accuracy and performance of the microcontroller with respect to used time, computational power and the intercommunication between sensors, microcontroller and other subsystems. The power consumption can also be measured to ensure that the ADS system does not draw more power then it has been given.

Chapter 8

Concluding Remarks and Recommendations

This report is the first work done within attitude determination for a student satellite made exclusively by students at Norwegian University of Science and Technology.

8.1 Conclusion

The presented Extended Kalman Filter with guass-Newton algorithm has been implemented in c-code. In addition a library of matrix operations that is used in the algorithm has been created. Implementation of sensor communication has not been fully completed and thus testing to see if the filter is sufficiently simplified with regards to computational power has not been performed.

The Sun sensor uses hardware previously untested for a satellite and hence testing is required before a final conclusion can be given. Preliminary testing shows however that the irradiance level in the designated orbit might be to high for the Sun sensor to produce usable data. A different kind of sensor might be required for measuring the vector to the Sun.

The Unscented Kalman filter shows promise with respect to estimation accuracy and should be further pursued to see if it can compete both with respect to accuracy and required computational power.

8.2 Recommendations

In this section recommendations on further work will be presented along with some discussions.

Hardware

If the Sun sensor hardware proves to be unfit for use in the Low Earth Orbit, the use of light dependent resistors can be implemented with relative few changes to the system. This type of sensor for measuring light levels has previously proved to give adequate results for use in attitude determination and control of a satellite.

In the time since this work started, it seems a great deal of work has been put into the new 32-bit microcontroller from Atmel, mentioned in chapter 5. Although many of its features are of no use to the satellite functionality, the computational power can prove to be very useful. The controller is also rated as a low power microcontroller and I believe it should be tested. At this time, when not all the subsystems such as sensors and communication hava been fully implemented, it would be the right time if any to change controller. The already implemented Kalman filter algorithm and matrix operations are independent of the microcontroller type and will not require any reprogramming unlike the sensors who rely on hardware within the controller.

Implementation

The benefits of using **malloc** is that it allows the use of functions from Numerical Recipes. Even with these benefits, it does not outweigh the negative consequences that might arise from its use. This means that the filter equations have to be reprogrammed to not use memory allocation when creating the matrices. This is however not a very huge job, but still very important in order to avoid run-time errors as explained in chapter 6. This change also means that the multidimensional mathematical operations must be rewritten. The work effort of doing this should however not be significant as the difference is only the indexation of the matrices.

There is still some work left to be done with regards to implementation on the microcontroller. This is mainly communication with the sensors as well

as bus communication protocols. This must be completed before extensive testing can be performed.

The unscented Kalman filter should be tested out in Simulink and implemented on a microcontroller for comparison with the improved extended Kalman filter. Testing performed by others suggest that at a slight increase in computation time, a significant improvement in estimates is obtainable.

Testing

As this project did not see time to perform tests on the implemented system, this will have priority in any further work that is done. There are different ways that these tests can be performed, one is to implement the filter as a S-function in Matlab and use it in the Simulink model. This requires some modification of the c-code, but as mentioned the help-files for simulink are of great help here.

The second test solution is to create a shell program and test the filter while running on the microcontroller. This will have to be done eventually to check for run-time errors.

Bibliography

Appel, P. [2004], 'Attitude estimation from magnetometer and earth-albedo-corrected coarse sun sensor measurement', **56, Issues 1-2**, 115–126.

Bak, T. [1999], 'Spacecraft attitude determination : A magnetometer approach', *PDF in institutional repository: http://vbn.aau.dk/ws/fbspretrieve/108233/fulltext* . Phd thesis, Aalborg University.

Egeland, O. and Gravdahl, J. T. [2002], *Modeling and Simulation for Automatic Control*, Marine Cybernetics AS.

Farrell, J. A. and Barth, M. [1999], *The Global Position System & Inertial Navigation*, McGraw-Hill.

Julier, S. J. and Uhlmann, J. K. [1997], 'A new extension of the kalman filter to nonlinear systems'.

Kahan, W. [1997], 'Ieee standard 754 for binary floating-point arithmetic', *Lecture Notes on the Status of IEEE 754* .

Kristiansen, R. [2000], 'Attitude control of mini satellite'. Master Thesis, Department of Engineering Cybernetics, NTNU.

Kyrkjebø, E. [2000], 'Three-axis attitude determination using magnetometers and a star tracker'. Master Thesis, Department of Engineering Cybernetics, NTNU.

Ma, G.-F. and Jiang, X.-Y. [2005], 'Unscented kalman filter for spacecraft attitude estimation and calibration using magnetometer measurements'. Proceedings of the Fourth International Conference on Machine Learning and Cybernetics,.

Marins, J. L., Yun, X., Bachmann, E. R., McGhee, R. B. and Zyda, M. J. [2000], 'An extended kalman filter for quaternion-based orientation using marg sensors'.

Narverud, E., Blom, E. K. and Birkeland, R. [Nov, 2006], 'Student satellite proposal from ntnu'. Project work at NTNU.

Nocedal, J. and Wright, S. J. [1999], *Numerical Optimization*, Springer-Verlag New York, Inc.

Ose, S. S. [2004], 'Attitude determination for the norwegian student satellite ncube'. Master Thesis, Department of Engineering Cybernetics, NTNU.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. and p. Flannery, B. [2002], *Numerical recipes in C++*, 2nd edn, The Press Syndicate of the University of Cambridge.

Sunde, B. O. [2004], 'Attitude determination for studentsatellitten ncubeii:kalmanfilter'. Master Thesis, Department of Engineering Cybernetics, NTNU.

Svartveit, K. [2003], 'Attitude determination for norwegian nano satellite ncube'. Master Thesis, Department of Engineering Cybernetics, NTNU.

Wikipedia [2007*a*], 'Ieee 754 floating point', Webpage.

Wikipedia [2007*b*], 'Keplerian elements', webpage. Last Viewed 8. June 2007.
URL: *http://en.wikipedia.org/wiki/Keplerian_ elements*

Appendix A

Linearized Angular Velocity Model

$$\mathbf{F}_{vel} = \begin{bmatrix} a51 & a52 & a53 & a54 & 0 & a56 & a57 \\ a61 & a62 & a63 & a64 & a65 & 0 & a67 \\ a71 & a72 & a73 & a74 & a75 & a76 & 0 \end{bmatrix} \tag{A.1}$$

The elements a_{if} are shown below.

$$\begin{aligned}
a_{51} &= 2k_x\omega_o(\epsilon_1(\omega^b_{ob,y} - c_{22}\omega_o) - \eta(\omega^b_{ob,z} - c_{32}\omega_o)) - 6k_x\omega_o^2(\epsilon_1 c_{33} + \eta c_{23}) + 2(\eta\omega^b_{ob,z} + \epsilon_1\omega^b_{ob,y})\omega_o \\
a_{52} &= 2k_x\omega_o(\eta(\omega^b_{ob,y} - c_{22}\omega_o) - \epsilon_1(\omega^b_{ob,z} - c_{32}\omega_o)) + 6k_x\omega_o^2(\epsilon_1 c_{23} - \eta c_{33}) + 2(\eta\omega^b_{ob,y} - \epsilon_1\omega^b_{ob,z})\omega_o \\
a_{53} &= -2k_x\omega_o(\epsilon_3(\omega^b_{ob,y} - c_{22}\omega_o) - \epsilon_2(\omega^b_{ob,z} - c_{32}\omega_o)) + 6k_x\omega_o^2(\epsilon_2 c_{23} - \epsilon_3 c_{33}) + 2(\epsilon_3\omega^b_{ob,z} - \epsilon_3\omega^b_{ob,y})\omega_o \\
a_{54} &= 2k_x\omega_o(\epsilon_3(\omega^b_{ob,z} - c_{32}\omega_o) - \eta(\omega^b_{ob,y} - c_{22}\omega_o)) - 6k_x\omega_o^2(\epsilon_2 c_{33} + \epsilon_3 c_{23}) - 2(\epsilon_3\omega^b_{ob,z} + \epsilon_2\omega^b_{ob,y})\omega_o \\
a_{56} &= k_x(\omega^b_{ob,z} - c_{32}\omega_o) - c_{32}\omega_o \\
a_{57} &= k_x(\omega^b_{ob,y} - c_{22}\omega_o) + c_{22}\omega_o
\end{aligned} \tag{A.2}$$

$$\begin{aligned}
a_{61} &= 2k_y\omega_o(\epsilon_3(\omega^b_{ob,z} - c_{32}\omega_o) - \epsilon_1(\omega^b_{ob,x} - c_{12}\omega_o)) + 6k_y\omega_o^2(\eta c_{13} + \epsilon_2 c_{33}) - 2(\epsilon_1\omega^b_{ob,x} + \epsilon_3\omega^b_{ob,z})\omega_o \\
a_{62} &= 2k_y\omega_o(\epsilon_2(\omega^b_{ob,z} - c_{32}\omega_o) - \eta(\omega^b_{ob,x} - c_{12}\omega_o)) + 6k_y\omega_o^2(\epsilon_3 c_{33} - \epsilon_1 c_{13}) - 2(\eta\omega^b_{ob,x} + \epsilon_2\omega^b_{ob,z})\omega_o \\
a_{63} &= 2k_y\omega_o(\epsilon_1(\omega^b_{ob,z} - c_{32}\omega_o) - \epsilon_3(\omega^b_{ob,x} - c_{12}\omega_o)) - 6k_y\omega_o^2(\eta c_{33} - \epsilon_2 c_{13}) + 2(\epsilon_3\omega^b_{ob,x} - \eta\omega^b_{ob,z})\omega_o \\
a_{64} &= 2k_y\omega_o(\eta(\omega^b_{ob,z} - c_{32}\omega_o) - \epsilon_2(\omega^b_{ob,x} - c_{12}\omega_o)) + 6k_y\omega_o^2(\epsilon_1 c_{33} + \epsilon_3 c_{13}) + 2(\epsilon_3\omega^b_{ob,z} - \eta\omega^b_{ob,z})\omega_o \\
a_{65} &= -k_y(\omega^b_{ob,z} - c_{32}\omega_o) + c_{32}\omega_o \\
a_{67} &= -k_y(\omega^b_{ob,x} - c_{12}\omega_o) - c_{12}\omega_o
\end{aligned} \tag{A.3}$$

$$
\begin{aligned}
a_{71} &= 2k_z\omega_o(\epsilon_3(\omega^b_{ob,y} - c_{22}\omega_o) + \eta(\omega^b_{ob,x} - c_{12}\omega_o)) + 6k_z\omega_o^2(\epsilon_1 c_{13} - \epsilon_2 c_{23}) + 2(\epsilon_3\omega^b_{ob,x} - \eta\omega^b_{ob,x})\omega_o \\
a_{72} &= 2k_z\omega_o(\epsilon_2(\omega^b_{ob,y} - c_{22}\omega_o) - \epsilon_1(\omega^b_{ob,x} - c_{12}\omega_o)) + 6k_z\omega_o^2(\epsilon_3 c_{23} + \eta c_{13}) + 2(\epsilon_1\omega^b_{ob,x} + \epsilon_2\omega^b_{ob,y})\omega_o \\
a_{73} &= 2k_z\omega_o(\epsilon_1(\omega^b_{ob,y} - c_{22}\omega_o) + \epsilon_2(\omega^b_{ob,x} - c_{12}\omega_o)) - 6k_z\omega_o^2(\epsilon_3 c_{13} - \eta c_{23}) + 2(\epsilon_1\omega^b_{ob,y} - \epsilon_2\omega^b_{ob,x})\omega_o \\
a_{74} &= 2k_z\omega_o(\eta(\omega^b_{ob,y} - c_{22}\omega_o) - \epsilon_3(\omega^b_{ob,x} - c_{12}\omega_o)) + 6k_z\omega_o^2(\epsilon_1 c_{23} + \epsilon_2 c_{13}) + 2(\epsilon_3\omega^b_{ob,x} - \eta\omega^b_{ob,y})\omega_o \\
a_{75} &= -k_z(\omega^b_{ob,y} - c_{22}\omega_o) - c_{22}\omega_o \\
a_{76} &= -k_z(\omega^b_{ob,x} - c_{12}\omega_o) + c_{12}\omega_o
\end{aligned}
\tag{A.4}
$$

Appendix B

Simulink Model

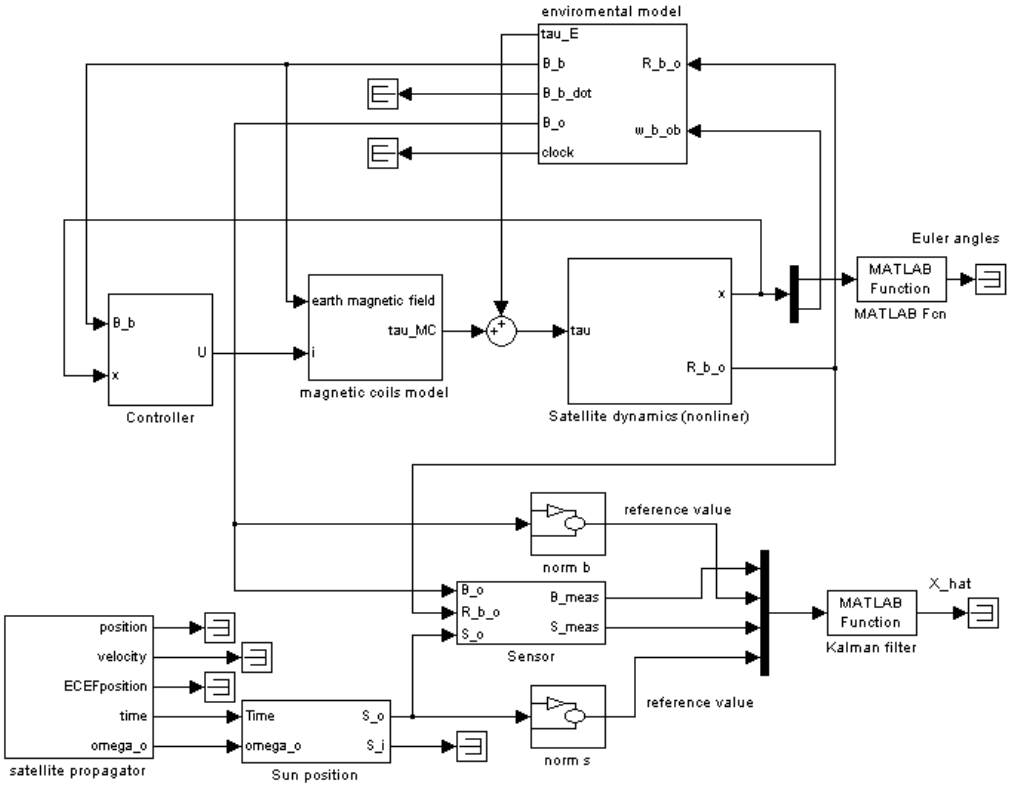

Figure B.1: Top view simulink model of satellite model and environment

Appendix C

CD Contents

The CD contains all the different files used and created in this project. Some files are on the root directory wheres other files are found in subdirectories on the CD.

Files on root directory:

- Student Satellite Proposal from NTNU

The different directories are presented next.

Directories

C files

Source code files for the implementation are presented here

- ads.c - The main file. Contain the state machine which controls the attitude determination system.
- ads_std_hdr.h - Standard headerfile included in all sourcefiles
- ekf.c - Contains all the functions of the implemented extended Kalman filter.
- magnetometer.c
- matrix_operations.c - Contains all the matrix operations used in the extended Kalman filter functions
- sunsensors.c
- twi.c - functions for handling TWI communication

- uart.c - functions for handling USART communication

Datasheets
This directory contains relevant datasheets in pfd-format.
Development board
Schematic files for the development board are found in this directory
Matlab
Files used in the Simulink model are presented here

- ECI2ECEF.m - transformation from ECI frame to ECEF frame
- EKF_init.m - Initialization for the filter
- EKF_simulation.mdl - Simulink file for the extended Kalman filter
- euler2q.m - creates a unit quaternion from euler angles
- genJacobian.m - Used to generate the Jacobian matrix for the Gauss-Newton method
- igrf2000.m
- kalman.m - The discrete extended Kalman filter algorithm
- LinearsystemR.m - Used to create the linear propagation matrix for the reduced system
- nonLinearProp.m - Used to create the non-linear propagation system for the system
- qProd.m - Performs a unit quaternion product
- qProdinv.m - Performs an inverse unit quaternion product
- Rquat.m
- Rxyx.m
- smtrx.m - computes the skew-symetric matrix of a vector
- suneci2orbit.m - Sun ECI to orbit frame