



**NTNU – Trondheim**  
Norwegian University of  
Science and Technology

# Development and Comparison of Estimation Methods for Attitude Determination

**Toril Bye Rinnan**

Master of Science in Engineering Cybernetics

Submission date: June 2012

Supervisor: Thor Inge Fossen, ITK

Norwegian University of Science and Technology  
Department of Engineering Cybernetics



# Development and Comparison of Estimation Methods for Attitude Determination

Toril Bye Rinnan

Department of Engineering Cybernetics

June 4, 2012





## MSC THESIS DESCRIPTION SHEET

**Name:** Toril Bye Rinnan  
**Department:** Engineering Cybernetics  
**Thesis title (Norwegian):** Utvikling og sammenligning av estimeringsmetoder for attitydebestemmelse  
**Thesis title (English):** Development and Comparison of Estimation Methods for Attitude Determination

**Thesis Description:** The purpose of the thesis is to investigate different estimation methods for satellite attitude determination and control. The considered satellite should be the NTNU CubeSat. The developed extended QUEST algorithm (EQUEST) should be implemented and compared to the nonlinear observer of Mahony. Several relevant satellite scenarios should be simulated.

The following items should be considered:

1. Define the scope of the thesis and clarify what your contributions are.
2. Literature study on estimation methods for attitude control including the QUEST, EQUEST and Mahony observer.
3. Develop a new extended quaternion estimator by changing the cost function to include quaternion products instead of quaternion subtractions.
4. Investigate the convergence and stability properties of the new EQUEST method and the Mahony observer.
5. Simulate the new extended quaternion estimator and compare it to the standard extended quaternion estimator as well as the extended Kalman filter. Verify the results by simulations.
6. Find realistic test scenarios for testing of the attitude estimators.
7. Conclude your results.

**Start date:** 2012-01-09  
**Due date:** 2012-06-04

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



## Abstract

The NTNU Test Satellite (NUTS) project, is part of the Norwegian Student Satellite program ANSAT. The goal of the project is to design and launch a double CubeSat by the end of 2014. During earlier satellite projects at NTNU, solid work on design of the attitude determination and control for a small satellite has been done.

One of the considered estimation methods for the attitude determination is the extended quaternion estimation method (EQUEST). Further development and testing of the method is described in this thesis. In addition to the new EQUEST method, a nonlinear observer has been implemented and tested. The simulation results for the two methods are compared in order to find the attitude estimation method best suited for the NUTS satellite.

The new EQUEST method has several advantages over the more common Kalman filtering for use in small CubeSats. It is less computationally costly, and has a fast start-up and settling time. Magnetorquers, which affect the local magnetic field, are used as actuators for the satellite. This makes a fast algorithm preferable, since the attitude estimation and the attitude control should be performed separately. The nonlinear observer is slower than the EQUEST method, but it can guarantee global exponential stability and it is less vulnerable to disturbances. It is therefore introduced as an alternative solution for the attitude determination problem.

The original EQUEST method builds upon the QUEST method which has been extended to include non-vectorized terms for gyroscope measurements and attitude prediction in the method's cost function. In these terms, subtractions between the estimated and measured quaternions are used. The result is not entirely mathematically correct, even though previous testing of the method has been successful. The subtractions in the included terms will not result in new attitude error quaternions.

In this thesis, the method is further developed by replacing the subtraction terms with quaternion products. The new method is tested and compared to the original EQUEST method and an extended Kalman filter. It is also compared to the implemented nonlinear observer. If the computational power of the NUTS satellite is sufficiently large, a combination of the developed EQUEST method and the nonlinear observer could be considered for the estimation. A combination of these two attitude estimation methods is implemented and the simulation results are analyzed.





## Sammendrag

NTNU's testsatellitt (NUTS) er en del av det nasjonale studentsatellittprogrammet ANSAT. Prosjektets mål er design og oppskyting av en dobbel kubesatellitt innen utgangen av 2014. I løpet av tidligere satellittprosjekter ved NTNU har mye arbeid blitt gjort innen utforming av attitydebestemmelse og attityderegulering for en liten satellitt.

En av estimeringsmetodene som vurderes for attitydebestemmelsen, er en utvidet kvaternionestimator, kalt EQUEST. Utvikling og testing av metoden er beskrevet i denne masteroppgaven. I tillegg til den nye EQUEST-metoden, har en ulineær observer blitt implementert og testet. Simuleringsresultatene for de to metodene er sammenlignet for å finne den metoden som egner seg best for NUTS-satellitten.

Den nye EQUEST-metoden har flere fordeler sammenlignet med det mer vanlige Kalmanfilteret for bruk i en liten kubesatellitt. Den krever færre regneoperasjoner, og har en kortere oppstarts- og innsvingningstid. Magnetiske spoler som vil påvirke det lokale magnetiske feltet, er brukt i attitydereguleringssystemet til satellitten. Dette gjør det ønskelig med en rask algoritme, siden attitydebestemmelsen og attitydereguleringen ikke bør utføres samtidig. Den ulineære observeren er tregere enn EQUEST-metoden, men den kan garantere global eksponentiell stabilitet, og den er mindre følsom for forstyrrelser. Den er derfor introdusert som en alternativ løsning for attitydeproblemet.

Den originale EQUEST-metoden bygger på QUEST-metoden, som har blitt utvidet til å inkludere ikke-vektoriserte ledd for gyroskopmålinger og attitydeprediksjon i metodens kostfunksjon. I disse leddene er subtraksjoner mellom de estimerte og målte kvaternionene brukt. Resultatet blir en metode som ikke er helt matematisk korrekt, selv om tidligere testing har vist gode resultater. Subtraksjonene i de inkluderte leddene, resulterer ikke i nye attitydefeil-kvaternioner.

I denne masteroppgaven er metoden utviklet videre ved å bytte ut subtraksjonsleddene med kvaternionmultiplikasjoner. Den nye metoden er testet og sammenlignet med den originale EQUEST-metoden og et utvidet Kalmanfilter. Den er også sammenlignet med den implementerte ulineære observeren. Hvis regnekraften til NUTS-satellitten er stor nok, kan en kombinasjon av den utviklede EQUEST-metoden og den ulineære observeren bli vurdert for estimeringen. En kombinasjon av disse to attitydeestimerings-metodene er implementert, og den nye metodens egenskaper er analysert.



## **Preface**

This thesis is the final work of my Masters degree provided by the Norwegian University of Science and Technology (NTNU). The work has been done at the Department of Engineering Cybernetics.

I would like to thank my supervisors, Thor Inge Fossen and Jan Tommy Gravdahl, for their guidance and support. The NUTS satellite project has been both educational and fun, and I would like to thank Roger Birkeland for his optimism, and the rest of the NUTS team members involved in the project the spring of 2012, for their help and enthusiasm.

I hope our knowledge will reach orbit.

Toril Bye Rinnan.  
Trondheim, June 2012.



# Contents

## 1 Introduction

- 1.1 Motivation . . . . .
- 1.2 The NTNU Test Satellite project . . . . .
- 1.3 Previous work . . . . .
- 1.4 Main contribution . . . . .
- 1.5 Outline of thesis . . . . .

## 2 Preliminaries

- 2.1 Coordinate frames . . . . .
  - 2.1.1 ECI frame . . . . .
  - 2.1.2 ECEF frame . . . . .
  - 2.1.3 NED frame . . . . .
  - 2.1.4 BODY frame . . . . .
- 2.2 Attitude representations . . . . .
  - 2.2.1 Euler angles . . . . .
  - 2.2.2 Quaternions . . . . .
- 2.3 Quaternion properties . . . . .
  - 2.3.1 Advantages of quaternions . . . . .
  - 2.3.2 Multiplication of quaternions . . . . .
  - 2.3.3 Quaternions and rotations . . . . .
- 2.4 Wahba's problem . . . . .
- 2.5 Cholesky factorization . . . . .
- 2.6 Lyapunov analysis . . . . .

## 3 Sensors

- 3.1 Magnetometer . . . . .
- 3.2 Gyroscope . . . . .
- 3.3 Sun sensor . . . . .
- 3.4 Time-varying reference vectors . . . . .
  - 3.4.1 Magnetic field model . . . . .
  - 3.4.2 Sun reference model . . . . .

## 4 Estimation methods

- 4.1 Quaternion estimation (QUEST) . . . . .
- 4.2 Extended Quaternion estimation (EQUEST) . . . . .
- 4.3 Optimization of the cost function . . . . .

4.4	Developed EQUEST convergence properties . . . . .	
4.5	Nonlinear observer . . . . .	
4.6	Stability properties for the nonlinear observer . . . . .	
4.7	Normalization . . . . .	
4.8	Quaternions to Euler angles . . . . .	
<b>5</b>	<b>Matlab implementation</b>	
5.1	Sensor data . . . . .	
5.2	Reference vector models . . . . .	
5.3	Prototype code . . . . .	
5.4	Extended Kalman filter . . . . .	
<b>6</b>	<b>Testing</b>	
6.1	Developed EQUEST . . . . .	
6.2	Developed EQUEST with loss of sun sensor data . . . . .	
6.3	Nonlinear observer . . . . .	
6.3.1	Sine wave . . . . .	
6.3.2	Step function . . . . .	
6.4	Nonlinear observer with loss of data . . . . .	
6.5	Nonlinear observer bias . . . . .	
6.6	Nonlinear observer with variable gains . . . . .	
6.7	Nonlinear observer with disturbances . . . . .	
6.8	Nonlinear observer with time-varying reference vectors . . . . .	
6.9	Combination of developed EQUEST and Nonlinear observer . . . . .	
6.10	Test cases conclusion . . . . .	
<b>7</b>	<b>Discussion</b>	
<b>8</b>	<b>Conclusions</b>	
<b>9</b>	<b>Further work</b>	
<b>A</b>	<b>Schmidt quasi-normalized Legendre function</b>	
<b>B</b>	<b>European CubeSat Symposium abstract and presentation</b>	
<b>C</b>	<b>Code</b>	
C.1	Code for the developed EQUEST method . . . . .	
C.2	Code for the nonlinear observer . . . . .	
C.2.1	Code for the projection algorithm used for the nonlinear observer	
C.3	Contents of CD . . . . .	

## Nomenclature

$\phi$	Euler angle, roll
$\theta$	Euler angle, pitch
$\psi$	Euler angle, yaw
$\mathbf{R}(\cdot)$	Rotation matrix using Euler angles
$\mathbf{q}$	Unit quaternion
$q_0$	Scalar part of unit quaternion
$\mathbf{q}_{vec}$	Vector part of unit quaternion
$\mathbf{Q}$	Quaternion matrix
$\nu$	General Euler angle
$\mathbf{n}$	Unit vector
$\mathbf{I}$	Identity matrix
$\mathbf{S}(\cdot)$	Skew symmetric matrix
$\mathbf{R}_n^b(\cdot)$	Rotation matrix representing a rotation from n to b
$\mathbf{M}$	Least squares estimate of a rotation matrix
$\mathbf{r}$	Known directional unit vector in the NED frame
$\mathbf{b}$	Known directional unit vector in the BODY frame
$\mathbf{A}$	Cholesky factor
$\mathbf{L}$	Lower triangular matrix
$x_e$	Equilibrium point
$V(\cdot)$	Lyapunov function
$\alpha$	Convergence rate gain
$\beta$	Convergence rate decrease constant
$r$	Geocentric radius for the satellite
$\mu_c$	Co-latitude ( $90^\circ - \mu$ ) of the satellite
$\mu$	Latitude of the satellite
$l$	Longitude of the satellite
$r_E$	Radius of the Earth
$g_n^m$	Vertical Gaussian coefficient
$h_n^m$	Horizontal Gaussian coefficient
$\mathcal{P}_n^m$	Schmidt quasi-normalized Legendre function
$\mathbf{B}$	Magnetic field vector
$\alpha_t$	Local sidereal time
$\epsilon_{sun}$	Elevation of the Sun
$T_{sun}$	Days since the Earth passed the vernal equinox
$\lambda_{sun}$	Azimuth angle towards the Sun
$J(\cdot)$	Cost function
$\sigma$	Standard deviation
$\mathbf{q}_{pre}$	Predicted unit quaternion
$\mathbf{q}_{gyro}$	Gyroscope unit quaternion
$\mathbf{D}$	Weight matrix for the gyroscope
$\mathbf{S}$	Weight matrix for the attitude prediction
$\mathbf{e}_{pre}$	Predicted error quaternion
$\mathbf{e}_{gyro}$	Gyroscope error quaternion

$\mathbf{N}_d$	Gyro weight matrix multiplied with gyro quaternion matrices
$\mathbf{N}_s$	Prediction weight matrix multiplied with predicted quaternion matrices
$\mathcal{L}(\cdot)$	Lagrange function
$\mathbf{G}$	Lagrange matrix
$\mathbf{x}$	State vector
$c$	Lagrange constant
$\mathbf{V}$	Lagrange matrix for the QUEST method
$\lambda$	Eigenvalue of the Lagrange matrix
$\mathbf{C}$	Orthogonal eigenvector matrix
$\mathbf{e}$	Eigenvector
$a$	Eigenvalue constant
$\lambda_{\mathbf{V}}$	Eigenvalue of $\mathbf{V}$
$\boldsymbol{\omega}_m$	Measured gyroscope data
$\boldsymbol{\omega}_{inj}$	Injection term
$\mathbf{b}_g$	Gyroscope bias
$k$	Injection term gain
$\mathbf{T}(\cdot)$	Nonlinear observer matrix
$w_p$	Bound for the projection term
$h$	Euler sampling time
$\gamma$	Symmetric positive definite projection matrix
$\mathbf{b}_{true}$	True unbiased vector measurement
$\mathbf{b}_{bias}$	Vector measurement bias
$\mathbf{\Gamma}$	Symmetric positive definite vector bias matrix
$M$	Bound for the gyro bias
$c_{obs}$	Constant Lyapunov parameter
$\epsilon$	Convergence bound for $\mathbf{V}$
$W$	Lyapunov function for the nonlinear observer
$\boldsymbol{\omega}$	Unbiased angular velocity
$\epsilon_{bias}$	Convergence bound for $W$
$T_b$	Integral time constant for vector bias
$N$	Bound for vector bias term
$V_{bias}$	Lyapunov function for the vector bias
$\lambda_{min}(\cdot)$	Minimum eigenvalue
$\lambda_{max}(\cdot)$	Maximum eigenvalue
$T_s$	Time constant for the implemented EKF
$\mathbf{b}_{bias}$	Bias for the vector measurements
$\mathbf{z}$	Measurement vector for the EKF
$\mathbf{P}$	Estimate covariance matrix for the EKF
$\mathbf{F}$	State estimate matrix for the EKF
$\mathbf{Q}_{kal}$	Process covariance matrix for the EKF
$\mathbf{K}$	Gain matrix for the EKF
$\mathbf{H}$	State estimate update matrix for the EKF
$\mathbf{R}_{kal}$	Measurement covariance matrix for the EKF
$\mathbf{q}_{ref}$	Implemented initial value for the reference quaternions
$\mathbf{q}_{method}$	Implemented initial value for the different methods
$\omega$	Scalar unbiased angular velocity
$w$	Measurement noise



## List of Figures

5.1	Test orbit for the NUTS satellite. . . . .
5.2	Test orbit for the NUTS satellite. . . . .
5.3	Graph for latitude, longitude and height. . . . .
6.1	Attitude estimation using the original EQUEST method and the EKF. . . . .
6.2	Attitude estimation using the developed EQUEST method and the EKF. . . . .
6.3	Sensor data and deviation for the extended Kalman filter. . . . .
6.4	Sensor data and deviation for the original EQUEST method. . . . .
6.5	Sensor data and deviation for the developed EQUEST method. . . . .
6.6	Attitude estimation using the developed EQUEST method and the EKF, with loss in sun sensor data along the x-axis. . . . .
6.7	Sensor data and deviation for the EKF, with loss in sun sensor data along the x-axis. . . . .
6.8	Sensor data and deviation for the developed EQUEST method, with loss in sun sensor data along the x-axis. . . . .
6.9	Attitude estimation using the developed EQUEST method and the EKF, with loss in sun sensor data along the y-axis. . . . .
6.10	Sensor data and deviation for the EKF, with loss in sun sensor data along the y-axis. . . . .
6.11	Sensor data and deviation for the developed EQUEST method, with loss in sun sensor data along the y-axis. . . . .
6.12	Attitude estimation using the developed EQUEST method and the EKF, with loss in sun sensor data along the z-axis. . . . .
6.13	Sensor data and deviation for the EKF, with loss in sun sensor data along the z-axis. . . . .
6.14	Sensor data and deviation for the developed EQUEST method, with loss in sun sensor data along the z-axis. . . . .
6.15	Attitude estimation sine wave response for the developed EQUEST method and the nonlinear observer. . . . .
6.16	Sensor data and deviation for the nonlinear observer. . . . .
6.17	Sensor data and deviation for the developed EQUEST method. . . . .
6.18	Attitude determination step response for the developed EQUEST method and the nonlinear observer. . . . .
6.19	Sensor data and deviation for the nonlinear observer. . . . .
6.20	Sensor data and deviation for the developed EQUEST method. . . . .

6.21	Attitude determination for the nonlinear observer and the developed EQUEST method with loss of sun sensor data along the x-axis. . . . .	
6.22	Sensor data and deviation for the nonlinear observer, with loss of sun sensor data along the x-axis. . . . .	
6.23	Sensor data and deviation for the developed EQUEST method, with loss of sun sensor data along the x-axis. . . . .	
6.24	Attitude determination for the nonlinear observer and the developed EQUEST method with loss in sun sensor data along the y-axis. . . . .	
6.25	Sensor data and deviation for the nonlinear observer, with loss in sun sensor data along the y-axis. . . . .	
6.26	Sensor data and deviation for the developed EQUEST method, with loss in sun sensor data along the y-axis. . . . .	
6.27	Attitude determination for the nonlinear observer and the developed EQUEST method with loss of sun sensor data around the z-axis. . . . .	
6.28	Sensor data and deviation for the nonlinear observer, with loss in sun sensor data along the z-axis. . . . .	
6.29	Sensor data and deviation for the developed EQUEST method, with loss in sun sensor data along the z-axis. . . . .	
6.30	Sine wave response for the estimated bias of the nonlinear observer. . . . .	
6.31	Step response for the estimated bias of the nonlinear observer. . . . .	
6.32	Gains in the injection term for the sun sensor and magnetometer measurement vectors. . . . .	
6.33	Attitude determination for the developed EQUEST method and the nonlinear observer with variable gains in the injection term. . . . .	
6.34	Sensor data and deviation for the nonlinear observer with variable gains in the injection term. . . . .	
6.35	Sensor data and deviation for the developed EQUEST method with variable gains in the injection term. . . . .	
6.36	Attitude determination for the developed EQUEST method, the EKF and the nonlinear observer with disturbances in the sensor data. . . . .	
6.37	Sensor data and deviation for the nonlinear observer with disturbances. . . . .	
6.38	Sensor data and deviation for the developed EQUEST method with disturbances. . . . .	
6.39	Sensor data and deviation for the EKF with disturbances. . . . .	
6.40	Attitude determination for the developed EQUEST method and the nonlinear observer with time-varying reference vectors. . . . .	
6.41	Sensor data and deviation for the nonlinear observer with time-varying reference vectors. . . . .	
6.42	Sensor data and deviation for the developed EQUEST method with time-varying reference vectors. . . . .	
6.43	Sensor data and deviation for the EKF with time-varying reference vectors. . . . .	
6.44	Attitude determination for the combination method and the EKF. . . . .	
6.45	Sensor data and deviation for the combination of the developed EQUEST and the nonlinear observer. . . . .	
6.46	Sensor data and deviation for the EKF. . . . .	

6.47	Attitude determination for the combination method and the EKF with disturbances in the sensor data. . . . .
6.48	Sensor data and deviation for the combination of the developed QUEST and the nonlinear observer with disturbances. . . . .
6.49	Sensor data and deviation for the EKF with disturbances. . . . .
6.50	Attitude determination for a combination method and the EKF with noise along all the axes for the sensor data. . . . .



# Chapter 1

## Introduction

This master thesis describes the design, analysis and development of two estimation methods for an attitude determination system used in a small CubeSat. One of the presented methods is a modification of the extended quaternion estimator (EQUEST), which builds on the quaternion estimator (QUEST). The other investigated method is a nonlinear observer, with stability properties based on Lyapunov analysis. The estimation methods are simulated in Matlab, and the test results are compared. The work has been part of the NTNU satellite project during the spring of 2012.

### 1.1 Motivation

An attitude determination and control system (ADCS) is important for orientation control of a satellite. Without reliable attitude estimates, mission objectives may be severely compromised. Estimation methods are needed to determine the current attitude. An extended quaternion estimation (EQUEST) method and a nonlinear observer have been developed and implemented for comparison of the test results. Due to limited space, weight and power, estimation methods used for larger satellites are less suited for implementation in CubeSats.

### 1.2 The NTNU Test Satellite project

The final estimation method is intended for use in the ADCS of a CubeSat. The satellite project at the Norwegian University of Science and Technology (NTNU) is part of the Norwegian Student Satellite Program run by the Norwegian Centre for Space-related Education (NAROM). The goal of the program is to build and launch three student satellites by the end of 2014. Students from NTNU, the University of Oslo and Narvik University College are each building a CubeSat. The NTNU Test Satellite (NUTS) is the last of the satellites to be launched, late in 2014.

The first students started working on the NUTS project in the spring semester of 2011, and 9 students from 5 different departments at NTNU have been involved in the project during the spring of 2012. Weekly meetings have enabled the group to share information and discuss the progress of the project. The team work required in the project has been

both interdisciplinary and international. In addition to weekly meetings, an effort to recruit new students to the project led to student involvement in design and production of stand material, PR work and the development of a new web site: [www.nuts.cubesat.no](http://www.nuts.cubesat.no). Each semester, the project and master work has been presented for other students, teachers and sponsors. In February, the annual European CubeSat Symposium were held in Brussels, and 5 students went to the conference to hold presentations and get feedback from space communities around the world. The submitted abstract and presentation can be found in appendix B.

The CubeSat dimensions are  $10 \times 10 \times 20$  cm, and the weight is limited to 2.6 kg. The CubeSat will have a polar orbit with an inclination close to  $90^\circ$ . The payload will be a small IR camera which will take pictures of the atmospheric gravity waves, in addition to a wireless internal databus.

### 1.3 Previous work

Two CubeSat projects at NTNU were launched unsuccessfully before the current NUTS project was initiated, and a lot of work has been done on the attitude estimation problem during these previous projects. Work on a discrete Kalman filter has been done by Svartveit [1] and Ose [2], and an extended Kalman filter was described by Rhode in 2007 [3]. In 2011, the difficulties related to these estimation methods led Jenssen and Yabar [4] to compare a new estimation method to the extended Kalman filter, described by Sabatini [5]. They extended the quaternion estimation (QUEST) method, based on work from Psiaki [6] and Markley [7], to include non-vectorized gyroscope measurements and prediction terms. The extension has been further developed with the introduction of quaternion products in the method's cost function. Analysis of the method and convergence properties is presented in this thesis.

Work on a nonlinear observer was first introduced in 2008 by Mahony et al. [41]. The method provides both attitude and gyro bias estimates. For stationary reference vector measurements or for unbiased gyro, this observer is globally exponentially stable. Further development has been done by Grip et al. [42]. By adding a parameter projection and a vector bias estimation, the result is a globally exponentially stable nonlinear observer for time-varying reference vectors. Such an observer is implemented and compared to the developed EQUEST method.

### 1.4 Main contribution

The work presented in this thesis, can be divided into two main parts. The first part is concerned with the further development of an extended quaternion estimator (EQUEST) [4], based on the quaternion estimator (QUEST), first introduced by Shuster and Oh in 1981 [34]. The EQUEST method was extended to include attitude predictions and gyroscope information in the cost function. The work done in this thesis is based on that extension. The included terms consist of quaternion subtractions, but in order to get a mathematically correct model, these subtractions must be replaced by multiplications. The cost function for the model has been modified, and the new method has been tested and compared to the

old original EQUEST method. The convergence properties of this new method has also been analyzed, and a proof for the result is presented.

The second part of the thesis includes a comparison of the new developed EQUEST method to another relatively new estimation method, a nonlinear observer, which was first introduced by Mahony et al. in 2008 [41]. This nonlinear observer has global exponential stability properties, and is therefore highly suitable for attitude estimation. Both methods are implemented and simulated for several test cases. The results are useful for determining which estimation method to use for the NTNU Test Satellite project. A final method, combining the developed EQUEST method and the nonlinear observer, has been implemented and tested.

## 1.5 Outline of thesis

**Chapter 1:** An introduction on the motivation for the thesis and the background work is given.

**Chapter 2:** Several aspects of attitude determination are introduced. The coordinate frames used for the estimation methods are described, and attitude representations including quaternion theory are presented. Understanding the behavior of quaternions is important when working with the estimation methods described in this thesis, particularly for the development of the extended quaternion estimator (EQUEST). Some background material for stability analysis is covered in this chapter.

**Chapter 3:** The chapter gives a presentation of the sensors used to obtain attitude information for the satellite. The magnetometer, gyroscope and sun sensor are described. When the satellite is in orbit, varying reference vector models must be used. Such models for the magnetometer and the sun sensor are presented.

**Chapter 4:** The design of the estimation methods considered for the NUTS CubeSat are described. Their convergence properties are investigated, along with comments on normalization and singularity problems for the quaternions and Euler angles.

**Chapter 5:** Parameters and models used for the Matlab implementation for the estimation methods are presented.

**Chapter 6:** Included test results from the implemented Matlab code presented in section 5 are shown, in addition to detailed descriptions of the test cases and the test results.

**Chapter 7:** The chapter contains a discussion and an evaluation of the work presented in this thesis.

**Chapter 8:** Concluding remarks are given.

**Chapter 9:** Future work on the attitude determination and control system for the NUTS CubeSat is described.





# Chapter 2

## Preliminaries

This chapter introduces several concepts regarding attitude determination. Frames and notations used for the estimation methods are presented. To fully understand the theory behind the estimation methods, quaternion theory and relevant quaternion properties are needed. Background information on some stability concepts used in this thesis is described.

### 2.1 Coordinate frames

For the attitude determination, the measurement vectors and the corresponding reference vectors used in the estimation methods are represented in different coordinate frames. A frame is a system which determines a point on a manifold using coordinates [8]. Definitions of the coordinate systems used in the attitude determination and control system for the NUTS satellite are described in the following pages. This theory can be found in many books, but most of the material has been obtained from Fossen [9].

#### 2.1.1 ECI frame

The Earth Centered Inertial frame has its x-axis pointing towards the vernal equinox, and its z-axis pointing along the rotation axis of the Earth at some initial time. The y-axis completes a right handed orthogonal coordinate system. The frame's origin is at the center of the Earth.

#### 2.1.2 ECEF frame

This frame also has its origin at the center of the Earth, but the Earth Centered Earth Fixed frame has its x-axis pointing towards the point where the intersection between the longitude and latitude have zero value. It can also be described as the intersection between the Greenwich meridian and the Equator. The frame's z-axis is pointing along the Earth's rotation axis. The y-axis completes the right handed orthogonal system. The ECEF frame is not an inertial frame, it rotates relative to the ECI frame along the Earth rotation.

### **2.1.3 NED frame**

The North East Down frame has its z-axis pointing downwards, perpendicular to the tangent plane of the Earth's reference ellipsoid. The ellipsoid is mathematically defined and fitted for approximation of the Earth. The x-axis points towards true north and the y-axis points East. The NED frame is an inertial frame.

### **2.1.4 BODY frame**

This frame is attached to the satellite, and is moving and rotating with it. The origin coincides with the origin of the NED frame. The axes coincide with the principle axes of inertia; the x-axis is pointing forwards, the y-axis is pointing to the right side and the z-axis is pointing downwards through the camera side of the satellite.

## 2.2 Attitude representations

Several representations for describing attitude are available, the most common being Euler angles. More complicated attitude representations are quaternions. Quaternions are used for all the estimation methods presented in this thesis. They are singular-free, and are therefore well suited for attitude determination.

### 2.2.1 Euler angles

Euler angles were first described by Leonhard Euler in 1776, and are used to represent the orientation of a body [10]. Three parameters are required for a full understanding of the orientation between two frames, one angle for the rotation around each of the axes. The angles are called roll, pitch and yaw and are usually written as  $\phi$ ,  $\theta$  and  $\psi$ . The Euler angles are often used for the definition of rotation matrices about the x, y and z-axis. In  $\mathbb{R}^3$ , the coordinate system rotations in a counter-clockwise direction looking towards the origin are given from:

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \quad (2.1)$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \quad (2.2)$$

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.3)$$

### 2.2.2 Quaternions

Quaternions were first described by Sir William Rowan Hamilton in 1843 [11]. His intention was to find an extension of vector algebra, and in 1845 Arthur Cayley published an article where he used multiplication of quaternions to describe rotations [12]. Three of the four elements of a quaternion give the coordinates for the axis of rotation, while the fourth is described by the angle of rotation [13].

A quaternion can be written as a four-dimensional vector:

$$\mathbf{q} := \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \quad (2.4)$$

The real part of the quaternion behaves like a scalar in the three-dimensional vector space. Using a rotation angle  $\nu$ , the real part can be written as:

$$q_0 = \cos(\nu/2) \quad (2.5)$$

The imaginary part uses a unit vector given from  $\mathbf{n} = \frac{\mathbf{n}}{\|\mathbf{n}\|}$  where the norm of the vector,  $\|\mathbf{n}\|$ , is defined as the square root of each of the squared elements of  $\mathbf{n}$  added together. Throughout the thesis, vectors and matrices will be written in bold print. The imaginary part can be written as a vector:

$$\mathbf{q}_{vec} := \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = [\mathbf{n} \sin(\nu/2)] \quad (2.6)$$

There are several ways to write quaternions. Sometimes it is convenient to think of a quaternion as the sum of a scalar and a vector written as:

$$\mathbf{q} := q_0 + \mathbf{q}_{vec} = q_0 + q_1i + q_2j + q_3k \quad (2.7)$$

Complex numbers can be represented as matrices, and so can quaternions. A quaternion describes a point in 4D space, and can be represented by a  $4 \times 4$  matrix by using a left-isoclinic rotation as proved in [14], and used in [16] and [17]:

$$\mathbf{Q} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \quad (2.8)$$

The transpose of the matrix is the same as the conjugate of the quaternion:

$$\mathbf{q}^* := q_0 - \mathbf{q}_{vec} = q_0 - q_1i - q_2j - q_3k \quad (2.9)$$

Two quaternions are conjugate if they are orthogonal with respect to their inner product. [15]. The inverse of a quaternion  $\mathbf{q}$  is defined as  $\mathbf{q}^{-1} = \frac{1}{\mathbf{q}}$ . In this report, the mathematics are based on unit quaternions, which satisfies the constraint:

$$\mathbf{q}^\top \mathbf{q} = 1 \quad (2.10)$$

The length of a unit quaternion is 1, which leads its inverse to be its conjugate.

## 2.3 Quaternion properties

### 2.3.1 Advantages of quaternions

The unit quaternion notation is compact, and round off errors are easier to handle than for matrix representation. The nearest orthonormal matrix to one that is not quite orthonormal, is difficult to find. Multiplying unit quaternions may similarly lead to quaternions that are no longer of unit length, but these can easily be normalized to make sure they correspond to valid rotations. The computational cost of normalizing a quaternion is much less than for normalizing a matrix.

Quaternions are safe from a phenomenon called gimbal lock. When the pitch angle in a pitch/roll/yaw-system is rotated  $90^\circ$  up or down, and the yaw and roll correspond to the same motion, a degree of freedom of rotation can be lost. In a gimbal-based aerospace inertial navigation system, this could have disastrous results if the aircraft is in a steep dive or ascent [16]. The quaternion elements vary continuously over the unit sphere in  $\mathbb{R}^4$ , (denoted by  $S^3$ ) as the orientation changes, avoiding this problem.

Due to the possible spin in the CubeSat, combined with the singularity problem for Euler angles, an attitude estimation method based on unit quaternions is preferred [4].

### 2.3.2 Multiplication of quaternions

In the developed EQUEST method, quaternion multiplications are used instead of quaternion subtractions. The result is a mathematically correct attitude quaternion. The quaternion product is more complicated than a vector or matrix product. In order to do the multiplication, the following rules for the imaginary operators are required [11]:

$$i^2 = j^2 = k^2 = -1 \quad (2.11)$$

$$ij = k \quad (2.12)$$

$$ji = -k \quad (2.13)$$

$$jk = i \quad (2.14)$$

$$kj = -i \quad (2.15)$$

$$ki = j \quad (2.16)$$

$$ik = -j \quad (2.17)$$

The multiplication of two quaternions is not commutative, and can be shown to consist of the cross product between the vector parts of the quaternions, and the dot product of the two quaternions [17].

The two quaternions used in the multiplication are defined as  $\mathbf{q} := q_0 + \underbrace{q_1i + q_2j + q_3k}_{\mathbf{q}_{vec}}$  and  $\mathbf{r} := r_0 + \underbrace{r_1i + r_2j + r_3k}_{\mathbf{r}_{vec}}$ .

The quaternion product is defined as:

$$\mathbf{q} \times \mathbf{r} = \mathbf{q}_{vec} \times \mathbf{r}_{vec} - \mathbf{q} \cdot \mathbf{r} \quad (2.18)$$

Written in matrix form, the multiplication of the two quaternions is given from [7] and [18]:

$$\mathbf{q} \otimes \mathbf{r} = \begin{bmatrix} q_0r_0 - \mathbf{q}_{vec} \cdot \mathbf{r} \\ q_0\mathbf{r}_{vec} + r_0\mathbf{q} + \mathbf{q} \times \mathbf{r} \end{bmatrix} = \begin{bmatrix} q_0r_0 - q_1r_1 - q_2r_2 - q_3r_3 \\ q_0r_1 + q_1r_0 + q_2r_3 - q_3r_2 \\ q_0r_2 + q_2r_0 + q_3r_1 - q_1r_3 \\ q_0r_3 + q_3r_0 + q_1r_2 - q_2r_1 \end{bmatrix} \quad (2.19)$$

### 2.3.3 Quaternions and rotations

Since quaternion multiplication is non-commutative, it is closely related to three-dimensional rotations. The multiplication of two quaternions can be substituted by rotation matrix multiplication for less computing [16]. A unit quaternion is defined as described in equations (2.5) and (2.6), and can be written as:

$$\mathbf{q}(\nu, \mathbf{n}) = \cos(\nu/2) + \mathbf{n} \sin(\nu/2) = e^{\mathbf{n}(\nu/2)} \quad (2.20)$$

where  $e$  is the exponential function defined as the solution of the integral  $\int_1^x \frac{dt}{t}$ . This gives a correspondence between unit quaternions and proper orthogonal matrices as representations of rotations:

$$\mathbf{q}(\nu, \mathbf{n}) \leftrightarrow \mathbf{R}(\nu, \mathbf{n}) \quad (2.21)$$

Rotation matrices using Euler angles are defined in equations (2.1),(2.1) and (2.1). Using a rotation angle  $\nu$ , a general rotation matrix can be written as:

$$\mathbf{R}(\nu, \mathbf{n}) = e^{\nu \mathbf{S}(\mathbf{n})} := \mathbf{I}_{3 \times 3} + (\sin \nu) \mathbf{S}(\mathbf{n}) + (1 - \cos \nu) (\mathbf{S}(\mathbf{n}))^2 \quad (2.22)$$

where  $\mathbf{I}_{3 \times 3}$  is the  $3 \times 3$  identity matrix consisting only of zeros, with elements of value 1 on the diagonal. In this thesis, the matrix  $\mathbf{S}$  will represent a skew symmetric matrix given from:

$$\mathbf{S}(\mathbf{n}) := \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix} \quad (2.23)$$

where  $\mathbf{n}$  is a unit vector defined as:

$$\mathbf{n} := \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} \quad (2.24)$$

A rotation matrix representing a rotation from frame  $\mathbf{b}$  to  $\mathbf{n}$ , can be expressed with unit quaternions as [19]:

$$\mathbf{R}_b^n(\mathbf{q}) = (q_0^2 - \|\mathbf{q}_{vec}\|^2) \mathbf{I}_{3 \times 3} + 2\mathbf{q}_{vec} \mathbf{q}_{vec}^\top - 2q_0 \mathbf{S}(\mathbf{q}_{vec}) = \Xi^\top(\mathbf{q}) \Psi(\mathbf{q}) \quad (2.25)$$

where

$$\Xi(\mathbf{q}) := \begin{bmatrix} q_0 \mathbf{I}_{3 \times 3} + \mathbf{S}(\mathbf{q}_{vec}) \\ -\mathbf{q}_{vec}^\top \end{bmatrix} \quad (2.26)$$

$$\Psi(\mathbf{q}) := \begin{bmatrix} q_0 \mathbf{I}_{3 \times 3} - \mathbf{S}(\mathbf{q}_{vec}) \\ -\mathbf{q}_{vec}^\top \end{bmatrix} \quad (2.27)$$

The skew symmetric matrix  $\mathbf{S}(\mathbf{q}_{vec})$  is given in the same way as equation (2.23) as:

$$\mathbf{S}(\mathbf{q}_{vec}) := \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix} \quad (2.28)$$

Using the the quaternion notation from equation (2.4) with  $\mathbf{q} := q_0 + \mathbf{q}_{vec}$ , where  $\mathbf{q}_{vec} :=$

$[q_1 \ q_2 \ q_3]^\top$ , another way of writing the rotation matrix is:

$$\mathbf{R}_n^b(\mathbf{q}) := \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \quad (2.29)$$

This matrix is orthogonal, which means it is a square matrix with its transpose equal to its inverse:  $\mathbf{R}^T = \mathbf{R}^{-1}$ . Note that this rotation matrix and the representation in equation (2.30) are transposed compared to the rotation matrix in (2.25). The transpose of a matrix is obtained by interchanging the elements of the rows and columns. This matrix represent therefore a rotation from frame n to b.

A third way of writing the rotation matrix used in the implementation of the nonlinear observer described in chapter (4.5) is [9]:

$$\mathbf{R}_n^b(\mathbf{q}) := \mathbf{I}_{3 \times 3} + 2q_0\mathbf{S}(\mathbf{q}_{vec}) + 2\mathbf{S}(\mathbf{q}_{vec})^2 \quad (2.30)$$

Note that the last rotation matrix notation in (2.29), along with the representation in equation (2.30) is the inverse of the rotation matrix in (2.25).

A quaternion can be found from the diagonal elements of a rotation matrix, by the following formulas [20]:

$$q_0 = \sqrt{\frac{1}{4}(1 + \mathbf{R}_{11} + \mathbf{R}_{22} + \mathbf{R}_{33})} \quad (2.31)$$

$$q_1 = \sqrt{\frac{1}{4}(1 + \mathbf{R}_{11} - \mathbf{R}_{22} - \mathbf{R}_{33})} \quad (2.32)$$

$$q_2 = \sqrt{\frac{1}{4}(1 - \mathbf{R}_{11} + \mathbf{R}_{22} - \mathbf{R}_{33})} \quad (2.33)$$

$$q_3 = \sqrt{\frac{1}{4}(1 - \mathbf{R}_{11} - \mathbf{R}_{22} + \mathbf{R}_{33})} \quad (2.34)$$

From the off-diagonal elements, the following formulas are obtained:

$$q_0q_1 = \frac{1}{4}(\mathbf{R}_{32} - \mathbf{R}_{23}) \quad (2.35)$$

$$q_0q_2 = \frac{1}{4}(\mathbf{R}_{13} - \mathbf{R}_{31}) \quad (2.36)$$

$$q_0q_3 = \frac{1}{4}(\mathbf{R}_{21} - \mathbf{R}_{12}) \quad (2.37)$$

$$q_1q_2 = \frac{1}{4}(\mathbf{R}_{12} - \mathbf{R}_{21}) \quad (2.38)$$

$$q_1q_3 = \frac{1}{4}(\mathbf{R}_{13} - \mathbf{R}_{31}) \quad (2.39)$$



$$q_2q_3 = \frac{1}{4}(\mathbf{R}_{23} - \mathbf{R}_{32}) \quad (2.40)$$

Since the quaternions  $\mathbf{q}$  and  $-\mathbf{q}$  are the same, either sign of the square root can be computed. After solving for one  $\mathbf{q}_k$  for  $k \in 0, 1, 2, 3$  in equations (2.31)-(2.34), the three equations which contain  $\mathbf{q}_k$  on the left hand side in equations (2.35)-(2.40) can be solved for the remaining elements of  $\mathbf{q}$ .

## 2.4 Wahba's problem

The developed extended QUEST method described in this report builds upon the principles of Wahba's problem. The problem was first stated by Grace Wahba in 1965 [21]. Given two sets of vector observations, a rotation matrix  $\mathbf{M}$  can be found which minimizes the orientation error. This is an optimization problem, where the cost function is:

$$\sum_j^n \|\mathbf{r}_j - \mathbf{M}\mathbf{b}_j\| \quad (2.41)$$

For satellite attitude determination, the vectors  $\mathbf{r}_j$  for  $j \in \{1, n\}$  are the reference sensor data given in the NED frame. The vectors  $\mathbf{b}_j$  for  $j \in \{1, n\}$  are the measured sensor data in the BODY frame.  $\mathbf{M}$  is the least squares estimate of the rotation matrix which carries the known frame of reference into the satellite fixed frame of reference.

The QUEST method uses this problem in order to minimize the attitude estimation error, as described in section 4.1.

## 2.5 Cholesky factorization

In order to find the eigenvalues of the matrices in the EQUEST method, Matlab uses Cholesky factorization. For a matrix  $\mathbf{A} \in \mathbb{R}^{n \times n}$  which is symmetric positive definite, a specialized factorization method called Cholesky factorization is possible. For a unique lower triangular matrix  $\mathbf{L} \in \mathbb{R}^{n \times n}$  with elements only in the lower left triangle, the Cholesky factor is produced as [22]:

$$\mathbf{A} := \mathbf{L}\mathbf{L}^\top \quad (2.42)$$

The cost of this factorization is  $\frac{n^3}{3} + O(n^2)$  flops and  $n$  square roots, where a flop represents any of the four elementary operations  $+$ ,  $-$ ,  $*$ ,  $/$ . The symmetry properties ensure that updating of the elements are only needed on and below the diagonal.

The algorithm is specified according to [23] as

```

for i = 1:n
   $\mathbf{L}_{ii} \leftarrow \sqrt{\mathbf{A}_{ii}}$ 
  for j = i+1:n
     $\mathbf{L}_{ji} \leftarrow \mathbf{A}_{ji}/\mathbf{L}_{ii}$ 
    for k = i+1:j
       $\mathbf{A}_{jk} \leftarrow \mathbf{A}_{jk} - \mathbf{L}_{ji}\mathbf{L}_{ki}$ 
    end
  end
end

```

Since the matrix  $\mathbf{A}$  is positive definite, the argument of the square root will always be positive, and the matrix  $\mathbf{L}$  has real and positive elements on the diagonal [24].

The Cholesky factorization is a numerically stable factorization algorithm [22]. Using the inequality  $l_{ij}^2 \leq \sum_{k=1}^i l_{ik}^2 = a_{ii}$ , the entries of  $\mathbf{L}$  are bounded [25]. Written in a different way, using norm notation, the Cholesky factor can never grow too large [24].

$$\|\mathbf{L}\| = \sqrt{\|\mathbf{A}\|} \quad (2.43)$$

## 2.6 Lyapunov analysis

The second estimation method described in this thesis is a nonlinear observer, which can be proven to be exponentially asymptotically stable. The proof of the stability properties makes use of Lyapunov analysis. This section describes the general theory of the Lyapunov stability concept.

The idea is that if all solutions that start near an equilibrium point stay close to this equilibrium point in the future, the equilibrium point is Lyapunov stable. If all the solutions that start near the equilibrium point, also converge to the equilibrium point, asymptotical stability is proved. If this convergence is faster than the rate  $\alpha \|x_e(0) - x_e\| e^{-\beta t}$ , the equilibrium point is exponentially asymptotically stable. The equilibrium point is denoted by  $x_e$  and the initial solution is  $x_e(0)$ . Time is denoted by  $t$ , and  $\alpha$  and  $\beta$  are constants.

The most common method for proving Lyapunov stability makes use of a Lyapunov function [26]. Such a function is given from  $V(x) : \mathbb{R}^n \rightarrow \mathbb{R}$  such that  $V(x) \geq 0$  and  $\dot{V}(x) \leq 0$ . Both equations have equality only if  $x = 0$ . It is required that  $V(0) = 0$ . A Lyapunov function candidate might not satisfy the conditions needed to prove stability. This does not mean that the equilibrium point is unstable, it just means that other Lyapunov functions might be tried, or that other methods for proving instability can be used.

# Chapter 3

## Sensors

Sensors are used to measure the vector components in the BODY and NED frames. An attitude determination system requires at least two vectors in order to estimate the attitude. Three types of sensors are used in the attitude determination and control system for the NUTS CubeSat. A magnetometer will be used, in addition to a sun sensor and a gyroscope. Sensor information is combined in the attitude determination system to provide the best possible estimates at all times. There are two classes of sensors commonly used in attitude determination systems; inertial sensors and reference sensors.

Inertial sensors are sensors that measure rotation or translational acceleration relative to an inertial frame. The sensors experience random drift and bias errors, and as a result, the errors are not bounded. Regular updates based on references such as the Sun, stars, or the Earth can be used in order to correct the errors. The gyroscope is an example of an inertial sensor.

Reference sensors typically provides noisy vector observations at a low frequency. A reference sensor measures the direction of a known vector. The vector measurement is a function of spacecraft attitude, but one sample from a reference sensor does not provide full attitude information. Therefore, two vector directions are needed for a complete estimation. Both the sun sensor and the magnetometer are reference sensors.

### 3.1 Magnetometer

A magnetometer measures the local magnetic field of the satellite. Magnetometers are inexpensive, reliable and light-weight which make them suitable for small satellites. The difference between computed and measured magnetic field components is a function of the spacecraft attitude, which means that the attitude vector can be estimated. The satellite must know the real magnetic field for every position in its orbit. The attitude is determined from a magnetometer by comparing the measured geomagnetic field with a reference field determined by a reference model [27]. Several models of the Earth's magnetic field exists, and a balance between a simple and accurate model is required for the estimation of attitude. The field model most often used is the International Geomagnetic Reference Field (IGRF) which is described in chapter (3.4.1) [28]. The Earth's magnetic field will influence the accuracy of the magnetometer. An altitude of 200 km will typically give an

accuracy of  $1^\circ$ , but the accuracy will improve for higher altitudes [29].

## 3.2 Gyroscope

Gyroscopes measure the angular velocity. Theoretically a gyroscope can track the orientation of the satellite by integrating the change in velocity. In order to provide good estimation the initial orientation must be correct and the measurement errors should be small. All gyroscopes have some measurement error, called bias. Biases are usually constant or slowly varying. Because of the bias, the orientation can not be tracked by only using gyros. With time, the attitude estimation will drift. The drift rate vary according to the choice of sensor, but will typically be less than 1 deg/s. [9]. An advantage is that many gyroscopes are small and can provide quite good measurement data.

## 3.3 Sun sensor

Sun sensors are popular, accurate and reliable, but require clear fields of view. The main idea is to measure the direction to the Sun. Small and cheap sun sensors can be bought and placed on each side of the satellite in order to detect the Sun angle. For the NUTS CubeSat, the solar cells might be used. Solar cells are not really sensors, but can be used to detect the direction to the Sun by monitoring the output current. The output from a solar cell depends on the angle between the solar panel and the sun rays. The sun sensors mounted on the satellite will be sensitive to every light source in space. Because of light reflected from the Earth, it is important to use an Earth albedo compensation when computing a sun vector, or else a large angular deviation might occur [30]. A sun sensor reference model is needed for the attitude determination, and such a model is presented in chapter (3.4.2) [31]. The typical field of view for a sun sensor is  $\pm 30^\circ$ , with an accuracy of approximately  $0,01^\circ$  [29].

## 3.4 Time-varying reference vectors

Previous testing of the implemented estimation methods has focused on the static case with the reference vectors as constant values. Since the CubeSat has been tested in Trondheim, the magnetometer reference vector has been implemented as the magnetic field vector for this site. The vector was found at the website for the National Geophysical Data Center (<http://www.ngdc.noaa.gov/>), as:  $\mathbf{r}_2 = [13598.5 \quad 444.7 \quad -49854.8]^\top$ . The sun sensor reference vector has been implemented as:  $\mathbf{r}_1 = [0 \quad 0 \quad 9.81]^\top$ . The vector  $\mathbf{r}_1$  represents the sum of the mass attraction gravity term and the centripetal term resulting from the Earth rotation along the z-axis [45]. This corresponds to an output from an accelerometer, which can be used for testing the prototype as seen in [4]. It will eventually be replaced by a sun sensor.

When the satellite is in orbit, the reference vectors will vary with the position of the satellite relative to the Earth. A model of the Earth's magnetic field and a sun sensor reference model are needed to get time-varying reference vectors in the NED frame. These

models are described below, and test results are presented in chapter 6.8. In addition to the sun sensor reference model, an albedo compensation for light reflections from the Earth is needed. Such a model is not investigated in this thesis.

### 3.4.1 Magnetic field model

The model for the Earth magnetic field is called the International Geomagnetic Reference Field, and the parameters for the model is updated every five years by IAGA (the International Association of Geomagnetism and Aeronomy) [32]. The version used in this thesis was updated in 2011, and is called IGRF11.

The magnetic field around the Earth cannot be modeled accurately, because of variations in the magnetic field. One of the most important variations consists of temporal changes, due to periods of solar activity facing the Earth. Diurnal variations are caused by particle movement in the ionosphere. In addition to these variations, magnetic storms appear during solar flares, and will have an impact on the magnetic field. For the most part such storms occur in the same periodic pattern as the temporal changes.

To describe the magnetic field, a harmonic model using the negative gradient of the scalar potential function  $V$  can be implemented [27].

$$V(r, \mu_c, l) = r_E \sum_{n=1}^k \left(\frac{r_E}{r}\right)^{n+1} \sum_{m=0}^n (g_n^m \cos(ml) + h_n^m \sin(ml)) P_n^m(\mu_c) \quad (3.1)$$

The reference radius of the Earth is denoted by  $a$  with the value of  $r_E = 6371200$  m. The geocentric coordinates are  $r, \mu_c$  and  $l$ , where  $r$  is the radius in km,  $\mu_c$  is the co-latitude (given by  $90^\circ - \mu$ , where  $\mu$  is the latitude) and  $l$  is the longitude. The geocentric model assumes that the Earth is the center of the universe, which can be useful when dealing with space-related problems or navigation issues. The Gaussian coefficients  $g_n^m$  and  $h_n^m$  are given from the IGRF11 model. The parameter  $P_n^m(\mu_c)$  is the Schmidt quasi-normalized Legendre function. The function has order  $m$  and degree  $n$ . The calculation of  $P_n^m(\mu_c)$  is described in appendix A.

The reference model consist of the following equations [27]:

$$B_r = \frac{-\partial V}{\partial r} = \sum_{n=1}^k \left(\frac{r_E}{r}\right)^{n+2} (n+1) \sum_{m=0}^n (g_n^m \cos(ml) + h_n^m \sin(ml)) P_n^m(\mu_c) \quad (3.2)$$

$$B_{\mu_c} = \frac{-1}{r} \frac{\partial V}{\partial \mu_c} = - \sum_{n=1}^k \left(\frac{r_E}{r}\right)^{n+2} \sum_{m=0}^n (g_n^m \cos(ml) + h_n^m \sin(ml)) \frac{\partial P_n^m(\mu_c)}{\partial \mu_c} \quad (3.3)$$

$$B_l = \frac{-1}{r \sin \mu_c} \frac{\partial V}{\partial l} = \frac{-1}{\sin \mu_c} \sum_{n=1}^k \left(\frac{r_E}{r}\right)^{n+2} \sum_{m=0}^n m (-g_n^m \cos(ml) + h_n^m \sin(ml)) P_n^m(\mu_c) \quad (3.4)$$

The parameters  $B_r, B_{\mu_c}$  and  $B_l$  are the field strength in local tangential coordinates. The output of the model is changed into geocentric inertial coordinates using the following equations:

$$B_x = (B_r \cos \mu + B_{\mu_c} \sin \mu) \cos \alpha_t - B_l \sin \alpha_t \quad (3.5)$$

$$B_y = (B_r \cos \mu + B_{\mu_c} \sin \mu) \sin \alpha_t + B_l \cos \alpha_t \quad (3.6)$$

$$B_z = (B_r \cos \mu + B_{\mu_c} \sin \mu) \quad (3.7)$$

where  $\mu$  is the latitude and  $\alpha_t$  is the local sidereal time of the location of the satellite. The magnetometer reference vector is therefore changed from the static vector into the time-varying vector:  $\mathbf{r}_2 := [B_x \ B_y \ B_z]^\top$ .

### 3.4.2 Sun reference model

The Sun's position relative to the satellite's orbital position is needed for utilization of the Sun vector measurement. The reference model described is based on work by Svartveit [1] and Sunde [31], and is simplified by assuming an Earth-based position (geocentric), where the Sun revolves around the Earth.

The elevation of the Sun varies periodically through the year and is given by:

$$\epsilon_{sun} = \frac{2\pi}{180} \sin\left(\frac{T_{sun}}{365} 2\pi\right) \quad (3.8)$$

The parameter  $T_{sun}$  is the time in days since the Earth passed the vernal equinox. For this thesis, the vernal equinox is defined as the spring equinox as seen from the Northern Hemisphere.

The The azimuth angle between the satellite and the Sun is given from:

$$\lambda_{sun} = \frac{T_{sun}}{365} 2\pi \quad (3.9)$$

The time-varying sun sensor reference vector is:

$$\mathbf{r}_1 := \mathbf{R}_\theta(\epsilon_{sun}) \mathbf{R}_\psi(\lambda_{sun}) \mathbf{r}_1^0 \quad (3.10)$$

where  $\mathbf{r}_1^0 = [1 \ 0 \ 0]^\top$  and the rotation matrices are expressed using Euler angles from equations (2.2) and (2.3) as:

$$\mathbf{R}_\theta = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \quad (3.11)$$

$$\mathbf{R}_\psi = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.12)$$

It should be noted that an albedo compensation model should be implemented to avoid errors caused by reflections from the Earth [30]. Such a model is not implemented in the test cases investigated in this thesis.



# Chapter 4

## Estimation methods

The attitude determination problem is unique, because it is both under- and overdetermined. Two vectors are needed for determination of the attitude, but because of the unit constraint, a unit vector will only contain two parameters. The requirement is therefore for more than one, but less than two vector measurements [29]. Accurate determination is not possible, but estimation methods must be utilized.

Numerous methods for attitude determination are based on minimizing the Wahba loss function described in section 2.4. These point estimation methods will fail when only one vector measurement is available, or when the observations are parallel [33].

As these solutions are relatively simple and as most spacecraft have at least two reference vector measurements available, such single frame solutions are widely used. They are, however, not easily adapted to handle faults or periods of poor observability.

The estimation methods considered for the NUTS CubeSat must be fast and have a limited number of operations. The sensors used in the satellite will have measurement noise for high frequencies, which means that the chosen estimation method should be robust and not too sensitive to disturbances. For these reasons, a new extended quaternion estimator (EQUEST) has been developed and compared to a nonlinear observer. The observer has a slower response, but is at the same time more robust than the developed EQUEST method.

This chapter describes the background theory for the following estimation methods:

- QUEST
- original EQUEST
- developed EQUEST
- Nonlinear observer

Potential problems with maintaining unit quaternions and converting quaternions into Euler angles are also discussed.

## 4.1 Quaternion estimation (QUEST)

The quaternion estimation (QUEST) method first introduced by Shuster and Oh [34] is a fast and popular estimation method. The QUEST method builds on the q-method developed by P. Davenport in 1968, cited in [35] and [36]. The main reason for developing the quaternion estimator, is to obtain a faster way of finding the optimal eigenvalue for the cost function. The method reformulates an eigenvalue problem to a problem of solving a characteristic equation. An algorithm is used to find the optimal rotation, which integrate all the different sensor measurements for the attitude [6]. Finding the optimal rotation matrix between two known vectors, can be written as an optimization problem as described in section 2.4. The idea is to minimize the cost function, which is defined as:

$$J(\mathbf{q}) = \frac{1}{2} \sum_{j=1}^n \frac{1}{\sigma_j^2} (\mathbf{b}_j - \mathbf{R}_b^i(\mathbf{q})\mathbf{r}_j)^\top (\mathbf{b}_j - \mathbf{R}_b^i(\mathbf{q})\mathbf{r}_j) = \frac{1}{2} \sum_{j=1}^n \frac{1}{\sigma_j^2} (\mathbf{b}_j^\top \mathbf{b}_j - 2\mathbf{b}_j^\top \mathbf{R}_b^i(\mathbf{q})\mathbf{r}_j + \mathbf{r}_j^\top \mathbf{r}_j) \quad (4.1)$$

In this equation,  $\mathbf{r}_j$  and  $\mathbf{b}_j$  are known sensor data vectors, and these vectors must be of equal length for the algorithm to work well. The vector,  $\mathbf{r}_j$  is given in the NED frame, while  $\mathbf{b}_j$  is given in the fixed BODY frame. The parameter  $\sigma_j$  is the standard deviation of the measurement error. Because of the unit length requirement, the cost function can be reduced to:

$$J(\mathbf{q}) = \sum_{j=1}^n \frac{1}{\sigma_j^2} (1 - \mathbf{b}_j^\top \mathbf{R}_b^i(\mathbf{q})\mathbf{r}_j) \quad (4.2)$$

One of the advantages with the QUEST method is that it does not depend on initial conditions, and the exact solution can be found after just one time-step. The method explicitly preserves the quaternion normalization.

One disadvantage is that the method can only estimate the attitude quaternions, which leave the sensor biases unknown. Because point estimation methods, like QUEST only utilize the vector measurements obtained for a single time-step to determine the attitude, information contained in past measurements is lost. This leads to a high sensitivity to noise. The QUEST method requires vectorized direction input vectors. Measurements from a gyroscope will be useless because this is non-vectorized data. This limits the range of application of the method.

## 4.2 Extended Quaternion estimation (EQUEST)

The QUEST method can only deal with very simple dynamic models, and cannot estimate anything apart from the attitude quaternion. An extension of the method, has been made by Psiaki [6], in order to include *a priori* information and effects of other state vector measurements. The improvements in performance led Jenssen and Yabar [4] to develop an extended quaternion estimator (EQUEST) method based on the results of Psiaki.

For the extension of the QUEST method, the cost function is modified to include terms with gyroscope measurements and attitude prediction. Two new quaternions are included in the equation. A linear prediction term,  $\hat{\mathbf{q}}_{pre}$  is based on previous samples. The other extension denoted by  $\hat{\mathbf{q}}_{gyro}$  is the next quaternion estimated by tracking the gyroscope. The original cost function described by equation (4.1) is extended to:

$$J(\mathbf{q}) = \frac{1}{2} \sum_{j=1}^n \left\{ \frac{1}{\sigma_j^2} (\mathbf{b}_j - \mathbf{R}_b^i(\mathbf{q})\mathbf{r}_j)^\top (\mathbf{b}_j - \mathbf{R}_b^i(\mathbf{q})\mathbf{r}_j) \right\} + \frac{1}{2} (\mathbf{q} - \hat{\mathbf{q}}_{gyro})^\top \mathbf{D} (\mathbf{q} - \hat{\mathbf{q}}_{gyro}) + \frac{1}{2} (\mathbf{q} - \hat{\mathbf{q}}_{pre})^\top \mathbf{S} (\mathbf{q} - \hat{\mathbf{q}}_{pre}) \quad (4.3)$$

Two new symmetric weight matrices,  $\mathbf{D}$  and  $\mathbf{S}$  are introduced. The added terms penalizes deviations both from the predicted attitude, and the rotation matrix estimated by the gyroscope measurements. It is possible to predict the orientation based on previous attitude calculations, as long as the change in attitude is slow. The change will be minimal for a short period of time. In this period, several attitude calculations are made, and a linear relation can be established between the change in attitude and time [4].

The extension of the QUEST method does not solve the problem of estimating the biases, but this could be solved by other techniques before subtracting the results from the measurements in the cost function. This will, however, be computationally expensive.

The method is developed and implemented for the use of an accelerometer, a gyroscope and a magnetometer, but the accelerometer will eventually be replaced by a sun sensor measuring the direction towards the sun. This replacement will only influence the input of the method, and the reference vector. Both are easily manipulated.

Jenssen and Yabar [4] used subtractions in the two added terms, which makes it easy to minimize the cost function because the minimum value will be zero. However, the subtraction of two quaternions does not result in a new attitude quaternion. To achieve this, the quaternion subtraction terms must be replaced with quaternion products. The mathematics for the replacement is presented below, and test results for the implemented method are presented in chapter 6.1. Throughout the rest of this thesis, the new EQUEST method with quaternion products instead of quaternion subtractions will be referred to as the developed EQUEST method. The method with subtraction terms, developed by Jenssen and Yabar, will be referred to as the original EQUEST method.

The quaternion representation leads to a useful formula for finding the shortest rotation from one orientation to another, as seen in [18] and [37]. By letting the error quaternions be  $\mathbf{e}_{gyro} = \hat{\mathbf{q}}_{gyro}^* \otimes \mathbf{q}$  and  $\mathbf{e}_{pre} = \hat{\mathbf{q}}_{pre}^* \otimes \mathbf{q}$ , the conjugated quaternions  $\hat{\mathbf{q}}_{gyro}^*$  and  $\hat{\mathbf{q}}_{pre}^*$  will rotate into  $\mathbf{q}$  by the shortest rotations. Intuitively, it is possible to think about multiplication of one quaternion by the conjugate of the other as a "subtraction", though remembering

that it is not commutative [38]. The quaternion product replacement, transforms the cost function of the extended QUEST method into:

$$J(\mathbf{q}) = \frac{1}{2} \sum_{j=1}^n \left\{ \frac{1}{\sigma_j^2} (\mathbf{b}_j - \mathbf{R}_b^i(\mathbf{q})\mathbf{r}_j)^\top (\mathbf{b}_j - \mathbf{R}_b^i(\mathbf{q})\mathbf{r}_j) \right\} + \frac{1}{2} (\hat{\mathbf{q}}_{gyro}^* \otimes \mathbf{q})^\top \mathbf{D} (\hat{\mathbf{q}}_{gyro}^* \otimes \mathbf{q}) + \frac{1}{2} (\hat{\mathbf{q}}_{pre}^* \otimes \mathbf{q})^\top \mathbf{S} (\hat{\mathbf{q}}_{pre}^* \otimes \mathbf{q}) \quad (4.4)$$

The quaternions and matrices in this new optimization problem are the same as before, given by the original EQUEST cost function in equation (4.3). As seen from equation (2.8), the conjugated quaternions in the product terms of equation (4.4), can be replaced by matrix representations:

$$J(\mathbf{q}) = \frac{1}{2} \sum_{j=1}^n \left\{ \frac{1}{\sigma_j^2} (\mathbf{b}_j - \mathbf{R}_b^i(\mathbf{q})\mathbf{r}_j)^\top (\mathbf{b}_j - \mathbf{R}_b^i(\mathbf{q})\mathbf{r}_j) \right\} + \frac{1}{2} (\hat{\mathbf{Q}}_{gyro}^* \mathbf{q})^\top \mathbf{D} (\hat{\mathbf{Q}}_{gyro}^* \mathbf{q}) + \frac{1}{2} (\hat{\mathbf{Q}}_{pre}^* \mathbf{q})^\top \mathbf{S} (\hat{\mathbf{Q}}_{pre}^* \mathbf{q}) \quad (4.5)$$

The order of the factors is reversed when transposed. This can be used for the attitude quaternion matrices, and the corresponding predicted quaternions. This general property can be written as:

$$(\hat{\mathbf{Q}}^* \mathbf{q})^\top = \mathbf{q}^\top \hat{\mathbf{Q}}^{*\top} \quad (4.6)$$

Using this for the two last terms, the cost function is rearranged into:

$$J(\mathbf{q}) = \frac{1}{2} \sum_{j=1}^n \left\{ \frac{1}{\sigma_j^2} (\mathbf{b}_j - \mathbf{R}_b^i(\mathbf{q})\mathbf{r}_j)^\top (\mathbf{b}_j - \mathbf{R}_b^i(\mathbf{q})\mathbf{r}_j) \right\} + \frac{1}{2} \mathbf{q}^\top (\hat{\mathbf{Q}}_{gyro}^{*\top} \mathbf{D} \hat{\mathbf{Q}}_{gyro}^*) \mathbf{q} + \frac{1}{2} \mathbf{q}^\top (\hat{\mathbf{Q}}_{pre}^{*\top} \mathbf{S} \hat{\mathbf{Q}}_{pre}^*) \mathbf{q} \quad (4.7)$$

The three matrices in each of the two last terms can be multiplied together using matrix multiplication:

$$J(\mathbf{q}) = \frac{1}{2} \sum_{j=1}^n \left\{ \frac{1}{\sigma_j^2} (\mathbf{b}_j - \mathbf{R}_b^i(\mathbf{q})\mathbf{r}_j)^\top (\mathbf{b}_j - \mathbf{R}_b^i(\mathbf{q})\mathbf{r}_j) \right\} + \frac{1}{2} \mathbf{q}^\top \mathbf{N}_d \mathbf{q} + \frac{1}{2} \mathbf{q}^\top \mathbf{N}_s \mathbf{q} \quad (4.8)$$

$$\mathbf{N}_d = \hat{\mathbf{Q}}_{gyro}^{*\top} \mathbf{D} \hat{\mathbf{Q}}_{gyro}^* \quad \text{and} \quad \mathbf{N}_s = \hat{\mathbf{Q}}_{pre}^{*\top} \mathbf{S} \hat{\mathbf{Q}}_{pre}^*$$

The minimization of the cost function can now be found, by utilizing the fact that the multiplication of two unit quaternions preserves their dot product. When the two quaternions in the error quaternion point in the same direction, the cost function is minimized. The dot product ensures maximal directional matching [38]. Minimization of the whole cost function is described in section (4.3).

### 4.3 Optimization of the cost function

Jenssen and Yabar [4] used Lagrange multipliers to solve the minimization problem for the original EQUEST method. This can also be used for the developed EQUEST method with the cost function given from equation (4.8).

The Lagrange method requires an equation on the form:

$$\mathcal{L}(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{G} \mathbf{x} + \mathbf{x}^\top \mathbf{c} \quad (4.9)$$

The first part of the developed EQUEST method cost function is identical to the QUEST method cost function in (4.1). It can be written on the quadratic form [7]:

$$g(\mathbf{q}) = -\mathbf{q}^\top \mathbf{V} \mathbf{q} \quad (4.10)$$

$\mathbf{V}$  is a symmetric matrix written as [4]:

$$\mathbf{V} := \begin{bmatrix} \mathbf{U} - \varphi \mathbf{I}_{3 \times 3} & \mathbf{Z} \\ \mathbf{Z}^\top & \varphi \end{bmatrix} \quad (4.11)$$

The matrix elements are:

$$\mathbf{U} := \mathbf{L}_s + \mathbf{L}_s^\top \quad (4.12)$$

$$\mathbf{L}_s := \sum_{j=1}^n \frac{1}{\sigma_j^2} (\mathbf{b}_j \mathbf{r}_j^\top) \quad (4.13)$$

$$\mathbf{Z} := \begin{bmatrix} L_{s,23} - L_{s,32} \\ L_{s,31} - L_{s,13} \\ L_{s,12} - L_{s,21} \end{bmatrix} \quad (4.14)$$

$$\varphi := \text{Trace}(\mathbf{L}_s) \quad (4.15)$$

where the trace of the matrix sums the elements on the main diagonal. Using the entire cost function in equation (4.8), the two last terms must be included. This leads the Lagrange equation without constraints to be:

$$\mathcal{L}(\mathbf{q}) = \frac{1}{2} \mathbf{q}^\top (\mathbf{N}_d + \mathbf{N}_s - \mathbf{V}) \mathbf{q} \quad (4.16)$$

The matrices  $\mathbf{N}_d$  and  $\mathbf{N}_s$  are symmetric because they can be written in the general form:

$$\mathbf{N} = \widehat{\mathbf{Q}}^{*\top} \mathbf{H} \widehat{\mathbf{Q}}^* \quad (4.17)$$

where  $\widehat{\mathbf{Q}}^*$  is a symmetric quaternion matrix given from equation (2.8), and  $\mathbf{H}$  represents a symmetric weight matrix. The weight matrices are given from equation (4.8), as  $\mathbf{D}$  and  $\mathbf{S}$ . From this result, it can be found that the matrix  $\mathbf{G}$  in the quadratic equation (4.9) is real and symmetric:

$$\mathbf{G} = \mathbf{N}_d + \mathbf{N}_s - \mathbf{V} \quad (4.18)$$

As each of the quaternions described throughout this thesis are unit quaternions, the constraint for the resulting cost function is still:

$$\mathbf{q}^\top \mathbf{q} = 1 \quad (4.19)$$

The entire resulting Lagrange equation is:

$$\mathcal{L}(\mathbf{q}) = \frac{1}{2} \mathbf{q}^\top \mathbf{G} \mathbf{q} - \frac{\lambda}{2} (\mathbf{q}^\top \mathbf{q} - 1) \quad (4.20)$$

Setting the derivative of this equation to zero, gives the eigenvalue problem for the function:

$$\frac{d\mathcal{L}}{d\mathbf{q}}(\mathbf{q}) = \mathbf{G} \mathbf{q} - \lambda \mathbf{I} \mathbf{q} = 0 \quad (4.21)$$

This leads to the standard eigenvalue equation:

$$\mathbf{G} \mathbf{q} = \lambda \mathbf{I} \mathbf{q} \quad (4.22)$$

The equation can be solved by using eigenvector mathematics [17]. The optimal unit quaternion which minimizes the function in (4.20), is the eigenvector corresponding to the most positive eigenvalue of the matrix  $\mathbf{G}$ , as seen in [4], [17] and [38]. This eigenvector will maximize the directional match between the quaternions.

The product of two unit quaternions is still a unit quaternion, which means that the quadratic term  $\frac{1}{2} \mathbf{q}^\top \mathbf{G} \mathbf{q}$  will be smaller than the highest eigenvalue of  $\mathbf{G}$  [39].

The attitude quaternions which produce stationary values of equation (4.20) are the eigenvectors of  $\mathbf{G}$ . Substitution of equation (4.22) into equation (4.16), results in:

$$\mathcal{L}(\mathbf{q}) = \frac{1}{2} \mathbf{q}^\top \mathbf{G} \mathbf{q} = \frac{1}{2} \mathbf{q}^\top \cdot [\lambda \quad \mathbf{q}] = \lambda \quad (4.23)$$

Since the quaternions consist of four elements,  $q_0, q_1, q_2$  and  $q_3$ , the stationary values in decreasing order will be:

$$\begin{aligned} \mathcal{L}(q_0) &= \lambda_0 \\ \mathcal{L}(q_1) &= \lambda_1 \\ \mathcal{L}(q_2) &= \lambda_2 \\ \mathcal{L}(q_3) &= \lambda_3 \end{aligned} \quad (4.24)$$

Another way of describing the solution of the minimization problem, is to see it as a sequential solution method for the cost function in (4.8). At each iteration all the quaternions are fixed, except one. This quaternion is determined in order to minimize the equation. The cost function is upper bounded by the sum of the largest eigenvalues of the fixed matrix  $\mathbf{G}$ , which ensures convergence towards a local minimum [40].

The matrix  $\mathbf{G}$  is a  $4 \times 4$  symmetric matrix, and will sometimes be full, which makes solving the eigenvalue problem computationally expensive. Since the matrix is symmetric, it can be decomposed to:

$$\mathbf{G} = \mathbf{C} \begin{bmatrix} -\lambda_0 & 0 & 0 & 0 \\ 0 & -\lambda_1 & 0 & 0 \\ 0 & 0 & -\lambda_2 & 0 \\ 0 & 0 & 0 & -\lambda_3 \end{bmatrix} \mathbf{C}^\top \quad (4.25)$$

where  $\mathbf{C}$  is an orthogonal eigenvector matrix. The elements on the diagonal are the eigenvalues of  $\mathbf{G}$ , and they will have corresponding eigenvectors  $\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2$  and  $\mathbf{e}_3$ . The eigenvectors span the four-dimensional space, and an arbitrary quaternion can be written as:

$$\mathbf{q} := a_0 \mathbf{e}_0 + a_1 \mathbf{e}_1 + a_2 \mathbf{e}_2 + a_3 \mathbf{e}_3 \quad (4.26)$$

Together with the constraint in (4.19), the following equation is obtained:

$$\mathbf{q}^\top \mathbf{q} = a_0^2 + a_1^2 + a_2^2 + a_3^2 = 1 \quad (4.27)$$

This results in:

$$\frac{1}{2} \mathbf{q}^\top \mathbf{G} \mathbf{q} = \frac{1}{2} (a_0^2 \lambda_0 + a_1^2 \lambda_1 + a_2^2 \lambda_2 + a_3^2 \lambda_3) \quad (4.28)$$

Note that the eigenvalues are arranged in the same way as for equation (4.24):

$$\lambda_0 \geq \lambda_1 \geq \lambda_2 \geq \lambda_3 \quad (4.29)$$

It can be shown that

$$\frac{(\lambda_0 + \lambda_1 + \lambda_2 + \lambda_3)}{\|(\lambda_0 + \lambda_1 + \lambda_2 + \lambda_3)\|} = 1 \quad (4.30)$$

The optimal value is obtained, by choosing  $a_0 = 1$ , and the rest zero. This corresponds to choosing  $\mathbf{q} = \mathbf{e}_0$ .

The computations for finding the right eigenvector can be simplified by using singular value decomposition, as seen in [38], or using the matrix of cofactors, as seen in [17]. In the implementation of the developed EQUEST method, Cholesky factorization is used as this is the standard eigenvalue solver used in Matlab. This method is proved to be stable according to section 2.5.

The matrix  $\mathbf{G}$  has been shown in equation (4.18) to be a symmetric matrix, and the identity matrix  $\mathbf{I}$  is of course both symmetric and positive definite. Solving the optimization problem is achieved by finding the largest eigenvalue and the corresponding eigenvector of  $\mathbf{G}$  for each iterate of the algorithm. It should be noted that the  $\mathbf{G}$  matrix, needs only to be symmetric, not positive definite when using the algorithm.

In order to understand how the eigenvalues and eigenvectors are computed, the algorithm for the Cholesky factorization for the minimization problem is described by the following equations. The eigenvalue problem is repeated again, since it is the starting point of the factorization:

$$\mathbf{G} \mathbf{q} = \lambda \mathbf{I} \mathbf{q} \quad (4.31)$$

The problem can be solved by using a factorization of the symmetric and positive definite identity matrix  $\mathbf{I}$ :

$$\mathbf{I} = \mathbf{L}\mathbf{L}^\top \quad (4.32)$$

where  $\mathbf{L}$  is a lower triangular matrix, as described in chapter 2.5. The eigenvectors of the equation are those of

$$\mathbf{H}_c \mathbf{e} = \lambda \mathbf{e} \quad (4.33)$$

where

$$\mathbf{H}_c = \mathbf{L}^{-1} \mathbf{G} \mathbf{L}^\top \quad (4.34)$$

and

$$\mathbf{e} = \mathbf{L}^\top \mathbf{q} \quad (4.35)$$

The eigenvectors of the original problem in (4.31), can then be retrieved by using

$$\mathbf{q} = \mathbf{L}^{-\top} \mathbf{e} \quad (4.36)$$

Since Cholesky factorization is used, the computations of eigenvalues and eigenvectors are efficient and stable. The requirement that  $\mathbf{L}$  from equation (4.32) should have positive diagonal entries, can be extended to include positive semi-definite matrices. The factorization will then not be unique in general.



## 4.4 Developed EQUEST convergence properties

The convergence properties of the developed EQUEST method are presented in this chapter. Local convergence is obtained, but a global result cannot be expected for all possible attitude determination scenarios.

In order to investigate the convergence properties, the trace of the matrices  $\mathbf{G}$  and  $\mathbf{V}$  is used. The trace of a matrix sums the elements on the main diagonal of the matrix, and it can also be used to find the sum of all the eigenvalues of the matrix. The characteristic equation  $\det(\mathbf{G} - \lambda\mathbf{I} = 0)$  can be found by using the trace of the matrix  $\mathbf{G}$  as:

$$\det(\mathbf{G} - \lambda\mathbf{I}_{3 \times 3}) = -\lambda^3 + \lambda^2 \text{Trace}(\mathbf{G}) + \lambda \frac{1}{2} (\text{Trace}(\mathbf{G}^2) - \text{Trace}^2(\mathbf{G})) + \det(\mathbf{G}) \quad (4.37)$$

According to [35], it can be shown from equations (4.10), (4.12) and (4.15), that the trace of  $\mathbf{V}$  is zero. The trace of the matrix  $\mathbf{G}$  is however not zero, because the matrices  $\mathbf{N}_d$  and  $\mathbf{N}_s$  have positive diagonal entries. This leads to the following equation:

$$\text{Trace}(\mathbf{G}) = \text{Trace}(\mathbf{N}_d + \mathbf{N}_s) = n_0 + n_1 + n_2 + n_3 \quad (4.38)$$

where the diagonal elements from the two matrices  $\mathbf{N}_d$  and  $\mathbf{N}_s$  are combined to get the elements  $n_0, n_1, n_2$  and  $n_3$ .

It can be shown that the eigenvalues of the matrix  $\mathbf{V}$  consist of combinations of three elements  $s_0 \geq s_1 \geq s_2 \geq 0$ :

$$\lambda_{\mathbf{V}_0} = s_0 + s_1 + s_2 \quad (4.39)$$

$$\lambda_{\mathbf{V}_1} = s_0 - s_1 - s_2 \quad (4.40)$$

$$\lambda_{\mathbf{V}_2} = -s_0 + s_1 - s_2 \quad (4.41)$$

$$\lambda_{\mathbf{V}_3} = -s_0 - s_1 + s_2 \quad (4.42)$$

where  $s_k$  for  $k = 0, 1, 2$  are given from  $s := [s_0 \ s_1 \ s_2]^\top = \sqrt{\text{eig}(\mathbf{L}_s^\top \mathbf{L}_s)}$ , where  $\mathbf{L}_s$  is defined in equation (4.13). Since the trace of  $\mathbf{V}$  is zero, the four eigenvalues must sum to zero:

$$\lambda_{\mathbf{V}_0} + \lambda_{\mathbf{V}_1} + \lambda_{\mathbf{V}_2} + \lambda_{\mathbf{V}_3} = 0 \quad (4.43)$$

The introduction of the two symmetric matrices  $\mathbf{N}_d$  and  $\mathbf{N}_s$ , leads to shifts in these eigenvalues. The lowest eigenvalue of this function becomes the largest eigenvalue of the developed EQUEST method cost function from equation (4.20). Each of the eigenvalues of  $\mathbf{G}$ , denoted by  $[\lambda_0, \dots, \lambda_3]$  are given from:

$$\lambda_k = n_k - \lambda_{\mathbf{V}_k} \quad (4.44)$$

for  $k = 0, 1, 2, 3$ .

The sum of the eigenvectors for the  $\mathbf{G}$  matrix is:

$$(n_1 0 - \lambda_{\mathbf{V}_0}) + (n_1 - \lambda_{\mathbf{V}_1}) + (n_2 - \lambda_{\mathbf{V}_2}) + (n_3 - \lambda_{\mathbf{V}_3}) = 0 \quad (4.45)$$

Note from equation (4.24) that the eigenvalues still have the property:  $\lambda_0 \geq \lambda_1 \geq \lambda_2 \geq \lambda_3$ . All the eigenvalues of the matrix  $\mathbf{G}$  cannot be zero, which leads  $\lambda_0 > 0$  and  $\lambda_3 < 0$ . The condition  $|\lambda_0| > |\lambda_k|$  for  $k \in 0, 1, 2, 3$  will be encountered for all situations except for the case where  $|\lambda_0| = |\lambda_3|$ .

Global convergence is only obtained if  $|\lambda_0| > \{|\lambda_1|, |\lambda_2|, |\lambda_3|\}$  as seen in [35]. By assuming diagonal positive elements of the matrices  $\mathbf{N}_s$  and  $\mathbf{N}_d$ , it can be seen from equations (4.39)-(4.42) that the absolute values of  $\lambda_0$  and  $\lambda_3$  are equal if only one reference vector of the developed EQUEST method exists, or if the two reference vectors are parallel. This means that the developed EQUEST method has global convergence in almost all cases, but cannot guarantee it. This is one of the reasons why the nonlinear observer is considered as an alternative estimation method for the CubeSat.

## 4.5 Nonlinear observer

A nonlinear observer has been implemented, and compared to the developed EQUEST method. The simulation results are presented in chapter 6.3.

The observer, described as an explicit complementary filter with stationary reference vectors was first presented by Mahony et al. in 2008 [41]. It was proven that this observer has global stability if the reference vectors are stationary, or if the gyroscope measurements are unbiased. An extension of the method was introduced by Grip et al. in 2012 [42]. Global exponential stability was proven even with the inclusion of time-varying reference vectors and gyro bias, provided an uncoupled vector estimation. Apart from the excellent stability properties, the nonlinear observer has the advantage that it is less sensitive to disturbances than the developed EQUEST method.

The observer equations are presented below with the same notation as in previous chapters, with  $q_0$  and  $\mathbf{q}_{vec} := [q_1 \ q_2 \ q_3]^\top$  as the scalar and vector parts of the quaternion. As described in equation (2.23),  $\mathbf{S}$  represents a skew symmetric matrix. For the nonlinear observer equations, the estimated parameters will be written with roof accents.

The estimated rotation matrix is given from equation (2.30):

$$\hat{\mathbf{R}}_n^b(\mathbf{q}) := \mathbf{I}_{3 \times 3} + 2q_0\mathbf{S}(\mathbf{q}_{vec}) + 2\mathbf{S}(\mathbf{q}_{vec})^2 \quad (4.46)$$

The observer equations are:

$$\dot{\hat{\mathbf{R}}}_n^b(\mathbf{q}) = \hat{\mathbf{R}}_n^b(\mathbf{q})\mathbf{S}(\boldsymbol{\omega}_m - \hat{\mathbf{b}}_g + \boldsymbol{\omega}_{inj}) \quad (4.47)$$

$$\dot{\hat{\mathbf{b}}}_g = \text{Proj}(\hat{\mathbf{b}}_g, -k_I\boldsymbol{\omega}_{inj}) \quad (4.48)$$

where  $\boldsymbol{\omega}_m$  is the measured gyroscope data and  $\hat{\mathbf{b}}_g$  is the estimated gyroscope bias. The parameter  $k_I$  is an injection gain to be designed. The implemented value for  $k_I$  is 0,1 for all the test scenarios in chapter 6.

The injection term,  $\boldsymbol{\omega}_{inj}$ , is given by:

$$\boldsymbol{\omega}_{inj} = -\text{vex}\left(\sum_{j=1}^n \frac{k_j}{2} (\mathbf{b}_j(\hat{\mathbf{b}}_j)^\top - \hat{\mathbf{b}}_j^\top(\mathbf{b}_j)^\top)\right) \quad (4.49)$$

where  $k_j \geq k_P > 0, j = 1, \dots, n$  are observer gains. The gains should be small, typically between 0 and 1. The observer is implemented and tested both with constant and varying values for the gains,  $k_1$  and  $k_2$  as described in chapter 6.6. The injection term uses the estimates of the sensor vector measurements in the BODY frame, which are given as:

$$\hat{\mathbf{b}}_1 = \hat{\mathbf{R}}_n^b(\mathbf{q})^\top \mathbf{r}_1 \quad (4.50)$$

$$\hat{\mathbf{b}}_2 = \hat{\mathbf{R}}_n^b(\mathbf{q})^\top \mathbf{r}_2 \quad (4.51)$$

The vectors  $\mathbf{r}_1$  and  $\mathbf{r}_2$  are the sensor data reference vectors in the NED frame. The function  $\text{vex}$  is defined as the inverse of the cross product. This can be illustrated by taking  $\text{vex}$  of a skew symmetric matrix, which gives:

$$\text{vex}(\mathbf{S}(\mathbf{n})) = \text{vex} \left( \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix} \right) = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} \quad (4.52)$$

One of the main improvements made by Grip et al. [42], is the improvement of the bias estimation. A parameter projection algorithm is introduced, which ensures semi-global exponential stability with time-varying reference vectors. Combination with an uncoupled vector estimation give a globally exponentially stable result. The projection algorithm was first described by Krstić et al. in 1995. It ensures that the gyroscope bias remains within a compact, convex set and is written as [43]:

$$\begin{aligned} \dot{\hat{\mathbf{b}}}_g &= \text{Proj} \left\{ \hat{\mathbf{b}}_g, \Psi \right\} \\ &= \left\{ \Psi \text{ if } \|\hat{\mathbf{b}}_g\| < w_p \text{ or } (\|\hat{\mathbf{b}}_g\| = w_p \text{ and } \nabla \mathcal{P}(\hat{\mathbf{b}}_g))\Phi \leq 0) \right. \\ &\quad \left. \left\{ \Psi - \frac{\Psi \gamma \nabla \mathcal{P}(\hat{\mathbf{b}}_g) \nabla \mathcal{P}(\hat{\mathbf{b}}_g)^\top}{\|\nabla \mathcal{P}(\hat{\mathbf{b}}_g)\|_\gamma^2} \text{ otherwise} \right. \right\} \quad (4.53) \end{aligned}$$

In the algorithm,  $\mathcal{P}(\hat{\mathbf{b}}_g) = \hat{\mathbf{b}}_g^\top \hat{\mathbf{b}}_g - w_p \leq 0$ , and  $\nabla \mathcal{P}(\hat{\mathbf{b}}_g)$  is the gradient of  $\mathcal{P}(\hat{\mathbf{b}}_g)$ . The constant  $w_p$  is chosen such that  $\|\hat{\mathbf{b}}_g\| \leq w_p$ . The constant matrix  $\gamma$  is symmetric and positive definite. The parameter  $\Psi$  is given from  $\Psi = -k_I \boldsymbol{\omega}_{inj}$ .

Implemented with the use of quaternions instead of rotation matrices, the observer becomes:

$$\dot{\hat{\mathbf{q}}} = \mathbf{T}(\hat{\mathbf{q}})(\boldsymbol{\omega}_m - \hat{\mathbf{b}}_g + \boldsymbol{\omega}_{inj}) \quad (4.54)$$

$$\dot{\hat{\mathbf{b}}}_g = \text{Proj}(\hat{\mathbf{b}}_g, -k_I \boldsymbol{\omega}_{inj}) \quad (4.55)$$

where

$$\mathbf{T}(\hat{\mathbf{q}}) = \frac{1}{2} \begin{bmatrix} -\hat{\mathbf{q}}_{vec}^\top \\ \hat{\mathbf{q}}_0 \mathbf{I}_{3 \times 3} + \mathbf{S}(\hat{\mathbf{q}}_{vec}) \end{bmatrix} \quad (4.56)$$

The injection term is still given from equation (4.49), and the projection algorithm is described in equation (4.53).

For the implementation, the observer equation must be discretized. Euler discretization with time-step  $h$  gives:

$$\hat{\mathbf{q}}(k+1) = \hat{\mathbf{q}}(k) + h\mathbf{T}(\hat{\mathbf{q}}(k)) \left[ \boldsymbol{\omega}_m(k) - \hat{\mathbf{b}}_g(k) + \boldsymbol{\omega}_{inj}(k) \right] \quad (4.57)$$

$$\hat{\mathbf{b}}_g(k+1) = \text{Proj}(\hat{\mathbf{b}}_g(k), -k_I \boldsymbol{\omega}_{inj}(k)) \quad (4.58)$$

The observer can also be implemented on an EKF-like corrector-predictor form, according to [44]. This way, the observer can handle different measurement sampling rates and dead-reckoning. Calibration of the magnetic field for the magnetometer will be less accurate for high frequencies. With a corrector-predictor implementation, a lower frequency for the sensor data from the magnetometer can be obtained without influencing the general sampling time. By using the nonlinear observer equations in (4.54) and (4.55), the same effect could be achieved by for instance setting the gain for the magnetometer in the injection term,  $k_2$ , to have a low value for every ten or twenty time-steps. After discretization, the corrector-predictor equations for the nonlinear observer are:

Predictor equations:

$$\hat{\mathbf{q}}(k) = \bar{\mathbf{q}}(k) + h\mathbf{T}(\bar{\mathbf{q}}(k))\boldsymbol{\omega}_{inj} \quad (4.59)$$

$$\hat{\mathbf{b}}_g(k) = \bar{\mathbf{b}}_g(k) - h\boldsymbol{\omega}_{inj} \quad (4.60)$$

Corrector equations:

$$\bar{\mathbf{q}}(k+1) = \hat{\mathbf{q}}(k) + h\mathbf{T}(\hat{\mathbf{q}}) \left[ \boldsymbol{\omega}_m(k) - \hat{\mathbf{b}}_g(k) \right] \quad (4.61)$$

$$\bar{\mathbf{b}}_g(k+1) = \hat{\mathbf{b}}_g(k) \quad (4.62)$$

Bias in the sensor data vector measurements can be handled by using a separate observer, independent from the gyroscope bias estimation. In order to describe the vector bias estimation, a single vector measurement  $\mathbf{b}_j$ , which consists of the true value  $\mathbf{b}_{true}$  and the bias  $\mathbf{b}_{bias}$ . The vector measurement is in the BODY frame, and it is not longer required that it has unit length.

The difference between the squared norms of the reference vectors and the corresponding measurement vectors can be written:

$$\mathbf{d} = \|\mathbf{r}_j\|^2 - \|\mathbf{b}_j\|^2 \quad (4.63)$$

Using the property  $\|\mathbf{r}_j\| = \left\| \mathbf{R}_n^b \mathbf{b}_j \right\| = \|\mathbf{b}_j\|$ , the difference is:

$$\mathbf{d} = \|\mathbf{r}_j\|^2 - \|\mathbf{b}_{true} + \mathbf{b}_{bias}\|^2 = \|\mathbf{b}_{bias}\|^2 - 2\mathbf{b}_j^\top \mathbf{b}_{bias} \quad (4.64)$$

A constant vector is defined as  $\boldsymbol{\rho} = \left[ \|\mathbf{b}_{bias}\|^2 \quad \mathbf{b}_{bias}^\top \right]^\top$  and a time-varying vector is given from  $\boldsymbol{\delta} = \left[ 1 \quad -2\mathbf{b}_j^\top \right]^\top$ . The difference can now be written as

$$\mathbf{d} = \boldsymbol{\delta}^\top \boldsymbol{\rho} \quad (4.65)$$

The observer equation is:

$$\dot{\hat{\boldsymbol{\rho}}} = \boldsymbol{\Gamma} \boldsymbol{\delta} (\mathbf{d} - \boldsymbol{\delta}^\top \hat{\boldsymbol{\rho}}) \quad (4.66)$$

where a symmetric positive definite matrix is denoted as  $\Gamma$ .

With this extra observer for the vector measurement bias estimation, it can be proved that the nonlinear observer is globally exponentially stable. This result is much stronger than for the developed EQUEST method, which cannot guarantee global convergence for all cases.

## 4.6 Stability properties for the nonlinear observer

Stability means how much the algorithm will be affected if small numerical errors are introduced and accumulated. Numerical stability is the central criterion for judging the usefulness of implementing an algorithm on a computer with roundoff.

The nonlinear observer with time-varying reference vectors and gyroscope bias estimation is globally exponentially stable, provided a parameter projection and a decoupled vector bias estimation is used for the implementation. The whole proof is given in [42]. A simplified proof is given in this chapter. For this, a Lyapunov function based on the error dynamics for the system is presented. The background theory for the Lyapunov analysis is given in chapter 2.6. The error dynamics of the nonlinear observer is described, using the same notation as in chapter (4.5):

$$\dot{\tilde{q}}_0 = \frac{1}{2} \mathbf{q}_{vec}^\top \mathbf{R}_n^b (\tilde{\mathbf{b}}_g + \boldsymbol{\omega}_{inj}) \quad (4.67)$$

$$\dot{\tilde{\mathbf{q}}}_{vec} = -\frac{1}{2} (\tilde{q}_0 \mathbf{I}_{3 \times 3} - \mathbf{S}(\tilde{\mathbf{q}}_{vec})) \mathbf{R}_n^b (\tilde{\mathbf{b}}_g + \boldsymbol{\omega}_{inj}) \quad (4.68)$$

$$\dot{\tilde{\mathbf{b}}}_g = -\text{Proj}(\hat{\mathbf{b}}_g, -k_I \boldsymbol{\omega}_{inj}) \quad (4.69)$$

A parametrized set is introduced:  $\tilde{\mathcal{R}} := \left\{ \mathbf{R}_n^b(\tilde{q}_0, \tilde{\mathbf{q}}_{vec}) \mid |\tilde{q}_0| \geq \epsilon \right\}$  where  $0 < \epsilon < 1$ . A stability theorem are given for the equations as [42]:

*Theorem 1.*

For each  $0 < \epsilon < 1$ , there exist a constant  $\bar{k}_P > 0$  such that, for all  $k_P > \bar{k}_P$ , the equilibrium point of equations (4.67),(4.68) and (4.69) is exponentially stable with all the initial conditions such that  $\tilde{\mathbf{R}}_n^b(0) \in \tilde{\mathcal{R}}$  and  $\hat{\mathbf{b}}_g(0) \in \hat{\mathcal{B}}_g$  contained in the region of attraction.

*Proof of Theorem 1.*

Let  $M$  be the bound on the compact set  $\tilde{\mathcal{B}}_g := \left\{ \tilde{\mathbf{b}}_g \mid \mathbf{b}_g \in \mathcal{B}_g, \hat{\mathbf{b}}_g \in \hat{\mathcal{B}}_g \right\}$ . Since  $\hat{\mathbf{b}}_g(0)$  must be contained by  $\hat{\mathcal{B}}_g$ , and because of the properties of the projection term, it follows that for all  $t \geq 0$ ,  $\hat{\mathbf{b}}_g(t) \in \mathcal{B}_g$  and therefore  $\tilde{\mathbf{b}}_g(t) \in \tilde{\mathcal{B}}_g$ .

Next, a Lyapunov function is defined as:

$$V = \|\tilde{\mathbf{q}}_{vec}\|^2 = 1 - \tilde{q}_0^2 \quad (4.70)$$

The derivative of the function is:

$$\dot{V} \leq -\tilde{q}_0 \tilde{\mathbf{q}}_{vec}^\top \mathbf{R}_n^b \tilde{\mathbf{b}}_g - k_P c_{obs}^2 \tilde{\mathbf{q}}_{vec}^2 (1 - \tilde{q}_0^2) \leq M - k_P c_{obs}^2 \tilde{q}_0^2 (1 - \tilde{q}_0^2) \quad (4.71)$$

where  $c_{obs}$  is a constant which fulfil the requirement that for each  $t \geq 0$ , the inequality  $\|\mathbf{r}_j^i \times \mathbf{r}_k^i\| \geq c_{obs}$  holds for a pair of indices  $j, k \in 1, \dots, n$ . It can be seen according to [42], that  $|\tilde{q}_0| = \epsilon$  gives  $\dot{V} < 0$ , and the property  $|\tilde{q}_0| \geq \epsilon$  is used throughout the rest of the proof.

A new Lyapunov function is introduced:

$$W = V + 2l_w \tilde{q}_0 \tilde{\mathbf{q}}_{vec}^\top \mathbf{R}_n^b \tilde{\mathbf{b}}_g + \frac{l_w}{2k_I} \tilde{\mathbf{b}}_g^\top \tilde{\mathbf{b}}_g \quad (4.72)$$

It can be shown that there exists positive constants  $\alpha_1$  and  $\alpha_2$  such that  $\alpha_1 \left\| (\tilde{\mathbf{q}}_{vec}, \tilde{\mathbf{b}}_g) \right\|^2 \leq W \leq \alpha_2 \left\| (\tilde{\mathbf{q}}_{vec}, \tilde{\mathbf{b}}_g) \right\|^2$ . After simplifications using different properties and bounds given from [42], the derivative of  $W$  can be written as:

$$\dot{W} = -(k_P a_w - l_w M^2) \|\tilde{\mathbf{q}}_{vec}\|^2 - l_w \epsilon^2 \|\tilde{\mathbf{b}}_g\|^2 + (1 + 2l_w \bar{\omega}) \|\tilde{\mathbf{q}}_{vec}\| \|\tilde{\mathbf{b}}_g\| \quad (4.73)$$

where  $\bar{\omega}$  is the angular velocity without bias and  $l_w$  is defined such that  $0 < l_w < \min \{1/(2k_I), c_{obs}^2 \epsilon^2 / (12nM + 8k_I n)\}$ . The parameter  $a_w$  is defined as  $a_w = c_{obs}^2 \epsilon^2 - l_w (12nM + 8k_I n)$ .

It can be shown that the expression is negative definite, but it will still give a semiglobal result. This is because Theorem 1 specifies that the initial angle must be less than  $180^\circ$  by a margin, and in addition the initial bias estimate must be in the set  $\hat{\mathcal{B}}_g$ . Wrong attitude estimates can occur if the initial error angle is close to  $180^\circ$ . One way of handling this is periodically resetting of the estimated rotation matrix  $\hat{\mathbf{R}}_n^b$ . For the entire proof, see appendix B of Grip et al. [42].

For the vector bias estimation, which can be included in order to achieve global stability, a second theorem is introduced:

*Theorem 2.*

If two constants  $\epsilon_{bias} > 0$  and  $T_b > 0$  exists, such that, for each  $t \geq 0$ ,

$$\int_t^{t+T_b} \boldsymbol{\delta}(\tau) \boldsymbol{\delta}^\top(\tau) d\tau \geq \epsilon_{bias} \mathbf{I} \quad (4.74)$$

Then the origin is a globally exponentially equilibrium point for (4.66).

*Proof of Theorem 2.* Let  $N$  be the bound on  $\|\boldsymbol{\delta}\|$ . The Lyapunov function candidate is chosen as:

$$V_{bias} = \frac{1}{2} \tilde{\boldsymbol{\rho}}^\top \left( \boldsymbol{\Gamma}^{-1} - l_{bias} \int_t^\infty e^{-\tau} \boldsymbol{\delta}(\tau) \boldsymbol{\delta}(\tau)^\top d\tau \right) \tilde{\boldsymbol{\rho}} \quad (4.75)$$

where  $l_{bias} < \min \{ \lambda_{min}(\boldsymbol{\Gamma})^{-1} / N^2, 2e^{-T} \epsilon_{bias} / (N^6 \lambda_{max}^2(\boldsymbol{\Gamma}) + e^{-T} \epsilon_{bias}) \}$ . A constant positive definite matrix is still denoted by  $\boldsymbol{\Gamma}$ . The largest and smallest eigenvalues of the matrix is written as  $\lambda_{max}$  and  $\lambda_{min}$ . With the bounds for  $V_{bias}$  given from [42], the derivative of the Lyapunov function is:

$$\dot{V}_{bias} \leq -\left(1 - \frac{1}{2} l_{bias}\right) \left\| \boldsymbol{\delta}^\top \tilde{\boldsymbol{\rho}} \right\|^2 + l_{bias} N^3 \lambda_{max}(\boldsymbol{\Gamma}) \|\tilde{\boldsymbol{\rho}}\| \left\| \boldsymbol{\delta}^\top \tilde{\boldsymbol{\rho}} \right\| - \frac{1}{2} l_{bias} e^{-T} \epsilon_{bias} \|\tilde{\boldsymbol{\rho}}\|^2 \quad (4.76)$$



This function is negative definite and will give global exponential stability. The error dynamics of the attitude estimation, the gyroscope estimation and the vector bias estimation in cascade will cause the attitude and gyroscope bias error to be influenced by the vector bias error, but not the other way around. A regional stability result can be achieved by the cascade implementation, as the attitude estimate can be outside the region of attraction.

## 4.7 Normalization

It is not guaranteed or likely that the measured quaternions are of unit length. One option is to normalize the result, which is simple, but not ideal because errors might occur in the result due to the mathematical complexity. In the EQUEST and QUEST algorithms, the constraint  $\mathbf{q}^\top \mathbf{q} = 1$  is included in the algorithm, and the search space will always be the set of quaternions of unit length according to [6] and [18].

A round-off error in the matrix representation will cause drift from an orthogonal matrix. Renormalization of this matrix will be computationally expensive, because of the large number of variables. The Gram-Schmidt algorithm [38] might be used for "renormalization" of the matrix, but the quaternions should be normalized before turned into matrix representations, when possible. It might also be noted that another method for finding the shortest rotation between matrices called interpolation, will be tricky, due to the six constraints needed to enforce three degrees of freedom.

The nonlinear observer also requires normalization of the quaternions, except for the vector estimation bias, which compares the norm of the measurement vector to the norm of the reference vector. The norms of the two vectors will be equal if the measurement vector is bias-free. Apart for this special requirement, care must be taken in the implementation of all the estimation methods to ensure that the quaternions have unit length.

## 4.8 Quaternions to Euler angles

The plotted estimated attitude will be given in Euler angles partly because this representation was used by Jenssen and Yabar in [4]. The developed EQUEST method is implemented and compared to their simulations. In addition, Euler angles are more intuitive and easy to understand than the quaternion representation. For continuity in the plotting, the implemented nonlinear observer and the developed EQUEST method are also compared using Euler angles.

When the calculations in quaternion representation are converted into Euler angles, care must be taken to ensure singularities do not appear. In other words, the pitch angle cannot be  $\pm 90^\circ$  in the following equation [45]:

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan2((2q_0q_1 + 2q_2q_3), 1 - (2q_1^2 + 2q_2^2)) \\ \arcsin(2q_0q_2 - 2q_3q_1) \\ \arctan2((2q_0q_3 + 2q_1q_2), 1 - (2q_2^2 + 2q_3^2)) \end{bmatrix} \quad (4.77)$$

To convert matrices to quaternions, linear combinations of the entries in the matrix can be used to find the square of the quaternion components. In order to convert matrices to Euler angles, the inverse of trigonometric functions are required. Only cosine and sine of the angles can be computed directly [17].

In the code for the original and developed EQUEST methods, the optimal eigenvectors determine the next optimal attitude quaternions used in the algorithms, and the conversion from quaternions to Euler angles is made without use of the matrix representations. The output from the nonlinear observer also gives four-dimensional quaternions which can be converted directly into Euler angles.

# Chapter 5

## Matlab implementation

Four estimation methods are implemented in Matlab and compared to each other. The implemented methods are

- original EQUEST
- developed EQUEST
- extended Kalman filter
- Nonlinear observer

The code for the original EQUEST method and the EKF was originally written by Jenssen and Yabar [4]. The code has been developed further to include quaternion products instead of subtractions. New code for the calculation of the optimal lambda value and the corresponding eigenvector has been written. Only the Matlab implementation is tested in this thesis, but hardware implementation of the developed EQUEST method and the nonlinear observer will be made by one of the other master students involved in the NUTS project. Testing of the prototype will eventually result in a final choice of estimation method for the satellite.

Note that the scalar element of the quaternion in the code is  $q_4$ , not  $q_0$ , with the notation  $\mathbf{q} := [q_1 \ q_2 \ q_3 \ q_4]^T$ , instead of  $\mathbf{q} := [q_0 \ q_1 \ q_2 \ q_3]^T$  which is used in the rest of the report. Both the EKF and the original EQUEST method have been modified and made more efficient. They are tested and compared to the new developed EQUEST method and the nonlinear observer. The methods have been implemented in Matlab (version R2010a), where different input signals and changes for parameters can easily be simulated.

### 5.1 Sensor data

A new data set for the vector measurements have been implemented for the comparison of the developed EQUEST method and the nonlinear observer. This has enabled testing of different input responses, which makes it possible to review several scenarios without too much reprogramming. For test cases 1-7 and test case 9, the new data set have stationary reference vectors  $\mathbf{r}_1 = [0 \ 0 \ 9.81]^T$  and  $\mathbf{r}_2 = [13598.5 \ 444.7 \ -49854.8]^T$ , where

$\mathbf{r}_1$  is the sun sensor vector and  $\mathbf{r}_2$  is the magnetometer vector. For testing of the satellite at the Earth, an accelerometer can be used instead of a sun sensor, but this will eventually have to be replaced. The replacement will however be trivial [4]. Therefore, the sensor is referred to as a sun sensor in this thesis, but it is denoted as an accelerometer in the Matlab code. In test case 8 in chapter 6, the reference vectors are varying as described in chapter 3.4.1 and 3.4.2. The angular velocity input has been modeled both as a sine wave and as a step input, as described in test case 3 of chapter 6.

The attitude dynamics is given from [45]:

$$\dot{\mathbf{q}} = \mathbf{T}(\mathbf{q})(\boldsymbol{\omega}_m - \mathbf{b}_g) \quad (5.1)$$

$$\dot{\mathbf{b}}_g = 0 \quad (5.2)$$

The implementation require discretized equations. Euler discretization with a sample time of 0.01s is implemented. This gives the attitude dynamics:

$$\mathbf{q}(k+1) = \mathbf{q}(k) + h\mathbf{T}(\mathbf{q})(\boldsymbol{\omega}_m(k) - \mathbf{b}_g(k)) \quad (5.3)$$

$$\mathbf{b}_g(k+1) = 0 \quad (5.4)$$

## 5.2 Reference vector models

Models for the magnetometer reference vector and the sun sensor reference vector have been implemented and tested. The nonlinear observer can handle time-varying reference vectors and still have good stability properties. The implemented models are described in chapter 3.4.1 and 3.4.2.

The magnetometer reference model requires latitude and longitude information from the satellite. In order to acquire this, a simulation program called STK (Satellite Tool Kit) has been used. The program is fairly intuitive and allows for several parameter settings, like an arbitrary height and inclination of a satellite. The STK is an analysis software, where the different aspects of a satellite mission can be modeled. One of the modeling options is to make an orbit or trajectory for the satellite, with different input parameters and get data for the position, speed and altitude as output.

The latitude and longitude have been computed with a satellite of inclination  $90^\circ$  and height of 500 km. The satellite was simulated with start values of  $0^\circ$  for the latitude and  $-30^\circ$  for the longitude. It was simulated to circulate 1,5 times around the Earth. A Julian time scale, J2000 is used by the simulator, and the corresponding UTC time for the simulation was May 22 from 10:00-12:00.

Screenshots from the STK program are shown in figures (5.1) and (5.2), where the green line is the simulated orbit, and the satellite is denoted by "NUTS". The graph in figure (5.3) shows the change in latitude (red graph), longitude (green graph) and height (blue graph) during the simulation. The orbit time is seen to be approximately 90 min, which correspond to the assumed orbital time for the NUTS CubeSat.

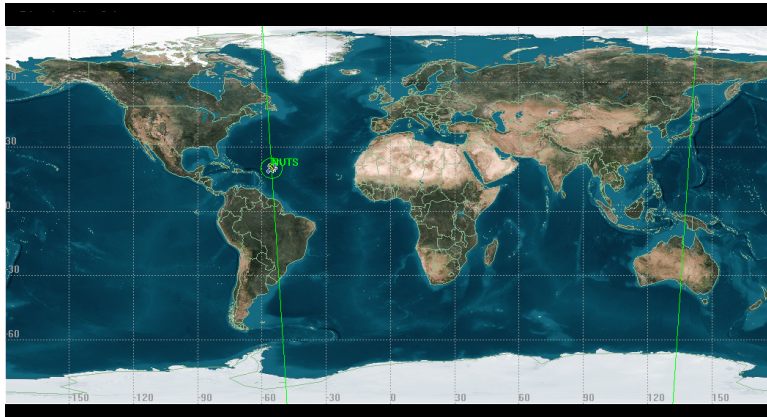


Figure 5.1: Test orbit for the NUTS satellite.

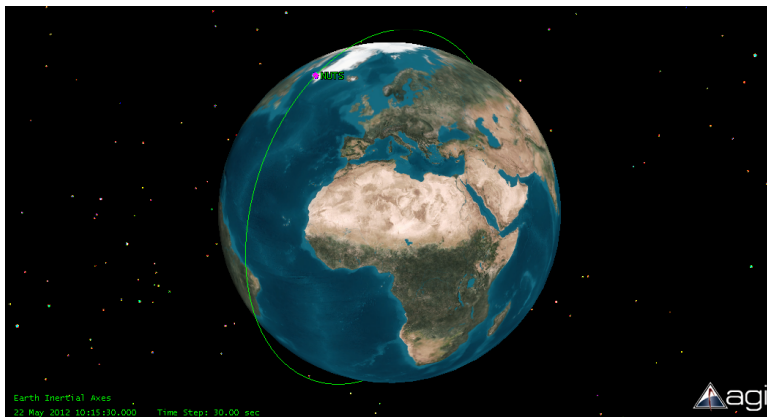


Figure 5.2: Test orbit for the NUTS satellite.

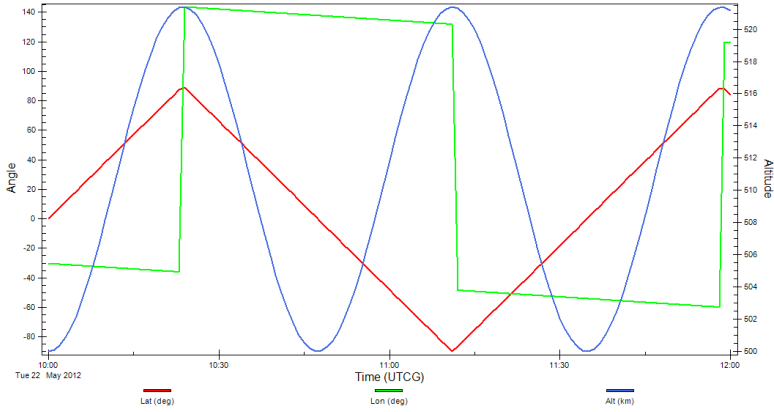


Figure 5.3: Graph for latitude, longitude and height.

### 5.3 Prototype code

The final implementation of the chosen estimation method will be written in C language, and implemented on a prototype with a microcontroller. The chosen microcontroller is an ATMEGA2561 running at 16MHz [4]. The sensor for the prototype is chosen as a CIMU IMU, and is powered through a USB connection to the computer. Another student involved in the satellite project will implement both the developed EQUEST method and the nonlinear observer for further real-time testing by the summer of 2012.

### 5.4 Extended Kalman filter

The nonlinear observer and both the original EQUEST method and the new developed EQUEST method are tested along with an extended Kalman filter. This filter is a normal Kalman filter extended to cope with nonlinear problems, through linearizations of the nonlinear terms for each iteration. The extended Kalman filter may diverge if the system model is inaccurate. This implementation was originally made by Jenssen and Yabar [4]. The implemented EKF uses a zero-order hold discretization with a time constant of  $T_s = 0.0011$ s. The Kalman filter equations are given for the state vector defined as

$$\mathbf{x} := [\mathbf{q} \quad \mathbf{b}_{bias_1} \quad \mathbf{b}_{bias_2}]^T \quad (5.5)$$

where  $\mathbf{b}_{bias_1}$  is the bias for the sun sensor and  $\mathbf{b}_{bias_2}$  is the bias for the magnetometer. Both are assumed constant. After discretization, the new state vector is given from

$$\mathbf{x}_{k+1} = \begin{bmatrix} e^{\left( \frac{1}{2} \begin{bmatrix} \mathbf{S}(\boldsymbol{\omega}) & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix} T_s \right)} & 0 & 0 \\ 0 & \mathbf{I}_{3 \times 3} & 0 \\ 0 & 0 & \mathbf{I}_{3 \times 3} \end{bmatrix} \mathbf{x}_k \quad (5.6)$$

where the parameter  $\omega$  in this equation is the angular velocity without bias. The measurement model which is linearized as described in appendix D of [4] is

$$\mathbf{z} = \begin{bmatrix} \mathbf{R}_n^b(\mathbf{q}) & 0 \\ 0 & \mathbf{R}_n^b(\mathbf{q}) \end{bmatrix} \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_{bias_1} \\ \mathbf{b}_{bias_2} \end{bmatrix} \quad (5.7)$$

The prediction equations for the EKF are:

$$\hat{\mathbf{x}}_k^- = f(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}) \quad (5.8)$$

$$\mathbf{P}_k^- = \mathbf{F}_k \mathbf{P}_k \mathbf{F}_k^\top + \mathbf{Q}_{kal} \quad (5.9)$$

The update equations are:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^\top (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^\top + \mathbf{R}_{kal})^{-1} \quad (5.10)$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-) \quad (5.11)$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^- \quad (5.12)$$





# Chapter 6

## Testing

Test results for the implemented estimation methods are presented in this chapter. Each section describes a test case for evaluation of the performance of the developed EQUEST method and the nonlinear observer. The test cases are chosen to be realistic scenarios for the NUTS CubeSat. Test case 1 and 2 compares the new developed EQUEST method with quaternion products to the original implementation with quaternion subtractions.

Test case 3 investigates the general performance of the implemented nonlinear observer, while test case 4-8 compares the behavior for the developed EQUEST method and nonlinear observer for different scenarios. Loss of sensor data is simulated, in addition to implementation of several time-varying parameters and sensor measurement noise.

Test case 9 tests the general behavior of a combination of the developed EQUEST method and the nonlinear observer, in order to get estimations where the strengths from both methods are utilized. All the test cases are described in detail below, and the results for each test case are discussed in chapter 6.10.

In all the cases, the reference quaternions have the initial values  $\mathbf{q}_{ref} = [0 \ 0 \ 0 \ 1]^\top$  and the implemented methods have initial values of  $\mathbf{q}_{method} = [0 \ 0 \ 1 \ 0]^\top$ , except for in test case 4, where the implemented methods have the same initial value as the reference quaternions. The gyroscope estimated bias has the initial value  $\hat{\mathbf{b}}_g = [0 \ 0 \ 0]^\top$ .

No sensor vector measurement noise for the sun sensor and magnetometer is assumed, except for in test case 7. This will ensure global exponential stability for the nonlinear observer, but for further testing, a bias estimation for the sensor vector measurements should be included. Gyroscope bias is implemented and estimated for the nonlinear observer in test case 3-9. The gyroscope measurements in test case 1 and 2 are assumed to be ideal.

All the simulations are made for a geostationary satellite placed over Trondheim, except for in test case 8, where reference vector models are implemented. The implementation is described in chapter 5.1. For all other test cases, the sensor reference inputs will have constant values.

It should be noted that the original and developed EQUEST methods are implemented without attitude prediction for all scenarios. This makes the method more sensitive to disturbances. The original EQUEST method implementation was made by Jenssen and Yabar [4], and for easy comparison of the methods, the developed EQUEST method is also implemented without the attitude prediction term. The extended Kalman filter is included

for comparison to the other estimation methods in test case 1-2 and test case 7-9.

The different test cases and descriptions of implemented parameters and models are presented in the following pages. The sample time used for the simulations is 0.01s, which corresponds to a frequency of 100Hz. The sample time may change for the final implementation on the prototype, which can have a maximum frequency of 16MHz. Using such a high frequency when running the code is not ideal, because it may lead to instabilities. For previous testing of the original EQUEST method done by Jenssen and Yabar [4], a frequency of 8MHz was used. The implemented Matlab code can easily be manipulated for other frequencies and corresponding sample times.

For the plotting of the simulations, Euler angles are used. It can be argued that errors in the rotation directions can be hidden by using Euler angles, but continuity in the testing is obtained, since the original implementation uses this representation. Euler angles are also much more intuitive to understand than quaternions, and this makes it easier to understand the plotted simulation outputs.

The different test cases used for the simulations are presented below. The test results are evaluated in chapter 6.10.

- **Case 1: Developed EQUEST**

The developed EQUEST method is implemented with quaternion products replacing the quaternion subtractions in the cost function, and compared to the original EQUEST method. The result is mathematically correct, and the performance of the new modified method is tested in order to see if there are any changes in performance from the original method. Both EQUEST implementations are compared to an implemented extended Kalman filter (EKF). The code for the original EQUEST method and the EKF builds on the implementation done by Jenssen and Yabar [4]. The data set for the test case is raw data from a 9-DOF Razor IMU, which was connected to the computer using a USB connection. The data was stored in a txt.-file, which makes it possible to compare the new developed EQUEST method to the old original EQUEST method and the EKF implementation.

- **Case 2: Developed EQUEST with loss of sun sensor data**

The CubeSat will have a sun sensor or use the solar panels to find one of the vector measurements needed for the attitude estimation. Such measurements often consist of only one or two vector elements, because parts of the satellite (and the sun sensor) will be in shade. Therefore, the developed EQUEST method have been tested with loss of sun sensor data along each of the three directional axes. Like for test case 1, the data is given from a 9-DOF Razor IMU. The developed EQUEST method response is compared to the EKF response for loss of the sensor data. It is very unlikely that loss in magnetometer data will occur, unless the sensor is faulty. Such a scenario is not investigated in this thesis.

- **Case 3: Nonlinear observer**

The implemented nonlinear observer is compared to the developed EQUEST method for two different angular velocity outputs. A gyroscope will give real-time values for this measurement when the satellite is in orbit. In test case 1 and 2, the data from a 9-DOF Razor IMU was used for the simulations. In the following cases a new data set is used, as described in chapter 5.1. The new data set makes it easier to test how the implemented methods will react for different sensor vector measurements. Two gyroscope measurement outputs are tested for the nonlinear observer, a sine wave output, given from  $\omega = 0.1 \sin(00.1t)$ , and a step response output where the value changes from  $-\pi$  to  $\pi$ , 20 seconds into the simulation time. After the detumbling phase, the CubeSat will be rotating slowly or not at all, and a slow change in the sine input is therefore more realistic than a fast changing sine wave. Both gyroscope measurement outputs are simulated with a constant bias of 0,007 degrees. The response for the EKF is not tested in this test case (along with test case 4-6), because the main focus is the difference in performance between the nonlinear observer and the developed EQUEST method. Note that the nonlinear observer is implemented with a bias estimation for the gyroscope, but not a vector estimation for the other sensor measurements. The measurements from the magnetometer and sun sensor are assumed to be bias-free, which still makes the method globally exponentially stable. For the final implementation, all the sensor biases should be estimated.

- **Case 4: Nonlinear observer with loss of sun sensor data**

This is the same scenario as in test case 2, where simulations of loss in sun sensor data along each axis are presented. The behavior for the developed EQUEST method is compared to the behavior for the nonlinear observer. The results are computed with the new data set described in chapter 5.1. This makes the response for the developed EQUEST method different from the response in test case 2, because of other reference quaternions. The gyroscope measurement output is modeled as a sine wave with period 0.001s as described in test case 3. Note that the implemented methods and the reference quaternions have the same initial values in this test case, given from  $\mathbf{q}_{method} = [0 \ 0 \ 0 \ 1]^T$ . This will give less deviations from the reference quaternions for the start-up phase of the simulations. By having equal initial values, the developed EQUEST method and the nonlinear observer get the same starting point for the simulations, and the effect of the sensor data loss can easily be seen.

- **Case 5: Nonlinear observer bias**

The estimated bias for the nonlinear observer is presented for the gyroscope measurement outputs described in test case 3. The gyroscope bias is a known constant value, chosen to be  $\mathbf{b}_g = 0.007 \text{ deg/s} = 0.0001 \text{ rad/s}$ . The initial guess for the estimated bias is  $\hat{\mathbf{b}}_g = [0 \ 0 \ 0]^T \text{ rad/s}$ , and this value is updated through the projection term in the nonlinear observer equations from chapter 4.5. The final bias

value is plotted, in addition to the estimated bias response.

- **Case 6: Nonlinear observer with variable gains.**

In the injection term used for the nonlinear observer equations from chapter 4.5, the influence of the magnetometer data and the sun sensor data are weighted with the gains  $k_1$  and  $k_2$ . By having large initial values for the gains with decreasing time responses, the speed of the nonlinear observer can be improved for the start-up phase of the simulation time [42]. Since a larger gain will make the method more vulnerable to disturbances, the gains are made exponentially decreasing towards their stationary values. The gain for the sun sensor is given from  $k_1 = e^{5-t}$  with a final stationary value of 0.5, and the magnetometer gain is given from  $k_2 = e^{3-t}$  with the same stationary value of 0.5. The response of the nonlinear observer with the varying gains are compared to the response of the developed EQUEST method. The gyroscope measurement output is modeled as a sine wave, as described in test case 3.

- **Case 7: Nonlinear observer with disturbances.**

One of the most important test objectives is to investigate how sensitive and vulnerable the estimation methods are to disturbances. The sensors in the IMU are likely to be sensitive to noise, especially for high frequencies [42]. The nonlinear observer is compared both to the developed EQUEST method and the extended Kalman filter where white noise has been implemented for the sun sensor and magnetometer measurement data. The white noise is modeled as  $w = 0.1 \text{randn}(1)$  in the sensor data along the x-axes for both sensor measurements. The gyroscope measurement output was modeled as a sine wave described in test case 3. The gyroscope output is modeled without any measurement noise. It should be noted that the developed EQUEST method is implemented without a prediction term, which will make it less robust. A better performance can be expected with the method implemented in its entirety. The reason for modeling the measurement noise along only one axis, is to get a more fair test scenario for the developed EQUEST method. The nonlinear observer and the developed EQUEST method are compared to the EKF, because this is a well-known and thoroughly tested estimation method. By comparing the test results to this method, more information on the performance can be obtained than for comparison of only the two relatively new methods.

- **Case 8: Nonlinear observer with time-varying reference vectors.**

For all the other test cases, the reference vectors have been assumed constant for a geostationary satellite. In reality they will be time-varying because the satellite will orbit around the Earth, which means that the magnetic field and the reference model for the sun sensor will change along with the position of the satellite. The implemented reference vector models are described in chapter 3.4.1 and 3.4.2. In this test

case, the input parameters for the models are latitude and longitude values for an example orbit, described in chapter 5.2. In addition, a time input for each of the models is required. The magnetometer reference model needs the time in days since January 1 2000, and the sun sensor reference model needs the sidereal time since the Earth passed the vernal equinox. The test orbit for the time-varying reference vectors was done at May 22 2012 from 10:00-12:00, and the corresponding inputs for this specific time have been implemented in Matlab. For a longer simulation period, the time input will be varying, but for this test case, only varying latitude and longitude parameters influence the reference vectors. The nonlinear observer is compared to both the developed EQUEST method and the implemented EKF.

- **Case 9: Combination of EQUEST and nonlinear observer.**

The developed EQUEST method has the advantage of being fast, and will find the optimal attitude quaternion in one time step. However, it is less robust towards disturbances than the nonlinear observer. This is the reason for the implementation of a combination of the two methods. The developed EQUEST method provides an initial condition for the nonlinear observer. This way, a fast response at the start-up phase, and robustness against disturbances can be achieved. The gyroscope measurement output was modeled as a sine wave as described in test case 3. The combination method has also been tested with disturbances in the sensor measurements. The disturbances have been implemented as described in test case 7. In addition to this, white noise along all the axes (not just the x-axis) for the sensor measurement vectors have been tested. The combination method has been compared to the implemented extended Kalman filter.

## 6.1 Developed EQUEST

The extended Kalman filter code and the EQUEST code have been modified in order to be more efficient and computationally less expensive. As described in chapter 4, the original EQUEST method has been implemented with quaternion products instead of quaternion subtractions. Both EQUEST algorithms are implemented without the prediction terms, implicating a higher sensitivity to disturbances [4]. The results are shown in figure (6.1) and (6.2). The red graph shows the performance of the implemented EQUEST method, the blue graph shows the performance of the extended Kalman filter and the green graph shows the correct attitude from the sensor data. As seen in the figures, there is no difference in performance for the EQUEST method and the developed EQUEST method. The new implementation is however theoretically correct, with the use of quaternion products instead of subtractions. Similar plots can be found in [4], but in this thesis different initial values in the implementations are used, leading to different responses for the start-up phase in the simulations.

In order to investigate the difference between the two methods more closely, the deviations from the sensor data for the three estimation methods, EKF, original EQUEST and developed EQUEST are shown in figures (6.3)-(6.5). It can be seen that the largest difference between the EKF and the EQUEST methods is in the start-up phase, where the performance of the EQUEST methods has a larger deviation from the sensor data than the EKF has. No difference between the two implemented EQUEST methods can be spotted.

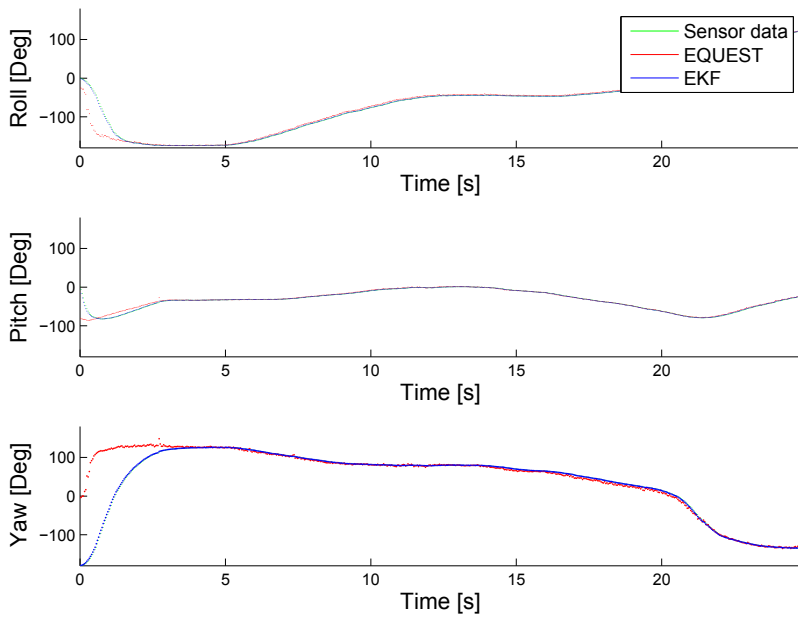


Figure 6.1: Attitude estimation using the original EQUEST method and the EKF.

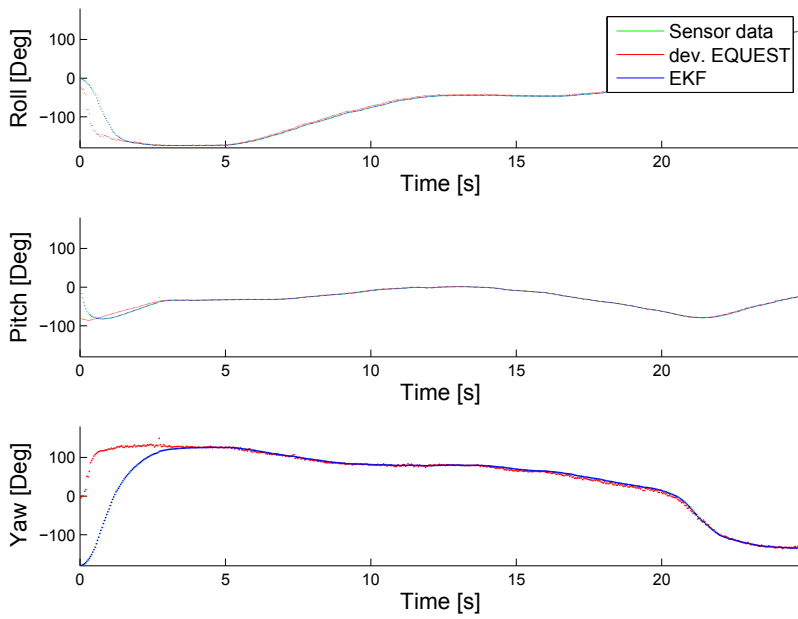


Figure 6.2: Attitude estimation using the developed EQUEST method and the EKF.



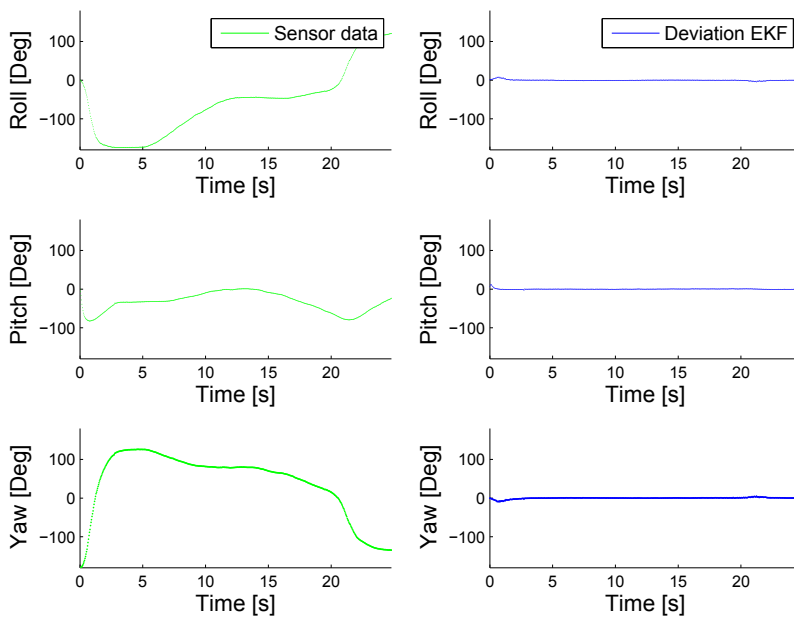


Figure 6.3: Sensor data and deviation for the extended Kalman filter.

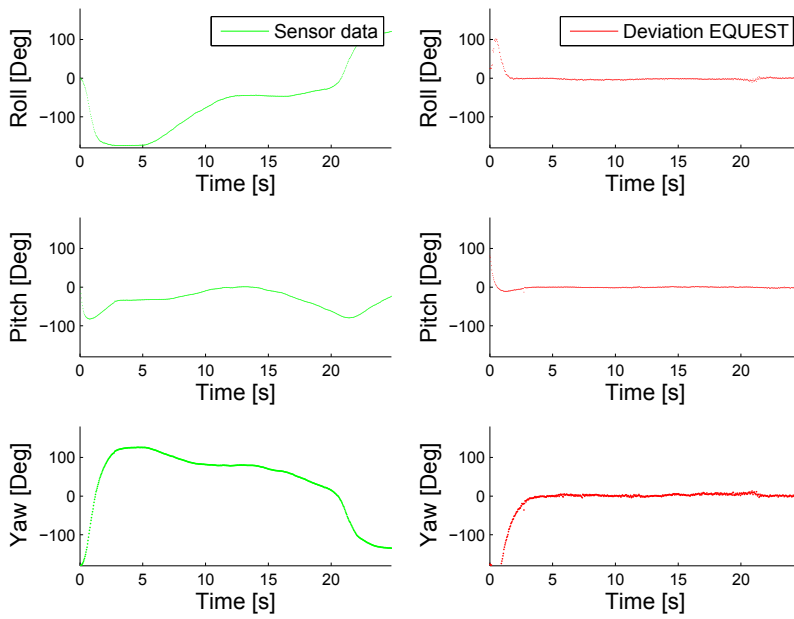


Figure 6.4: Sensor data and deviation for the original EQUEST method.

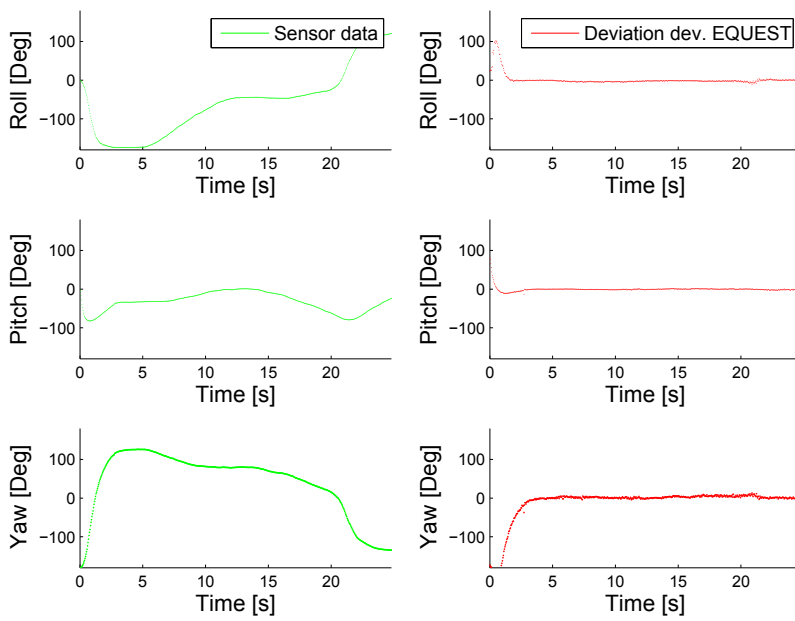


Figure 6.5: Sensor data and deviation for the developed EQUEST method.

## 6.2 Developed EQUEST with loss of sun sensor data

The developed EQUEST method and the EKF are tested with loss in the measurements from the sun sensor data. If the solar panels are used, no sensor data from the side where the panels are in shadow, can be used. This may cause loss in data from one, or several of the directional axes. The plots below, show the performance of the EKF (blue graph) and the developed EQUEST methods, (red graph) for loss of data along the x-, y- and z- axis. The sensor data is also plotted (green graph).

It can easily be seen that the EKF handles the data loss better than the developed EQUEST method. The deviation between the sensor data and the developed EQUEST method is plotted for each of the three axes in figure (6.8), (6.11) and (6.14). The deviation for the EKF is shown in figure (6.7), (6.10) and (6.13).

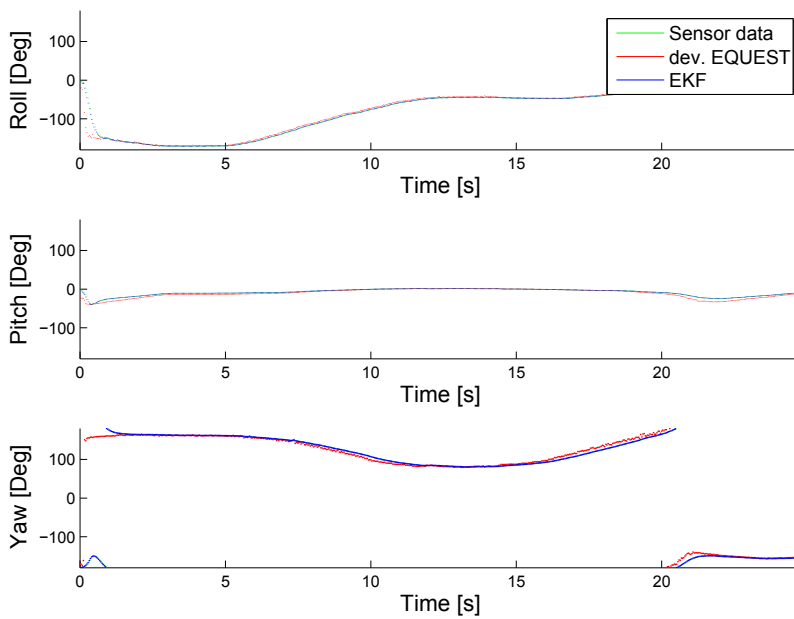


Figure 6.6: Attitude estimation using the developed EQUEST method and the EKF, with loss in sun sensor data along the x-axis.

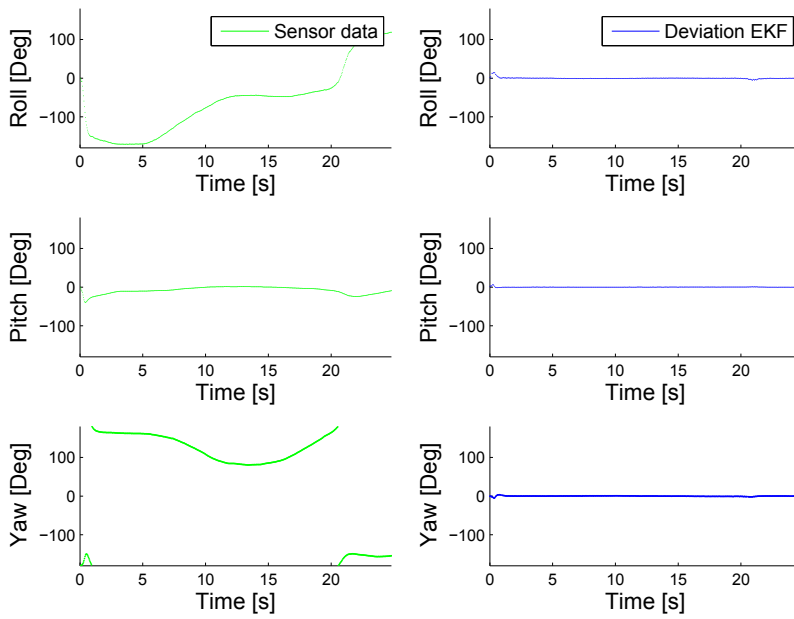


Figure 6.7: Sensor data and deviation for the EKF, with loss in sun sensor data along the x-axis.

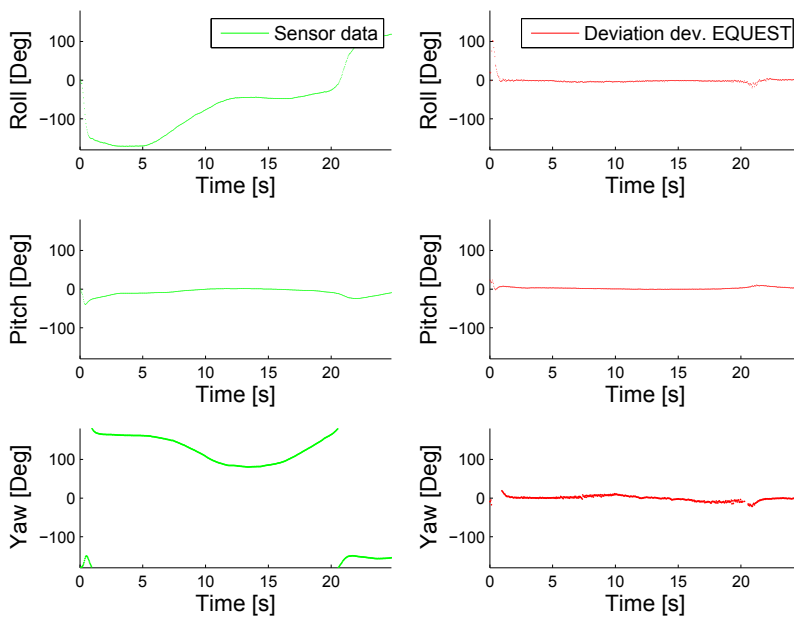


Figure 6.8: Sensor data and deviation for the developed EQUEST method, with loss in sun sensor data along the x-axis.

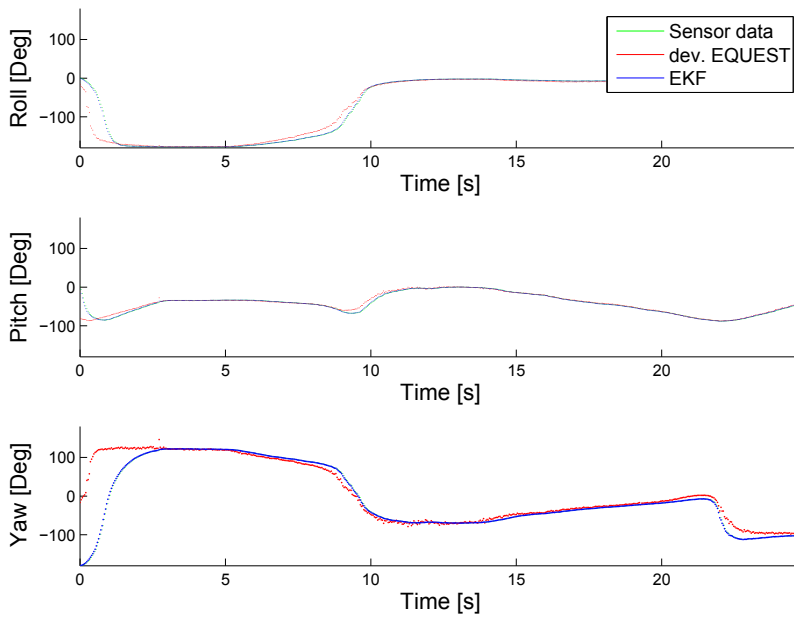


Figure 6.9: Attitude estimation using the developed EQUEST method and the EKF, with loss in sun sensor data along the y-axis.



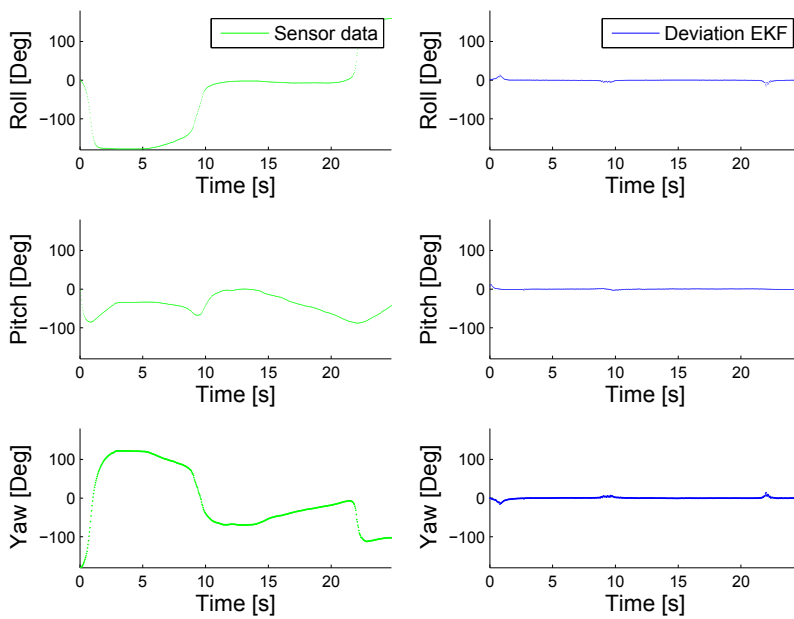


Figure 6.10: Sensor data and deviation for the EKF, with loss in sun sensor data along the y-axis.

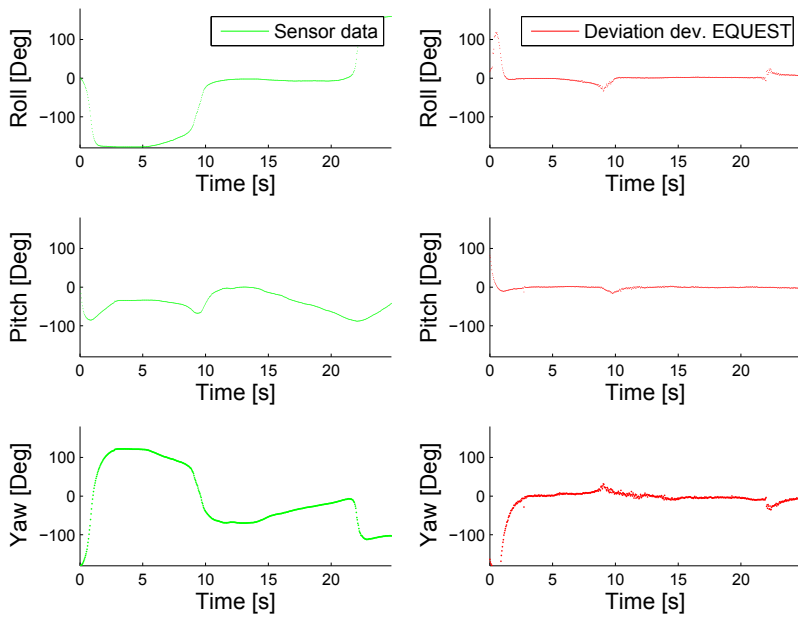


Figure 6.11: Sensor data and deviation for the developed EQUEST method, with loss in sun sensor data along the y-axis.

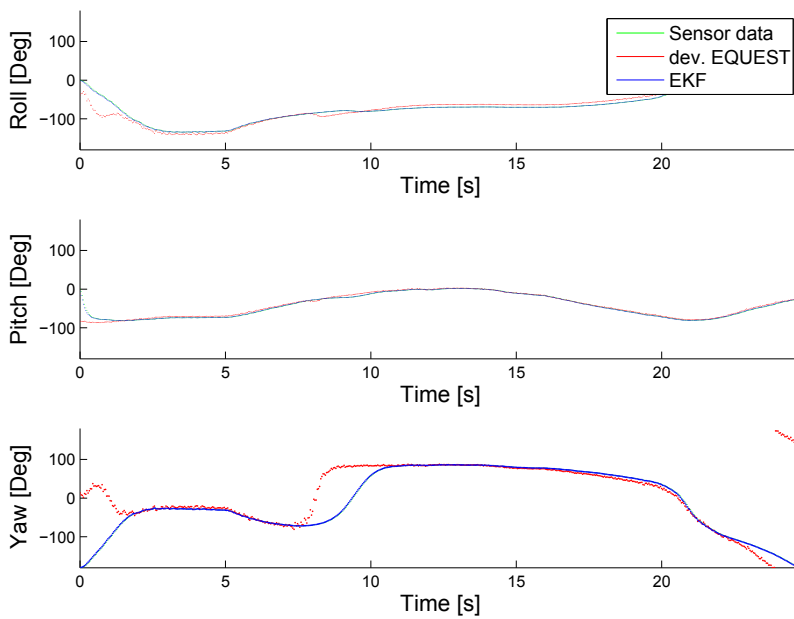


Figure 6.12: Attitude estimation using the developed EQUEST method and the EKF, with loss in sun sensor data along the z-axis.

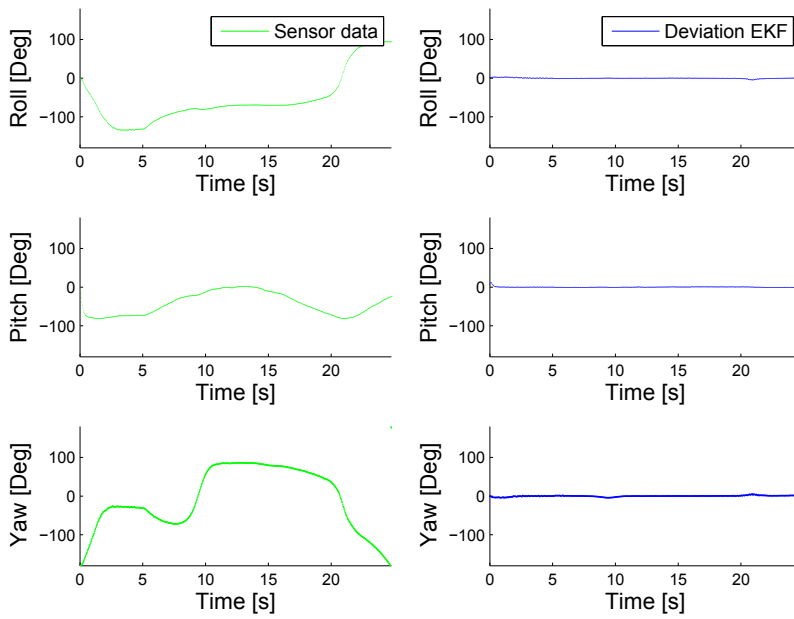


Figure 6.13: Sensor data and deviation for the EKF, with loss in sun sensor data along the z-axis.

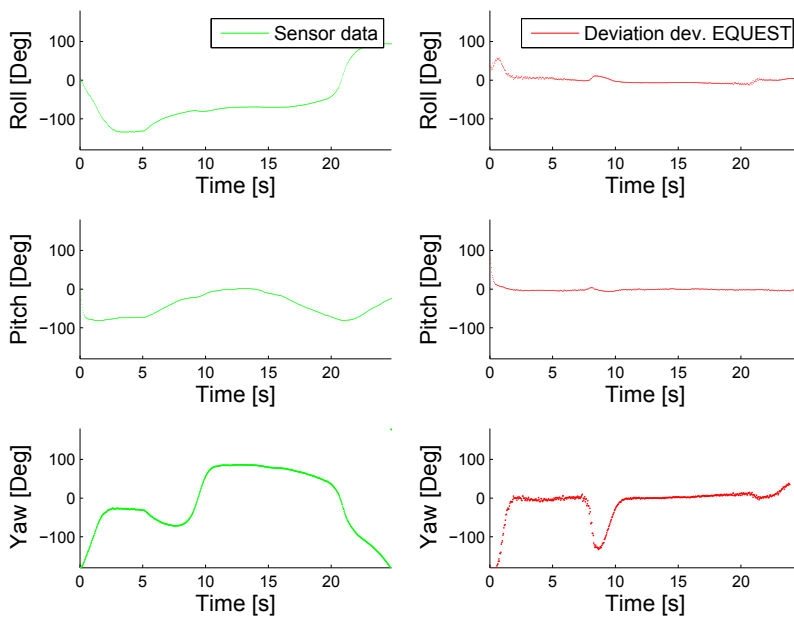


Figure 6.14: Sensor data and deviation for the developed EQUEST method, with loss in sun sensor data along the z-axis.

## 6.3 Nonlinear observer

The nonlinear observer from [41] and [42] is implemented in Matlab and compared to the performance of the developed EQUEST method. A new data set has been implemented in order to control the sample time and be able to test different kinds of satellite movements.

### 6.3.1 Sine wave

A sine wave with period 0.01s was implemented to simulate the gyroscope measurement vector, and the results are shown in figure (6.15). The red graph shows the performance for the developed EQUEST method, and the black graph shows the performance for the nonlinear observer. The green graph shows the plotted sensor data.

It can be seen from this plot and the deviation from the sensor data for the nonlinear observer and the developed EQUEST method in figure (6.16) and (6.17) that the developed EQUEST method finds the correct value faster than the nonlinear observer. The method computes the correct value in just one time-step, which will give a faster response in the start-up phase. Note that the scaling of the plots for the deviation are different than the plots for the sensor data.

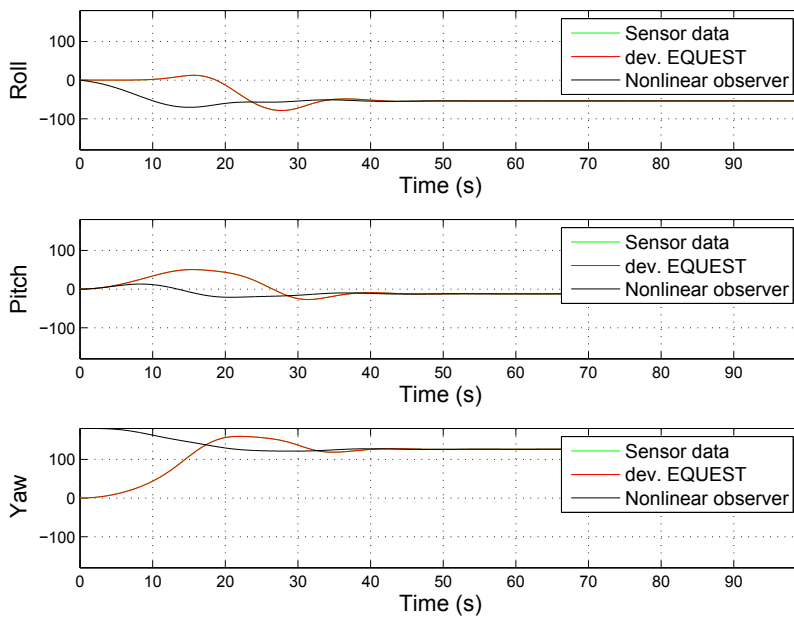


Figure 6.15: Attitude estimation sine wave response for the developed EQUEST method and the nonlinear observer.

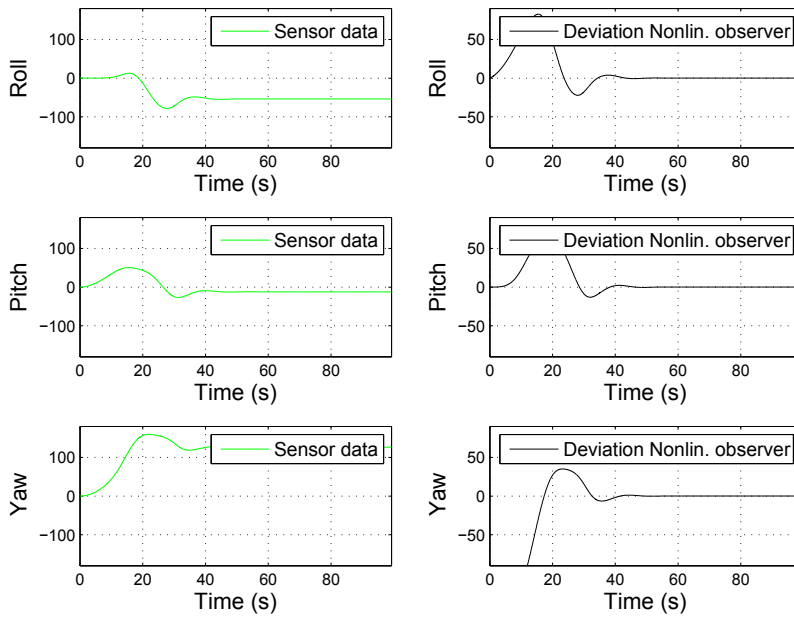


Figure 6.16: Sensor data and deviation for the nonlinear observer.



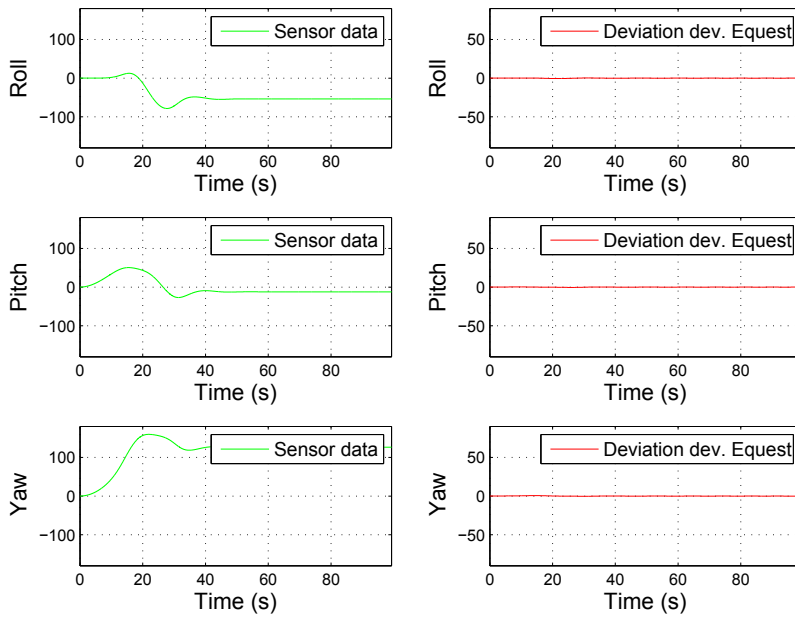


Figure 6.17: Sensor data and deviation for the developed EQUEST method.

### 6.3.2 Step function

The gyroscope measurements has also been modeled with a step from  $-\pi$  to  $\pi$  at time  $t = 20$ . The developed EQUEST method is compared to the nonlinear observer. The results are presented in figure (6.18). The developed EQUEST method is plotted with a red graph, the nonlinear observer with a black graph and the sensor data with a green graph. The plots for the deviation from sensor data in figure (6.3.2) and (6.3.2) show that the nonlinear observer has less overshoot when adjusting to the new stationary value, but that the developed EQUEST method finds the new value faster. Note that the scaling in the plots for the sensor data and the deviations are the same, unlike for the other test cases.

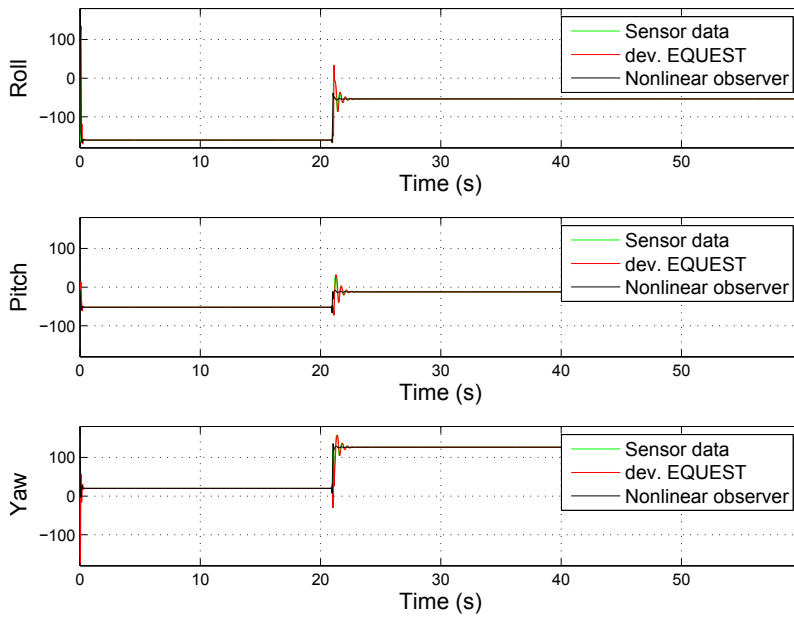


Figure 6.18: Attitude determination step response for the developed EQUEST method and the nonlinear observer.

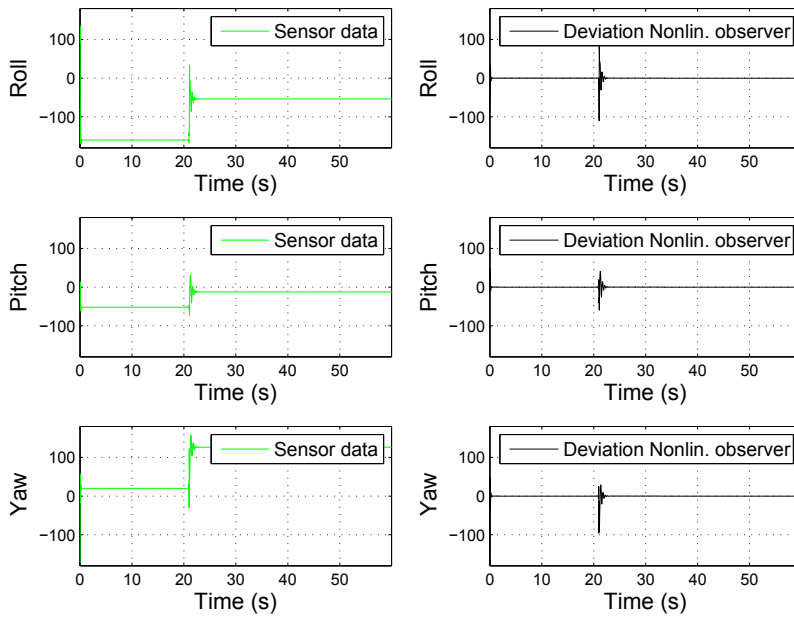


Figure 6.19: Sensor data and deviation for the nonlinear observer.

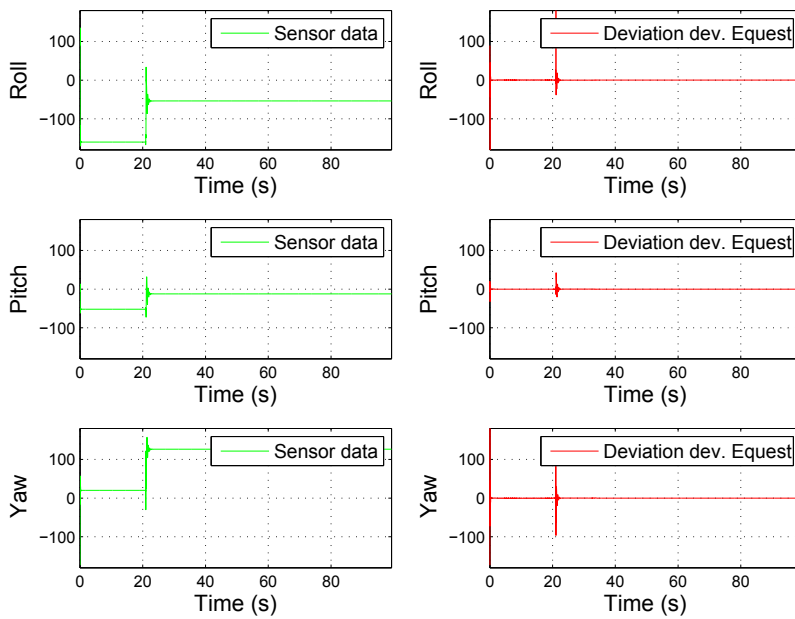


Figure 6.20: Sensor data and deviation for the developed EQUEST method.

## 6.4 Nonlinear observer with loss of data

The nonlinear observer is compared to the developed EQUEST method, with loss in the sun sensor data along each of the three directional axes. Plots of the methods and the sensor data are presented in figure (6.21), (6.24) and (6.27). The deviations from the sensor data for the nonlinear observer are shown in the figures (6.22), (6.25) and (6.28). The deviations from the sensor data for the developed EQUEST method are shown in figure (6.23), (6.26) and (6.29). The red graph shows the response for the developed EQUEST method, the black graph shows the response for the nonlinear observer and the green graph shows the sensor data.

The nonlinear observer has hardly any change in performance for the loss of sensor data, but the developed EQUEST method is less robust. Note that the scaling in the plots for the deviation are different than in the plots for the sensor data. The reason for less deviations from the reference quaternions for early time-steps are the choice of initial values which are different from the other test cases. Both methods have the same initial value as the sensor data, which give them a better response for the start-phase of the simulations.

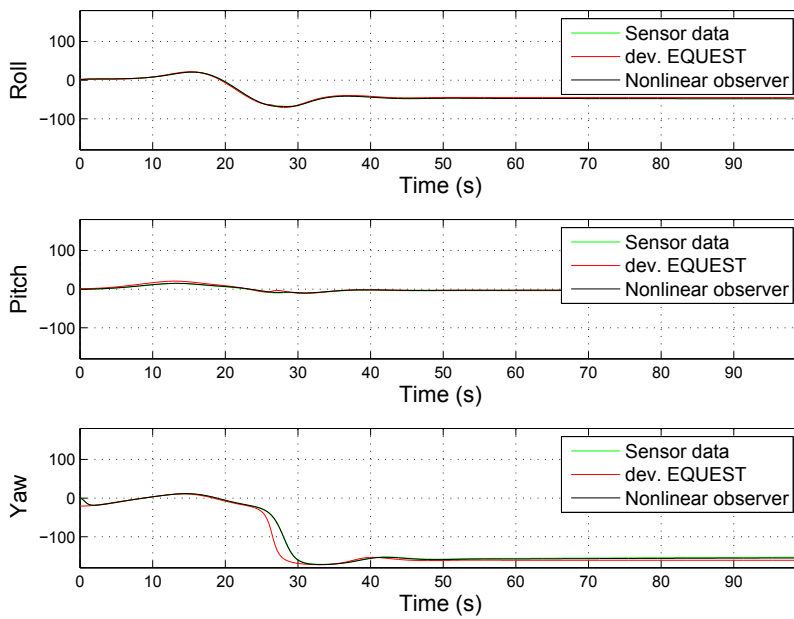


Figure 6.21: Attitude determination for the nonlinear observer and the developed EQUEST method with loss of sun sensor data along the x-axis.

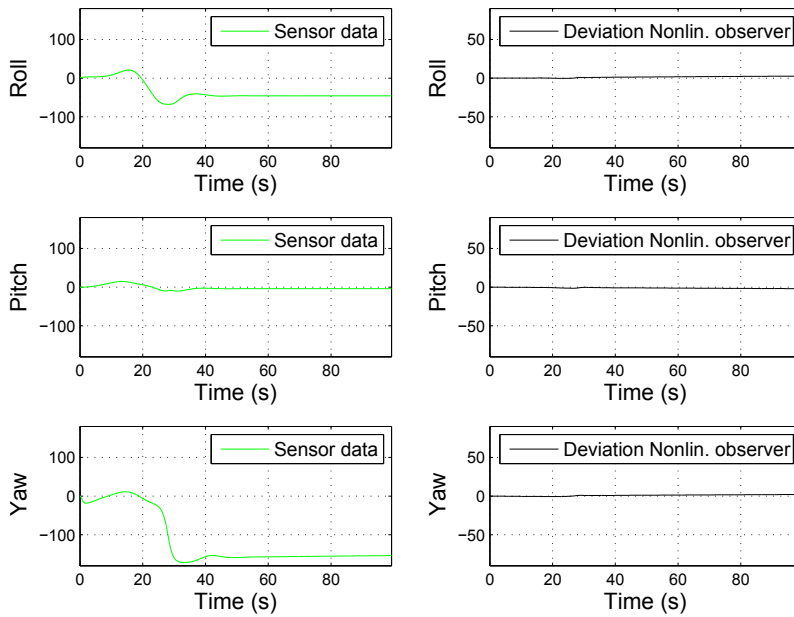


Figure 6.22: Sensor data and deviation for the nonlinear observer, with loss of sun sensor data along the x-axis.



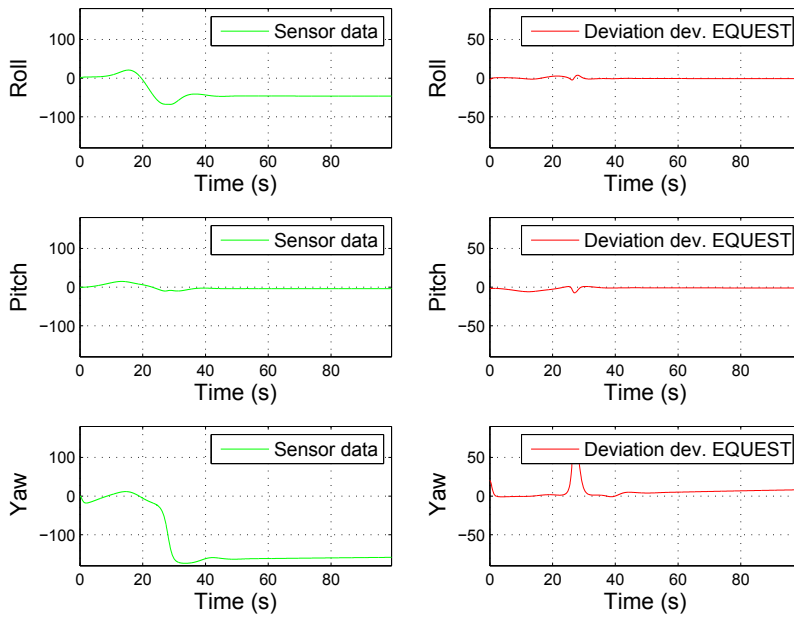


Figure 6.23: Sensor data and deviation for the developed EQUEST method, with loss of sun sensor data along the x-axis.

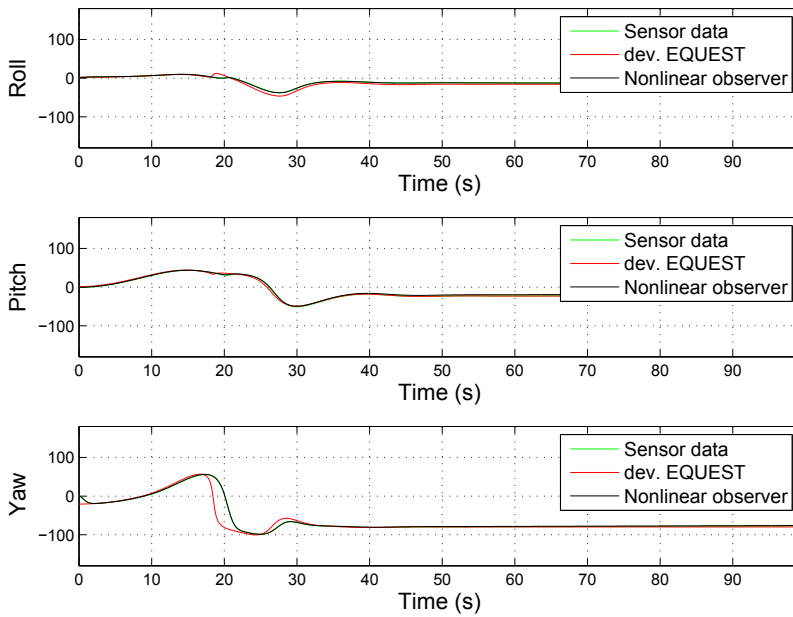


Figure 6.24: Attitude determination for the nonlinear observer and the developed EQUEST method with loss in sun sensor data along the y-axis.

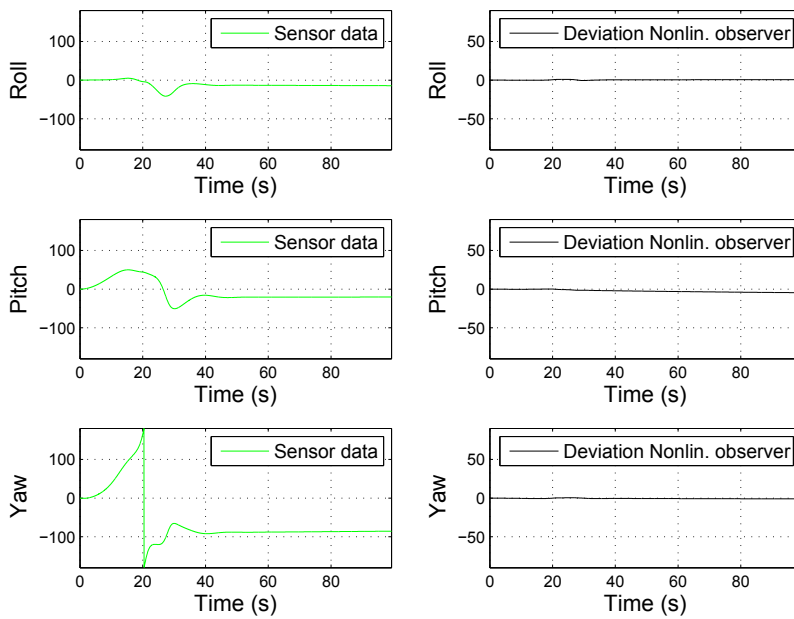


Figure 6.25: Sensor data and deviation for the nonlinear observer, with loss in sun sensor data along the y-axis.

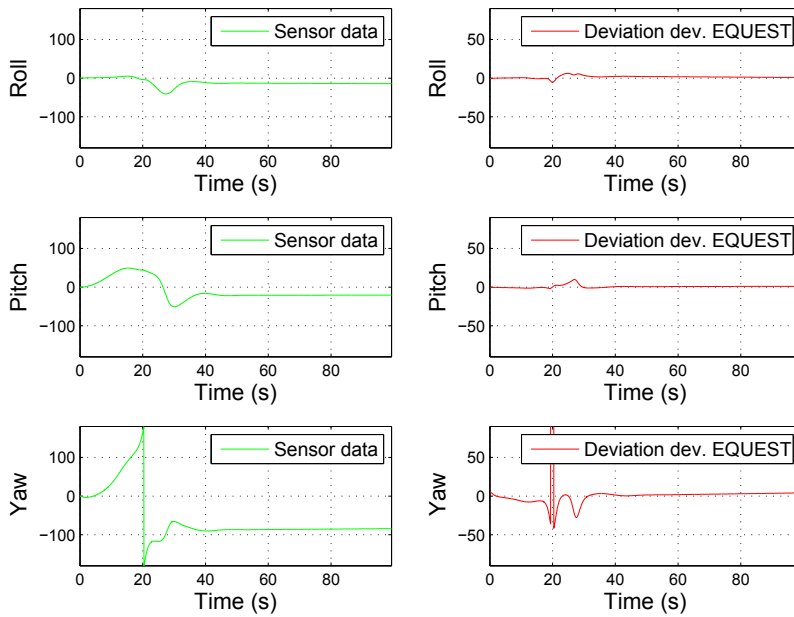


Figure 6.26: Sensor data and deviation for the developed EQUEST method, with loss in sun sensor data along the y-axis.

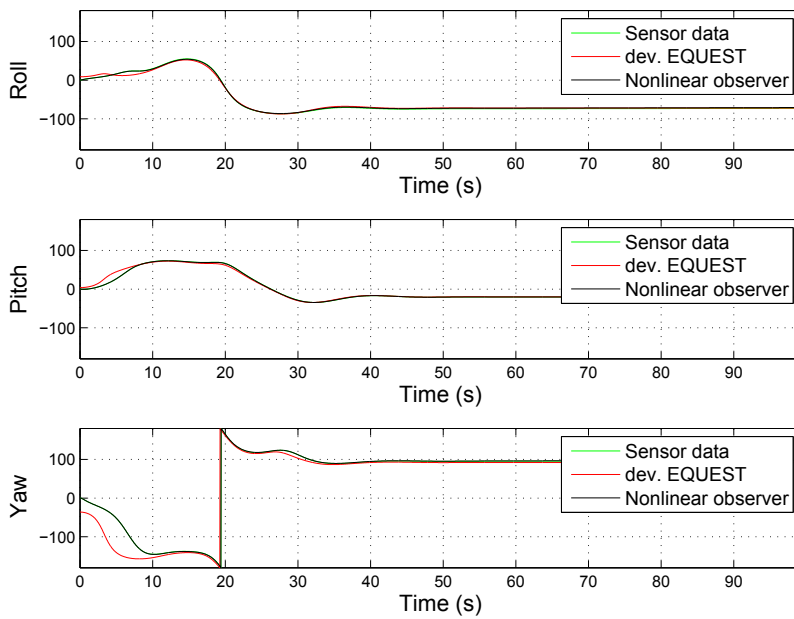


Figure 6.27: Attitude determination for the nonlinear observer and the developed EQUEST method with loss of sun sensor data around the z-axis.

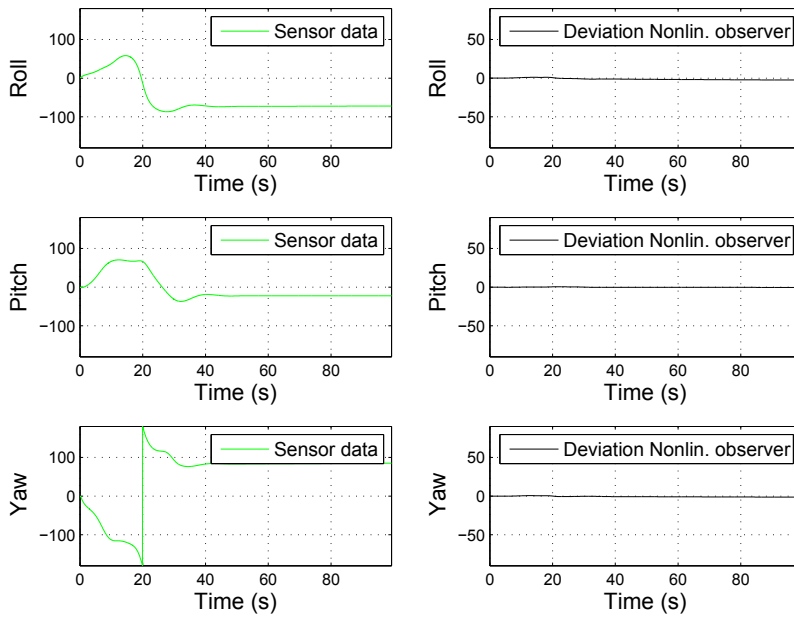


Figure 6.28: Sensor data and deviation for the nonlinear observer, with loss in sun sensor data along the z-axis.

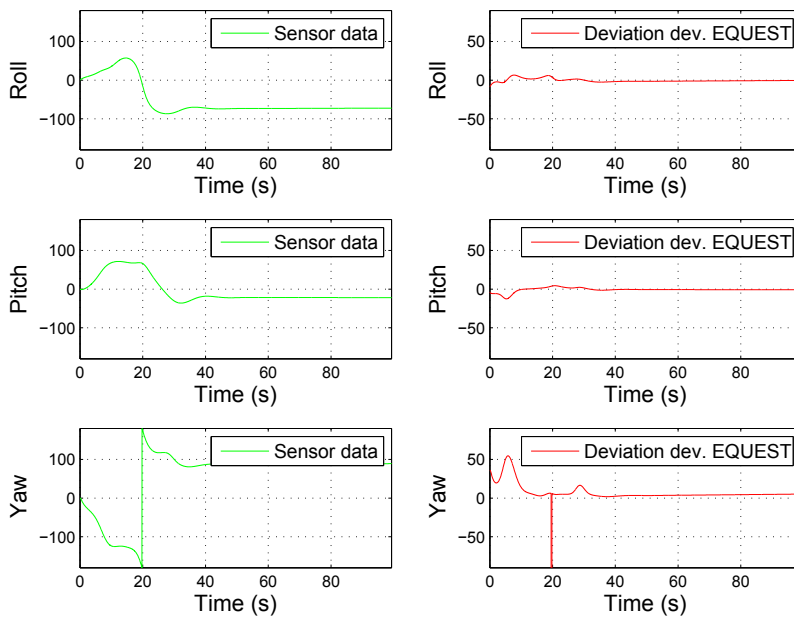


Figure 6.29: Sensor data and deviation for the developed EQUEST method, with loss in sun sensor data along the z-axis.

## 6.5 Nonlinear observer bias

The nonlinear observer provides a gyro bias estimation, which makes it more suitable for attitude determination than the developed EQUEST method. Plots for the estimated bias for the nonlinear observer are presented in figure (6.30) and (6.31). The blue dotted lines show the actual gyroscope bias and the black graphs show the estimated bias. The bias estimation for the sine wave response and the step response described in test case 3 is plotted. Both bias estimations converge towards the correct values of 0.007 degrees.



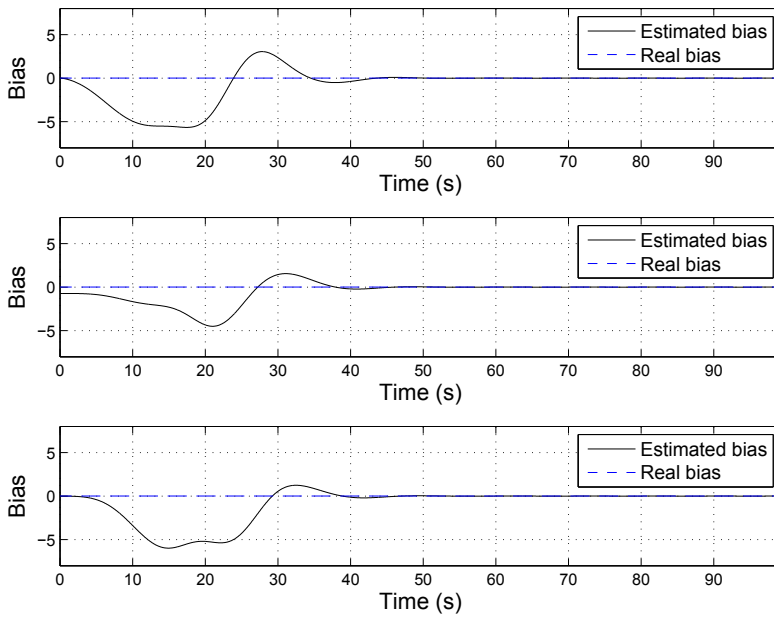


Figure 6.30: Sine wave response for the estimated bias of the nonlinear observer.

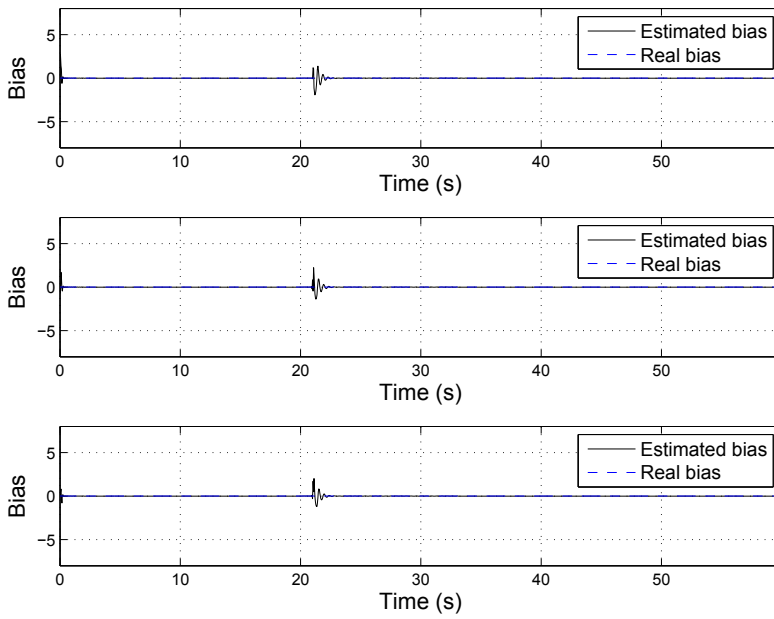


Figure 6.31: Step response for the estimated bias of the nonlinear observer.

## 6.6 Nonlinear observer with variable gains

The nonlinear observer with variable gains in the injection term is compared to the response of the developed EQUEST method in figure (6.33). The red graph shows the developed EQUEST method response, the black graph shows the nonlinear observer response and the green graph shows the sensor data. A plot for the varying gains is presented in figure (6.32). The blue line shows the gain for the sun sensor and the black line shows the gain for the magnetometer. Both gains start at fixed values ( $e^5$  and  $e^3$ ), before decreasing exponentially towards value 0.5. The deviation between the sensor data and the two estimation methods are given in figure (6.34) and (6.35). Note that the scaling in the plots for the deviation are different than in the plots for the sensor data.

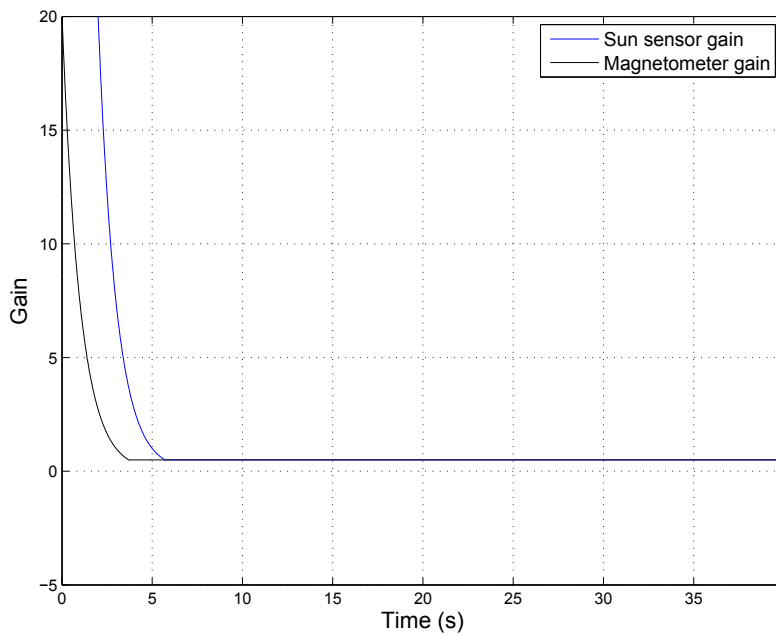


Figure 6.32: Gains in the injection term for the sun sensor and magnetometer measurement vectors.

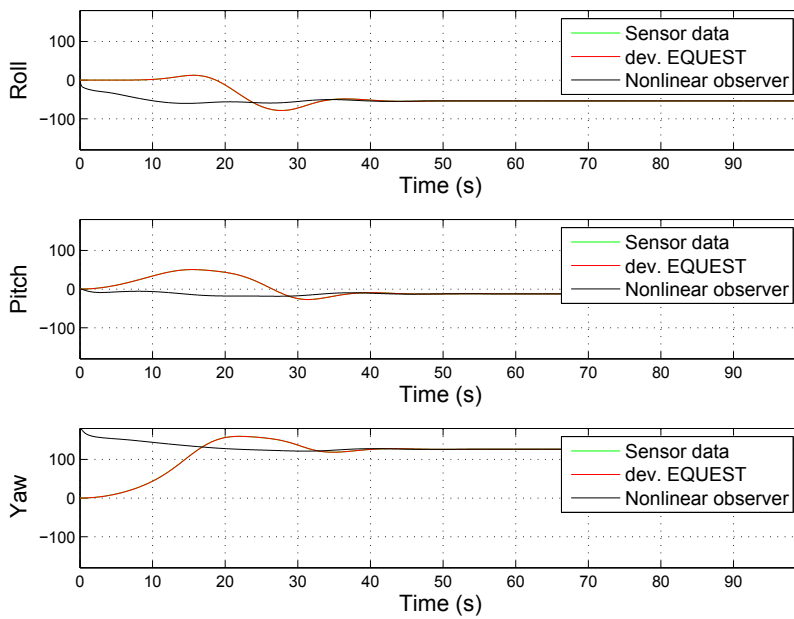


Figure 6.33: Attitude determination for the developed EQUEST method and the nonlinear observer with variable gains in the injection term.

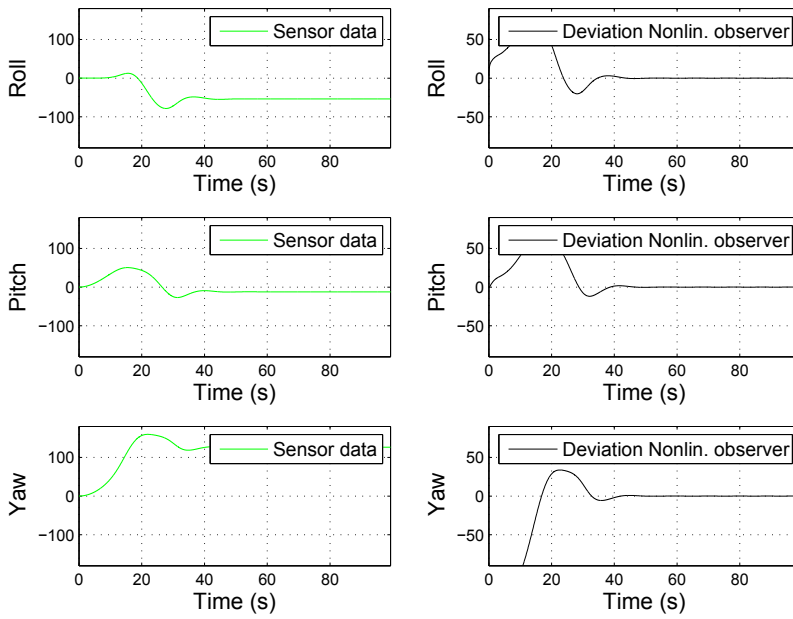


Figure 6.34: Sensor data and deviation for the nonlinear observer with variable gains in the injection term.

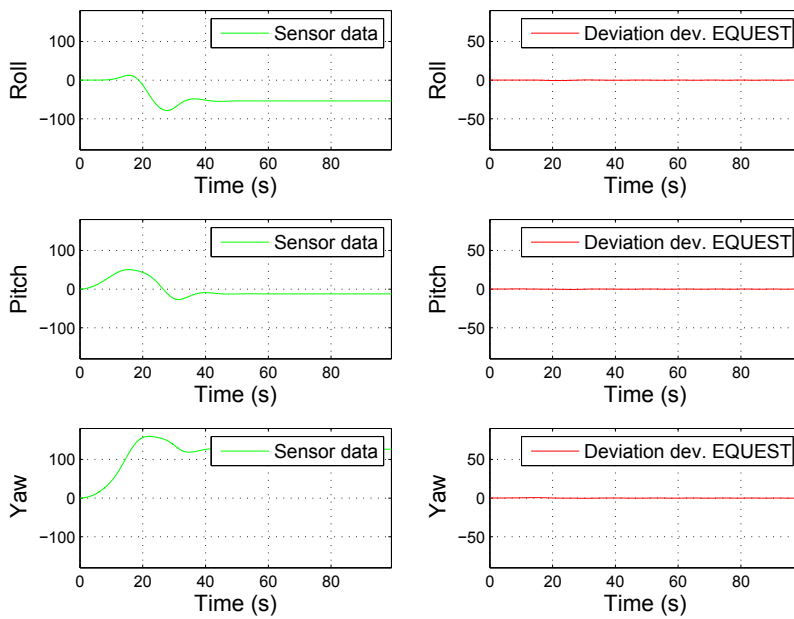


Figure 6.35: Sensor data and deviation for the developed EQUEST method with variable gains in the injection term.

## 6.7 Nonlinear observer with disturbances

The nonlinear observer is compared to both the developed EQUEST method and the EKF with disturbances for the sensor vectors in figure (6.36). The blue graph shows the response for the EKF, the red graph shows the response for the developed EQUEST method and the black graph shows the response for the nonlinear observer. The green graph shows the sensor data. The deviations between the estimation methods and the sensor data are given in figure (6.37), (6.38) and (6.39). The nonlinear observer performs remarkably much better than both the EKF and the developed EQUEST method when disturbances are introduced. Especially the developed EQUEST will have erratic results. Note that the scaling in the plots for the deviation are different than in the plots for the sensor data.



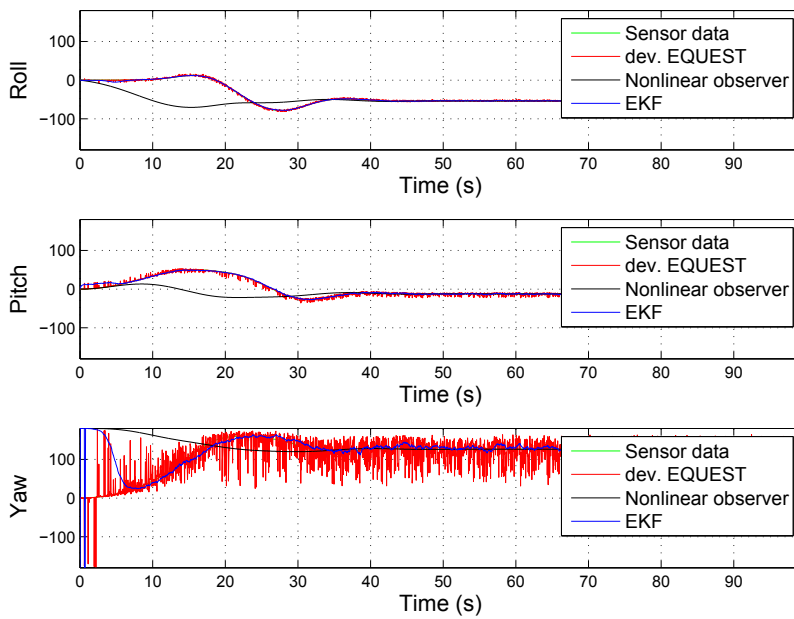


Figure 6.36: Attitude determination for the developed EQUEST method, the EKF and the nonlinear observer with disturbances in the sensor data.

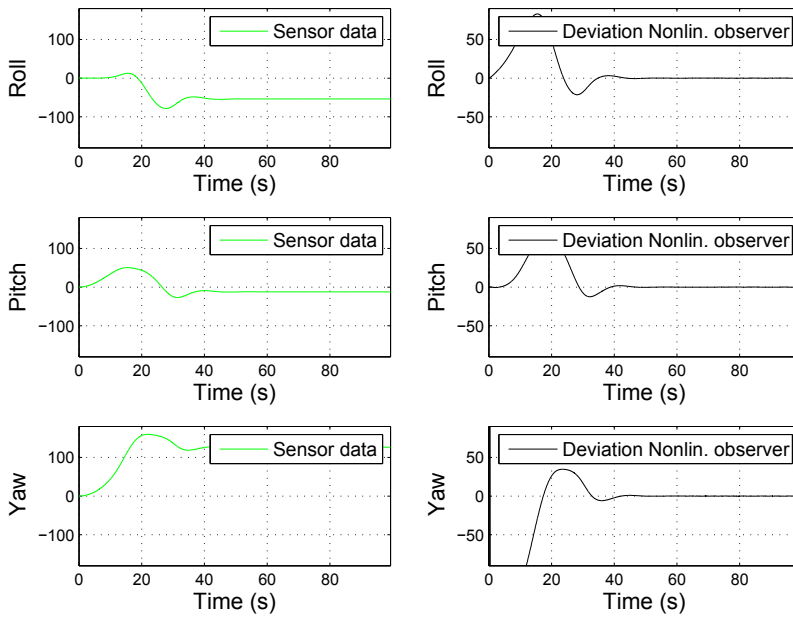


Figure 6.37: Sensor data and deviation for the nonlinear observer with disturbances.

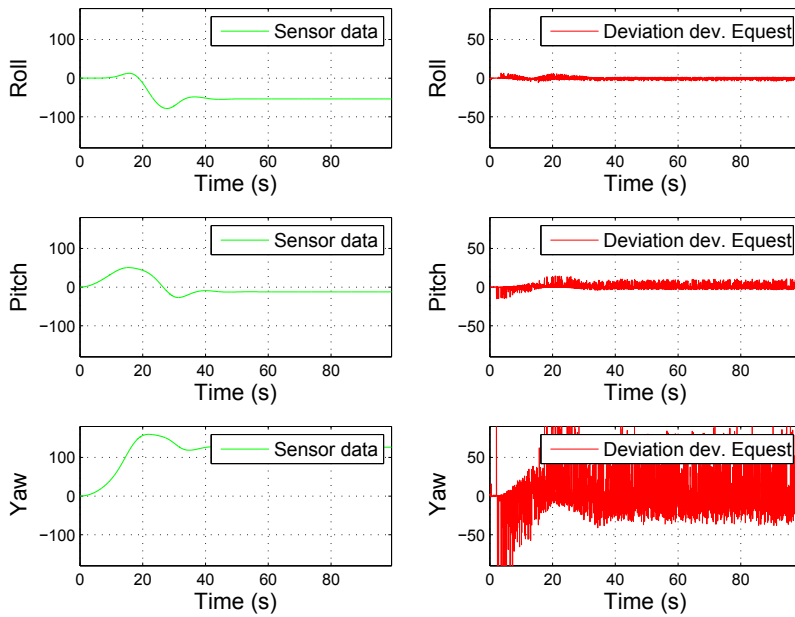


Figure 6.38: Sensor data and deviation for the developed EQUEST method with disturbances.

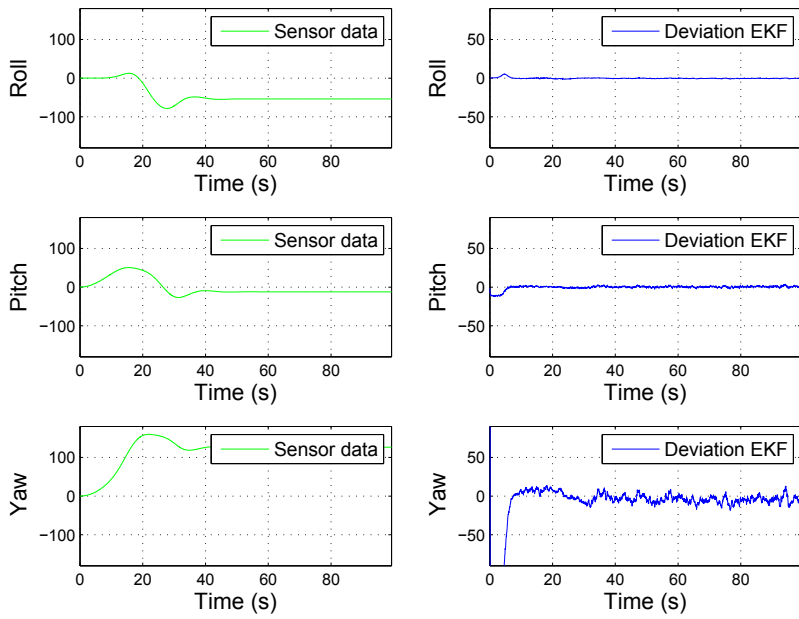


Figure 6.39: Sensor data and deviation for the EKF with disturbances.

## **6.8 Nonlinear observer with time-varying reference vectors**

The nonlinear observer with time-varying reference vectors is compared to both the EKF and the developed EQUEST method in figure (6.40). The blue graph shows the EKF, the red graph shows the developed EQUEST method and the black graph shows the nonlinear observer. The green graph shows the sensor data. The deviations from the sensor data for the estimations methods are presented in figure (6.41), (6.42) and (6.43). The time-varying reference vector implementation does not influence the simulation results to a large degree because they are changing very slowly. The largest change in behavior can be seen for the implemented EKF.

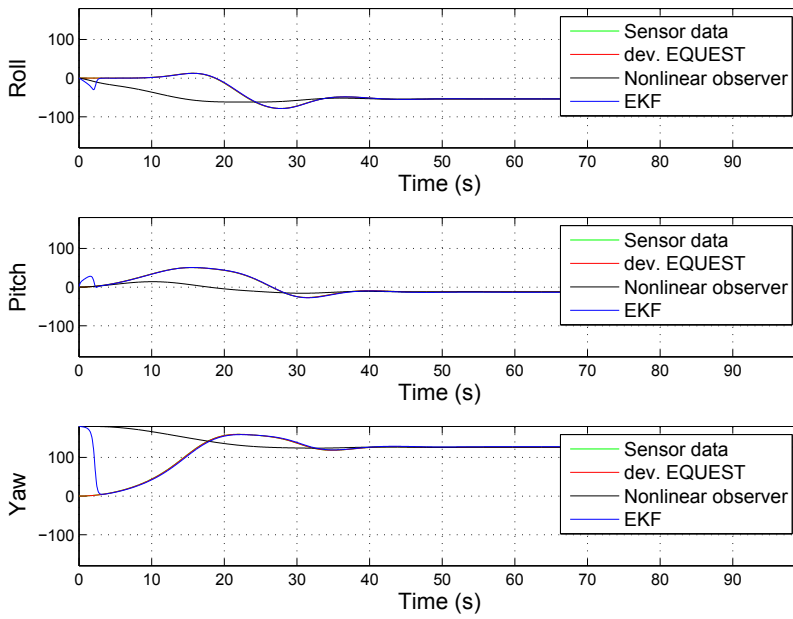


Figure 6.40: Attitude determination for the developed EQUEST method and the nonlinear observer with time-varying reference vectors.

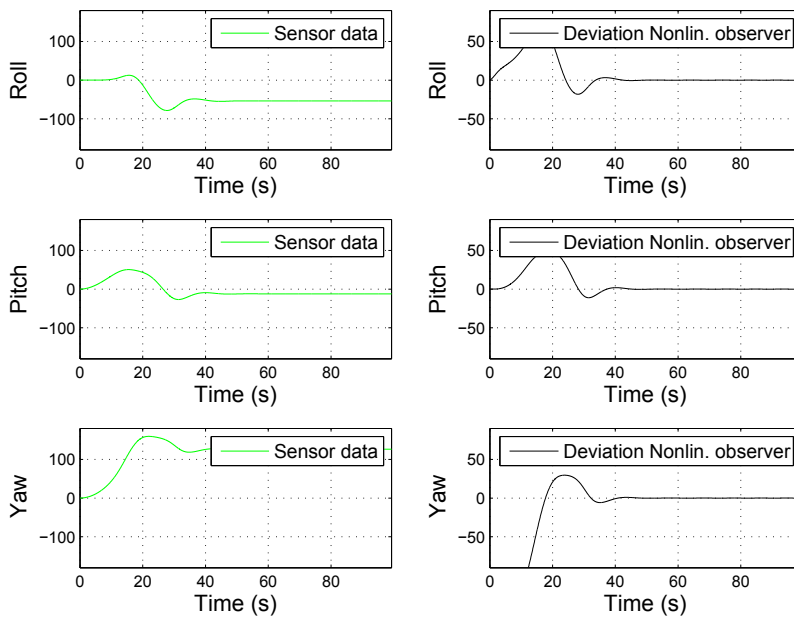


Figure 6.41: Sensor data and deviation for the nonlinear observer with time-varying reference vectors.

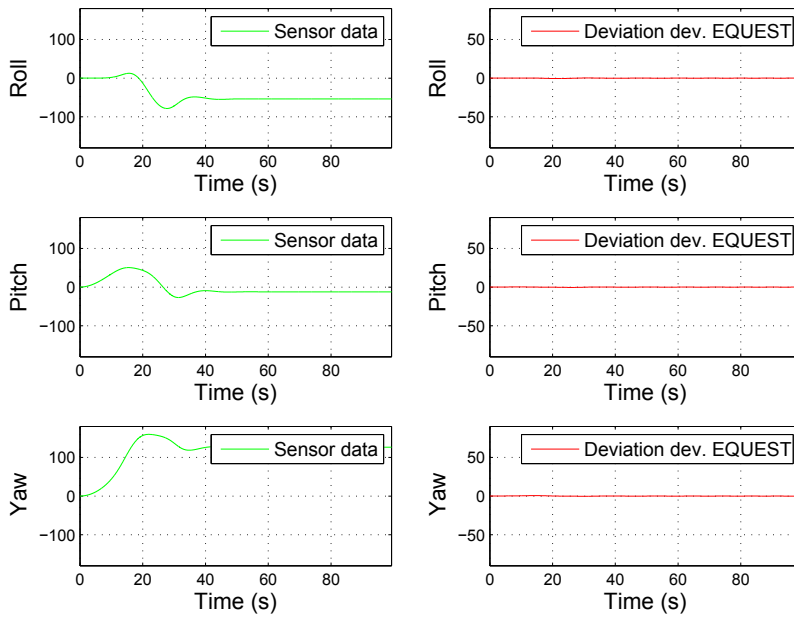


Figure 6.42: Sensor data and deviation for the developed EQUEST method with time-varying reference vectors.



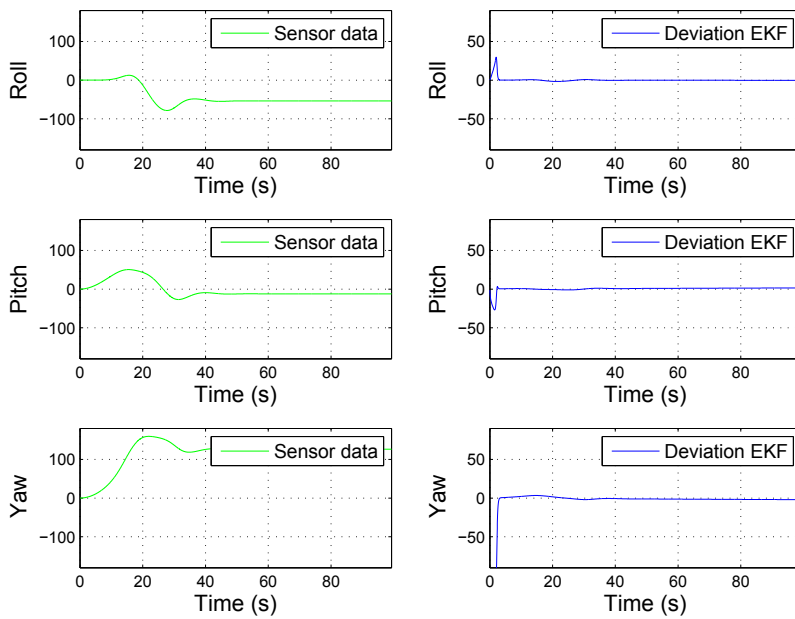


Figure 6.43: Sensor data and deviation for the EKF with time-varying reference vectors.

## 6.9 Combination of developed EQUEST and Nonlinear observer

A combination of the developed EQUEST method and the nonlinear observer is compared to the EKF. The red graph shows the combination method response, the blue graph shows the EKF response and the green graph shows the sensor data. The deviation for the combination method from the sensor data are plotted in figure (6.45) and the deviation for the EKF from the sensor data is plotted in figure (6.46). The response for the combination method can be seen to be faster than the nonlinear observer response, since the developed EQUEST method is used for finding the initial value. Simulation results for the combination method and the EKF with disturbances along only the x-axis of the sensor data is presented in figure (6.47). An extra simulation with noise along all the axes is presented in figure(6.50). The response for the nonlinear observer is slower with the disturbances, but it converges towards the correct value and is more smooth than the EKF. Deviation from the sensor data both for the EKF and the combination method are presented in figure (6.48) and (6.49). An extra plot with disturbances along each of the axes in the sensor data is shown in figure (6.50). Note that the scaling in the plots for the deviation are different than than in the plots for the sensor data.

6.9. COMBINATION OF DEVELOPED EQUEST AND NONLINEAR OBSERVER103

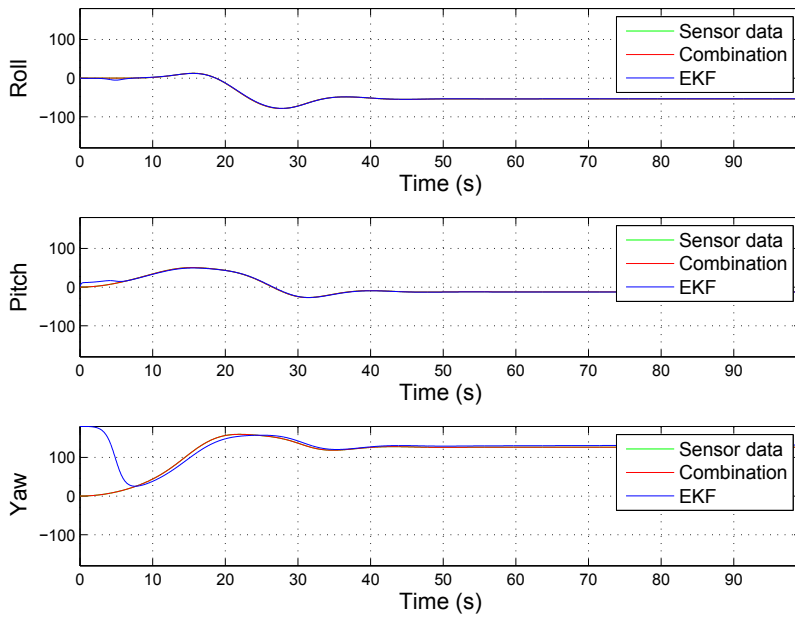


Figure 6.44: Attitude determination for the combination method and the EKF.

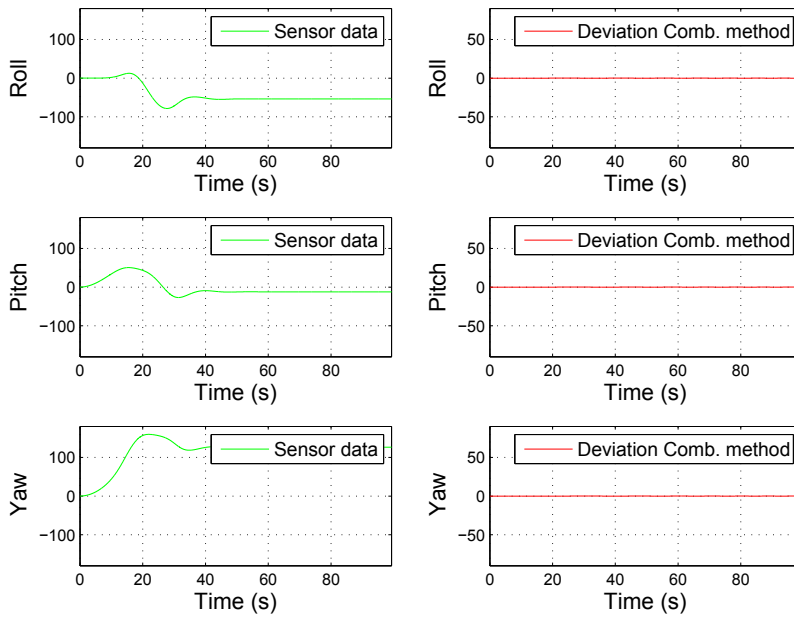


Figure 6.45: Sensor data and deviation for the combination of the developed EQUEST and the nonlinear observer.

6.9. COMBINATION OF DEVELOPED EQUEST AND NONLINEAR OBSERVER105

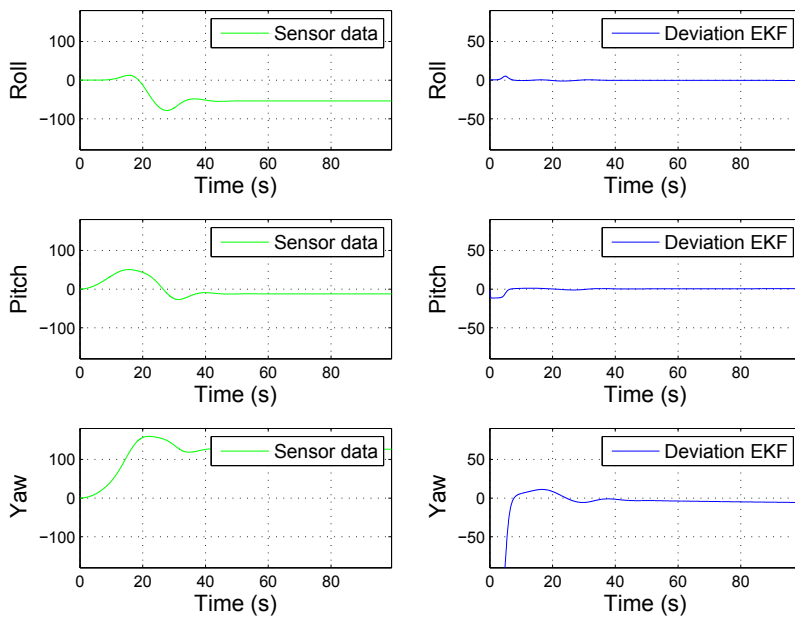


Figure 6.46: Sensor data and deviation for the EKF.

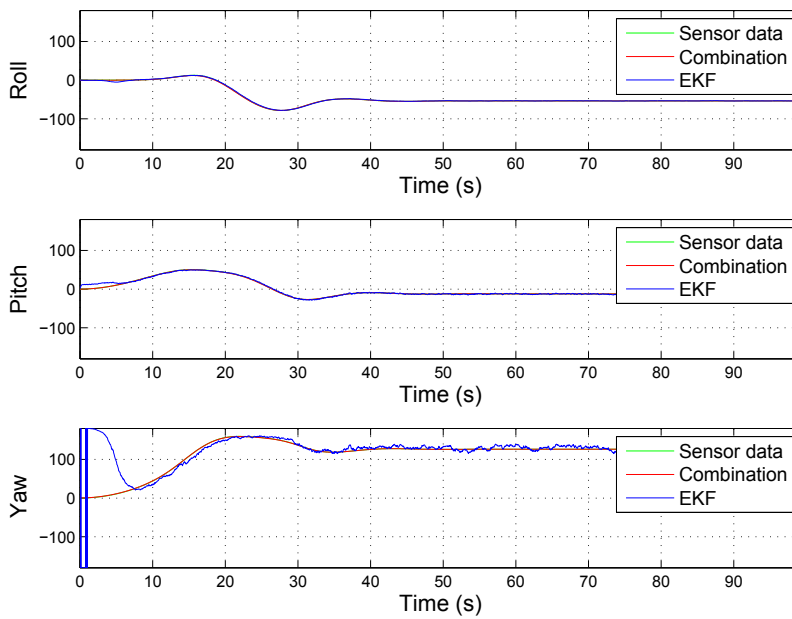


Figure 6.47: Attitude determination for the combination method and the EKF with disturbances in the sensor data.

6.9. COMBINATION OF DEVELOPED EQUEST AND NONLINEAR OBSERVER107

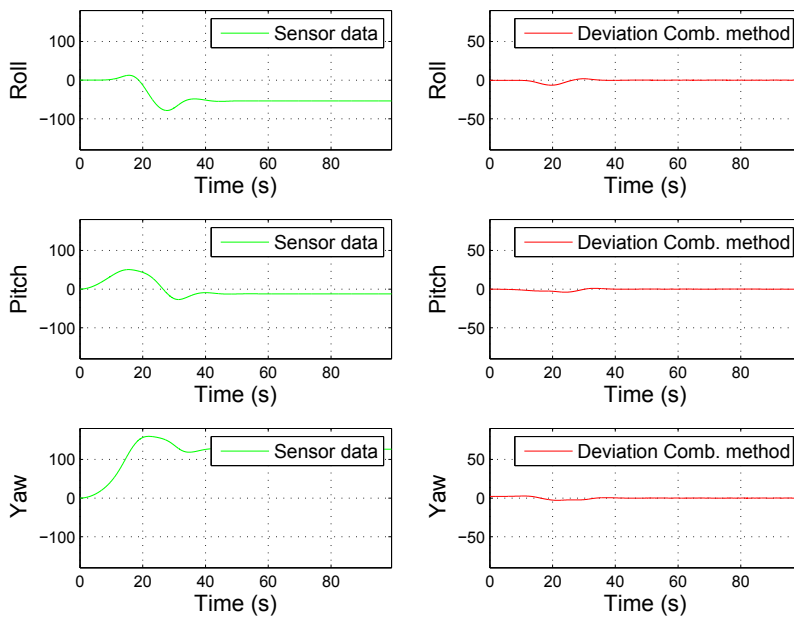


Figure 6.48: Sensor data and deviation for the combination of the developed EQUEST and the nonlinear observer with disturbances.

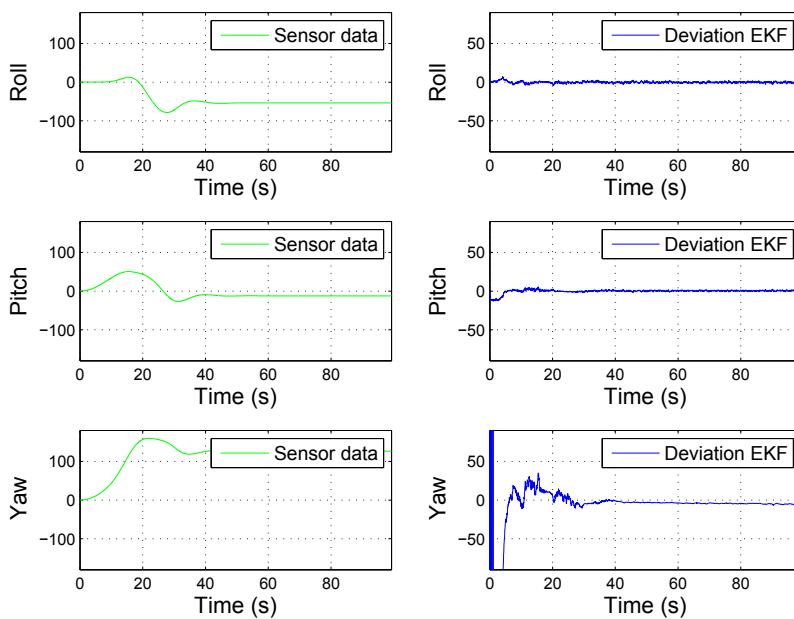


Figure 6.49: Sensor data and deviation for the EKF with disturbances.



6.9. COMBINATION OF DEVELOPED EQUEST AND NONLINEAR OBSERVER109

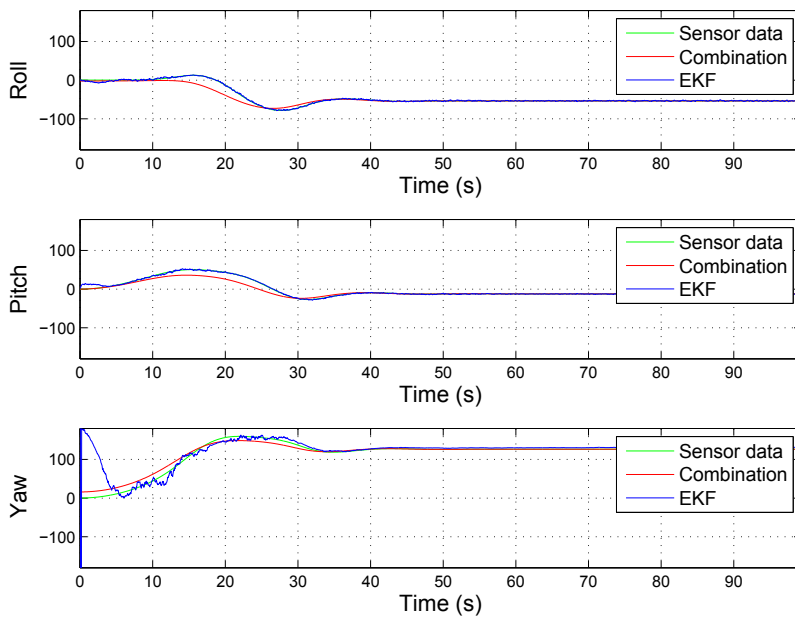


Figure 6.50: Attitude determination for a combination method and the EKF with noise along all the axes for the sensor data.

## 6.10 Test cases conclusion

The test results for all the test cases are analyzed in the following pages. The main focus is comparison of the different implemented methods in order to find the estimation method best suited for the NUTS CubeSat mission.

- **Case 1: Developed EQUEST**

The developed EQUEST method is compared to the original EQUEST method and an extended Kalman filter. From chapter 6.1, the plots show that no difference in performance for the two methods can be seen. Both the new developed EQUEST method with quaternion products in the cost function, and the old original EQUEST method with quaternion subtractions, perform slightly worse than the extended Kalman filter for the implemented data set. No change in performance can be achieved with quaternion products instead of quaternion subtractions. The new method with quaternion products can be written on the same form as the old method, only with different matrices in the cost function. This will cause a shift in the eigenvalues of the method, but it will not change the next quaternion used in the solution. It should be noted that only one test for these two methods is done in this thesis, and more tests should be made in order to cover all possible scenarios, for instance the response for multiple eigenvalues or parallel eigenvectors.

The sensor data is plotted with the developed EQUEST method, the original EQUEST method and the Kalman filter. The deviations from the sensor data for the methods show that the EKF has the best results, and it finds the correct value faster than the two EQUEST methods. Using the same initial values for both the reference quaternions and the implemented methods will give better performance for the EQUEST methods as shown in [4]. However, the main object of this test case is to compare the developed and the original EQUEST methods, and see if the behavior changes with the mathematical correct notation.

- **Case 2: Developed EQUEST with loss of sun sensor data**

The developed EQUEST method response is compared to the response for the EKF with loss of sun sensor data along each of the directional axes. The two methods are similar for loss of data along the x-axis, but figures (6.9), (6.11), (6.12) and (6.14) show that the EKF performs better for loss along the y-axis and the z-axis than the developed EQUEST method. This is probably because the loss in sun sensor data along these axes cause a steeper change in the sensor data. The EKF can handle such fast changes better than the developed EQUEST method because it uses the previous attitude when calculating the next attitude quaternions. The developed EQUEST method starts over for each time-step and will be slower to perceive the fast attitude change.

- **Case 3: Nonlinear observer**

The implemented nonlinear observer is compared to the developed EQUEST method for different models of the angular velocities. For a sine wave gyroscope measurement, the performance of the developed EQUEST method is better in the start-up phase than the nonlinear observer. The EQUEST method is a point estimation method which solves for the next input quaternion in one single time-step. The nonlinear observer will however use more time before converging to the correct attitude. This might cause problems for the CubeSat as mentioned earlier 4.3. Since the magnetic field of the satellite will be influenced by the magnetorquers, the estimation of the attitude and the control of the attitude must be separated. The performance of the chosen estimation method should be a trade-off between a short start-up time and robustness against disturbances. From the plots in the figures (6.15) and (6.16), it can be seen that the nonlinear observer uses between 30 and 40 seconds before converging towards the correct value, which makes the developed EQUEST method a more optimal method for this scenario.

For the step response, the difference in the performance of the two methods are more subtle. The nonlinear observer has slightly smaller overshoot before finding the right value, than the developed EQUEST method. No real preference of one method over the other can be found in this case, but it should be noted that the overshoot for the developed EQUEST is more than  $180^\circ$ , which might be problematic in certain situations, for instance when the satellite is supposed to point the infrared camera payload towards the Earth.

- **Case 4: Nonlinear observer with loss of data**

This is the same test scenario as in test case 2, with loss of data along each axis for the sensor data. It should be noted that the data set is different than the one in test case 2, and that the initial values are the same for the reference quaternions and the implemented methods, making the deviations for the start-up time smaller than in the other test cases. The performance of the developed EQUEST method is worse for loss of data than the nonlinear observer. The nonlinear observer performs just as good without the sensor data along each of the axes, but the developed EQUEST method struggles with the fast changes, especially along the z-axis. Since the initial values for this test case are the same for both the estimation methods and the reference quaternions, the nonlinear observer performs better than it might have done for a different start value. Even with another initial value for the nonlinear observer, the performance for the developed EQUEST method is not good enough. It can be concluded that the preferred choice for the estimation method for such a test case would be the nonlinear observer.

- **Case 5: Nonlinear observer bias**

This test case is included to show the performance of the bias estimation for the nonlinear observer. The gyroscope bias is chosen as 0.007 degrees, and for the sine

wave, and the step function implementation, the estimated values can be seen to converge towards their true values. The gyroscope bias estimation for the nonlinear observer is one of the main arguments for using it for the satellite instead of the developed EQUEST method, which is unable to estimate any of the sensor measurement biases.

- **Case 6: Nonlinear observer with variable gains**

The injection term for the nonlinear observer is described in chapter 4.5. The sensor measurements from the magnetometer and the sun sensor are weighted with the gains  $k_1$  and  $k_2$ . This test case shows how the gains are set to be time-varying and how the nonlinear observer performance changes. The main idea is to make the nonlinear observer converge faster, but it can be seen from the figures (6.33) and (6.34) that the nonlinear observer has a faster response of only a few seconds by using this approach. The overshoot for the nonlinear observer is slightly smaller than with constant gains. The developed EQUEST method is still much faster in the start-up phase than the nonlinear observer, at least for this test case with a modeled sine wave gyroscope measurement.

- **Case 7: Nonlinear observer with disturbance**

Since the developed EQUEST method is implemented without the attitude prediction term, it is more sensitive towards disturbances and noise. Therefore, only the x-axis sensor data is implemented and tested to illustrate a mild influence from measurement noise. The results show that even if the noise for the measurement vectors are small, the performance of the developed EQUEST method is highly erratic. Because the method computes the solution in one time-step, it discards all previous information, and will not be able to filter out noise. The response for the EKF is also shown in the plots. It performs better than the developed EQUEST method, but slightly worse than the nonlinear observer. The nonlinear observer would be more prone to disturbances, if the convergence time was made shorter. This test case shows how much the choice of estimation method will influence the attitude determination. To avoid erroneous attitudes, the nonlinear observer should be the implemented method for the NUTS satellite.

- **Case 8: Nonlinear observer with time-varying reference vectors**

The time-varying reference vectors will not have much influence on the simulated estimation methods. The change of the vectors will be slow and not much change is visible in the behavior for the methods. The stability properties will change for the nonlinear observer, but the implemented method will still be globally exponentially stable as long as the measurement vectors remain unbiased. The largest difference can be seen for the EKF, which have a larger deviation in the beginning of the simulation period, because it filters out some of the signals. For the final implementation,

time-varying reference vectors must be tested more extensively, especially for the nonlinear observer since it is a relatively new method for use in satellites.

- **Case 9: Combination of EQUEST and nonlinear observer**

A combination of the developed EQUEST method and the nonlinear observer give very successful results. For all the other test cases (except for test case 4), the initial value for both the developed EQUEST method and the nonlinear observer and the initial value for the reference quaternions were different. The developed EQUEST method finds the correct initial value for the nonlinear observer during the first time-step and the nonlinear observer follows the sensor data without any deviations.

By introducing noise into the implementation, no deviation for the nonlinear observer can be seen, unless there is so much noise that the developed EQUEST method is unable to find the right initial value, as seen in figure (6.50). The nonlinear observer will automatically behave as if the initial value is wrong, and converge towards the correct reference. The effects of much noise is therefore much smaller than for the developed EQUEST method alone. Compared to the EKF, the combination method would be preferred because of the simplicity of the implementation and its good performance.



# Chapter 7

## Discussion

The thesis is divided into two main parts. The first part is concerned with changing the cost function for the original EQUEST method developed by Jenssen and Yabar [4]. The convergence properties of this method is inspected, and the simulation results for the new developed EQUEST are compared to the original EQUEST method and an extended Kalman filter. Testing of the implementation of this method is made, for comparison to the original EQUEST method and an implemented extended Kalman filter.

The developed EQUEST method requires a few more matrix multiplications than the original EQUEST method, because of the matrix representations in the cost function. The number of operations is still less than for the extended Kalman filter, even though the eigenvalues and eigenvectors of a  $4 \times 4$  matrix must be found. The added number of operations are not expected to be a problem, since the previously developed EQUEST method requires only about 8% of the operations of the EKF.

When substituting quaternion products for quaternion subtractions, several issues arise. The minimization problem is not intuitively as easy, because the minimal value for the product is no longer zero, but has value 1. By making use of a matrix representation, the cost function can still be written as a minimization problem. This problem can be solved by use of eigenvector mathematics, to find the optimal eigenvalue and the corresponding eigenvector. The original EQUEST method was implemented using Lagrange multipliers, which means a large part of the existing code can be used when implementing the new developed EQUEST method.

The convergence analysis shows that the method converges locally. The eigenvalues are shifted for the new method, but convergence for the method can still only be proven under certain conditions. Global results can only be assumed if there are more than one vector measurement and these are non-parallel. This result is mathematically weak, but testing of the method in this thesis and in [4] show that the developed EQUEST method most often will converge towards the correct reference quaternion. However, using this method means that stability for the entire attitude determination and control system cannot easily be proved.

The simulation results show that there is no real difference in behavior for the developed EQUEST method and the original EQUEST method. The performance for the two methods appear to be identical, but more testing for the developed EQUEST method on the microcontroller on the prototype is expected to give an unequivocal result. The

method is compared to the well-known extended Kalman filter, which can be seen to have better simulation results, especially for fast change in the attitude. The new developed EQUEST method is also compared to the implemented extended Kalman filter for the situation where loss in the sensor data along one of the directional axes occurs. Especially in the z-direction, the results for the EKF are better, but these results are not critical. The extended Kalman filter will be too complex to implement for the satellite, and the developed EQUEST method can be shown to respond fast in the start-up phase and the deviation from the sensor data is within reasonable bounds.

Part two of the thesis uses the new developed EQUEST method and compares the simulation results to a nonlinear observer described in Grip et al. [42]. Several test scenarios are simulated in Matlab. The main goal is to investigate the developed EQUEST method properties, and discuss the qualities of the method compared to this relatively new, but simple estimation method. Because of the few equations needed for the observer, and the lack of matrix multiplications, fewer operations are needed. This will save even more power for the ADCS of the satellite. The nonlinear observer does not require the computation of eigenvalues, and it is proved to be globally exponentially stable for unbiased reference vectors or if a vector bias estimation is implemented. This makes it a relevant alternative estimation method for use in the NUTS CubeSat.

The investigated test scenarios are described in chapter (6). Several responses for angular velocities are simulated, in addition to test cases with implemented noise and time-varying gains or reference vectors. The most important tests for comparison between the methods are the test cases which show the responses for typical satellite movements. Test case 3 shows the response for a typical satellite scenario with slow change in the attitude, and an example of a typical detumbling phase where the satellite suddenly need to change its orientation. The influence of disturbances are described in test case 7, and test case 2 and 4 illustrate what will happen in the simulations for loss of sun sensor data. The nonlinear observer performs better than the developed EQUEST method in almost all these cases, but the main drawback with the observer is the slow adaptation to the reference quaternions in the start-up phase of the simulations. The developed EQUEST method is much quicker, but it is not as robust against disturbances and fast change.

The sensor vector measurements will most likely be affected by noise, which tends to be particularly severe at high frequencies. The implemented methods are simulated with a sample time of 0,001s which corresponds to 100Hz. The microcontroller used can have frequencies of up to 16MHz. Disturbances in the gyroscope measurements or model noise might also occur, but only sensor vector measurement noise is tested in this thesis. The robustness towards noise is a very important quality for the estimation method, because it might be unable to estimate the correct attitude if the performance is too affected by noise. This is one of the reasons why Kalman filtering is used so extensively in attitude determination systems for larger satellites. This method has good filtering properties and the performance is not compromised by the influence of disturbances. The test results show that the developed EQUEST method will be very erratic, and does not cope well with the introduction of noise from the sensor measurements. The performance is so compromised that the nonlinear observer seem an obvious choice for the chosen estimation method. It can be argued that the developed EQUEST method might be preferred when the satellite is rotating slowly, but only if low sensitivities for the sensors are assumed.



Since the developed EQUEST method calculates the solution in one time-step, it can be used to find the initial value for the nonlinear observer. Such a combination method is implemented, and the results are shown in test case 9. A combination of the two methods will be more computationally expensive, but the methods combined will not exceed the number of operations of the extended Kalman filter. The original EQUEST method uses only about 8% of its number of operations, and few operations for the new method and the nonlinear observer will be added to this. The test results for the combination method is very promising, since the response is fast and still robust towards disturbances. It has also been considered to have the developed EQUEST method as a fall-back method for the attitude determination, but if both methods are implemented, the method might as well be used to find the correct start value for the nonlinear observer.

Another important challenge when using quaternions are the conversion back to Euler angles. Care must be taken to ensure no singularities appear during the transformation. Both in the developed EQUEST method and the nonlinear observer, precautions are made in the implemented code in order to prevent this.



# Chapter 8

## Conclusions

The quaternion subtraction terms in the previous EQUEST method have been replaced by quaternion product terms. The result is a mathematically correct model. Testing shows that there is no difference in the simulation results for the new developed EQUEST method and the old original implementation made in [4]. The methods are compared to an implemented extended Kalman filter, and can be seen to perform slightly worse than this method.

The convergence properties for the developed EQUEST method is investigated and the method can be proven to converge globally, except for certain situations which will rarely be experienced in real life.

A nonlinear observer is implemented and tested. The results are compared to the developed EQUEST method and for some scenarios it is also compared to the extended Kalman filter. The test results show that the developed EQUEST method is faster in the start-up phase, but the nonlinear observer performs better for loss of sensor data and when disturbances are introduced.

A combination of the developed EQUEST method and the nonlinear observer is implemented and tested. The combination method reacts fast, and is robust against noise. For too much disturbances, the developed EQUEST method will be unable to find the correct initial value for the nonlinear observer. The method will still converge towards the correct reference quaternions, but it will have a slower response for the start-up phase.

It can be concluded that the nonlinear observer seems like the best choice of estimation method for the attitude determination and control system in the NUTS CubeSat. A point estimation algorithm like the developed EQUEST method for calculation of the initial value will improve the start-up time. More testing for the implementation on the prototype, and calculations for how many operations the estimation method can have should be investigated before making a final decision.



# Chapter 9

## Further work

Further work involves testing and analyzing the performance of the developed EQUEST method and the nonlinear observer on the actual prototype with code written in the C programming language. Work on this has been done by another NUTS project member during the spring of 2012. Rewriting the code for the original EQUEST method from floating to fixed point variables has been done. This code works as a basis for the new implementation of the developed EQUEST method. For the nonlinear observer code, a vector bias estimation must be implemented to ensure global exponential stability for all possible scenarios. An albedo compensation for the sun sensor must also be implemented for the chosen estimation method. The implemented new data set is only used for the Matlab testing, and the prototype will be tested with real-time data from an IMU. Realistic scenarios should be chosen.

Tests for the chosen estimation method together with the rest of the attitude determination and control system must be performed and analyzed. Communication with other modules in the satellite must also be tested, especially with regard to the sleep mode code implemented for the microcontroller as described in [4].

For the control and actuator parts of the ADCS for the NUTS satellite, a lot of work still remains. The coil design for the actuators are yet to be decided. The system needs a switching algorithm which changes from the detumbling controller to the reference controller at the correct time. A power connection for the prototype is also needed, since a USB connection cannot be used in flight mode.

A final choice of sensors must be made, with focus on robustness against temperature changes, vacuum and radiation. The sensors must also be small and be in a reasonable price-range. It has been suggested by previous NUTS project members to try to estimate the local magnetic field created by all the different electronic components of the satellite. This way, the estimate can be feed-forwarded to the estimation method and subtracted for the sensor output. This solution will be complicated and not very flexible for changes in the design of the satellite modules. The final results from such an estimation cannot be used until a late stage of the satellite development, and might be erroneous even then. A more robust estimation method which can filter out noise is therefore preferred.



# Bibliography

- [1] K. Svartveit, "Attitude determination of the NCUBE satellite", Department of Engineering Cybernetics, NTNU, Master Thesis, June 2003.
- [2] S.S. Ose, "Attitude determination for the Norwegian student satellite nCube", Master Thesis, Department of Engineering Cybernetics, NTNU, June 2004.
- [3] J. Rohde, "Kalman filter for attitude determination of student satellite", Master Thesis, Department of Engineering Cybernetics, NTNU, July 2007.
- [4] K. L. Jenssen and K. H. Yabar, "Development, Implementation and Testing of Two Attitude Estimation Methods for Cube Satellites", Master Thesis, Department of Engineering Cybernetics, NTNU, May 2011.
- [5] A. M. Sabatini, "Quaternion-Based Extended Kalman Filter for Determining Orientation by Inertial and Magnetic Sensing", IEEE Transaction on Biomedical Engineering, Vol.53, No. 7, July 2006.
- [6] Mark L. Psiaki, "Attitude-Determination Filtering via Extended Quaternion Estimation", Journal of Guidance, Control and Dynamics, Vol.23, 2nd edition, March-April 2000.
- [7] L. F. Markley and D. Mortari, "Quaternion Attitude Estimation Using Vector Observations", The Journal of the Astronautical Sciences, Vol.48, April-September 2000.
- [8] E. W. Weisstein, "Coordinate system", MathWorld-A Wolfram Web Resource, <http://mathworld.wolfram.com/CoordinateSystem.html>.
- [9] T. I. Fossen, "Handbook of marine craft hydrodynamics and motion control", 2011, John Wiley and Sons, Ltd, Chichester, UK.
- [10] L. Eulero (Leonhard Euler), "Formulae generales pro translatione quacunque corporum rigidorum (General formulas for the translation of arbitrary rigid bodies)", presented to the St. Petersburg Academy on October 9, 1775, and first published in *Novi Commentarii academiae scientiarum Petropolitanae* 20, pp. 189-207, 1776.
- [11] Sir W. R. Hamilton, "On Quaternions; or on a new system of imaginaries in algebra", *Philosophical magazine*, Vol.25, July 1844.

- [12] A. Cayley, "On certain results relating to quaternions", *Philosophical magazine*, Vol.6, February 1845.
- [13] R. Courant and D. Hilbert, "Methods of Mathematical Physics", Vol. 1, 1953, Interscience Publishers, Inc, New York, USA.
- [14] J. E. Mebius, "A matrix-based proof of the quaternion representation theorem for four-dimensional matrices", arXiv General Mathematics 0501249 v.1, January 2005.
- [15] M. Barile, "Conjugate elements", MathWorld-A Wolfram Web Resource, <http://mathworld.wolfram.com/ConjugateElements.html>
- [16] K. Shoemake, "Animating Rotation with Quaternion Curves", *Siggraph*, Vol. 19, No. 3, July, 1985.
- [17] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions", *Journal of the Optical Society of America*, Vol. 4, p. 629-642, April 1987.
- [18] P. J. From, "Offshore robotics, robust and optimal solutions for autonomous operation", Ph.D Thesis, Department of Engineering Cybernetics, NTNU, May 2010.
- [19] J. L. Crassidis, F. L. Markley and Y. Cheng, "Survey of nonlinear attitude estimation methods", *Journal of guidance, control and dynamics*, Vol. 30, No.1, January-February 2007.
- [20] E. Salamin, "Application of quaternions to computation with rotations", Working paper, Stanford AI Lab, Department of Computer Science, Stanford University, USA, 1979.
- [21] G. Wahba, "A least squares estimate of satellite attitude", *SIAM Review*, Vol. 7, No.3, p.409, July 1965.
- [22] S. Tarca, "Performance optimization of symmetric factorization algorithms", Master Thesis, Department of Computer Science, Cornell University, 2010.
- [23] J. Nocedal, S. J. Wright, "Numerical Optimization", Sec. edition, 2007, Springer Science + Business Media LLC, New York, USA.
- [24] N. J. Higham, "Cholesky factorization", *WIREs Comp Stat*, Vol.1, No.2, pp. 251-254, 2009.
- [25] G. H. Golub and C. F. van Loan, "Matrix computations", Third edition, 1996, The John Hopkins University Press, Baltimore, MD.
- [26] H. K Khalil, "Nonlinear Systems", Third edition, Prentice Hall, New Jersey, 2002.
- [27] J. Davis, "Mathematical modeling of Earth's magnetic field", Technical note, Virginia Tech, Blacksburg, VA 24061, May 12, 2004.
- [28] A. J. Zmuda, "The International Geomagnetic Reference Field: Introduction", *Bulletin International Association of Geomagnetism and Aeronomy*, No. 28, pp. 148-152, 1971.



- [29] C. D. Hall, "Spacecraft attitude dynamics and control", Lecture notes, Department of Aerospace and Ocean Engineering, Virginia Tech, March 2003.
- [30] P. Appel, "Attitude estimation from magnetometer and Earth-albedo-corrected coarse sun sensor measurement", *Acta Astronautica*, IAA, 2004.
- [31] B. O. Sunde, "Sensor modeling and attitude determination for micro-satellite", Master Thesis, Department of Engineering Cybernetics, NTNU, June 2005.
- [32] R. A. Langel, "The main geomagnetic field", *Geomagnetism*, Chap. 4, pp. 249-512. Academic Press, 1987.
- [33] Y. Cheng and M. D. Schuster, "An improvement to the QUEST algorithm", *The Journal of Astronautical Sciences*, JAS-1294, 2008.
- [34] M. D. Shuster and S. D. Oh, "Three-axis attitude determination from vector observations", *Journal of Guidance and Control Dynamics*, Vol. 4, pp. 70-77, January-February, 1981.
- [35] J. Keat, "Analysis of least-squares attitude determination Routine DOAOP", Computer Sciences Corporation, CSM/TM-77/6034, February 1977.
- [36] G. M. Lerner, "Three-Axis Attitude Determination", *Spacecraft Attitude Determination and Control*, ed. by James R. Wertz, Holland, 1978.
- [37] A. M. Sabatini, "Quaternion based attitude estimation algorithm applied to signals from body-mounted gyroscopes", *Electronic Letters*, Vol. 40, No. 10, May 2004.
- [38] M. P. Johnson, "Exploiting Quaternions to Support Expressive Interactive Character Motion", Ph.D. Thesis, Department of Media arts and Sciences, Massachusetts Institute of Technology, 1995.
- [39] R. Benjemaa and F. Schmidt, "A solution for the registration of multiple 3D point sets using unit quaternions", *European Conference on Computer Vision*, Vol. 2, p.34-50, 1998.
- [40] T. D. Howell and J. C. Lafon, "The complexity of the quaternion product", Technical report TR 75-245, Department of Computer Science, Cornell University, New York, June 1975.
- [41] R. Mahony, T. Hamel and J.M. Pfimlin, "Nonlinear complementary filters on the special orthogonal group", *IEEE Transactiona on Automatic Control*, Vol. 53, No. 5, pp. 1203-1218, 2008.
- [42] H. F. Grip, T. I. Fossen, T. A. Johansen and A. Saberi, "Attitude estimation using biased gyro and vector measurements with time-varying reference vectors", *IEEE transactions on automatic control*, Vol.7, No. 5, pp.1332-1338, May 2012.
- [43] M. Krstić, I. Kanellakopoulos and P. V. Kokotović, "Nonlinear and adaptive control design", John Wiley and Sons, Inc, New York, 1995.

- [44] H. Nøkland, "Nonlinear observer design for GNSS and IMU integration", Master thesis, Department of Engineering Cybernetics, NTNU, June 2011.
- [45] B. Vik, "Integrated satellite and inertial systems", Lecture Compendium, Department of Engineering Cybernetics, NTNU, 2009.

## Appendix A

# Schmidt quasi-normalized Legendre function

The magnetic field equations uses the Schmidt quasi-normalized Legendre functions, which are defined according to [27] as:

$$P_n^m = \left[ \frac{2(n-m)!}{(n+m)!} \right]^{1/2} P_{n,m} \quad (\text{A.1})$$

where the associated Legendre polynomial without normalization is given from

$$P_{n,m}(v) = (1-v^2)^{1/2m} \frac{d^m}{dv^m} (P_n(v)) \quad (\text{A.2})$$

The cosine of the co-latitude for the magnetic field calculations is denoted by  $v$ , and the regular Legendre polynomial  $P_n(v)$  can be written as:

$$P_n(v) = \frac{1}{2^n n!} \left( \frac{d}{dv} \right)^n (v^2 - 1)^n \quad (\text{A.3})$$

The derivatives of the associated Legendre functions must be calculated, since the derivative of the Schmidt quasi-normalized Legendre function is used in the calculation of the magnetic field strength for the co-latitude  $\mu_c$ . The derivation can be done recursively by using the following three equations:

$$\frac{\partial P^{0,0}}{\partial \mu_c} = 0 \quad (\text{A.4})$$

$$\frac{\partial P^{n,n}}{\partial \mu_c} = \sin \mu_c \frac{\partial P^{n-1,n-1}}{\partial \mu_c} + \cos \mu_c P^{n-1,n-1}, n \geq 1 \quad (\text{A.5})$$

$$\frac{\partial P^{n,m}}{\partial \mu_c} = \cos \mu_c \frac{\partial P^{n-1,m}}{\partial \mu_c} - \sin \mu_c P^{n-1,m} - K^{n,m} \frac{\partial P^{n-2,m}}{\partial \mu_c} \quad (\text{A.6})$$



## **Appendix B**

# **European CubeSat Symposium abstract and presentation**

This abstract was submitted to the European CubeSat Symposium in Brussels 30 January-1 February 2012. The abstract was accepted, and a 15 min oral presentation was prepared for the session: Attitude determination and control. More information can be found on the website: <https://www.vki.ac.be/CubeSatWorkshop/index.php>

## Development on the EQUEST method for attitude determination

*Toril Bye Rinnan, Roger Birkeland, Jan Tommy Gravdahl*

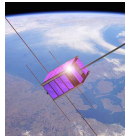
Department of Engineering Cybernetics  
Norwegian University of Science and Technology (NTNU)  
N-7491 Trondheim, Norway

A new extended quaternion estimator (EQUEST) method is being developed for the Norwegian University Test Satellite (NUTS). The attitude is calculated by minimization of a cost function relating sensor measurements with known references. The method is intended for use in the ADCS of a CubeSat.

The satellite is being built in a student CubeSat project at the Norwegian University of Science and Technology. The project was started in September 2010 and is part of the Norwegian student satellite program run by NAROM (Norwegian Centre for Space-related Education). The NUTS project goals are to design, manufacture and launch a double CubeSat by 2014. As payload, an IR-camera observing waves in the air-glow layer is planned, as well as a short-range RF experiment. The satellite will fly two transceivers in the amateur radio bands. Final year master students from several departments are the main contributors in the project.

The work is based on a quaternion estimator (QUEST) method which is extended to include non-vectorized measurements, such as gyroscope data. The main idea behind this extension is to modify the cost function by including terms for the gyroscope measurements and attitude prediction. An early version of the algorithm used quaternion subtraction for evaluating error in attitude in the cost function. This cost function has been developed further by changing the quaternion subtraction terms into quaternion products. When multiplied, the quaternions will represent proper rotations, and form new attitude quaternions. Ways of minimizing the new cost function are analyzed. Because of the limitations of CubeSats, the electronic components need to be optimized in regard to size, financial budgets and power use. This in turn influence which estimation method to use. The quaternion product is more computational expensive than the quaternion subtraction, but the former leads to a more accurate mathematical model. For the intended hardware and sampling time for the satellite, the added number of software operations for the quaternion product are not expected to be a problem.

## Development on the EQUEST method for attitude determination



Toril Bye Rinnan  
Norwegian University of Science and Technology

## Introduction

### The NTNU Test Satellite (NUTS)

- Double cubesat
- Launch planned in 2014
- 10-15 master students at NTNU
- Infrared camera payload for observation of gravity waves

### Outline:

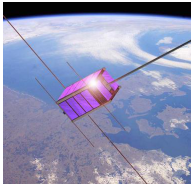
- Quaternions
- QUEST and EQUEST
- Developed EQUEST
- Simulations
- EKF vs EQUEST

Toril Bye Rinnan, torilbye@stud.ntnu.no



<http://nuts.iet.ntnu.no>

## Attitude determination



### Attitude:

- Orientation control
- Determine attitude

### Estimation methods:

- Kalman filter
- Quaternion estimator
- Developed extended quaternion estimator

## Quaternions



### Quaternions:

- Avoid singularities
- Represent rotations
- A quaternion product gives a new attitude quaternion

Toril Bye Rinnan, torilbye@stud.ntnu.no



<http://nuts.iet.ntnu.no>

Toril Bye Rinnan, torilbye@stud.ntnu.no



<http://nuts.iet.ntnu.no>

QUEST- quaternion estimation

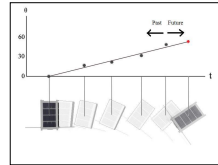
QUEST method cost function:

$$J(q) = \frac{1}{2} \sum_{j=1}^n \frac{1}{\sigma_j^2} (b_j - R_b^i(q)r_j)^T (b_j - R_b^i(q)r_j)$$

Parameters:

- $b$  - measured direction values in spacecraft coordinates
- $r$  - known direction values in inertial coordinates
- $\sigma$  - standard deviation of measurement error

EQUEST- extended quaternion estimation



Motivation:

- Include terms for gyroscope measurements and linear attitude prediction

EQUEST- extended quaternion estimation

Extended QUEST method cost function:

$$J(q) = \frac{1}{2} \sum_{j=1}^n \left\{ \frac{1}{\sigma_j^2} (b_j - R_b^i(q)r_j)^T (b_j - R_b^i(q)r_j) \right\} + \frac{1}{2} (q - \hat{q}_{gyro})^T D (q - \hat{q}_{gyro}) + \frac{1}{2} (q - \hat{q}_{pre})^T S (q - \hat{q}_{pre})$$

Developed EQUEST

Developed EQUEST method cost function:

$$J(q) = \frac{1}{2} \sum_{j=1}^n \left\{ \frac{1}{\sigma_j^2} (b_j - R_b^i(q)r_j)^T (b_j - R_b^i(q)r_j) \right\} + \frac{1}{2} (\hat{q}_{gyro}^* \otimes q)^T D (\hat{q}_{gyro}^* \otimes q) + \frac{1}{2} (\hat{q}_{pre}^* \otimes q)^T S (\hat{q}_{pre}^* \otimes q)$$

- Subtraction terms replaced with quaternion products



### Developed EQEST

Developed EQEST method cost function:

$$J(q) = \frac{1}{2} \sum_{j=1}^n \left\{ \frac{1}{\sigma_j^2} (b_j - R_b^i(q)r_j)^T (b_j - R_b^i(q)r_j) \right\} + \frac{1}{2} (\widehat{Q}_{gyro}^* q)^T D (\widehat{Q}_{gyro}^* q) + \frac{1}{2} (\widehat{Q}_{pre}^* q)^T S (\widehat{Q}_{pre}^* q)$$

- Quaternions replaced with matrix representation

### Developed EQEST

Developed EQEST method cost function:

$$J(q) = \frac{1}{2} \sum_{j=1}^n \left\{ \frac{1}{\sigma_j^2} (b_j - R_b^i(q)r_j)^T (b_j - R_b^i(q)r_j) \right\} + \frac{1}{2} q^T (\widehat{Q}_{gyro}^{*T} D \widehat{Q}_{gyro}^*) q + \frac{1}{2} q^T (\widehat{Q}_{pre}^{*T} S \widehat{Q}_{pre}^*) q$$

- Rearranging of terms and matrix multiplication

### Developed EQEST

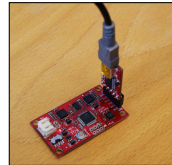
Developed EQEST method cost function:

$$J(q) = \frac{1}{2} \sum_{j=1}^n \left\{ \frac{1}{\sigma_j^2} (b_j - R_b^i(q)r_j)^T (b_j - R_b^i(q)r_j) \right\} + \frac{1}{2} q^T N_d q + \frac{1}{2} q^T N_s q$$

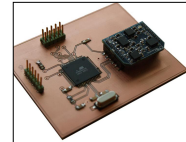
$$N_d = \widehat{Q}_{gyro}^{*T} D \widehat{Q}_{gyro}^*$$

$$N_s = \widehat{Q}_{pre}^{*T} S \widehat{Q}_{pre}^*$$

### Implementation

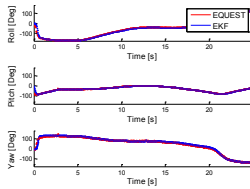


9-DOF IMU

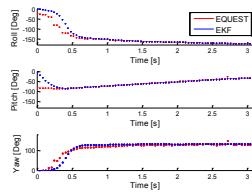


Prototype

EKF vs developed EQUEST



EKF vs developed EQUEST



EKF vs developed EQUEST

**EKF:**


- Well known
- Estimates sensor bias
- Good filtering effect
- Requires about 4000 arithmetic operations

**Developed EQUEST:**

- Finds solution in one operation
- Requires about 3200 arithmetic operations
- About 5 times faster than implemented EKF
- More intuitive tuning parameters

QUESTIONS?

**Sponsors:**

 Kongsberg Seatex, supporting the trip to this conference



# Appendix C

## Code

### C.1 Code for the developed EQUEST method

The following code is an extension of the code made by Jenssen and Yabar [4]. The file `optimal_lambda` has been eliminated by using the function `eig` for calculating the eigenvalues of the developed EQUEST method cost function. Computing the matrix  $\mathbf{G}$  from chapter 4.3, denoted by `KK` in the code has been modified. The extended Kalman filter code is mainly unchanged. The code is also included on a CD.

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %   Toril Bye Rinnan
3  %   Engineering Cybernetics - June 2012
4  %
5  %   This code is based on the implementation for the
6  %   original EQUEST method made by Yabar & Jenssen in April 2011.
7  %
8  %   Further development of the algorithm is implemented, with the
9  %   use of multiplications instead of subtractions in the
10 %   method's cost function.
11 %
12 %   This file runs the new developed EQUEST method and the EKF
13 %   on saved data from the sparkfun IMU
14 %   "data_read2.m" is required to run the file.
15 %   This files draws the Euler angles for the estimated attitude
16 %   both for the EKF and the new developed EQUEST method
17 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
18
19 data_read2
20
21 %Local magnetic field for Trondheim
22 magnetometer=[13598.5;444.7;-49854.8];
23
24 %Normalized reference vector magnetometer
25 magnetometer=magnetometer/norm(magnetometer);
26 m1 = magnetometer(1);
27 m2 = magnetometer(2);
28 m3 = magnetometer(3);
29
```

```

30 %Normalized gravity vector
31 a=[0 0 1];
32 a1=a(1);
33 a2=a(2);
34 a3=a(3);
35
36 %Initialization
37 q=[0,0,0,1]';
38 x = zeros(10,1);
39 x(1:4,1) = [0,0,1,0]';
40 qq=[0,0,1,0]';
41 P = 0*eye(10);
42 go = true;
43 count=1;
44 t=0;
45 euler_ang1=[0,0,0]';
46 euler_ang2=[0,0,0]';
47 Ts = 0.0011;
48
49 %Covariance matrix for process and sensors
50 r1 = 0.05;
51 r2 = 0.08;
52 R = diag([r1,r1,r1,r2,r2,r2]);
53
54 % %Estimate only for each 4. step
55 for i=1:4:2483
56
57     %Normalized vector measurements read from txt- file
58     acc = dataM(3:5,i);
59     acc = acc/norm(acc);
60
61     mag = dataM(9:11,i);
62     mag = mag/norm(mag);
63
64     omega_real = dataM(6:8,i)-[260 257 260]';
65     omega_real = omega_real*0.92*(pi/180);
66
67     %EKF
68     z = [acc;mag];
69     omega_skew=[0 -omega_real(3) omega_real(2);
70               omega_real(3) 0 -omega_real(1);
71               -omega_real(2) omega_real(1) 0];
72     Omega = 0.5 * [omega_skew omega_real; -omega_real' 0];
73
74     A = [expm(Omega*Ts) zeros(4,3) zeros(4,3);
75         zeros(3,4) eye(3) zeros(3,3);
76         zeros(3,4) zeros(3,3) eye(3)];
77
78     %Extended Kalman Filter Equations
79     %Time update eq
80     x = A*x;
81     x(1:4)=x(1:4)/norm(x(1:4));
82
83     q1 = x(1);
84     q2 = x(2);
85     q3 = x(3);
86     q4 = x(4);

```

```

87     q_13=[0 -q3 q2;q3 0 -q1;-q2 q1 0];
88     E=[q4*eye(3)+q_13;-q1 -q2 -q3];
89     Y=[q4*eye(3)-q_13;-q1 -q2 -q3];
90     Rot=E'*Y;
91
92     %Linearization of measurment matrix
93     num = (q1^2+q2^2+q3^2+q4^2)^(3/2);
94
95     daxdq1 = a1*q1^3+3*a1*q1*q2^2+3*a1*q1*q3^2+a1*q1*q4^2-
96             2*a2*q1*q3*q4+2*a2*q2^3+2*a2*q2*q3^2+2*a2*q2*q4^2+
97             2*a3*q1*q2*q4+2*a3*q2^2*q3+2*a3*q3^3+2*a3*q3*q4^2;
98
99     daxdq2 = -3*a1*q1^2*q2-a1*q2^3-a1*q2*q3^2-3*a1*q2*q4^2+
100            2*a2*q1^3+2*a2*q1*q3^2+2*a2*q1*q4^2-2*a2*q2*q3*q4-
101            2*a3*q1^2*q4-2*a3*q1*q2*q3-2*a3*q3^2*q4-2*a3*q4^3;
102
103     daxdq3 = -3*a1*q1^2*q3-a1*q2^2*q3-a1*q3^3-3*a1*q3*q4^2+
104            2*a2*q1^2*q4-2*a2*q1*q2*q3+2*a2*q2^2*q4+2*a2*q4^3+
105            2*a3*q1^3+2*a3*q1*q2^2+2*a3*q1*q4^2+2*a3*q2*q3*q4;
106
107     daxdq4 = -(-a1*q1^2*q4-3*a1*q2^2*q4-3*a1*q3^2*q4-
108            a1*q4^3-2*a2*q1^2*q3+2*a2*q1*q2*q4-2*a2*q2^2*q3-
109            2*a2*q3^3+2*a3*q1^2*q2+2*a3*q1*q3*q4+2*a3*q2^3+
110            2*a3*q2*q3^2);
111
112     daydq1 = 2*(a1*(q1*q3*q4+q2^3+q2*(q3^2+q4^2)) +
113            a3*(q4*(q2^2+q3^2+q4^2)-q1*q2*q3)) -
114            a2*q1*(q1^2+3*q2^2+q3^2+3*q4^2);
115
116     daydq2 = 2*a1*q1^3+2*a1*q1*q3^2+2*a1*q1*q4^2+
117            2*a1*q2*q3*q4+3*a2*q1^2*q2+a2*q2^3+3*a2*q2*q3^2+
118            a2*q2*q4^2+2*a3*q1^2*q3-2*a3*q1*q2*q4+2*a3*q3^3+
119            2*a3*q3*q4^2;
120
121     daydq3 = -2*a1*q1^2*q4-2*a1*q1*q2*q3-2*a1*q2^2*q4-
122            2*a1*q4^3-a2*q1^2*q3-3*a2*q2^2*q3-a2*q3^3-3*a2*q3*q4^2+
123            2*a3*q1^2*q2-2*a3*q1*q3*q4+2*a3*q2^3+2*a3*q2*q4^2;
124
125     daydq4 = -2*a1*q1^2*q3-2*a1*q1*q2*q4-2*a1*q2^2*q3-
126            2*a1*q3^3+3*a2*q1^2*q4+2*a2*q2^2*q4+3*a2*q3^2*q4+
127            a2*q4^3+2*a3*q1^3+2*a3*q1*q2^2+2*a3*q1*q3^2-2*a3*q2*q3*q4;
128
129     dazdq1 = -(2*a1*q1*q2*q4-2*a1*q2^2*q3-2*a1*q3^3-
130            2*a1*q3*q4^2+2*a2*q1*q2*q3+2*a2*q2^2*q4+2*a2*q3^2*q4+
131            2*a2*q4^3+a3*q1^3+a3*q1*q2^2+3*a3*q1*q3^2+3*a3*q1*q4^2);
132
133     dazdq2 = 2*(a1*(q1^2*q4-q1*q2*q3+q3^2*q4+q4^3) +
134            a2*(q1^2*q3+q1*q2*q4+q3^3+q3*q4^2)) -
135            a3*q2*(q1^2+q2^2+3*(q3^2+q4^2));
136
137     dazdq3 = 2*a1*q1^3+2*a1*q1*q2^2+2*a1*q1*q4^2-
138            2*a1*q2*q3*q4+2*a2*q1^2*q2+2*a2*q1*q3*q4+2*a2*q2^3+
139            2*a2*q2*q4^2+3*a3*q1^2*q3+3*a3*q2^2*q3+3*a3*q3^3+3*a3*q3*q4^2;
140
141     dazdq4 = -(-2*a1*q1^2*q2+2*a1*q1*q3*q4-2*a1*q2^3-
142            2*a1*q2*q3^2+2*a2*q1^3+2*a2*q1*q2^2+2*a2*q1*q3^2+
143            2*a2*q2*q3*q4-3*a3*q1^2*q4-3*a3*q2^2*q4-a3*q3^2*q4-a3*q4^3);

```

```

144
145      dmxdq1 = m1*q1^3+m1*q1*q2^2+3*m1*q1*q3^2+m1*q1*q4^2-
146      2*m2*q1*q3*q4+2*m2*q2^3+2*m2*q2*q3^2+2*m2*q2*q4^2+
147      2*m3*q1*q2*q4+2*m3*q2^2*q3+2*m3*q3^3+2*m3*q3*q4^2;
148
149      dmxdq2 = -3*m1*q1^2*q2-m1*q2^3-m1*q2*q3^2-3*m1*q2*q4^2+
150      2*m2*q1^2*q3+2*m2*q1*q3^2+2*m2*q1*q4^2-2*m2*q2*q3*q4-
151      2*m3*q1^2*q4-2*m3*q1*q2*q3-2*m3*q3^2*q4-2*m3*q4^3;
152
153      dmxdq3 = -3*m1*q1^2*q3-m1*q2^2*q3-m1*q3^3-3*m1*q3*q4^2+
154      2*m2*q1^2*q4-2*m2*q1*q2*q3+2*m2*q2^2*q4+2*m2*q4^3+2*m3*q1^3+
155      2*m3*q1*q2^2+2*m3*q1*q4^2+2*m3*q2*q3*q4;
156
157      dmxdq4 = -(m1*q1^2*q4-3*m1*q2^2*q4-3*m1*q3^2*q4-m1*q4^3-
158      2*m2*q1^2*q3+2*m2*q1*q2*q4-2*m2*q2^2*q3-2*m2*q3^3+
159      2*m3*q1^2*q2+2*m3*q1*q3*q4+2*m3*q2^3+2*m3*q2*q3^2);
160
161      dmydq1 = 2*(m1*(q1*q3*q4+q2^3+q2*(q3^2+q4^2))+
162      m3*(q4*(q2^2+q3^2+q4^2)-q1*q2*q3))-
163      m2*q1*(q1^2+3*q2^2+q3^2+3*q4^2);
164
165      dmydq2 = 2*m1*q1^3+2*m1*q1*q3^2+2*m1*q1*q4^2+
166      2*m1*q2*q3*q4+3*m2*q1^2*q2+m2*q2^3+3*m2*q2*q3^2+
167      2*m2*q2*q4^2+2*m3*q1^2*q3-2*m3*q1*q2*q4+2*m3*q3^3+
168      2*m3*q3*q4^2;
169
170      dmydq3 = -2*m1*q1^2*q4-2*m1*q1*q2*q3-2*m1*q2^2*q4-
171      2*m1*q4^3-2*m2*q1^2*q3-3*m2*q2^2*q3-2*m2*q3^3-3*m2*q3*q4^2+
172      2*m3*q1^2*q2-2*m3*q1*q3*q4+2*m3*q2^3+2*m3*q2*q4^2;
173
174      dmydq4 = -2*m1*q1^2*q3-2*m1*q1*q2*q4-2*m1*q2^2*q3-
175      2*m1*q3^3+3*m2*q1^2*q4+2*m2*q2^2*q4+3*m2*q3^2*q4+m2*q4^3+
176      2*m3*q1^3+2*m3*q1*q2^2+2*m3*q1*q3^2-2*m3*q2*q3*q4;
177
178      dmzdq1 = -(2*m1*q1*q2*q4-2*m1*q2^2*q3-2*m1*q3^3-
179      2*m1*q3*q4^2+2*m2*q1*q2*q3+2*m2*q2^2*q4+2*m2*q3^2*q4+
180      2*m2*q4^3+m3*q1^3+m3*q1*q2^2+3*m3*q1*q3^2+3*m3*q1*q4^2);
181
182      dmzdq2 = 2*(m1*(q1^2*q4-q1*q2*q3+q3^2*q4+q4^3)+
183      m2*(q1^2*q3+q1*q2*q4+q3^3+q3*q4^2))-
184      m3*q2*(q1^2+q2^2+3*(q3^2+q4^2));
185
186      dmzdq3 = 2*m1*q1^3+2*m1*q1*q2^2+2*m1*q1*q4^2-
187      2*m1*q2*q3*q4+2*m2*q1^2*q2+2*m2*q1*q3*q4+2*m2*q2^3+
188      2*m2*q2*q4^2+3*m3*q1^2*q3+3*m3*q2^2*q3+m3*q3^3+m3*q3*q4^2;
189
190      dmzdq4 = -(2*m1*q1^2*q2+2*m1*q1*q3*q4-2*m1*q2^3-
191      2*m1*q2*q3*q4+2*m2*q1^3+2*m2*q1*q2^2+2*m2*q1*q3^2+
192      2*m2*q2*q3*q4-3*m3*q1^2*q4-3*m3*q2^2*q4-m3*q3^2*q4-m3*q4^3);
193
194
195      H = 1/num*[daxdq1 daxdq2 daxdq3 daxdq4 num 0 0 0 0 0;
196      daydq1 daydq2 daydq3 daydq4 0 num 0 0 0 0;
197      dazdq1 dazdq2 dazdq3 dazdq4 0 0 num 0 0 0;
198      dmxdq1 dmxdq2 dmxdq3 dmxdq4 0 0 0 num 0 0;
199      dmydq1 dmydq2 dmydq3 dmydq4 0 0 0 0 num 0;
200      dmzdq1 dmzdq2 dmzdq3 dmzdq4 0 0 0 0 0 num];

```

```

201
202     %(Ts/2)^2+E*10*eye(3)*E'
203     Q = [0.001*eye(4) zeros(4,3) zeros(4,3);
204         zeros(3,4) 0.00000001*eye(3) zeros(3,3);
205         zeros(3,4) zeros(3,3) eye(3)*0.00000001];
206
207     P = A*P*A'+Q;
208
209     %Measurement update eq
210     %Iteratively updating of the estimated quaternion
211     K = P*H'*inv(H*P*H'+R);
212     x = x+K*(z-[Rot*[0 0 1]';Rot*magnetometer]-x(5:10));
213     P = (eye(size(K*H))-K*H)*P;
214
215
216     %Converting from quaternions to
217     %euler angles(pitch,roll,yaw)
218     qq = x(1:4)/norm(x(1:4));
219     euler_ang1=qua_to_euler(qq);
220
221     %Developed EQUEST
222     q_13=[0 -q(3) q(2);q(3) 0 -q(1);-q(2) q(1) 0];
223     E=[q(4)*eye(3)+q_13;-q(1) -q(2) -q(3)];
224     Δ_q=0.5*E*omega_real;
225     q=q+Δ_q*0.01;
226     q=q/norm(q);
227
228     B=(1/0.001)*acc*[0 0 1] + (1/0.001)*mag*magnetometer';
229     %(1/0.05)*(mag/norm(mag))*magnetometer
230     S=B+B';
231     Z=[B(2,3)-B(3,2);B(3,1)-B(1,3);B(1,2)-B(2,1)];
232     o=trace(B);
233     V=[S-o*eye(3) Z;Z' o];
234     %Q'*N*Q
235     Q = [q(4) -q(1) -q(2) -q(3); q(1) q(4) -q(3) q(2);
236         q(2) q(3) q(4) -q(1); q(3) -q(2) q(1) q(4)];
237     N = Q'*eye(4)*Q*100;
238     %KK = -V + N
239     KK = -V+N;
240
241     %Using eigenvalues to find optimal q
242     I = eye(4);
243     [v,d] = eig(KK, I, 'chol');
244     y = v(:,1);
245
246     %Converting from quaternions to euler
247     %angles(pitch,roll,yaw)
248     q=y(1:4);
249     q = q/norm(q);
250     euler_ang2 = qua_to_euler(q);
251
252
253     %Plot for Euler angles
254     subplot(3,1,1)
255     hold on;
256     plot(t,euler_ang2(1)*180/pi,'color','red');
257     plot(t,euler_ang1(1)*180/pi,'color','blue');

```

```
258     axis([0 inf -180 180]);
259     h_legend = legend('dev. EQUEST', 'EKF');
260     xlabel('Time [s]', 'FontSize', 14);
261     ylabel('Roll [Deg]', 'FontSize', 14);
262     set(h_legend, 'FontSize', 12);
263
264     subplot(3,1,2)
265     hold on;
266     plot(t, euler_ang2(2)*180/pi, 'color', 'red');
267     plot(t, euler_ang1(2)*180/pi, 'color', 'blue');
268     axis([0 inf -180 180]);
269     xlabel('Time [s]', 'FontSize', 14);
270     ylabel('Pitch [Deg]', 'FontSize', 14);
271
272     subplot(3,1,3)
273     hold on;
274     plot(t, euler_ang2(3)*180/pi, 'LineWidth', 1, 'color', 'red');
275     plot(t, euler_ang1(3)*180/pi, 'LineWidth', 1, 'color', 'blue');
276     axis([0 inf -180 180]);
277     xlabel('Time [s]', 'FontSize', 14);
278     ylabel('Yaw [Deg]', 'FontSize', 14);
279
280     %Time update
281     t=t+0.04;
282
283     end
```



## C.2 Code for the nonlinear observer

New code for a nonlinear observer has been implemented in addition to the modifications of the EQUEST code. A new data set is implemented, which makes it easier to model different vector measurement responses, and compare the simulation results. The code for the nonlinear observer with the developed EQUEST method and the EKF is included below. The code is also included on a CD. The code for a combination of the nonlinear observer and the developed EQUEST method can also be tested by accessing the files on this CD.

```

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %   Toril Bye Rinnan
3  %   Engineering Cybernetics - June 2012
4  %
5  %   A nonlinear observer is implemented, along with
6  %   the developed EQUEST method and the EKF.
7  %   This files draws the Euler angles for the estimated
8  %   attitude for all three methods and the sensor data
9  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
10
11 %Local magnetic field for Trondheim
12 magnetometer=[13598.5;444.7;-49854.8];
13
14 %Normalized reference vector magnetometer
15 magnetometer=magnetometer/norm(magnetometer);
16 m1 = magnetometer(1);
17 m2 = magnetometer(2);
18 m3 = magnetometer(3);
19
20 %Normalized gravity vector
21 accelerometer=[0 0 1]';
22 a1=a(1);
23 a2=a(2);
24 a3=a(3);
25
26
27 %Initialization
28 qr = [0,0,0,1]';
29
30 x = zeros(10,1);
31 x(1:4,1) = [0 0 1 0]';
32 Ts = 0.0011;
33 go = true;
34 count=1;
35
36 q=[0,0,1,0]';
37 qq=[0,0,1,0]';
38 q_hat =[0,0,1,0]';
39 b_g = [0.0001 0.0001 0.0001]';
40 b_g_hat = [0,0,0]';
41
42 euler_ang1=[0,0,0]';
43 euler_ang2=[0,0,0]';
44 euler_ang3=[0,0,0]';

```

```

45 euler_ang4=[0,0,0]';
46
47 k = [1 1];
48 w = 0.1*rand(1);
49 simdata = zeros(2483,13);
50
51 %Covariance matrix for process and sensors
52 r1 = 0.05;
53 r2 = 0.08;
54 R = diag([r1,r1,r1,r2,r2,r2]);
55 P = 0*eye(10);
56
57 %Sampling time
58 dt = 0.01;
59 t=0;
60 N = 2484;
61
62 %%Estimate only for each 4. step
63 for i=1:1:N-1
64
65     %Gyroscope measurements
66     omega = [0.1*sin(0.01*t) 0.1*sin(0.01*t) 0.1*sin(0.01*t)]';
67     omega_real = omega + b_g;
68     omega_real = omega_real*180/pi;
69
70     q_vecr = [qr(1); qr(2); qr(3)];
71     S=[0 -qr(3) qr(2);qr(3) 0 -qr(1);-qr(2) qr(1) 0];
72     Rr = eye(3) + 2*qr(4)*S + 2*(S*S);
73
74     %Normalized vector measurements
75     mag = Rr'*magnetometer;
76     acc = Rr'*accelerometer;
77     mag = mag/norm(mag);
78     acc = acc/norm(acc);
79
80     %Attitude dynamics
81     T = 0.5*[-q_vecr'; (qr(4)*eye(3) + S)];
82
83     qr = qr + dt*T*(omega_real-b_g);
84     qr = qr/norm(qr);
85
86     euler_ang1 = qua_to_euler(qr);
87
88     %Observer gains
89     k_i = 0.1;
90     k = [0.5; 0.5];
91     q_vec_hat = [q_hat(1); q_hat(2); q_hat(3)];
92     S_hat = [0 -q_hat(3) q_hat(2);q_hat(3) 0 -q_hat(1);
93             -q_hat(2) q_hat(1) 0];
94     R_hat = eye(3) + 2*q_hat(4)*S_hat + 2*(S_hat*S_hat);
95
96     v_hat_1= R_hat'*accelerometer;
97     v_hat_2= R_hat'*magnetometer;
98     v_1 = acc;
99     v_2 = mag;
100
101     %Injection term

```

```

102     sum1 = 0.5*k(1)*((v_1*v_hat_1')-(v_hat_1*v_1'));
103     sum2 = 0.5*k(2)*((v_2*v_hat_2')-(v_hat_2*v_2'));
104     sum = sum1 + sum2;
105
106     sigma = -[sum(3,2) sum(1,3) sum(2,1)]';
107     %Observer equations
108
109     T_hat = 0.5*[-q_vec_hat'; q_hat(4)*eye(3) + S_hat];
110     q_hat = q_hat + dt*T_hat*(omega_real - b_g_hat + sigma);
111     b_g_hat = proj(b_g_hat, (-k_i*sigma));
112
113     q_hat = q_hat/norm(q_hat);
114     euler_ang2 = qua_to_euler(q_hat);
115
116     %EKF
117     z = [acc;mag];
118     omega_skew=[0 -omega_real(3) omega_real(2);
119                omega_real(3) 0 -omega_real(1);
120                -omega_real(2) omega_real(1) 0];
121     Omega = 0.5 * [omega_skew omega_real; -omega_real' 0];
122
123     A = [expm(Omega*Ts) zeros(4,3) zeros(4,3);
124          zeros(3,4) eye(3) zeros(3,3);
125          zeros(3,4) zeros(3,3) eye(3)];
126
127     %Extended Kalman Filter Equations
128
129     %Time update eq
130     x = A*x;
131     x(1:4)=x(1:4)/norm(x(1:4));
132
133     q1 = x(1);
134     q2 = x(2);
135     q3 = x(3);
136     q4 = x(4);
137     q_13=[0 -q3 q2;q3 0 -q1;-q2 q1 0];
138     E=[q4*eye(3)+q_13;-q1 -q2 -q3];
139     Y=[q4*eye(3)-q_13;-q1 -q2 -q3];
140     Rot=E'*Y;
141
142     %Linearization of measurment matrix
143     num = (q1^2+q2^2+q3^2+q4^2)^(3/2);
144
145     daxdq1 = a1*q1^3+3*a1*q1*q2^2+3*a1*q1*q3^2+a1*q1*q4^2-
146             2*a2*q1*q3*q4+2*a2*q2^3+2*a2*q2*q3^2+2*a2*q2*q4^2+
147             2*a3*q1*q2*q4+2*a3*q2^2*q3+2*a3*q3^3+2*a3*q3*q4^2;
148
149     daxdq2 = -3*a1*q1^2*q2-a1*q2^3-a1*q2*q3^2-3*a1*q2*q4^2+
150             2*a2*q1^3+2*a2*q1*q3^2+2*a2*q1*q4^2-2*a2*q2*q3*q4-
151             2*a3*q1^2*q4-2*a3*q1*q2*q3-2*a3*q3^2*q4-2*a3*q4^3;
152
153     daxdq3 = -3*a1*q1^2*q3-a1*q2^2*q3-a1*q3^3-3*a1*q3*q4^2+
154             2*a2*q1^2*q4-2*a2*q1*q2*q3+2*a2*q2^2*q4+2*a2*q4^3+
155             2*a3*q1^3+2*a3*q1*q2^2+2*a3*q1*q4^2+2*a3*q2*q3*q4;
156
157     daxdq4 = -(-a1*q1^2*q4-3*a1*q2^2*q4-3*a1*q3^2*q4-
158             a1*q4^3-2*a2*q1^2*q3+2*a2*q1*q2*q4-2*a2*q2^2*q3-

```

```

159      2*a2*q3^3+2*a3*q1^2*q2+2*a3*q1*q3*q4+2*a3*q2^3+
160      2*a3*q2*q3^2);
161
162      daydq1 = 2*(a1*(q1*q3*q4+q2^3+q2*(q3^2+q4^2)) +
163      a3*(q4*(q2^2+q3^2+q4^2)-q1*q2*q3)) -
164      a2*q1*(q1^2+3*q2^2+q3^2+3*q4^2);
165
166      daydq2 = 2*a1*q1^3+2*a1*q1*q3^2+2*a1*q1*q4^2+
167      2*a1*q2*q3*q4+3*a2*q1^2*q2+a2*q2^3+3*a2*q2*q3^2+
168      a2*q2*q4^2+2*a3*q1^2*q3-2*a3*q1*q2*q4+2*a3*q3^3+
169      2*a3*q3*q4^2;
170
171      daydq3 = -2*a1*q1^2*q4-2*a1*q1*q2*q3-2*a1*q2^2*q4-
172      2*a1*q4^3-a2*q1^2*q3-3*a2*q2^2*q3-a2*q3^3-3*a2*q3*q4^2+
173      2*a3*q1^2*q2-2*a3*q1*q3*q4+2*a3*q2^3+2*a3*q2*q4^2;
174
175      daydq4 = -2*a1*q1^2*q3-2*a1*q1*q2*q4-2*a1*q2^2*q3-
176      2*a1*q3^3+3*a2*q1^2*q4+2*a2*q2^2*q4+3*a2*q3^2*q4+
177      a2*q4^3+2*a3*q1^3+2*a3*q1*q2^2+2*a3*q1*q3^2-2*a3*q2*q3*q4;
178
179      dazdq1 = -(2*a1*q1*q2*q4-2*a1*q2^2*q3-2*a1*q3^3-
180      2*a1*q3*q4^2+2*a2*q1*q2*q3+2*a2*q2^2*q4+2*a2*q3^2*q4+
181      2*a2*q4^3+a3*q1^3+a3*q1*q2^2+3*a3*q1*q3^2+3*a3*q1*q4^2);
182
183      dazdq2 = 2*(a1*(q1^2*q4-q1*q2*q3+q3^2*q4+q4^3) +
184      a2*(q1^2*q3+q1*q2*q4+q3^3+q3*q4^2)) -
185      a3*q2*(q1^2+q2^2+3*(q3^2+q4^2));
186
187      dazdq3 = 2*a1*q1^3+2*a1*q1*q2^2+2*a1*q1*q4^2-
188      2*a1*q2*q3*q4+2*a2*q1^2*q2+2*a2*q1*q3*q4+2*a2*q2^3+
189      2*a2*q2*q4^2+3*a3*q1^2*q3+3*a3*q2^2*q3+a3*q3^3+a3*q3*q4^2;
190
191      dazdq4 = -(-2*a1*q1^2*q2+2*a1*q1*q3*q4-2*a1*q2^3-
192      2*a1*q2*q3^2+2*a2*q1^3+2*a2*q1*q2^2+2*a2*q1*q3^2+
193      2*a2*q2*q3*q4-3*a3*q1^2*q4-3*a3*q2^2*q4-a3*q3^2*q4-a3*q4^3);
194
195      dmxdq1 = m1*q1^3+3*m1*q1*q2^2+3*m1*q1*q3^2+m1*q1*q4^2-
196      2*m2*q1*q3*q4+2*m2*q2^3+2*m2*q2*q3^2+2*m2*q2*q4^2+
197      2*m3*q1*q2*q4+2*m3*q2^2*q3+2*m3*q3^3+2*m3*q3*q4^2;
198
199      dmxdq2 = -3*m1*q1^2*q2-m1*q2^3-m1*q2*q3^2-3*m1*q2*q4^2+
200      2*m2*q1^3+2*m2*q1*q3^2+2*m2*q1*q4^2-2*m2*q2*q3*q4-
201      2*m3*q1^2*q4-2*m3*q1*q2*q3-2*m3*q3^2*q4-2*m3*q4^3;
202
203      dmxdq3 = -3*m1*q1^2*q3-m1*q2^2*q3-m1*q3^3-3*m1*q3*q4^2+
204      2*m2*q1^2*q4-2*m2*q1*q2*q3+2*m2*q2^2*q4+2*m2*q4^3+2*m3*q1^3+
205      2*m3*q1*q2^2+2*m3*q1*q4^2+2*m3*q2*q3*q4;
206
207      dmxdq4 = -(-m1*q1^2*q4-3*m1*q2^2*q4-3*m1*q3^2*q4-m1*q4^3-
208      2*m2*q1^2*q3+2*m2*q1*q2*q4-2*m2*q2^2*q3-2*m2*q3^3+
209      2*m3*q1^2*q2+2*m3*q1*q3*q4+2*m3*q2^3+2*m3*q2*q3^2);
210
211      dmydq1 = 2*(m1*(q1*q3*q4+q2^3+q2*(q3^2+q4^2)) +
212      m3*(q4*(q2^2+q3^2+q4^2)-q1*q2*q3)) -
213      m2*q1*(q1^2+3*q2^2+q3^2+3*q4^2);
214
215      dmydq2 = 2*m1*q1^3+2*m1*q1*q3^2+2*m1*q1*q4^2+

```

```

216     2*m1*q2*q3*q4+3*m2*q1^2*q2+m2*q2^3+3*m2*q2*q3^2+
217     m2*q2*q4^2+2*m3*q1^2*q3-2*m3*q1*q2*q4+2*m3*q3^3+
218     2*m3*q3*q4^2;
219
220     dmydq3 = -2*m1*q1^2*q4-2*m1*q1*q2*q3-2*m1*q2^2*q4-
221     2*m1*q4^3-m2*q1^2*q3-3*m2*q2^2*q3-m2*q3^3-3*m2*q3*q4^2+
222     2*m3*q1^2*q2-2*m3*q1*q3*q4+2*m3*q2^3+2*m3*q2*q4^2;
223
224     dmydq4 = -2*m1*q1^2*q3-2*m1*q1*q2*q4-2*m1*q2^2*q3-
225     2*m1*q3^3+3*m2*q1^2*q4+2*m2*q2^2*q4+3*m2*q3^2*q4+m2*q4^3+
226     2*m3*q1^3+2*m3*q1*q2^2+2*m3*q1*q3^2-2*m3*q2*q3*q4;
227
228     dmzdq1 = -(2*m1*q1*q2*q4-2*m1*q2^2*q3-2*m1*q3^3-
229     2*m1*q3*q4^2+2*m2*q1*q2*q3+2*m2*q2^2*q4+2*m2*q3^2*q4+
230     2*m2*q4^3+m3*q1^3+m3*q1*q2^2+3*m3*q1*q3^2+3*m3*q1*q4^2);
231
232     dmzdq2 = 2*(m1*(q1^2*q4-q1*q2*q3+q3^2*q4+q4^3)+
233     m2*(q1^2*q3+q1*q2*q4+q3^3+q3*q4^2))-
234     m3*q2*(q1^2+q2^2+3*(q3^2+q4^2));
235
236     dmzdq3 = 2*m1*q1^3+2*m1*q1*q2^2+2*m1*q1*q4^2-
237     2*m1*q2*q3*q4+2*m2*q1^2*q2+2*m2*q1*q3*q4+2*m2*q2^3+
238     2*m2*q2*q4^2+3*m3*q1^2*q3+3*m3*q2^2*q3+m3*q3^3+m3*q3*q4^2;
239
240     dmzdq4 = -(-2*m1*q1^2*q2+2*m1*q1*q3*q4-2*m1*q2^3-
241     2*m1*q2*q3^2+2*m2*q1^3+2*m2*q1*q2^2+2*m2*q1*q3^2+
242     2*m2*q2*q3*q4-3*m3*q1^2*q4-3*m3*q2^2*q4-m3*q3^2*q4-m3*q4^3);
243
244
245     H = 1/num*[daxdq1 daxdq2 daxdq3 daxdq4 num 0 0 0 0 0;
246     daydq1 daydq2 daydq3 daydq4 0 num 0 0 0 0;
247     dazdq1 dazdq2 dazdq3 dazdq4 0 0 num 0 0 0;
248     dmxdq1 dmxdq2 dmxdq3 dmxdq4 0 0 0 num 0 0;
249     dmydq1 dmydq2 dmydq3 dmydq4 0 0 0 0 num 0;
250     dmzdq1 dmzdq2 dmzdq3 dmzdq4 0 0 0 0 0 num];
251
252     %(Ts/2)^2*E*10*eye(3)*E'
253     Q = [0.001*eye(4) zeros(4,3) zeros(4,3);
254     zeros(3,4) 0.00000001*eye(3) zeros(3,3);
255     zeros(3,4) zeros(3,3) eye(3)*0.00000001];
256
257     P = A*P*A'+Q;
258     %Measurement update eq
259     %Iteratively updating of the estimated quaternion
260     K = P*H'*inv(H*P*H'+R);
261     x = x+K*(z-[Rot*[0 0 1]';Rot*magnetometer]-x(5:10));
262     P = (eye(size(K*H))-K*H)*P;
263
264
265     %Converting from quaternions to Euler
266     %angles(pitch,roll,yaw)
267     qq = x(1:4)/norm(x(1:4));
268     euler_ang3=qua_to_euler(qq);
269
270     %developed EQUEST
271     q_13=[0 -q(3) q(2);q(3) 0 -q(1);-q(2) q(1) 0];
272     E=[q(4)*eye(3)+q_13;-q(1) -q(2) -q(3)];

```

```

273     Δq=0.5*E*omega_real;
274     q=q+Δq*0.01;
275     q=q/norm(q);
276
277     B=(1/0.001)*acc*[0 0 1] + (1/0.001)*mag*magnetometer';
278     %(1/0.05)*(mag/norm(mag))*magnetometer
279     S=B+B';
280     Z=[B(2,3)-B(3,2);B(3,1)-B(1,3);B(1,2)-B(2,1)];
281     o=trace(B);
282     V=[S-o*eye(3) Z;Z' o];
283     %Q'*N*Q
284     Q = [q(4) -q(1) -q(2) -q(3);
285          q(1) q(4) -q(3) q(2);
286          q(2) q(3) q(4) -q(1);
287          q(3) -q(2) q(1) q(4)];
288     N = Q'*eye(4)*Q*100;
289     %KK = -V + N
290     KK = -V*20+N;
291
292     %Using eigenvalues to find optimal q
293     I = eye(4);
294     [v,d] = eig(KK, I, 'chol');
295     y = v(:,1);
296
297     %Converting from quaternions to Euler
298     %angles(pitch,roll,yaw)
299     q=y(1:4);
300     q = q/norm(q);
301     euler_ang4 = qua_to_euler(q);
302
303     simdata(i,:) = [t euler_ang4'*180/pi euler_ang3'*180/pi
304                   euler_ang1'*180/pi euler_ang2'*180/pi];
305     t=t+0.04;
306
307     end
308
309     %Plot for Euler angles
310
311     t = simdata(:,1);
312     Equest = simdata(:,2:4);
313     Kalman = simdata(:,5:7);
314     q = simdata(:,8:10);
315     Nonlinear = simdata(:,11:13);
316
317     clf
318     figure(gcf)
319
320     subplot(311),plot(t,q(:,1),'g',t,Equest(:,1),'r',
321                     t,Nonlinear(:,1),'k',t,Kalman(:,1),'b')
322     xlabel('Time (s)', 'FontSize',14),
323     ylabel('Roll', 'FontSize',14),grid
324     h_legend = legend('Sensor data',
325                     'dev. EQUEST', 'Nonlinear observer', 'EKF');
326     axis([0 inf -180 180]);
327     set(h_legend, 'FontSize',12);
328
329

```

```
330 subplot(312),plot(t,q(:,2),'g',t,Equest(:,2),'r',
331                 t,Nonlinear(:,2),'k',t,Kalman(:,2),'b')
332 xlabel('Time (s)', 'FontSize',14),
333 ylabel('Pitch', 'FontSize',14),grid
334 h_legend = legend('Sensor data',
335                 'dev. EQUEST', 'Nonlinear observer', 'EKF');
336 axis([0 inf -180 180]);
337 set(h_legend, 'FontSize',12);
338
339 subplot(313),plot(t,q(:,3),'g',t,Equest(:,3),'r',
340                 t,Nonlinear(:,3),'k',t,Kalman(:,3),'b')
341 xlabel('Time (s)', 'FontSize',14),
342 ylabel('Yaw', 'FontSize',14),grid
343 h_legend = legend('Sensor data',
344                 'dev. EQUEST', 'Nonlinear observer', 'EKF');
345 axis([0 inf -180 180]);
346 set(h_legend, 'FontSize',12);
```

### C.2.1 Code for the projection algorithm used for the nonlinear observer

The nonlinear observer requires a projection algorithm for updating the gyroscope bias estimation. The code for the projection function is included below, and on the handed in CD.

```

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %           Toril Bye Rinnan
3  %           June, 2012
4  %
5  %           Projection algorithm used for the nonlinear observer
6  %
7  %           Input to the fuction is the estimated gyro bias
8  %           and the negative value of k_I multiplied
9  %           with the injection term
10 %
11 %           Output is the next estimated bias value
12 %
13 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
14
15 function [bg] = proj(b_g_hat,neg_ksigma)
16
17 x = (norm(b_g_hat))^2;
18 k = neg_ksigma;
19 wm = 1;
20 if (x > wm)
21     wm = x;
22 end
23
24 P_grad = b_g_hat;
25
26 if (x<wm) || (x==wm && P_grad'*k<0)
27     p = k;
28 else
29     p = k - (k*(P_grad'*P_grad))*((norm(P_grad)^2))^(-1);
30 end
31 bg = p;

```



## **C.3 Contents of CD**

combination.m  
data\_read2.m  
igrf.m  
igrfSg.txt  
igrfSh.txt  
LatLongTabell.txt  
msph2cart.m  
msph2inert.m  
nonlinear.m  
proj.m  
qua\_to\_euler.m  
spark\_Kalman\_Equest\_new.m  
sparkfun.txt