Nonlinear Observer Design for GNSS and IMU integration

Harald Nøkland

June 2011

Master's Thesis for the Degree of
MSc in Engineering Cybernetics

Department of Engineering Cybernetics
Faculty of Information Technology,
Mathematics and Electrical Engineering
Norwegian University of Science and Technology

NTNU
Norwegian University of
Science and Technology

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

MSC THESIS DESCRIPTION SHEET

Name:	Harlad Nøkland
Department:	Engineering Cybernetics
Thesis Title (English):	Nonlinear Observer Design for GNSS and IMU integration
Thesis Title (Norwegian):	Ulinære tilstandsestimatorer for GNSS og IMU integrasjon

Thesis Description: The purpose of the thesis is to develop nonlinear observers for integration of global navigation satellite system (GNSS) and low-cost inertial measurement unit (IMU) data to be used onboard unmanned vehicles. This involves development of fault-tolerant solutions and experimental testing of the algorithms.

The following items must be considered:

1. *Literature study:* Give an overview of different methods for GNSS and IMU integration.
2. *Methods and software:* Develop software for simulation and testing of the extended Kalman filter (EKF) and nonlinear algorithms of Mahoney et al. and Hua. The observers should make use of the IMU accelerometers, gyros and magnetometer measurements, GNSS positions or pseudo-range measurements and optionally measurements for wind speed and altitude.
3. *Fault tolerance:* Include low-level quality check on signals such that the system is capable of handling signal freeze, drop out, drift and wild points. Other fault-tolerant scenario should also be considered, for instance dead-reckoning capabilities.
4. *Simulator:* Develop an unmanned aerial vehicle (UAS) simulator that generates all measurements needed for numerical testing of the observer algorithms. Faults and error handling should be simulated as events which can be triggered by an operator.
5. *Experimental testing and verification:* Mount the Xsens MTi-G attitude and heading reference system and GPS antenna on a car for logging of time-series. Compare the performance of the nonlinear observers with the build-in EKF solution (navigation filter).
6. *Report:* Present your findings and experimental results in the report.

Start date:	2011-01-17
Due date:	2011-06-13

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

Abstract

In order to efficiently control an unmanned vehicle, knowledge about the position, velocity and attitude (orientation) is needed. This thesis address this problem, and designs a navigation system for local navigation using low-cost sensors. Two loosely coupled GNSS/IMU integration filters are developed using a direct state estimation approach.

The first is a quaternion based multiplicative extended Kalman filter (MEKF). A multiplicative filter differs from the usual EKF in how the attitude is represented, which is done by a quaternion product. The filter avoids the singular covariance matrix caused by the constraint on the quaternion. Two versions of the filter are developed: one using the q-method to get a measurement of the attitude; and one using vector measurements directly.

The second is a nonlinear observer, termed HuaMahony. It is derived by combining two nonlinear algorithms proposed by Mahony et al. and Hua. The resulting nonlinear observer is able to estimate the linear acceleration as well as gyro bias. The nonlinear observer is written on an EKF-like discrete-time corrector-predictor formulation.

Both the multiplicative extended Kalman filter and the nonlinear observer are tested and verified through simulations and experimental data. Tests are carried out to examine how different disturbances affects the estimates and to compare the performance results.

Simulation results shows an average dynamic accuracy of <0.5 deg RMS for the attitude estimates, for both observers. The results from the experimental tests shows an average roll, pitch and yaw accuracy of (0.3 0.3 2.0) and (0.8 0.7 4.4) deg RMS for the MEKF and HuaMahony observer respectively, where it has been assumed that the Xsens MTi-G built-in EKF estimates are the true values.

An advantage of the MEKF is quicker convergence from initial errors, while an advantage of the HuaMahony observer is less computational load. Results show that the HuaMahony is three times faster than the MEKF when it comes to execution time.

Preface

This thesis is the final work of the Master of Science (MSc) program provided by the Norwegian University of Science and Technology (NTNU). It has been carried out at the Department of Engineering Cybernetics (ITK).

I would like to thank my supervisor professor Thor I. Fossen for his guidance. Also I would like to thank my brother Arild Nøkland for many valuable discussions. Finally, I would like to thank my fellow student Jakob Jakobsen for lending me the Xsens MTi-G unit, which have been used to record experimental data.

Harald Nøkland
Trondheim, June 2011

Contents

Chapter 1

Introduction

Today all sorts of unmanned vehicles are becoming increasingly popular. Applications include data acquisition, surveillance or hazardous missions without the risk of human lives. Currently it is widely used for military purposes, however a great potential also exist for civilian or research purposes.

In order to efficiently control an unmanned vehicle, knowledge about the position, velocity and attitude (orientation) is needed. This is generally provided by a navigation system, however commercial high-quality navigation systems are rather expensive. This motivates the development of a navigation system using low-cost sensors. Typical inertial sensors that are used for navigation purposes are accelerometers, gyros and magnetometers. Together they form an inertial measurement unit (IMU). In addition, global navigation satellite systems (GNSS) are often used, such as GPS.

1.1 Motivation

The purpose of this thesis is to develop a GNSS/IMU integrated navigation system, using a low-cost IMU. It is to be used onboard an unmanned aerial vehicle (UAV) in order to provide estimates of position, velocity and attitude. It is an important part of a motion control system, which usually is divided into the following tasks (see Figure 1.1):

- Guidance system: Path planning and trajectory generation, providing setpoints to the control system.
- Control system: Manipulating the actuators such that the setpoints are reached.

- Navigation system: Keeping track of the current position, velocity and attitude. Provides feedback to the control system such that it knows when the setpoints are reached. Also it may provide feedback to the guidance system such that it for instance can recalculate the path if the current position is way of.

1.2 State-of-the-Art Navigation Systems

Traditional inertial navigation systems (INS) integrate the angular velocity output from a gyro to get the attitude. Similarly the acceleration output from an accelerometer is integrated once to get the velocity, and twice to get the position. However the signal output of a low-cost IMU typically contains high noise levels and time-varying biases, which cause an unlimited drift in the estimates (Gade 2009). To remedy this problem different aiding techniques are applied to estimate the biases and update the position, velocity and attitude estimates.

1.2.1 Attitude Aiding

A common attitude aiding technique is to apply vector measurements of known inertial directions. Such measurements can be obtained from: a magnetometer which measure the Earth's magnetic field; an accelerometer which measure the gravity together with the acceleration of the vehicle; a star tracker which measure the direction of a star; or a sun sensor which measure the direction of the sun.

In fact the attitude can be algebraically reconstructed if measurements of two of more known nonparallel inertial directions are available. Wahba (1965) formulated this as an optimization problem. An approximate solution is given by the QUEST algorithm (Shuster and Oh 1981), while Davenport's q-method provides an exact solution to the optimization problem (Keat 1977). Either the algebraically reconstructed attitude or the vector measurements can be used as an aiding technique.

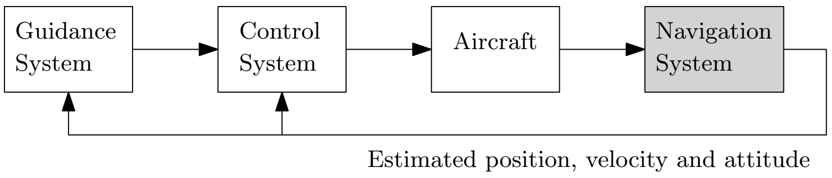


Figure 1.1: Block diagram showing how the navigation system relates to the other tasks of a motion control system. Figure is reproduced from Fossen (2011)

1.2.2 Position and Velocity Aiding

Advances made on global navigation satellite systems (GNSS) makes it possible to obtain accurate measurements of position and velocity. However it depends on good satellite coverage, and fails to give measurements during periods of signal loss. Integrating GNSS and IMU gives a redundant navigation system capable of handling GNSS dropouts, at least for a short period. The level of integration of a GNSS/IMU integrated navigation system is often divided into three main architectures based on how tight the coupling is, namely (Vik and Fossen 2001, Schmidt and Phillips 2010):

loosely coupled is a an integration filter which makes use of GNSS position and velocity measurements. A drawback of this design is that when the number of visible satellites is less than four, GNSS measurements are unavailable. Examples of loosely coupled integration filters can be found in: Fiorenzani et al. (2008), Ellingsen (2008), Vasconcelos, Silvestre and Oliveira (2011), Vasconcelos, Cardeira, Silvestre, Oliveira and Batista (2011), Hua (2010), Vik and Fossen (2001).

tightly coupled is a an integration filter which makes use of satellite pseudo range measurements. An advantage of this design is that the pseudo range measurements will provide information although there is less than four visible satellites. Another key feature is velocity feedback to the GNSS receiver correlator loops, making a more robust system. Examples of tightly coupled integration filter can be found in: Wendel and Trommer (2004), Yi and Grejner-Brzezinska (2006).

deeply coupled (also named ultra-tightly) is a an integration filter which encapsulating the GNSS correlator loops. This allow the most optimal use of GNSS and IMU raw data, making a highly robust system. Examples of deeply coupled integration filter can be found in: Abdel-Hafez (2010), Babu et al. (2008)

Generally the performance and robustness increase with tighter integration, at the cost of increased complexity and possible lack of redundancy (Vik n.d.).

1.2.3 Direct and Indirect Integration

In addition to the different levels of integration, there are two different ways of choosing the modeling variables for the position, velocity and attitude known as *direct* and *indirect* integration (Vik and Fossen 2001, Maybeck 1979):

indirect integration estimates the errors based on error models of the position, velocity and attitude. The error estimates provided by the integration filter are then used to update the position, velocity and attitude acquired by integration of accelerometer and gyro outputs. The integration filter can be designed with a slow update rate due to the slow dynamics of the error model. This method is computational advantageous if the integration filter runs on a separate computer. (Vik and Fossen 2001)

direct integration estimates the position, velocity and attitude directly in the integration filter. This method is advantageous when only one computer is used because it avoids the additional propagation of an error model. However, if a Kalman filter is used for integration, the *indirect* method may still be more efficient, as a high update rate of the covariance matrix is computational intensive. (Vik and Fossen 2001)

1.2.4 Integration Filter

The the body-fixed measurements of angular velocity and acceleration is related to the position, velocity and attitude through differential equations, known as the strapdown inertial navigation equations (Vik n.d.). The equations are nonlinear, thus nonlinear algorithms are most suitable for integration of GNSS and IMU. Usually an extended Kalman filter (EKF) is used, but other methods are also used.

A survey of modern nonlinear attitude estimation methods is presented in Crassidis et al. (2007). There has recently been an increasingly interest for nonlinear observers (Vik and Fossen 2001, Mahony et al. 2008, Martin and Salaün 2008, Hua 2010, Fossen 2011, Vasconcelos, Cardeira, Silvestre, Oliveira and Batista 2011). Nonlinear observers offer an easier to tune and less computational alternative to the EKF. In addition stronger proofs of stability and convergence can be achieved.

1.3 This Thesis

This thesis designs a navigation system for local navigation using low-cost sensors. Two *loosely* coupled GNSS/IMU integration filters are developed using a *direct* state estimation approach.

The first is a quaternion based multiplicative extended Kalman filter (Markley 2003). A multiplicative filter differs from the usual EKF in how the attitude is represented, which is done by a quaternion product. The filter avoids the singular covariance matrix caused by the constraint on the quaternion. Two versions of the filter are developed: one using the q-method to get a measurement of the attitude; and one using vector measurements directly.

The second is a nonlinear observer. It is derived by combining two nonlinear algorithms proposed by Hua (2010) and Mahony et al. (2008). The advantage of Hua (2010), which is linear acceleration estimation, is combined with the advantage of Mahony et al. (2008), which is gyro bias estimation. The nonlinear observer is formulated on an EKF-like discrete-time corrector-predictor form.

Both the extended Kalman filter and the nonlinear observer are tested and verified through simulations and experimental data. Tests are carried out to examine how different disturbances affects the estimates. and to compare the performance results. It is desirable to find out whether the nonlinear observer can provide competitive performance compared to the extended Kalman filter.

1.4 Contributions

The main contributions and work done in this thesis are:

Chapter 1: Literature study and learning about existing methods.

Chapter 3: Coding of a C-program which can decode and extract data from the experimentally recorded binary log files.

Chapter 4: Derivation of the q-method in a slightly different way compared to Keat (1977). Coding of the q-method in Matlab.

Chapter 5: The multiplicative extended Kalman filter is described in Markley (2003). However this thesis derives a different attitude model based on a discrete-time approach. Another contribution is the derivation of an analytic vector expression (5.21) and (5.17) for the partial derivative of the rotation matrix with respect to the vector part of the quaternion, when the rotation matrix is parametrized by the error quaternion (5.2). I suppose someone already have done this, but I have not been able to find it.

Chapter 6: Design of a nonlinear observer by using nonlinear algorithms of Hua (2010) and Mahony et al. (2008). In addition writing the nonlinear observer on a discrete-time corrector-predictor formulation.

Chapter 7: Simulink implementation of the simulator.

Chapter 8: Matlab implementation of the nonlinear observers, including methods to handle different sampling rates of GNSS and IMU, dead-reckoning and other fault tolerant solutions.

Chapter 9: Testing and verification through simulations.

Chapter 10: Recording of experimental data and testing.

Chapter 2

Background

This chapter contains a brief summary of navigation fundamentals. This is based on Fossen (2011) and Vik (n.d.).

2.1 Reference Frames

The different reference frames used here are described as follows:

ECI

The Earth-Centered Inertial (ECI) frame $\{i\} = (x_i, y_i, z_i)$ is an inertial frame fixed in space. The origin of $\{i\}$ is located at the center of the Earth. It is defined with the x-axis pointing towards the *vernal equinox*, and the z-axis pointing along the Earth's rotation axis. The y-axis completes the right handed orthogonal coordinate system.

ECEF

The Earth-Centered Earth-Fixed (ECEF) frame $\{e\} = (x_e, y_e, z_e)$ has its origin fixed to the center of the Earth. It is defined with the x-axis pointing towards the intersection of 0° longitude (Greenwich meridian) and 0° latitude (Equator). The z-axis points along the Earth's rotation axis, and the y-axis complete the right handed orthogonal coordinate system. The ECEF frame rotates relative to the ECI frame with the Earth rotation rate $\omega_e = 7.2921 \cdot 10^{-5}$ rad/s. Both Cartesian and ellipsoidal coordinates (longitude, latitude, height) are used to represent position in the ECEF frame.

Table 2.1: Overview over velocity and angular velocity vector

Frame	Vector	Comment
BODY	$\boldsymbol{v}_{b/n}^{b} = [u, v, w]$	Surge, sway and heave.
BODY/NED	$\boldsymbol{\omega}_{b/n}^{b} = [p, q, r]$	Roll, pitch and yaw rate
NED	$\boldsymbol{v}_{b/n}^{n} = [v_N, v_E, v_D]$	North, east and down velocity
NED/ECEF	$\boldsymbol{\omega}_{n/e}^{n} = [\omega_N, \omega_E, \omega_D]$	Function of change in long. and lat.
ECEF	$\boldsymbol{v}_{b/e}^{e} = [v_{x_e}, v_{y_e}, v_{z_e}]$	Velocity in frame given by GPS
ECEF/ECI	$\boldsymbol{\omega}_{e/i}^{e} = [0, 0, \omega_e]$	ω_e is angular velocity of Earth

NED

The North East Down (NED) frame $\{n\} = (x_n, y_n, z_n)$ is defined relative to the Earth's reference ellipsoid (World Geodetic System 1984). The z-axis points downward perpendicularly to the tangent plane of the ellipsoid, and the x-axis points towards true north. The y-axis point towards east to complete the orthogonal coordinate system.

BODY

The body-fixed reference frame $\{b\} = (x_b, y_b, z_b)$ is a moving and rotating coordinate frame that is fixed to the vehicle. The x-axis points in the forward direction, the y-axis to the right side and the z-axis downward. The position and orientation of the vehicle are described relative to the inertial frame (approximated by $\{n\}$ for local navigation).

2.2 Notation

Throughout this thesis vectors are written in bold, while matrices are written in capital letters, non-bold. The vectors have superscripts denoting which frame it is decomposed in. In addition some vectors have subscripts showing which frames the vector is relative to. Consider the following examples

$$\begin{aligned}
\boldsymbol{v}_{b/n}^{n} &= \text{linear velocity of \{b\} with respect to \{n\} expressed in \{n\}} \\
\boldsymbol{\omega}_{b/n}^{b} &= \text{angular velocity of \{b\} with respect to \{n\} expressed in \{b\}} \\
\boldsymbol{p}_{b/e}^{e} &= \text{position of \{b\} relative to \{e\} expressed in \{e\}}
\end{aligned}$$

An overview of some common vectors are given in the Tables 2.1 and 2.2.

Table 2.2: Overview over position and attitude vectors

Frame	Vector	Comment
BODY/NED	$\boldsymbol{q} = [\eta, \epsilon_1, \epsilon_2, \epsilon_3]$	Orientation of BODY relative to NED
NED	$\boldsymbol{p}^n_{b/n} = [N, E, D]$	NED position
NED/ECEF	$\boldsymbol{\Theta}_{en} = [l, \mu]$	Orientation of NED relative to ECEF
ECEF	$\boldsymbol{p}^e_{b/e} = [x, y, z]$	ECEF position

2.3 Transformation between BODY and NED

The attitude is represented by the singularity free unit quaternion

$$\boldsymbol{q} = \begin{bmatrix} \eta \\ \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} = \begin{bmatrix} \eta \\ \boldsymbol{\epsilon} \end{bmatrix} \tag{2.1}$$

where η is the real part and $\boldsymbol{\epsilon}$ is the imaginary part. A unit quaternion satisfies $\boldsymbol{q}^T\boldsymbol{q} = 1$. The inverse unit quaternion

$$\boldsymbol{q}^{-1} = \begin{bmatrix} \eta \\ -\boldsymbol{\epsilon} \end{bmatrix} \tag{2.2}$$

represent a rotation in the opposite direction. The quaternion product of two unit quaternions is also a unit quaternion

$$\boldsymbol{q}_1 \otimes \boldsymbol{q}_2 = \begin{bmatrix} \eta_1\eta_2 - \boldsymbol{\epsilon}_1^T\boldsymbol{\epsilon}_2 \\ \eta_1\boldsymbol{\epsilon}_2 + \eta_2\boldsymbol{\epsilon}_1 + \boldsymbol{\epsilon}_1^\times\boldsymbol{\epsilon}_2 \end{bmatrix} \tag{2.3}$$

which represent a composite rotation. A rotation matrix $R \in SO(3)$ has the following properties

$$R^TR = RR^T = I, \qquad \det R = 1, \qquad R^{-1} = R^T \tag{2.4}$$

The rotation matrix from BODY to NED is given by

$$R^n_b(\boldsymbol{q}) = I_{3\times3} + 2\eta S(\boldsymbol{\epsilon}) + 2S^2(\boldsymbol{\epsilon}) \tag{2.5}$$

where S is the skew-symmetric matrix

$$S(\boldsymbol{x}) = -S^T(\boldsymbol{x}) = \boldsymbol{x}^\times = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \tag{2.6}$$

It is worth mentioning that $\boldsymbol{q}$ and$-\boldsymbol{q}$ represent the same rotation matrix. Hence, the unit quaternions have double coverage of $SO(3)$.

2.3.1 Quaternion Differential Equation

Given the body-fixed angular velocity vector $\boldsymbol{\omega}^b_{b/n} = [\ p \quad q \quad r\ ]^T$, the differential equation for the quaternion can be written as

$$\dot{\boldsymbol{q}} = \frac{1}{2}T(\boldsymbol{q})\boldsymbol{\omega}^b_{b/n} \tag{2.7}$$

$$T(\boldsymbol{q}) = \begin{bmatrix} -\epsilon_1 & -\epsilon_2 & -\epsilon_3 \\ \eta & -\epsilon_3 & \epsilon_2 \\ \epsilon_3 & \eta & -\epsilon_1 \\ -\epsilon_2 & \epsilon_1 & \eta \end{bmatrix} = \begin{bmatrix} -\boldsymbol{\epsilon}^T \\ \eta I_{3\times 3} + S(\boldsymbol{\epsilon}) \end{bmatrix} \tag{2.8}$$

or alternatively

$$\dot{\boldsymbol{q}} = \frac{1}{2}\Omega(\boldsymbol{\omega}^b_{b/n})\boldsymbol{q} \tag{2.9}$$

$$\Omega(\boldsymbol{\omega}^b_{b/n}) = \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} = \begin{bmatrix} 0 & -(\boldsymbol{\omega}^b_{b/n})^T \\ \boldsymbol{\omega}^b_{b/n} & -S(\boldsymbol{\omega}^b_{b/n}) \end{bmatrix} \tag{2.10}$$

or alternatively

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \oplus \begin{bmatrix} 0 \\ \boldsymbol{\omega}^b_{b/n} \end{bmatrix} \tag{2.11}$$

2.4 Transformation between NED and ECEF

The rotation matrix from NED to ECEF is given by

$$R^e_n(\boldsymbol{\Theta}_{en}) = \begin{bmatrix} -\sin(\mu)\cos(l) & -\sin(l) & -\cos(\mu)\cos(l) \\ -\sin(\mu)\sin(l) & \cos(l) & -\cos(\mu)\sin(l) \\ \cos(\mu) & 0 & -\sin(\mu) \end{bmatrix} \tag{2.12}$$

where the vector $\boldsymbol{\Theta}_{en} = [l, \mu]^T$ consist of the longitude l and latitude μ.

Table 2.3: WGS-84 parameters

Parameter	Comments
$r_e = 6\ 378\ 137$ m	Equatorial radius of ellipsoid (semi-major axis)
$r_p = 6\ 356\ 752$ m	Polar axis radius of ellipsoid (semi-minor axis)
$\omega_e = 7.292115 \cdot 10^{-5}$ rad/s	Angular velocity of the Earth
$\mu_g = 398\ 600.5 \cdot 10^9\ \mathrm{m}^3/\mathrm{s}^2$	Gravitational constant of Earth
$e = 0.0818$	Eccentricity of ellipsoid

2.4.1 Transformation from Cartesian to Ellipsoidal ECEF Coordinates

The measurements of satellite navigation systems (GPS, etc) are often presented as the ellipsoidal parameters longitude l, latitude μ and height h. The reference ellipsoid used for satellite navigation systems is WGS-84. The most important parameters of the ellipsoid are listed in Table 2.3. The transformation from Cartesian coordinates (x_e, y_e, z_e) to longitude, latitude and height (l, μ, h) is usually done iteratively. The longitude can easily be found from

$$l = \arctan\left(\frac{y_e}{x_e}\right) \tag{2.13}$$

The latitude and height can be found using the following algorithm:

1. Compute $p = \sqrt{x_e^2 + y_e^2}$ and $e = \sqrt{1 - \frac{r_p^2}{r_e^2}}$
2. Compute the approximate value $\mu_{(0)}$ from

$$\tan(\mu_{(0)}) = \frac{z_e}{p}\left(1 - e^2\right)^{-1} \tag{2.14}$$

3. Compute an approximate value $N_{(0)}$ from

$$N_{(0)} = \frac{r_e^2}{\sqrt{r_e^2 \cos^2(\mu_{(0)}) + r_p^2 \sin^2(\mu_{(0)})}} \tag{2.15}$$

4. Compute the ellipsoidal height by

$$h = \frac{p}{\cos(\mu_{(0)})} - N_{(0)} \tag{2.16}$$

5. Compute an improved value for the latitude by

$$\tan(\mu) = \frac{z_e}{p}\left(1 - e^2 \frac{N_{(0)}}{N_{(0)} + h}\right)^{-1} \tag{2.17}$$

6. Check for another iteration step: if $\mu = \mu_{(0)}$ then the iteration is completed. Otherwise set $\mu_{(0)} = \mu$ and continue with step 3.

2.4.2 Transformation from Ellipsoidal to Cartesian ECEF Coordinates

The transformation from longitude, latitude and height (l, μ, h) to Cartesian coordinates (x_e, y_e, z_e) is given by

$$\begin{bmatrix} x_e \\ y_e \\ z_e \end{bmatrix} = \begin{bmatrix} (N+h)\cos(\mu)\cos(l) \\ (N+h)\cos(\mu)\sin(l) \\ (\frac{r_p^2}{r_e^2}N+h)\sin(\mu) \end{bmatrix} \tag{2.18}$$

where N is the radius of curvature in prime vertical obtained from

$$N = \frac{r_e^2}{\sqrt{r_e^2\cos^2(\mu) + r_p^2\sin^2(\mu)}} \tag{2.19}$$

2.5 Transformation between ECEF and ECI

The rotation matrix from ECEF to ECI is given by

$$R_e^i(\omega_{ie}^e t) = \begin{bmatrix} \cos(\omega_e t) & -\sin(\omega_e t) & 0 \\ \sin(\omega_e t) & \cos(\omega_e t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.20}$$

where ω_e is the Earth rotation rate (see Table 2.3).

Chapter 3

Sensor and Navigation Systems

Typical inertial sensors that are used for navigation purposes are accelerometers, gyros and magnetometers. Together they form an inertial measurement unit (IMU). In addition, global navigation satellite systems (GNSS) are often used. Several GNSS systems exists, such as GPS (American), GLONASS (Russian) and GALILEO (European). This chapter describes what the IMU and GPS measures and the mathematical modeling of those measurements. The device used in this thesis is the Xsens MTi-G unit (Xsens 2009d).

3.1 Inertial Measurement Unit (IMU)

The IMU contains a cluster of three gyros, three accelerometers and three magnetometers that measures angular velocity, acceleration and magnetic field respectively (see Figure 3.1).

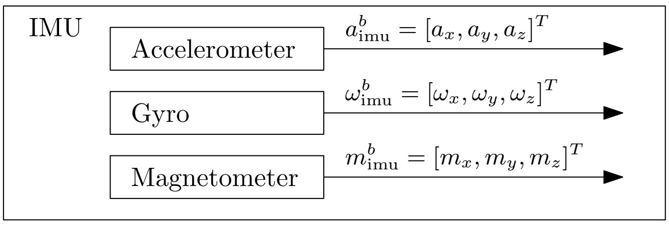

IMU
Accelerometer
$a^b_{\text{imu}} = [a_x, a_y, a_z]^T$
Gyro
$\omega^b_{\text{imu}} = [\omega_x, \omega_y, \omega_z]^T$
Magnetometer
$m^b_{\text{imu}} = [m_x, m_y, m_z]^T$

Figure 3.1: Block diagram showing the IMU assembly and its signals.

3.1.1 Gyro measurement

An error free gyro (gyroscope) measure the angular velocity of the body frame relative to the inertial frame about the sensor axis. Thus a cluster of three gyros with the sensor axes mounted orthogonal and aligned with the body axes measures the three component vector

$$\boldsymbol{\omega}^b_{b/i} = \boldsymbol{\omega}^b_{e/i} + \boldsymbol{\omega}^b_{n/e} + \boldsymbol{\omega}^b_{b/n} \tag{3.1}$$

where $\boldsymbol{\omega}^b_{e/i}$ is the Earth's rotation rate, $\boldsymbol{\omega}^b_{n/e}$ is a function of change in longitude and latitude, and $\boldsymbol{\omega}^b_{b/n}$ is the roll, pitch and yaw rate. However the gyro used in this thesis is not accurate enough to measure the Earth's rotation rate nor the transport rate over the curved Earth surface if the gyro has a velocity (Xsens 2009d). For local navigation we consider NED to be the inertial frame, that is the NED frame is not moving, and use the approximation

$$\boldsymbol{\omega}^b_{b/i} \approx \boldsymbol{\omega}^b_{b/n} \tag{3.2}$$

3.1.2 Gyro error model

Assuming small scale-factor errors and small misalignment errors, the gyro output can be modeled as (Fossen 2011):

$$\boldsymbol{\omega}^b_{\text{imu}} \approx \boldsymbol{\omega}^b_{b/n} + \boldsymbol{b}^b_{\text{gyro}} + \boldsymbol{w}_{\text{gyro}} \tag{3.3}$$

where $\boldsymbol{b}^b_{\text{gyro}} = [\begin{array}{ccc} b_p & b_q & b_r \end{array}]^T$ represent gyro bias in roll, pitch and yaw rate, and $\boldsymbol{w}_{\text{gyro}} \in \mathbb{R}^3$ is bounded unmodeled errors and measurement noise. The bias is modeled as a slowly time-varying disturbance

$$\dot{\boldsymbol{b}}^b_{\text{gyro}} = \boldsymbol{w}_{\text{bgyro}} \tag{3.4}$$

where $\boldsymbol{w}_{\text{bgyro}} \in \mathbb{R}^3$ is Gaussian white noise.

3.1.3 Accelerometer measurement

An error free accelerometer measure the specific force on the body frame along the sensor axis. Thus a cluster of three accelerometers with the sensor axes mounted orthogonal and aligned with the body axes measures the three component vector

$$\boldsymbol{f}^b = \boldsymbol{a}^b_{b/i} - R^b_n \boldsymbol{g}^n \tag{3.5}$$

where $\boldsymbol{a}^b_{b/i}$ is the linear acceleration of the moving body with respect to $\{i\}$ expressed in $\{b\}$, and $\boldsymbol{g}^n \approx \begin{bmatrix} 0 & 0 & 9,81 \end{bmatrix}^T$ m/s^2 is known as plumb bob gravity. For local navigation we consider NED to be the inertial frame, that is the NED frame is not moving, and use the approximation

$$\boldsymbol{a}^b_{b/i} \approx \boldsymbol{a}^b_{b/n} \tag{3.6}$$

3.1.4 Accelerometer error model

Assuming small scale-factor errors and small misalignment errors, the accelerometer output can be modeled as (Fossen 2011):

$$\boldsymbol{f}^b_{\text{imu}} \approx \boldsymbol{a}^b_{b/n} - R^b_n \boldsymbol{g}^n + \boldsymbol{b}^b_{\text{acc}} + \boldsymbol{w}_{\text{acc}} \tag{3.7}$$

or alternatively

$$\boldsymbol{f}^b_{\text{imu}} \approx R^b_n[\dot{\boldsymbol{v}}^n_{b/n} - \boldsymbol{g}^n] + \boldsymbol{b}^b_{\text{acc}} + \boldsymbol{w}_{\text{acc}} \tag{3.8}$$

where $\boldsymbol{b}^b_{\text{acc}} = \begin{bmatrix} b_u & b_v & b_w \end{bmatrix}^T$ represent accelerometer bias in x, y and z direction, and $\boldsymbol{w}_{\text{acc}} \in \mathbb{R}^3$ is bounded unmodeled errors and measurement noise. The bias is modeled as a slowly time-varying disturbance

$$\dot{\boldsymbol{b}}^b_{\text{acc}} = \boldsymbol{w}_{\text{bacc}} \tag{3.9}$$

where $\boldsymbol{w}_{\text{bacc}} \in \mathbb{R}^3$ is Gaussian white noise.

3.1.5 Magnetometer measurement

An error free magnetometer measure the strength of the magnetic field along the sensor axis. Thus a cluster of three magnetometers with the sensor axes mounted orthogonal and aligned with the body axes measures the three component vector

$$\boldsymbol{m}^b = R^b_n \boldsymbol{m}^n \tag{3.10}$$

where $\boldsymbol{m}^n = \begin{bmatrix} m_N & m_E & m_D \end{bmatrix}^T$ represent the magnitude and direction of the Earth's magnetic field. The magnetic field is different around the globe and in fact time-varying as well. In our area (Trondheim, Norway) it is approximately $\boldsymbol{m}^n \approx \begin{bmatrix} 13\,605 & 439 & 49\,864 \end{bmatrix}^T$ nT, which is found from an online calculator[1].

[1] `http://ngdc.noaa.gov/geomagmodels/IGRFWMM.jsp`

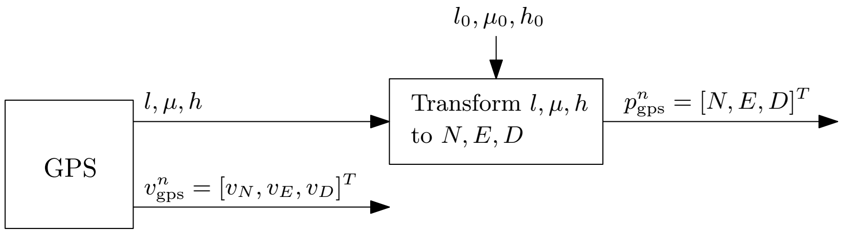

l_0, μ_0, h_0
l, μ, h
Transform l, μ, h to N, E, D
$p^n_{\text{gps}} = [N, E, D]^T$
GPS
$v^n_{\text{gps}} = [v_N, v_E, v_D]^T$

Figure 3.2: Block diagram showing the GPS signals, and the transformation to NED coordinates.

3.1.6 Magnetometer error model

Assuming small scale-factor errors and small misalignment errors, the magnetometer output can be modeled as (Fossen 2011):

$$\boldsymbol{m}^b_{\text{imu}} \approx R^b_n \boldsymbol{m}^n + \boldsymbol{b}^b_{\text{mag}} + \boldsymbol{w}_{\text{mag}} \tag{3.11}$$

where $\boldsymbol{b}^b_{\text{mag}}$ is the local magnetic disturbance, and $\boldsymbol{w}_{\text{mag}} \in \mathbb{R}^3$ is bounded unmodeled errors and measurement noise. The local magnetic disturbance is modeled as a slowly time-varying disturbance

$$\dot{\boldsymbol{b}}^b_{\text{mag}} = \boldsymbol{w}_{\text{bmag}} \tag{3.12}$$

where $\boldsymbol{w}_{\text{bmag}} \in \mathbb{R}^3$ is Gaussian white noise.

3.2 Global Positioning System (GPS)

A GPS measures/calculates the position and velocity in the ECEF frame. The position is usually presented as the ellipsoidal parameters longitude l, latitude μ and height h over ellipsoid (WGS84), while the velocity is usually presented in NED. For local navigation we consider NED to be the inertial frame, thus we need to transform the GPS position measurement to NED coordinates. This can be done by assigning the origin of the NED frame to a fixed point in ECEF. The following GPS measurements are then calculated with respect to this point (see Figure 3.2).

3.2.1 NED Coordinates from Longitude and Latitude

The GPS measurement can be transformed to NED coordinates by the following steps:

1. Determine longitude, latitude and height of reference point (l_0, μ_0, h_0) and calculate the corresponding ECEF coordinates $\boldsymbol{p}_0^e := \boldsymbol{p}_{n/e}^e$ by using (2.18). This will be the origin of the NED frame.

2. Transform the GPS measurement to ECEF coordinates $\boldsymbol{p}_{b/n}^e$ by using (2.18) and calculate the displacement in NED by

$$\boldsymbol{p}_{b/n}^n = R_n^e(l_0, \mu_0)^T[\boldsymbol{p}_{b/e}^e - \boldsymbol{p}_0^e] \tag{3.13}$$

3.2.2 GPS error model

Assuming that the GPS position measurement has been transformed to NED coordinates, the GPS output can be modeled as (Fossen 2011):

$$\boldsymbol{p}_{\text{gps}}^n = \boldsymbol{p}_{b/n}^n + R_b^n \boldsymbol{r}_{\text{ant}}^b + \boldsymbol{w}_{\text{pos}} \tag{3.14}$$

$$\boldsymbol{v}_{\text{gps}}^n = \boldsymbol{v}_{b/n}^n + R_b^n S(\boldsymbol{\omega}_{b/n}^b)\boldsymbol{r}_{\text{ant}}^b + \boldsymbol{w}_{\text{vel}} \tag{3.15}$$

where $\boldsymbol{r}_{\text{ant}}^b$ is the location of the antenna and $\boldsymbol{w}_{\text{pos}}, \boldsymbol{w}_{\text{vel}} \in \mathbb{R}^3$ is bounded unmodeled errors and measurement noise.

3.3 Xsens MTi-G

The device used in this thesis is the Xsens MTi-G (see Figure 3.3). It contains a three-axis IMU (see Table 3.1), a GPS receiver and a barometer. Sampling rates for the IMU and GPS is 100 Hz and 4 Hz respectively The unit also have a built-in extended Kalman filter (EKF) which provides estimates of position, velocity and attitude (see Table 3.2). These estimates is considered to be the true values during experimental testing of the nonlinear observers developed in this thesis. The MT Manger application can log data from the unit to a binary file (.mtb). MTB files can later be exported to ASCII text data. More information about the unit can be found in the user manual Xsens (2009d).

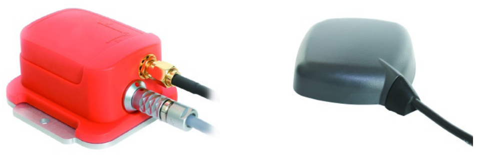


Figure 3.3: The Xsens MTi-G with GPS antenna

Table 3.1: Calibrated IMU data performance specification. Reproduced from Xsens (2009d).

	Gyro	Accelerometer	Magnetometer
Dimensions	3 axes	3 axes	3 axes
Full Scale	± 300 deg/s	± 50 m/s^2	± 750 mGauss
Linearity	0.1 %	0.2 %	0.2 %
Bias stability (1σ)	1 deg/s	0.02 m/s^2	0.1 mGauss
Scale-factor stability (1σ)	-	0.03 %	0.5 %
Noise	0.05 deg/s/$\sqrt{}$Hz	0.002 m/s^2/$\sqrt{}$Hz	0.5 mGauss (1σ)
Alignment error	0.1 deg	0.1 deg	0.1 deg
Bandwidth	40 Hz	30 Hz	10 Hz

Table 3.2: Performance specification of the Xsens MTi-G built-in EKF. Reproduced from Xsens (2009d).

	Roll/Pitch	Yaw	Position
Static accuracy	< 0.5 deg RMS	< 1 deg RMS	2.5 m CEP
Dynamic accuracy	1 deg RMS	2 deg RMS	

3.3.1 Configuration

MT Manager has been configured to log RAW inertial and GPS PVT data messages. This is the most powerful option as it is possible to reprocess the data with different EKF scenarios. The main difference between these scenarios is which sensors and assumptions are used in the EKF. The *aerospace nobaro*[2] scenario has been chosen. It makes use of the accelerometer, gyro, magnetometer and GPS. The sampling rate for the EKF is set to 100 Hz.

3.3.2 Exporting Data

It is easy to export MTB files to ASCII text data with the MT Manger (Xsens 2009c). Table 3.3 contains a overview of the data directly exported from MT Manger. The IMU measurements are factory calibrated with respect to misalignment, scale-factor and temperature. The calibrated data is unprocessed, i.e. only the physical calibration model is applied to the AD-converters. There is no additional filtering applied to the data.

MT Manger does not support export of all data of interest, i.e. unprocessed GPS position and velocity data. Therefore, a custom C-program that decodes the messages in the binary file and exports them to ASCII text data was made (source code is included on the CD). Table 3.4 contains an overview of the data of interest.

[2]see Xsens (2009d)

Table 3.3: Data exported from MT Manager

Data	Comment/Unit
Cal Accelerometer	acceleration (m/s^2)
Cal Gyro	angular velocity (rad/s)
Cal Magnetometer	magnetic field (arbitrary units) normalized to earth field strength
EKF Attitude	quaternion
EKF Position	longitude (deg), latitude (deg) and height (m)
EKF Velocity	NED velocity (m/s)

Table 3.4: Data extracted with custom C-program

Data	Comment/Unit
GPS Position	longitude (deg), latitude (deg) and height (m)
GPS Velocity	NED velocity (m/s)
GPS Fix	true/false
GPS age	when the value decrease, new GPS data is available

For more information about the structure of the MTB file and different messages see Xsens (2009b).

Chapter 4

The q-method

Davenport's q-method is an algorithm which determine the attitude from a set of vector measurements (Keat 1977, Shuster and Oh 1981). It can be described as finding a rotation matrix which satisfies

$$\boldsymbol{r}_i^n = R_b^n \boldsymbol{b}_i^b \qquad\qquad i = 1, ..., m \tag{4.1}$$

where $\boldsymbol{r}_1^n, ..., \boldsymbol{r}_m^n$ are a set of reference unit vectors, which are m known directions (e.g. the direction of the gravity force or the Earth's magnetic field) in the NED coordinate system, and $\{\boldsymbol{b}_1^b, ..., \boldsymbol{b}_n^b\}$ are the observation unit vectors, which are the same m directions measured in the body coordinate system. In order for the rotation matrix to be fully determined there must be at least two nonparallel measurements. This can be formulated as an optimization problem where we wish to find an optimal R_b^n that minimizes the objective function

$$f(R_b^n) = \frac{1}{2} \sum a_i |\boldsymbol{r}_i^n - R_b^n \boldsymbol{b}_i^b|^2 \tag{4.2}$$

where $a_1, ..., a_m$ are a set of non-negative weights.

4.1 Derivation of the q-method algorithm

The derivation shown here is inspired by Keat (1977) and Shuster and Oh (1981), but it is not identical. Superscripts will be dropped in the following for simplicity. The q-method transforms the objective function into a quadratic function in the quaternion, then solves this to get the optimal quaternion describing the attitude. First we can rewrite the objective function as:

$$\begin{aligned}
f(R) &= \frac{1}{2}\sum a_i|\boldsymbol{r}_i - R\boldsymbol{b}_i|^2 \\
&= \frac{1}{2}\sum a_i[\boldsymbol{r}_i - R\boldsymbol{b}_i]^T[\boldsymbol{r}_i - R\boldsymbol{b}_i] \\
&= \frac{1}{2}\sum a_i(\boldsymbol{r}_i^T\boldsymbol{r}_i - \boldsymbol{r}_i^T R\boldsymbol{b}_i - \boldsymbol{b}_i^T R^T\boldsymbol{r}_i - \boldsymbol{b}_i R^T R\boldsymbol{b}_i) \\
&= \frac{1}{2}\sum a_i(-2\boldsymbol{r}_i^T R\boldsymbol{b}_i) \\
&= -\sum a_i\boldsymbol{r}_i^T R\boldsymbol{b}_i
\end{aligned} \tag{4.3}$$

where it is used that $RR^T = I$. Terms independent of R has been canceled since they don't affect the optimal R. The rotation matrix R can be expressed as a function of the quaternions (2.5):

$$\begin{aligned}
R(\boldsymbol{q}) &= I_{3\times3} + 2\eta\boldsymbol{\epsilon}^\times + 2\boldsymbol{\epsilon}^\times\boldsymbol{\epsilon}^\times \\
&= I_{3\times3} + 2\eta\boldsymbol{\epsilon}^\times + 2\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T - 2\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}I_{3\times3} \\
&= \eta^2 I_{3\times3} + 2\eta\boldsymbol{\epsilon}^\times + 2\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T - \boldsymbol{\epsilon}^T\boldsymbol{\epsilon}I_{3\times3}
\end{aligned} \tag{4.4}$$

where it is used that $\boldsymbol{\epsilon}^\times\boldsymbol{\epsilon}^\times = \boldsymbol{\epsilon}\boldsymbol{\epsilon}^T - \boldsymbol{\epsilon}^T\boldsymbol{\epsilon}I_{3\times3}$ and $\eta^2 + \boldsymbol{\epsilon}^T\boldsymbol{\epsilon} = 1$. By inserting (4.4) into (4.3) the objective function can now be written as a function of the quaternions:

$$\begin{aligned}
f(\boldsymbol{q}) &= -\sum a_i\boldsymbol{r}_i^T[\eta^2 I_{3\times3} + 2\eta\boldsymbol{\epsilon}^\times + 2\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T - \boldsymbol{\epsilon}^T\boldsymbol{\epsilon}I_{3\times3}]\boldsymbol{b}_i \\
&= -\sum a_i(\eta^2\boldsymbol{r}_i^T\boldsymbol{b}_i + 2\eta\boldsymbol{r}_i^T\boldsymbol{\epsilon}^\times\boldsymbol{b}_i + 2\boldsymbol{r}_i^T\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T\boldsymbol{b}_i - \boldsymbol{\epsilon}^T\boldsymbol{\epsilon}\boldsymbol{r}_i^T\boldsymbol{b}_i) \\
&= -\sum a_i(\eta^2\boldsymbol{r}_i^T\boldsymbol{b}_i - \eta(\boldsymbol{r}_i^\times\boldsymbol{b}_i)^T\boldsymbol{\epsilon} - \eta\boldsymbol{\epsilon}^T(\boldsymbol{r}_i^\times\boldsymbol{b}_i) + ... \\
&\qquad \boldsymbol{\epsilon}^T\boldsymbol{r}_i\boldsymbol{b}_i^T\boldsymbol{\epsilon} + \boldsymbol{\epsilon}^T\boldsymbol{b}_i\boldsymbol{r}_i^T\boldsymbol{\epsilon} - \boldsymbol{\epsilon}^T\boldsymbol{\epsilon}\boldsymbol{r}_i^T\boldsymbol{b}_i) \\
&= -\eta^2\sigma + \eta\boldsymbol{z}^T\boldsymbol{\epsilon} + \eta\boldsymbol{\epsilon}^T\boldsymbol{z} - \boldsymbol{\epsilon}^T S\boldsymbol{\epsilon} + \boldsymbol{\epsilon}^T\boldsymbol{\epsilon}\sigma I_{3\times3} \\
&= \begin{bmatrix} \eta & \boldsymbol{\epsilon}^T \end{bmatrix} \begin{bmatrix} -\sigma & \boldsymbol{z}^T \\ \boldsymbol{z} & -S + \sigma I_{3\times3} \end{bmatrix} \begin{bmatrix} \eta \\ \boldsymbol{\epsilon} \end{bmatrix} \\
&= \boldsymbol{q}^T K\boldsymbol{q}
\end{aligned} \tag{4.5}$$

where the introduced variables are

$$B = \sum a_i \boldsymbol{r}_i \boldsymbol{b}_i^T \tag{4.6}$$

$$\sigma = tr(B) = \sum a_i \boldsymbol{r}_i^T \boldsymbol{b}_i \tag{4.7}$$

$$S = B + B^T = \sum a_i(\boldsymbol{r}_i \boldsymbol{b}_i^T + \boldsymbol{b}_i \boldsymbol{r}_i^T) \tag{4.8}$$

$$\boldsymbol{z} = \sum a_i(\boldsymbol{r}_i^\times \boldsymbol{b}_i) \tag{4.9}$$

$$K = K^T = \begin{bmatrix} -\sigma & \boldsymbol{z}^T \\ \boldsymbol{z} & -S + \sigma I_{3\times 3} \end{bmatrix} \tag{4.10}$$

The optimization problem has been reduced to

$$\min_{\boldsymbol{q}\in\mathbb{R}^4} f(\boldsymbol{q}) = \boldsymbol{q}^T K \boldsymbol{q} \tag{4.11}$$

$$s.t.$$

$$\boldsymbol{q}^T \boldsymbol{q} - 1 = 0 \tag{4.12}$$

The Lagrangian function and it's gradient for the problem is

$$\mathcal{L}(\boldsymbol{q}, \lambda) = \boldsymbol{q}^T K \boldsymbol{q} - \lambda(\boldsymbol{q}^T \boldsymbol{q} - 1) \tag{4.13}$$

$$\nabla_q \mathcal{L}(\boldsymbol{q}^*, \lambda^*) = 2K\boldsymbol{q}^* - 2\lambda^* \boldsymbol{q}^* \tag{4.14}$$

where the superscript $*$ denotes the optimal value and λ is the Lagrange multiplier. The first order optimality conditions often known as the KKT-conditions becomes (Nocedal and Wright 2006)

$$K\boldsymbol{q}^* = \lambda^* \boldsymbol{q}^* \tag{4.15}$$

$$(\boldsymbol{q}^*)^T \boldsymbol{q}^* = 1 \tag{4.16}$$

Thus, the optimal point $\boldsymbol{q}^*$ must be an eigenvector of K with λ^* as the corresponding eigenvalue. Equation (4.15) is independent of the normalization of $\boldsymbol{q}^*$ and, therefore, (4.16) does not determine λ^*. However, by examining the objective function in the optimal point

$$f(\boldsymbol{q}^*) = (\boldsymbol{q}^*)^T K \boldsymbol{q}^* = \lambda^* (\boldsymbol{q}^*)^T \boldsymbol{q}^* = \lambda^* \tag{4.17}$$

it's seen that f will be minimized when λ^* is the smallest eigenvalue of K. Hence the optimal point $\boldsymbol{q}^*$ is the eigenvector of K belonging to the smallest eigenvalue of K.

$$K\boldsymbol{q}^* = \lambda_{\min} \boldsymbol{q}^* \tag{4.18}$$

The q-method algorithm can be summarized as follows:

- Compute the symmetric 4×4 matrix K
- Compute the normalized eigenvector belonging to the smallest eigenvalue of K

4.2 Using the q-method

The direction of the gravity $\boldsymbol{g}^n$ and the Earth's magnetic field $\boldsymbol{m}^n$ is known, and can be assumed to be constant for local navigation. Under a weak acceleration assumption ($\dot{\boldsymbol{v}}^n_{b/n} \approx 0$), the magnetometer and accelerometer gives us two observation vectors $\boldsymbol{m}^b$ and $\boldsymbol{g}^b$ respectively. By solving the objective function

$$f(R^n_b(\boldsymbol{q})) = a_1|\boldsymbol{m}^n - R^n_b(\boldsymbol{q})\boldsymbol{m}^b|^2 + a_2|\boldsymbol{g}^n - R^n_b(\boldsymbol{q})\boldsymbol{g}^b|^2 \tag{4.19}$$

we get a measurement of the optimal unit quaternion describing the attitude (see Figure 4.1). Afterwards a loosely coupled Kalman filter can be used to filter the measurements. It is important that the corresponding vectors have equal length. Preferable should all vectors be unit vectors, because this gives more intuitive weighting between the terms.

The q-method, as it is derived here, has been implemented and tested in Matlab. The code is given in the Appendix. To verify the method, a wide range of attitudes with their associated observation vectors have been generated. In Figure 4.2 it can be seen that the calculated quaternion follows the true quaternion exactly. In this case it was not added any noise to the observation vectors. When the observation vectors are corrupted by noise, the determined attitude is noisy as well (see Figure 4.3), which shows the need of additional filtering.

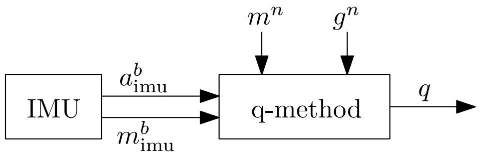


Figure 4.1: Block diagram showing the q-method signal flow.

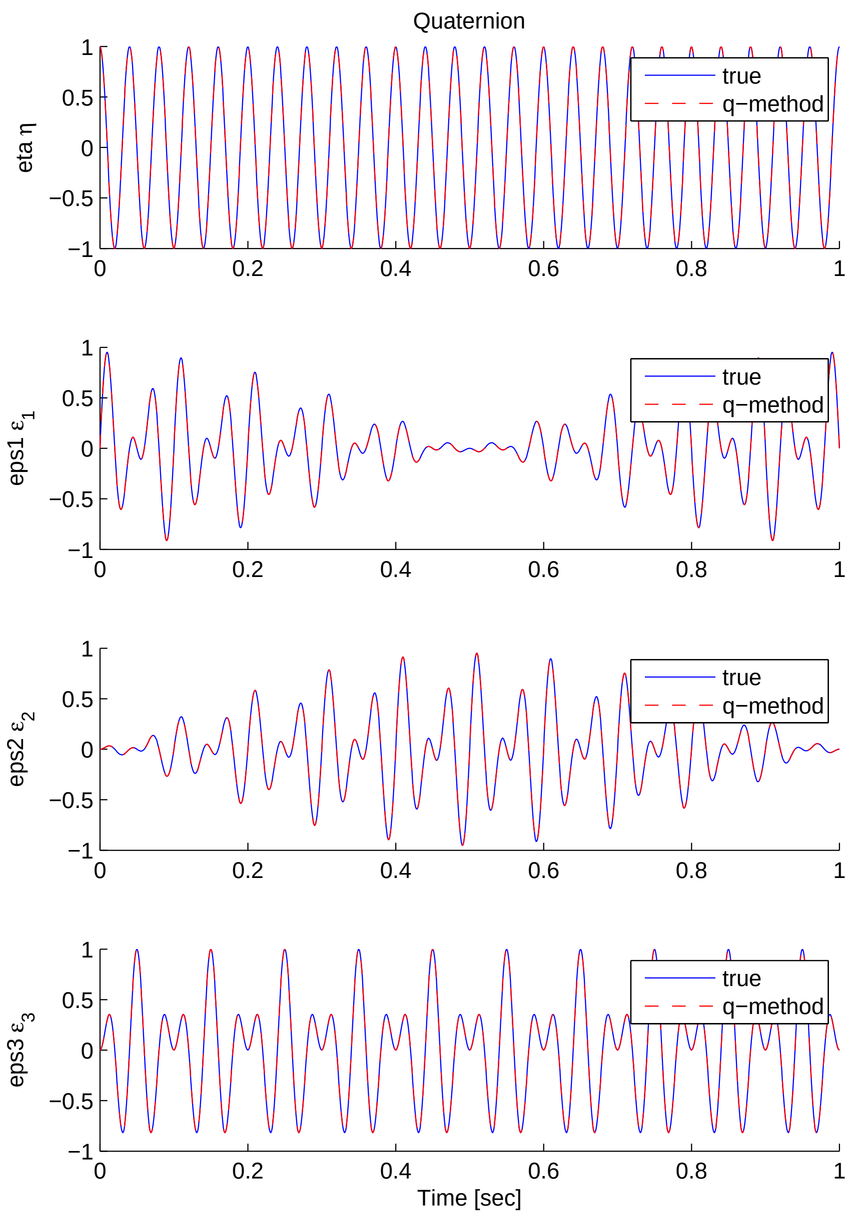


Figure 4.2: Verification of the q-method implementation.

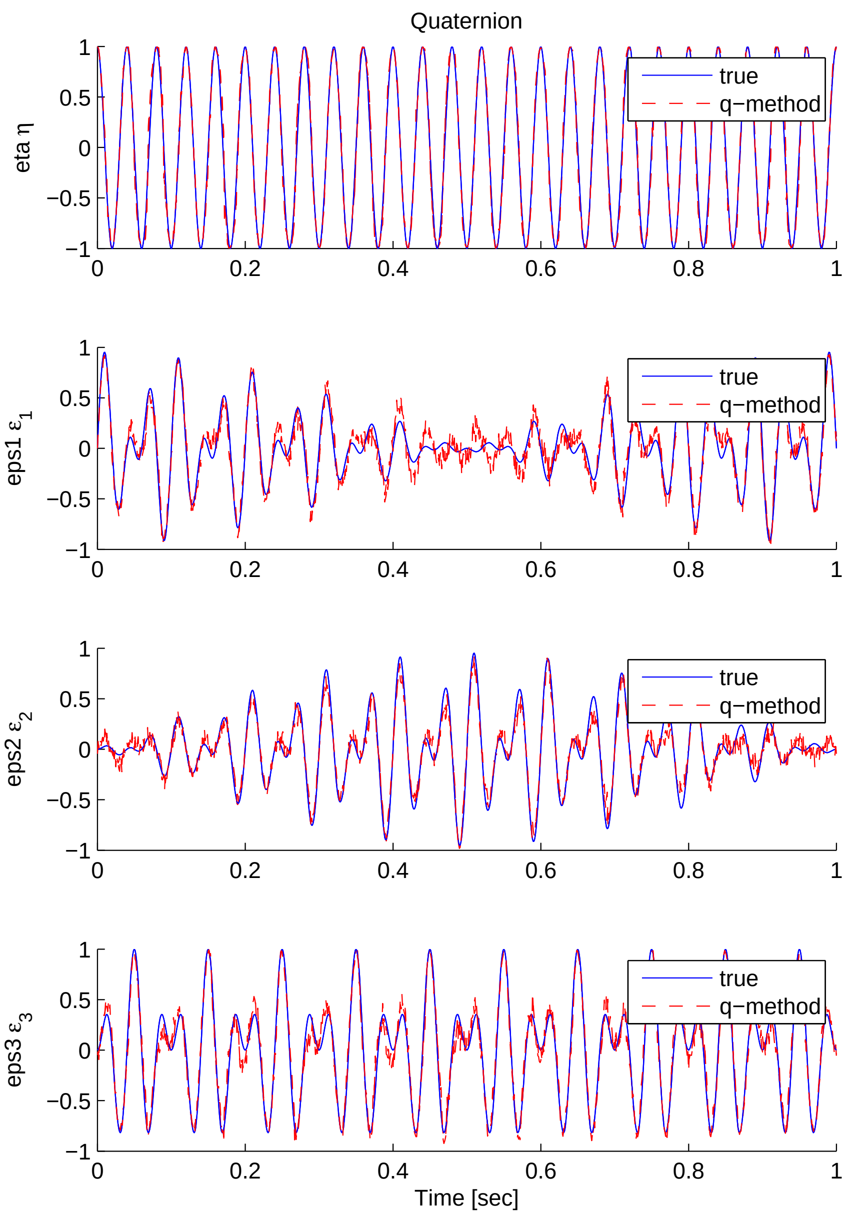


Figure 4.3: The effect of noisy observation vectors.

Chapter 5

Extended Kalman Filter Design

The *linear* Kalman filter (KF) is simply an algorithm which produce optimal minimum variance estimates, under the assumptions of Gaussian white noise and a linear observable system. However, as the inertial navigation equations are nonlinear the *extended* Kalman filter (EKF) will be used, which is the nonlinear variant of the usual KF. In the EKF the nonlinear system is linearized about the currently best estimate. It is commonly used in inertial navigation and many other fields. Stability and convergence of the EKF has been proven in Jouffroy and Fossen (2010), under the assumption of a lower and upper bounded covariance matrix.

In this chapter an extended Kalman filter for integration of GPS and IMU is derived. The goal is to estimate the position, velocity and attitude. The attitude will be represented by the singularity free unit quaternion. Special care must be taken when designing a quaternion based extended Kalman filter (Vik n.d.). Because of the constraint on the quaternions, the covariance matrix P will be singular, and hence has an eigenvalue equal to zero. To maintain this singularity is very difficult due to the accumulation of round-off errors in a numerical filter. In fact, this zero eigenvalue can even become negative and the filter becomes unstable. There exist several methods to deal with this singular covariance matrix. Most of them tries to reduce the dimension so that three parameters are used instead of four to describe the covariance of the four quaternions. The *multiplicative* extended Kalman filter (MEKF) handles this in an elegant way.

5.1 Discrete Multiplicative Extended Kalman Filter

This section contains a brief introduction to the multiplicative extended Kalman filter (MEKF), a more in depth explanation can be found in Markley (2003). The attitude is represented as the quaternion product

$$\boldsymbol{q} = \hat{\boldsymbol{q}} \otimes \delta\boldsymbol{q}(\delta\boldsymbol{\epsilon}) \qquad \Leftrightarrow \qquad R_b^n(\boldsymbol{q}) = R_{\hat{b}}^n(\hat{\boldsymbol{q}})R_b^{\hat{b}}(\delta\boldsymbol{q}) \tag{5.1}$$

where $\hat{\boldsymbol{q}}$ is some unit reference quaternion, representing the rotation from the reference frame $\{\hat{b}\}$ to the NED frame $\{n\}$. Moreover

$$\delta\boldsymbol{q}(\delta\boldsymbol{\epsilon}) = \begin{bmatrix} \sqrt{1-\delta\boldsymbol{\epsilon}^T\delta\boldsymbol{\epsilon}} \\ \delta\boldsymbol{\epsilon} \end{bmatrix} \tag{5.2}$$

is the attitude error, representing the rotation from the body frame $\{b\}$ to the reference frame $\{\hat{b}\}$. The MEKF computes an unconstrained estimate of the three-component $\delta\boldsymbol{\epsilon}$, while using the four-component $\hat{\boldsymbol{q}}$ to provide a globally non-singular attitude representation.

The filter proceeds in three steps: time propagation, measurement update, and reset. The discrete measurement update assigns a finite post-update value to $\delta\hat{\boldsymbol{\epsilon}}_k$. In order to avoid the need to propagate two representations of the attitude, the reset operation moves the attitude information from $\delta\hat{\boldsymbol{\epsilon}}_k$ to $\hat{\boldsymbol{q}}_k$, then $\delta\hat{\boldsymbol{\epsilon}}_k$ is reset to zero. Since true quaternion is not changed by this operation, (5.1) requires

$$\hat{\boldsymbol{q}}_{k-1} \otimes \delta\boldsymbol{q}(\delta\hat{\boldsymbol{\epsilon}}_k) = \hat{\boldsymbol{q}}_k \otimes \delta\boldsymbol{q}(\boldsymbol{0}) = \hat{\boldsymbol{q}}_k \tag{5.3}$$

Given the discrete nonlinear system on the form

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}_k(\boldsymbol{x}_k, \boldsymbol{u}_k) + \Gamma_k \boldsymbol{w}_k \tag{5.4}$$

$$\boldsymbol{y}_k = \boldsymbol{h}_k(\boldsymbol{x}_k) + \boldsymbol{v}_k \tag{5.5}$$

the discrete multiplicative extended Kalman filter algorithm can be summarized in Table 5.1.

5.2 Attitude Estimation

In this section we want to design a discrete multiplicative extended Kalman filter for attitude estimation. By integrating attitude measurements and gyro measurements, we achieve two things:

Table 5.1: Discrete Multiplicative Extended Kalman Filter

Design matrices	$Q_k = E[\boldsymbol{w}_k \boldsymbol{w}_k^T] = \text{cov}(\boldsymbol{w}_k) \approx \sigma_{w_k}^2 I$ $R_k = E[\boldsymbol{v}_k \boldsymbol{v}_k^T] = \text{cov}(\boldsymbol{v}_k) \approx \sigma_{v_k}^2 I$
Initial conditions	$\bar{\boldsymbol{x}}_0 = \boldsymbol{x}_0$ $\bar{P}_0 = P_0 = E[(\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0)(\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0)^T]$ $\hat{\boldsymbol{q}}_0 = \boldsymbol{q}_0$
 Kalman gain matrix State estimate update Error covariance update	$H_k = \left.\frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{x}_k}\right\|_{\boldsymbol{x}_k = \bar{\boldsymbol{x}}_{k\_}}$ $K_k = \bar{P}_k H_k^T [H_k \bar{P}_k H_k^T + R_k]^{-1}$ $\hat{\boldsymbol{x}}_k = \bar{\boldsymbol{x}}_k + K_k[\boldsymbol{y} - \boldsymbol{h}(\bar{\boldsymbol{x}}_k)]$ $\hat{P}_k = [I - K_k H_k]\bar{P}_k[I - K_k H_k]^T + K_k R_k K_k^T$
Move error Reset error	$\hat{\boldsymbol{q}}_k = \hat{\boldsymbol{q}}_{k-1} \otimes \delta \boldsymbol{q}(\delta \hat{\boldsymbol{\epsilon}}_k)$ $\delta \hat{\boldsymbol{\epsilon}}_k = 0$
 State estimation propagation Error covariance propagation	$\Phi_k = \left.\frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{x}_k}\right\|_{\boldsymbol{x}_k = \hat{\boldsymbol{x}}_k}$ $\bar{\boldsymbol{x}}_{k+1} = \boldsymbol{f}_k(\hat{\boldsymbol{x}}_k, \boldsymbol{u}_k)$ $\bar{P}_{k+1} = \Phi_k \hat{P}_k \Phi_k^T + \Gamma_k Q_k \Gamma_k^T$

- Low-pass filtering of the attitude measurements
- High-pass filtering of the gyro measurements

5.2.1 Assumptions

For this section we make the following assumptions:

- The vehicle has constant speed $\dot{\boldsymbol{v}}^n_{b/n} = 0$ and the accelerometer bias is zero $\boldsymbol{b}^b_{\text{acc}} = 0$. These states are not observable without an additional position or velocity measurement. Later these assumptions will be relaxed.
- The local magnetic disturbance is zero $\boldsymbol{b}^b_{\text{mag}} = 0$. For a well calibrated magnetometer, this is a reasonable assumption.

Consequently, we can write the accelerometer (3.8) and magnetometer (3.11) model as

$$\boldsymbol{f}^b_{\text{imu}} = -\boldsymbol{g}^b + \boldsymbol{w}_{\text{acc}} \tag{5.6}$$

$$\boldsymbol{m}^b_{\text{imu}} = \boldsymbol{m}^b + \boldsymbol{w}_{\text{mag}} \tag{5.7}$$

5.2.2 Attitude Model

The time-derivative of $\delta\boldsymbol{q}$ is found by differentiating

$$\delta\boldsymbol{q} = \hat{\boldsymbol{q}}^{-1} \otimes \boldsymbol{q} \tag{5.8}$$

which is from the definition in (5.1). Considering that the resulting model will be used in a discrete-time filter, one can argue that because there is no time propagation of $\hat{\boldsymbol{q}}$ in the discrete filter, $\hat{\boldsymbol{q}}$ will be constant between each sample. Hence we can treat $\hat{\boldsymbol{q}}$ as a constant when differentiating. This means that the change in the error quaternion equals the change in the true quaternion. The following show the derivation of the time-derivative of $\delta\boldsymbol{q}$:

$$\begin{aligned}
\delta\dot{\boldsymbol{q}} = \begin{bmatrix} \delta\dot{\eta} \\ \delta\dot{\boldsymbol{\epsilon}} \end{bmatrix} &= \hat{\boldsymbol{q}}^{-1} \otimes \dot{\boldsymbol{q}} \\
&= \frac{1}{2}\hat{\boldsymbol{q}}^{-1} \otimes \boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega}^b_{b/n} \end{bmatrix} \\
&= \frac{1}{2}\delta\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega}^b_{b/n} \end{bmatrix} \\
&= \frac{1}{2}\Omega(\boldsymbol{\omega}^b_{b/n})\delta\boldsymbol{q} \\
&= \frac{1}{2}\begin{bmatrix} 0 & -(\boldsymbol{\omega}^b_{b/n})^T \\ \boldsymbol{\omega}^b_{b/n} & -S(\boldsymbol{\omega}^b_{b/n}) \end{bmatrix} \begin{bmatrix} \sqrt{1-\delta\boldsymbol{\epsilon}^T\delta\boldsymbol{\epsilon}} \\ \delta\boldsymbol{\epsilon} \end{bmatrix}
\end{aligned} \tag{5.9}$$

which gives the following vector part

$$\begin{aligned}
\delta\dot{\boldsymbol{\epsilon}} &= \frac{1}{2}[\boldsymbol{\omega}^b_{b/n}\sqrt{1-\delta\boldsymbol{\epsilon}^T\delta\boldsymbol{\epsilon}} - S(\boldsymbol{\omega}^b_{b/n})\delta\boldsymbol{\epsilon}] \\
&= \frac{1}{2}[I_{3\times3}\sqrt{1-\delta\boldsymbol{\epsilon}^T\delta\boldsymbol{\epsilon}} + S(\delta\boldsymbol{\epsilon})]\boldsymbol{\omega}^b_{b/n}
\end{aligned} \tag{5.10}$$

Because of the unit constraint on the error quaternion, it can be constructed from the vector part an any time. Thus it is sufficient to only estimate the vector part in the filter. In fact, this kind of model reduction is the goal in order to avoid the singular covariance matrix.

For each iteration in the filter, the attitude information in the error quaternion will be moved to $\hat{\boldsymbol{q}}$, in the reset operation. This ensures that the real part of the error quaternion will be small. Consequently, the possible problems of a model reduction discussed in Vik (n.d.), Lefferts and Schuster (1982) are efficiently avoided.

Finally, the state variables in the filter are chosen to be $\boldsymbol{x} = [\delta\epsilon_1, \delta\epsilon_2, \delta\epsilon_3, b_p, b_q, b_r]$ $= [\delta\boldsymbol{\epsilon}, \boldsymbol{b}^b_{\text{gyro}}]$ and the input is $\boldsymbol{u} = \boldsymbol{\omega}^b_{\text{imu}}$. Substituting (3.3) into (5.10), together with (3.4) the attitude model can be written as

$$\underbrace{\begin{bmatrix} \delta\dot{\boldsymbol{\epsilon}} \\ \dot{\boldsymbol{b}}^b_{\text{gyro}} \end{bmatrix}}_{\dot{\boldsymbol{x}}} = \underbrace{\begin{bmatrix} \frac{1}{2}[I_{3\times3}\sqrt{1-\delta\boldsymbol{\epsilon}^T\delta\boldsymbol{\epsilon}} + S(\delta\boldsymbol{\epsilon})][\boldsymbol{\omega}^b_{\text{imu}} - \boldsymbol{b}^b_{\text{gyro}}] \\ 0_{3\times1} \end{bmatrix}}_{\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})} + \underbrace{I_{6\times6}}_{\Gamma} \underbrace{\begin{bmatrix} \boldsymbol{w}_{\text{gyro}} \\ \boldsymbol{w}_{\text{bgyro}} \end{bmatrix}}_{\boldsymbol{w}} \tag{5.11}$$

where $\boldsymbol{w} \in \mathbb{R}^6$ is Gaussian white noise with zero mean and variance σ^2_w.

5.2.3 Measurement Equation

By using the q-method, we get a measurement of the attitude $\boldsymbol{q}$. Input to the q-method is the magnetometer and accelerometer (recall from section 4.2). We calculate $\delta\boldsymbol{\epsilon}$ by using (5.8), then extracting the three last elements. Another approach is to change the reference vectors to $\boldsymbol{m}^{\hat{b}} = R_n^{\hat{b}}(\hat{\boldsymbol{q}}^{-1})\boldsymbol{m}^n$ and $\boldsymbol{g}^{\hat{b}} = R_n^{\hat{b}}(\hat{\boldsymbol{q}}^{-1})\boldsymbol{g}^n$, then the q-method will calculate $\delta\boldsymbol{\epsilon}$ directly. In either way we get the following measurement equation

$$\boldsymbol{y} = \delta\boldsymbol{\epsilon}_{\text{qmetod}} = \underbrace{\delta\boldsymbol{\epsilon}}_{\boldsymbol{h}(\boldsymbol{x})} + \underbrace{\boldsymbol{w}_\epsilon}_{\boldsymbol{v}} \tag{5.12}$$

where $\boldsymbol{v} \in \mathbb{R}^3$ is Gaussian white noise with zero mean and variance σ_v^2.

5.2.4 Discrete-Time Matrices

Discretizing the system with the Euler method gives

$$\boldsymbol{x}_{k+1} = \underbrace{\boldsymbol{x}_k + h\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k)}_{\boldsymbol{f}_k(\boldsymbol{x}_k, \boldsymbol{u}_k)} + \underbrace{h\Gamma}_{\Gamma_k} \boldsymbol{w}_k \tag{5.13}$$

$$\boldsymbol{y}_k = \underbrace{\boldsymbol{h}(\boldsymbol{x}_k)}_{\boldsymbol{h}_k(\boldsymbol{x}_k)} + \boldsymbol{v}_k \tag{5.14}$$

The discrete-time matrices needed in the filter becomes

$$\begin{aligned}
Q_k &\approx \sigma_{w_k}^2 I_{6\times6} \\
R_k &\approx \sigma_{v_k}^2 I_{3\times3} \\
\boldsymbol{f}_k(\hat{\boldsymbol{x}}_k, \boldsymbol{u}_k) &= \hat{\boldsymbol{x}}_k + h \begin{bmatrix} \frac{1}{2}[\boldsymbol{\omega}_{\text{imu}}^b - \hat{\boldsymbol{b}}_{\text{gyro}}^b] \\ 0_{3\times1} \end{bmatrix} \\
\Phi_k &= \left.\frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{x}_k}\right|_{\boldsymbol{x}_k = \hat{\boldsymbol{x}}_k} = I_{6\times6} + h \begin{bmatrix} -\frac{1}{2}S(\boldsymbol{\omega}_{\text{imu}}^b - \hat{\boldsymbol{b}}_{\text{gyro}}^b) & -\frac{1}{2}I_{3\times3} \\ 0_{3\times3} & 0_{3\times3} \end{bmatrix} \\
\Gamma_k &= hI_{6\times6} \\
H_k &= \left.\frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{x}_k}\right|_{\boldsymbol{x}_k = \bar{\boldsymbol{x}}_k} = \begin{bmatrix} I_{3\times3} & 0_{3\times3} \end{bmatrix}
\end{aligned}$$

where the filter property that $\delta\hat{\boldsymbol{\epsilon}}_k = 0$ has been used.

5.2.5 Alternative Measurement Equation

Instead of using the q-method, we can use the magnetometer (5.7) and accelerometer (5.6) directly in the measurement equation. After inserting (5.1) it looks like this

$$\boldsymbol{y} = \begin{bmatrix} \mathbf{m}^b_{\text{imu}} \\ \boldsymbol{f}^b_{\text{imu}} \end{bmatrix} = \underbrace{\begin{bmatrix} R^{\hat{b}}_b(\delta\boldsymbol{q})^T R^n_{\hat{b}}(\hat{\boldsymbol{q}})^T \mathbf{m}^n \\ -R^{\hat{b}}_b(\delta\boldsymbol{q})^T R^n_{\hat{b}}(\hat{\boldsymbol{q}})^T \boldsymbol{g}^n \end{bmatrix}}_{\boldsymbol{h}(\boldsymbol{x})} + \underbrace{\begin{bmatrix} \boldsymbol{w}_{\text{mag}} \\ \boldsymbol{w}_{\text{acc}} \end{bmatrix}}_{\boldsymbol{v}} \tag{5.15}$$

where $\boldsymbol{v} \in \mathbb{R}^6$ is Gaussian white noise with zero mean and variance σ_v^2. We need to develop an expression for the linearized measurement matrix H_k. This involves linearizing the transposed rotation matrix, which is a bit complicated. To make it easier to see how the Jacobian is calculated we can temporarily write $\boldsymbol{h}$ as

$$\boldsymbol{h}(\boldsymbol{x}) = \begin{bmatrix} R^{\hat{b}}_b(\delta\boldsymbol{q})^T \mathbf{m}^{\hat{b}} \\ -R^{\hat{b}}_b(\delta\boldsymbol{q})^T \boldsymbol{g}^{\hat{b}} \end{bmatrix} = \begin{bmatrix} R(\delta\boldsymbol{q})^T \boldsymbol{m} \\ -R(\delta\boldsymbol{q})^T \boldsymbol{g} \end{bmatrix} \tag{5.16}$$

We now define

$$\begin{aligned} W(\boldsymbol{\epsilon},\boldsymbol{v}) :=& \frac{\partial}{\partial\boldsymbol{\epsilon}} \left\{ R(\delta\boldsymbol{q}(\boldsymbol{\epsilon}))^T \boldsymbol{v} \right\} \qquad (5.17) \\ =& \frac{\partial}{\partial\boldsymbol{\epsilon}} \left\{ [I_{3\times3} - 2\sqrt{1-\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}} S(\boldsymbol{\epsilon}) + 2\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T - 2\boldsymbol{\epsilon}^T\boldsymbol{\epsilon} I_{3\times3}]\boldsymbol{v} \right\} \\ =& 2\frac{\partial}{\partial\boldsymbol{\epsilon}} \left\{ \frac{1}{2}\boldsymbol{v} + \sqrt{1-\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}} S(\boldsymbol{v})\boldsymbol{\epsilon} + (\boldsymbol{v}^T\boldsymbol{\epsilon})\boldsymbol{\epsilon} - (\boldsymbol{\epsilon}^T\boldsymbol{\epsilon})\boldsymbol{v} \right\} \\ =& 2\left\{ +\sqrt{1-\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}} S(\boldsymbol{v}) - S(\boldsymbol{v})\boldsymbol{\epsilon}\frac{\boldsymbol{\epsilon}^T}{\sqrt{1-\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}}} + \boldsymbol{v}^T\boldsymbol{\epsilon} I_{3\times3} + \boldsymbol{\epsilon}\boldsymbol{v}^T - \boldsymbol{v}2\boldsymbol{\epsilon}^T \right\} \\ =& + 2\sqrt{1-\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}} S(\boldsymbol{v}) - \frac{2}{\sqrt{1-\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}}} S(\boldsymbol{v})\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T + 2\boldsymbol{v}^T\boldsymbol{\epsilon} I_{3\times3} + 2\boldsymbol{\epsilon}\boldsymbol{v}^T - 4\boldsymbol{v}\boldsymbol{\epsilon}^T \end{aligned}$$

This expression is derived by hand, but it is double checked with Matlab to verify the correctness. The updated discrete-time matrices becomes

$$\begin{aligned} R_k &\approx \sigma^2_{v_k} I_{6\times6} \\ H_k &= \left.\frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{x}_k}\right|_{\boldsymbol{x}_k = \bar{\boldsymbol{x}}_k} = \begin{bmatrix} W(\delta\bar{\boldsymbol{\epsilon}}_k, R^n_{\hat{b}}(\hat{\boldsymbol{q}})^T \mathbf{m}^n) & 0_{3\times3} \\ -W(\delta\bar{\boldsymbol{\epsilon}}_k, R^n_{\hat{b}}(\hat{\boldsymbol{q}})^T \boldsymbol{g}^n) & 0_{3\times3} \end{bmatrix} \end{aligned}$$

5.3 Position, Velocity and Attitude Estimation

In this section we want to design a discrete multiplicative extended Kalman filter for position, velocity and attitude estimation. This will be done by expanding the attitude filter from previous section to include position and velocity as well. By integrating GPS measurements and accelerometer measurements we achieve two things:

- Low-pass filtering of the GPS measurements
- High-pass filtering of the accelerometer measurements

5.3.1 Assumptions

When introducing GPS measurements, the assumptions about constant speed and zero accelerometer bias made in previous section can be removed.

5.3.2 Position, Velocity and Attitude Models

The state variables in the filter are chosen to be $\boldsymbol{x} = [\delta\boldsymbol{\epsilon},\, \boldsymbol{b}^b_{\text{gyro}},\, \boldsymbol{p}^n_{b/n},\, \boldsymbol{v}^n_{b/n},\, \boldsymbol{b}^b_{\text{acc}}]$ and the input is $\boldsymbol{u} = [\boldsymbol{\omega}^b_{\text{imu}},\, \boldsymbol{f}^b_{\text{imu}}]$. From (3.9) and by inserting (5.1) to (3.8) we get a model for the position and velocity. Together with the attitude model (5.11) we can write the full model as

$$\underbrace{\begin{bmatrix} \delta\dot{\boldsymbol{\epsilon}} \\ \dot{\boldsymbol{b}}^b_{\text{gyro}} \\ \dot{\boldsymbol{p}}^n_{b/n} \\ \dot{\boldsymbol{v}}^n_{b/n} \\ \dot{\boldsymbol{b}}^b_{\text{acc}} \end{bmatrix}}_{\dot{\boldsymbol{x}}} = \underbrace{\begin{bmatrix} \frac{1}{2}[I_{3\times3}\sqrt{1-\delta\boldsymbol{\epsilon}^T\delta\boldsymbol{\epsilon}} + S(\delta\boldsymbol{\epsilon})][\boldsymbol{\omega}^b_{\text{imu}} - \boldsymbol{b}^b_{\text{gyro}}] \\ 0_{3\times1} \\ \boldsymbol{v}^n_{b/n} \\ R^n_{\hat{b}}(\hat{\boldsymbol{q}})R^{\hat{b}}_b(\delta\boldsymbol{q})[\boldsymbol{f}^b_{\text{imu}} - \boldsymbol{b}^b_{\text{acc}}] + \boldsymbol{g}^n \\ 0_{3\times1} \end{bmatrix}}_{\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})} + \underbrace{I_{15\times15}}_{\Gamma} \underbrace{\begin{bmatrix} \boldsymbol{w}_{\text{gyro}} \\ \boldsymbol{w}_{\text{bgyro}} \\ 0_{3\times1} \\ \boldsymbol{w}_{\text{acc}} \\ \boldsymbol{w}_{\text{bacc}} \end{bmatrix}}_{\boldsymbol{w}}$$

where $\boldsymbol{w} \in \mathbb{R}^{15}$ is Gaussian white noise with zero mean and variance σ^2_w.

5.3.3 Measurement Equation

By using the q-method, we get a measurement of the state $\delta\boldsymbol{\epsilon}$ (recall from section 5.2.3). A feed forward term from the states is needed to cancel $R^b_n\dot{\boldsymbol{v}}^n_{b/n} + \boldsymbol{b}^b_{\text{acc}}$ in (3.8) before using the q-method. Expanding (5.12) with GPS position (3.14) and velocity (3.15) measurement, then inserting (5.1), gives the following measurement equation

$$\boldsymbol{y} = \begin{bmatrix} \delta\boldsymbol{\epsilon}_{\text{qmetod}} \\ \boldsymbol{p}^n_{\text{gps}} \\ \boldsymbol{v}^n_{\text{gps}} \end{bmatrix} = \underbrace{\begin{bmatrix} \delta\boldsymbol{\epsilon} \\ \boldsymbol{p}^n_{b/n} + R^n_{\hat{b}}(\hat{\boldsymbol{q}})R^{\hat{b}}_b(\delta\boldsymbol{q})\boldsymbol{r}^b_{\text{ant}} \\ \boldsymbol{v}^n_{b/n} + R^n_{\hat{b}}(\hat{\boldsymbol{q}})R^{\hat{b}}_b(\delta\boldsymbol{q})S(\boldsymbol{\omega}^b_{\text{imu}} - \boldsymbol{b}^b_{\text{gyro}})\boldsymbol{r}^b_{\text{ant}} \end{bmatrix}}_{\boldsymbol{h}(\boldsymbol{x})} + \underbrace{\begin{bmatrix} \boldsymbol{w}_\epsilon \\ \boldsymbol{w}_{\text{pos}} \\ \boldsymbol{w}_{\text{vel}} \end{bmatrix}}_{\boldsymbol{v}} \tag{5.18}$$

where $\boldsymbol{v} \in \mathbb{R}^9$ is Gaussian white noise with zero mean and variance σ_v^2.

5.3.4 Discrete-Time Matrices

Discretizing the system with the Euler method gives

$$\boldsymbol{x}_{k+1} = \underbrace{\boldsymbol{x}_k + h\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k)}_{\boldsymbol{f}_k(\boldsymbol{x}_k,\boldsymbol{u}_k)} + \underbrace{h\Gamma}_{\Gamma_k} \boldsymbol{w}_k \tag{5.19}$$

$$\boldsymbol{y}_k = \underbrace{\boldsymbol{h}(\boldsymbol{x}_k)}_{\boldsymbol{h}_k(\boldsymbol{x}_k)} + \boldsymbol{v}_k \tag{5.20}$$

We need to develop an expression for the linearized model Φ_k. This involves linearizing the rotation matrix. It looks a lot like the expression in (5.17), but this time it is not the *transposed* rotation matrix. To make it easier to calculate the Jacobian Φ_k we can define

$$\begin{aligned} V(\boldsymbol{\epsilon}, \boldsymbol{v}) :=& \frac{\partial}{\partial \boldsymbol{\epsilon}} \left\{ R(\delta\boldsymbol{q}(\boldsymbol{\epsilon}))\boldsymbol{v} \right\} \\ =& \frac{\partial}{\partial \boldsymbol{\epsilon}} \left\{ [I_{3\times3} + 2\sqrt{1-\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}}S(\boldsymbol{\epsilon}) + 2\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T - 2\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}I_{3\times3}]\boldsymbol{v} \right\} \\ =& 2\frac{\partial}{\partial \boldsymbol{\epsilon}} \left\{ \frac{1}{2}\boldsymbol{v} - \sqrt{1-\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}}S(\boldsymbol{v})\boldsymbol{\epsilon} + (\boldsymbol{v}^T\boldsymbol{\epsilon})\boldsymbol{\epsilon} - (\boldsymbol{\epsilon}^T\boldsymbol{\epsilon})\boldsymbol{v} \right\} \\ =& 2\left\{ -\sqrt{1-\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}}S(\boldsymbol{v}) + S(\boldsymbol{v})\boldsymbol{\epsilon}\frac{\boldsymbol{\epsilon}^T}{\sqrt{1-\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}}} + \boldsymbol{v}^T\boldsymbol{\epsilon}I_{3\times3} + \boldsymbol{\epsilon}\boldsymbol{v}^T - \boldsymbol{v}2\boldsymbol{\epsilon}^T \right\} \\ =& -2\sqrt{1-\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}}S(\boldsymbol{v}) + \frac{2}{\sqrt{1-\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}}}S(\boldsymbol{v})\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T + 2\boldsymbol{v}^T\boldsymbol{\epsilon}I_{3\times3} + 2\boldsymbol{\epsilon}\boldsymbol{v}^T - 4\boldsymbol{v}\boldsymbol{\epsilon}^T \end{aligned} \tag{5.21}$$

Now consider the case for $\boldsymbol{\epsilon} = 0$, that gives

$$V(0, \boldsymbol{v}) = -2S(\boldsymbol{v}) \tag{5.22}$$

which is exactly the case we get when calculating $\partial \dot{\boldsymbol{v}}^n_{b/n}/\partial\delta\boldsymbol{\epsilon}_k$, after the filter property $\delta\hat{\boldsymbol{\epsilon}}_k = 0$ is inserted. The discrete-time matrices needed in the filter becomes

$$
\begin{aligned}
Q_k &\approx \sigma^2_{w_k} I_{15\times15}\\
R_k &\approx \sigma^2_{v_k} I_{9\times9}\\
\boldsymbol{f}_k(\hat{\boldsymbol{x}}_k, \boldsymbol{u}_k) &= \hat{\boldsymbol{x}}_k + h\begin{bmatrix} \frac{1}{2}[\boldsymbol{\omega}^b_{\text{imu}} - \hat{\boldsymbol{b}}^b_{\text{gyro}}] \\ 0_{3\times1} \\ \hat{\boldsymbol{v}}^n_{b/n} \\ R^n_b(\hat{\boldsymbol{q}})[\boldsymbol{f}^b_{\text{imu}} - \hat{\boldsymbol{b}}^b_{\text{acc}}] + \boldsymbol{g}^n \\ 0_{3\times1} \end{bmatrix}\\
\Phi_k &= \left.\frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{x}_k}\right|_{\boldsymbol{x}_k = \hat{\boldsymbol{x}}_k} =\\
I_{15\times15} + h &\begin{bmatrix} -\frac{1}{2}S(\boldsymbol{\omega}^b_{\text{imu}} - \hat{\boldsymbol{b}}^b_{\text{gyro}}) & -\frac{1}{2}I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_{3\times3} & 0_{3\times3} \\ -2R^n_b(\hat{\boldsymbol{q}})S(\boldsymbol{f}^b_{\text{imu}} - \hat{\boldsymbol{b}}^b_{\text{acc}}) & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & -R^n_b(\hat{\boldsymbol{q}}) \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix}\\
\Gamma_k &= hI_{15\times15}\\
H_k &= \left.\frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{x}_k}\right|_{\boldsymbol{x}_k = \bar{\boldsymbol{x}}_k} =
\end{aligned}
$$

$$
\begin{bmatrix} I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ R^n_{\hat{b}}(\hat{\boldsymbol{q}})V(\delta\bar{\boldsymbol{\epsilon}}_k, \boldsymbol{r}^b_{\text{ant}}) & 0_{3\times3} & I_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ R^n_{\hat{b}}(\hat{\boldsymbol{q}})V(\delta\bar{\boldsymbol{\epsilon}}_k, S(\boldsymbol{\omega}^b_{\text{imu}} - \bar{\boldsymbol{b}}^b_{\text{gyro}})\boldsymbol{r}^b_{\text{ant}}) & R^n_{\hat{b}}(\hat{\boldsymbol{q}})R^{\hat{b}}_b(\delta\bar{\boldsymbol{q}})S(\boldsymbol{r}^b_{\text{ant}}) & 0_{3\times3} & I_{3\times3} & 0_{3\times3} \end{bmatrix}
$$

where the filter property that $\delta\hat{\boldsymbol{\epsilon}}_k = 0$ has been used.

5.3.5 Alternative Measurement Equation

Instead of using the q-method, we can use the magnetometer (3.11), accelerometer (3.8), GPS position (3.14) and velocity (3.15) measurement directly in the measurement equation. After inserting (5.1) it looks like this

$$
\boldsymbol{y} = \begin{bmatrix} \mathbf{m}^b_{\text{imu}} \\ \boldsymbol{f}^b_{\text{imu}} \\ \boldsymbol{p}^n_{\text{gps}} \\ \boldsymbol{v}^n_{\text{gps}} \end{bmatrix} = \underbrace{\begin{bmatrix} R^{\hat{b}}_b(\delta\boldsymbol{q})^T R^n_{\hat{b}}(\hat{\boldsymbol{q}})^T \mathbf{m}^n \\ R^{\hat{b}}_b(\delta\boldsymbol{q})^T R^n_{\hat{b}}(\hat{\boldsymbol{q}})^T[\dot{\boldsymbol{v}}^n_{b/n} - \boldsymbol{g}^n] + \boldsymbol{b}^b_{\text{acc}} \\ \boldsymbol{p}^n_{b/n} + R^n_{\hat{b}}(\hat{\boldsymbol{q}})R^{\hat{b}}_b(\delta\boldsymbol{q})\boldsymbol{r}^b_{\text{ant}} \\ \boldsymbol{v}^n_{b/n} + R^n_{\hat{b}}(\hat{\boldsymbol{q}})R^{\hat{b}}_b(\delta\boldsymbol{q})S(\boldsymbol{\omega}^b_{\text{imu}} - \boldsymbol{b}^b_{\text{gyro}})\boldsymbol{r}^b_{\text{ant}} \end{bmatrix}}_{\boldsymbol{h}(\boldsymbol{x})} + \underbrace{\begin{bmatrix} \boldsymbol{w}_{\text{mag}} \\ \boldsymbol{w}_{\text{acc}} \\ \boldsymbol{w}_{\text{pos}} \\ \boldsymbol{w}_{\text{vel}} \end{bmatrix}}_{\boldsymbol{v}} \tag{5.23}
$$

where $\boldsymbol{v} \in \mathbb{R}^{12}$ is Gaussian white noise with zero mean and variance σ_v^2. The updated discrete-time matrices becomes

$$
\begin{aligned}
R_k &\approx \sigma_{v_k}^2 I_{12\times 12} \\
H_k &= \left.\frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{x}_k}\right|_{\boldsymbol{x}_k = \bar{\boldsymbol{x}}_k} =
\end{aligned}
$$

$$
\begin{bmatrix}
W(\delta\bar{\boldsymbol{\epsilon}}_k, R_{\hat{b}}^n(\hat{\boldsymbol{q}})^T \mathbf{m}^n) & 0_{3\times 3} & 0_{3\times 3} & 0_{3\times 3} & 0_{3\times 3} \\
-W(\delta\bar{\boldsymbol{\epsilon}}_k, R_{\hat{b}}^n(\hat{\boldsymbol{q}})^T \boldsymbol{g}^n) & 0_{3\times 3} & 0_{3\times 3} & 0_{3\times 3} & I_{3\times 3} \\
R_{\hat{b}}^n(\hat{\boldsymbol{q}}) V(\delta\bar{\boldsymbol{\epsilon}}_k, \boldsymbol{r}_{\text{ant}}^b) & 0_{3\times 3} & I_{3\times 3} & 0_{3\times 3} & 0_{3\times 3} \\
R_{\hat{b}}^n(\hat{\boldsymbol{q}}) V(\delta\bar{\boldsymbol{\epsilon}}_k, S(\boldsymbol{\omega}_{\text{imu}}^b - \bar{\boldsymbol{b}}_{\text{gyro}}^b)\boldsymbol{r}_{\text{ant}}^b) & R_{\hat{b}}^n(\hat{\boldsymbol{q}}) R_b^{\hat{b}}(\delta\bar{\boldsymbol{q}}) S(\boldsymbol{r}_{\text{ant}}^b) & 0_{3\times 3} & I_{3\times 3} & 0_{3\times 3}
\end{bmatrix}
$$

Chapter 6

Nonlinear Observer Design

Tuning of the Kalman filter is a difficult and time consuming task. The elements in the measurement covariance matrix R can often be found in the data sheet for the sensors, but the elements in the process covariance matrix Q is often difficult to relate to physical quantities. Also the Riccati equation is computational extensive. This is some of the reasons why there has been an increasing interest for nonlinear observer design the last decades (Fossen 2011).

In this chapter a nonlinear observer for integration of IMU and GPS is derived. The observer is made of a combination of two nonlinear algorithms recently proposed by Hua (2010) and Mahony et al. (2008). Later the performance of the nonlinear observer will be compared with the extended Kalman filter to see whether it has competitive results.

6.1 Attitude Observer

Mahony et al. (2008) has proposed an observer, termed the *explicit complementary filter*, that provides attitude estimates as well as gyro bias estimates. Vectorial measurements, such as gravitational and magnetic field directions, are used directly without algebraic reconstruction of the attitude. It remains well conditioned in the case where only a single vector direction is measured. The observer is well suited for implementation on embedded hardware platforms. Local exponential and almost global stability of the observer error dynamics have been shown with Lyapunov analysis, for 2 or more independent inertial directions. The accelerometer measurement is approximated to be a gravity measurement ($\dot{\boldsymbol{v}}^n_{b/n} \approx 0$). The quaternion representation of the *explicit complementary filter* by Mahony et al. (2008) is:

$$\boldsymbol{\omega}^b_{\text{mes}} = -\text{vex}\left(\sum_{i=1}^{n} \frac{k_i}{2}\left(\boldsymbol{v}_i^b(\hat{\boldsymbol{v}}_i^b)^T - \hat{\boldsymbol{v}}_i^b(\boldsymbol{v}_i^b)^T\right)\right) \tag{6.1}$$

$$\dot{\hat{\boldsymbol{q}}} = \frac{1}{2}T(\hat{\boldsymbol{q}})\left[\boldsymbol{\omega}^b_{\text{imu}} - \hat{\boldsymbol{b}}^b_{\text{gyro}} + K_{\text{quat}}\boldsymbol{\omega}^b_{\text{mes}}\right] \tag{6.2}$$

$$\dot{\hat{\boldsymbol{b}}}^b_{\text{gyro}} = -K_{\text{bgyr}}\boldsymbol{\omega}^b_{\text{mes}} \tag{6.3}$$

where the operator vex is the inverse of the cross-product operator

$$\boldsymbol{a} \times \boldsymbol{b} = S(\boldsymbol{a})\boldsymbol{b} \tag{6.4}$$

$$\text{vex}(S(\boldsymbol{a})) = \boldsymbol{a} \tag{6.5}$$

The following expression corresponding to (6.1) is derived in Fossen (2011):

$$\text{vex}(\boldsymbol{a}\boldsymbol{b}^T - \boldsymbol{b}\boldsymbol{a}^T) = \begin{bmatrix} a_3b_2 - a_2b_3 \\ a_1b_3 - a_3b_1 \\ a_2b_1 - a_1b_2 \end{bmatrix} \tag{6.6}$$

Moreover $\boldsymbol{v}_i^b$ represent normalized vectorial measurements, $\hat{\boldsymbol{v}}_i^b$ represent normalized estimated vectorial measurements

$$\boldsymbol{v}_1^b = \frac{\boldsymbol{m}^b_{\text{imu}}}{\|\boldsymbol{m}^b_{\text{imu}}\|} \tag{6.7}$$

$$\boldsymbol{v}_2^b = \frac{\boldsymbol{f}^b_{\text{imu}}}{\left\|\boldsymbol{f}^b_{\text{imu}}\right\|} \tag{6.8}$$

$$\hat{\boldsymbol{v}}_1^b = R_b^n(\hat{\boldsymbol{q}})^T\frac{\boldsymbol{m}^n}{\|\boldsymbol{m}^n\|} \tag{6.9}$$

$$\hat{\boldsymbol{v}}_2^b = -R_b^n(\hat{\boldsymbol{q}})^T\frac{\boldsymbol{g}^n}{\|\boldsymbol{g}^n\|} \tag{6.10}$$

and $k_i = \{k_{\text{mag}}, k_{\text{acc}}\} > 0$ is the weighting between the sensors. $K_{\text{quat}}, K_{\text{bgyr}} > 0$ are positive gain matrices.

6.1.1 Discrete-Time Corrector-Predictor Formulation

In order to handle different measurement sampling rates and dead-reckoning it is advantageous to write the observer on a EKF-like corrector-predictor form. Discretizing the explicit complementary filter by Mahony et al. (2008) using Euler integration, gives the following vectorial measurements

$$\boldsymbol{v}_1^b = \frac{\boldsymbol{m}_{\text{imu}}^b(k)}{\left\|\boldsymbol{m}_{\text{imu}}^b(k)\right\|} \tag{6.11}$$

$$\boldsymbol{v}_2^b = \frac{\boldsymbol{f}_{\text{imu}}^b(k)}{\left\|\boldsymbol{f}_{\text{imu}}^b(k)\right\|} \tag{6.12}$$

$$\bar{\boldsymbol{v}}_1^b = R_b^n(\bar{\boldsymbol{q}}(k))^T \frac{\boldsymbol{m}^n}{\|\boldsymbol{m}^n\|} \tag{6.13}$$

$$\bar{\boldsymbol{v}}_2^b = -R_b^n(\bar{\boldsymbol{q}}(k))^T \frac{\boldsymbol{g}^n}{\|\boldsymbol{g}^n\|} \tag{6.14}$$

and the corrector becomes

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

$$\hat{\boldsymbol{q}}(k) = \bar{\boldsymbol{q}}(k) + h \cdot \frac{1}{2}T(\bar{\boldsymbol{q}}(k))K_{\text{quat}}\boldsymbol{\omega}_{\text{mes}}^b \tag{6.16}$$

$$\hat{\boldsymbol{b}}_{\text{gyro}}^b(k) = \bar{\boldsymbol{b}}_{\text{gyro}}^b(k) - h \cdot K_{\text{bgyr}}\boldsymbol{\omega}_{\text{mes}}^b \tag{6.17}$$

and the predictor becomes

$$\bar{\boldsymbol{q}}(k+1) = \hat{\boldsymbol{q}}(k) + h \cdot \frac{1}{2}T(\hat{\boldsymbol{q}}(k))\left[\boldsymbol{\omega}_{\text{imu}}^b(k) - \hat{\boldsymbol{b}}_{\text{gyro}}^b(k)\right] \tag{6.18}$$

$$\bar{\boldsymbol{b}}_{\text{gyro}}^b(k+1) = \hat{\boldsymbol{b}}_{\text{gyro}}^b(k) \tag{6.19}$$

6.2 Position, Velocity and Attitude Observer

Hua (2010) has recently proposed an advanced observer, that provides estimates of attitude and velocity. The observer makes use of accelerometer, magnetometer and linear velocity measurements, which makes it better adapted for vehicles subjected to important linear accelerations. Stability analysis results state semi-global convergence. The observer proposed by Hua (2010) is

$$\dot{\hat{\boldsymbol{v}}}^n_{b/n} = Q\boldsymbol{f}^b_{\text{imu}} + \boldsymbol{g}^n + k_{\text{vel}}(\boldsymbol{v}^n_{\text{gps}} - \hat{\boldsymbol{v}}^n_{b/n}) \tag{6.20}$$

$$\dot{Q} = QS(\boldsymbol{\omega}^b_{\text{imu}}) - \rho Q + k_Q(\boldsymbol{v}^n_{\text{gps}} - \hat{\boldsymbol{v}}^n_{b/n})(\boldsymbol{f}^b_{\text{imu}})^T \tag{6.21}$$

$$\rho = k_{\text{norm}}\max(0, \|Q\| - \sqrt{3}) \tag{6.22}$$

$$\dot{\hat{R}}^n_b = \hat{R}^n_b S(\boldsymbol{\omega}^b_{\text{imu}} + \boldsymbol{\sigma}) \tag{6.23}$$

$$\boldsymbol{\sigma} = k_{\text{mag}}\boldsymbol{m}^b_{\text{imu}} \times (\hat{R}^n_b)^T\boldsymbol{m}^n + k_{\text{acc}}\boldsymbol{f}^b_{\text{imu}} \times (\hat{R}^n_b)^T(Q\boldsymbol{f}^b_{\text{imu}} + k_{\text{vel}}(\boldsymbol{v}^n_{\text{gps}} - \hat{\boldsymbol{v}}^n_{b/n})) \tag{6.24}$$

where $k_{\text{vel}}, k_{\text{mag}}, k_{\text{acc}}, k_Q, k_{\text{norm}}$ is some positive constant gains, and $Q \in \mathbb{R}^{3\times3}$ is a virtual matrix that allows an estimation of the specific acceleration in the inertial frame $\{n\}$. It is not a rotation matrix but still such that $Q\boldsymbol{f}^b_{\text{imu}} - R^n_b\boldsymbol{f}^b_{\text{imu}}$ tends to zero. $\|Q\|$ is the Frobenius norm, i.e. $\|Q\| = \sqrt{\text{tr}(Q^TQ)}$. One can view $Q\boldsymbol{f}^b_{\text{imu}} + k_{\text{vel}}(\boldsymbol{v}^n_{\text{gps}} - \hat{\boldsymbol{v}}^n_{b/n})$ as an estimate of $\boldsymbol{f}^n$.

The observer of Hua (2010) does not provide estimates of the gyro bias. But as the observer of Mahony et al. (2008) has gyro bias estimation, the attitude part, (6.23) and (6.24), can be removed and replaced with the attitude observer of Mahony et al. (2008). This will be a combination of the advantage of the linear acceleration estimation from Hua together with gyro bias estimation from Mahony. It is straight forward to modify the observer to include position as well by adding

$$\dot{\hat{\boldsymbol{p}}}^n_{b/n} = \boldsymbol{v}^n_{b/n} + k_{\text{pos}}(\boldsymbol{p}^n_{\text{gps}} - \hat{\boldsymbol{p}}^n_{b/n}) \tag{6.25}$$

where k_{pos} is a positive gain.

6.2.1 Discrete-Time Corrector-Predictor Formulation

In the following the discrete-time corrector-predictor formulation for the combined Hua (2010) and Mahony et al. (2008) observer will be derived using Euler integration. As the specific acceleration is estimated the vectorial measurements will be

$$\boldsymbol{v}_1^b = \frac{\boldsymbol{m}_{\text{imu}}^b(k)}{\left\|\boldsymbol{m}_{\text{imu}}^b(k)\right\|} \tag{6.26}$$

$$\boldsymbol{v}_2^b = \frac{\boldsymbol{f}_{\text{imu}}^b(k)}{\left\|\boldsymbol{f}_{\text{imu}}^b(k)\right\|} \tag{6.27}$$

$$\bar{\boldsymbol{v}}_1^b = R_b^n(\bar{\boldsymbol{q}}(k))^T \frac{\boldsymbol{m}^n}{\|\boldsymbol{m}^n\|} \tag{6.28}$$

$$\bar{\boldsymbol{v}}_2^b = R_b^n(\bar{\boldsymbol{q}}(k))^T \frac{\bar{Q}(k)\boldsymbol{f}_{\text{imu}}^b(k) + k_{\text{vel}}(\boldsymbol{v}_{\text{gps}}^n(k) - \bar{\boldsymbol{v}}_{b/n}^n(k))}{\left\|\bar{Q}(k)\boldsymbol{f}_{\text{imu}}^b(k) + k_{\text{vel}}(\boldsymbol{v}_{\text{gps}}^n(k) - \bar{\boldsymbol{v}}_{b/n}^n(k))\right\|} \tag{6.29}$$

and the corrector becomes

$$\hat{\boldsymbol{p}}_{b/n}^n(k) = \bar{\boldsymbol{p}}_{b/n}^n(k) + h \cdot k_{\text{pos}}(\boldsymbol{p}_{\text{gps}}^n(k) - \bar{\boldsymbol{p}}_{b/n}^n(k)) \tag{6.30}$$

$$\hat{\boldsymbol{v}}_{b/n}^n(k) = \bar{\boldsymbol{v}}_{b/n}^n(k) + h \cdot k_{\text{vel}}(\boldsymbol{v}_{\text{gps}}^n(k) - \bar{\boldsymbol{v}}_{b/n}^n(k)) \tag{6.31}$$

$$\hat{Q}(k) = \bar{Q}(k) + h \cdot k_Q(\boldsymbol{v}_{\text{gps}}^n(k) - \bar{\boldsymbol{v}}_{b/n}^n(k))(\boldsymbol{f}_{\text{imu}}^b(k))^T \tag{6.32}$$

together with (6.15), (6.16) and (6.17). The predictor becomes

$$\bar{\boldsymbol{p}}_{b/n}^n(k+1) = \hat{\boldsymbol{p}}_{b/n}^n(k) + h \cdot \hat{\boldsymbol{v}}_{b/n}^n(k) \tag{6.33}$$

$$\bar{\boldsymbol{v}}_{b/n}^n(k+1) = \hat{\boldsymbol{v}}_{b/n}^n(k) + h \cdot (\hat{Q}(k)\boldsymbol{f}_{\text{imu}}^b(k) + \boldsymbol{g}^n) \tag{6.34}$$

$$\bar{Q}(k+1) = \hat{Q}(k) + h \cdot \hat{Q}(k)S(\boldsymbol{\omega}_{\text{imu}}^b(k)) \tag{6.35}$$

together with (6.18) and (6.19). The "normalization" term ρQ has been removed. It should be replaced by the discrete variant

$$Q(k) = \frac{Q(k)}{\|Q(k)\|}\sqrt{3} \tag{6.36}$$

and used after any calculations on Q.

Chapter 7

Simulator

It is convenient to have a simulator that can generate IMU and GPS measurements during development, testing and verification of the nonlinear observers. For instance faults can be simulated to test the observer error handling. This chapter contains a unmanned aerial vehicle (UAV) model as well as sensor and navigation system models. The simulator has been implemented in Simulink.

7.1 Simulator Model

This section contains the equations needed to implement the simulator that generates IMU and GPS measurements. The 6 DOF kinematic equations are given by Fossen (2011):

$$\dot{\boldsymbol{p}}^n_{b/n} = R^n_b(\boldsymbol{q})\boldsymbol{v}^b_{b/n} \tag{7.1}$$

$$\dot{\boldsymbol{q}} = \frac{1}{2}T(\boldsymbol{q})\boldsymbol{\omega}^b_{b/n} + \frac{\gamma}{2}(1 - \boldsymbol{q}^T\boldsymbol{q})\boldsymbol{q} \tag{7.2}$$

where a feedback term has been added to (7.2). The feedback term drives the quaternion to a unit quaternion in order to compensate for numerical round of errors. The normalization gain can be set to $\gamma = 100$. The 6 DOF equations of motion are given by Fossen (2011):

$$m[\dot{\boldsymbol{v}}^b_{b/n} + S(\boldsymbol{\omega}^b_{b/n})\boldsymbol{v}^b_{b/n}] + D_1\boldsymbol{v}^b_{b/n} = \boldsymbol{f}^b \tag{7.3}$$

$$I_b\dot{\boldsymbol{\omega}}^b_{b/n} + S(\boldsymbol{\omega}^b_{b/n})I_b\boldsymbol{\omega}^b_{b/n} + D_2\boldsymbol{\omega}^b_{b/n} = \boldsymbol{m}^b \tag{7.4}$$

where $\boldsymbol{f}^b$ and $\boldsymbol{m}^b$ is forces and moments respectively, D_1, D_2 are linear damping matrices and I_b is the inertia matrix. The center of origin (CO) for the body frame has been chosen such that it coincides with the center of gravity (CG) . The gyro measurement is modeled according to (3.3) and (3.4)

$$\begin{aligned} \boldsymbol{y}_{\text{gyro}} &= \boldsymbol{\omega}^b_{b/n} + \boldsymbol{b}^b_{\text{gyro}} + \boldsymbol{w}_{\text{gyro}} && (7.5) \\ \dot{\boldsymbol{b}}^b_{\text{gyro}} &= \boldsymbol{w}_{\text{bgyro}} && (7.6) \end{aligned}$$

The accelerometer model is given by (3.7) and (3.9)

$$\begin{aligned} \boldsymbol{y}_{\text{acc}} &= \boldsymbol{a}^b_{b/n} - R^n_b(\boldsymbol{q})^T \boldsymbol{g}^n + \boldsymbol{b}^b_{\text{acc}} + \boldsymbol{w}_{\text{acc}}, \qquad \boldsymbol{a}^b_{b/n} = \dot{\boldsymbol{v}}^b_{b/n} + S(\boldsymbol{\omega}^b_{b/n})\boldsymbol{v}^b_{b/n} && (7.7) \\ \dot{\boldsymbol{b}}^b_{\text{acc}} &= \boldsymbol{w}_{\text{bacc}} && (7.8) \end{aligned}$$

The magnetometer model is given by (3.11)

$$\boldsymbol{y}_{\text{mag}} = R^n_b(\boldsymbol{q})^T \frac{\boldsymbol{m}^n}{\|\boldsymbol{m}^n\|} + \boldsymbol{b}^b_{\text{mag}} + \boldsymbol{w}_{\text{mag}} \tag{7.9}$$

The GPS model is given by (3.14) and (3.15)

$$\begin{aligned} \boldsymbol{y}_{\text{pos}} &= \boldsymbol{p}^n_{b/n} + R^n_b(\boldsymbol{q})\boldsymbol{r}^b_{\text{ant}} + \boldsymbol{w}_{\text{pos}} && (7.10) \\ \boldsymbol{y}_{\text{vel}} &= \boldsymbol{v}^n_{b/n} + R^n_b(\boldsymbol{q})S(\boldsymbol{\omega}^b_{b/n})\boldsymbol{r}^b_{\text{ant}} + \boldsymbol{w}_{\text{vel}} && (7.11) \end{aligned}$$

7.2 Simulator Parameters and Measurement Noise

The simulator parameters can be set in the various parameter dialog boxes. See Figure 7.2 for an example of the gyro parameters dialog box. For simplicity the following parameters are used for the vehicle model

$$m = 1, \quad I_b = I_{3\times3}, \quad D_1 = I_{3\times3}, \quad D_2 = I_{3\times3} \tag{7.12}$$

The simulator has three different options for the measurement noise $\boldsymbol{w}_{\text{gyro}}$, $\boldsymbol{w}_{\text{acc}}$, $\boldsymbol{w}_{\text{mag}}$, $\boldsymbol{w}_{\text{pos}}$ and $\boldsymbol{w}_{\text{vel}}$ (see Figure 7.2):

Off The measurement noise is off, i.e. $\boldsymbol{w} = 0$

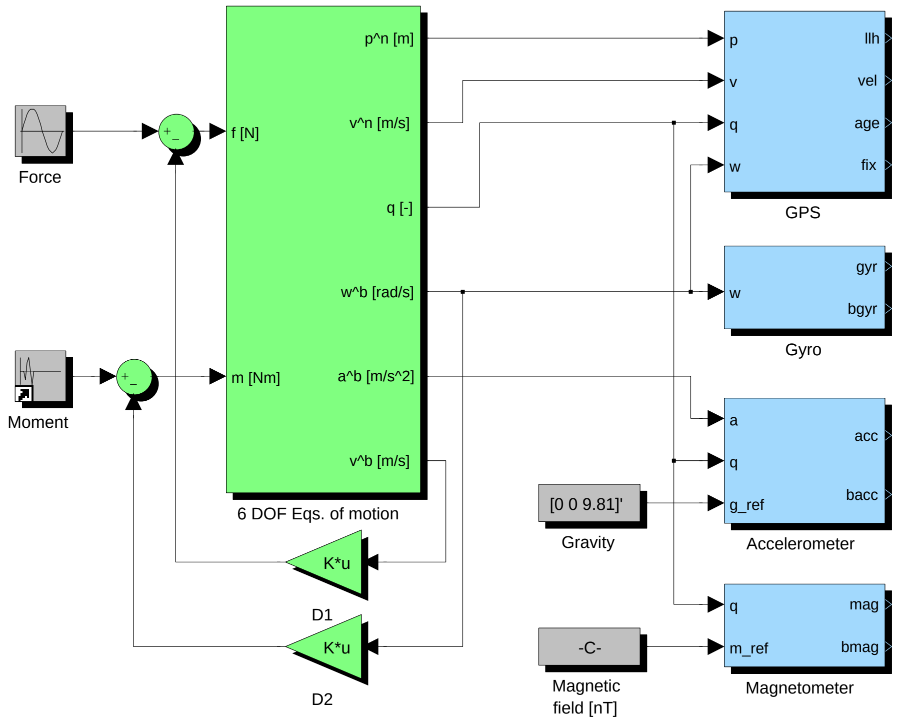
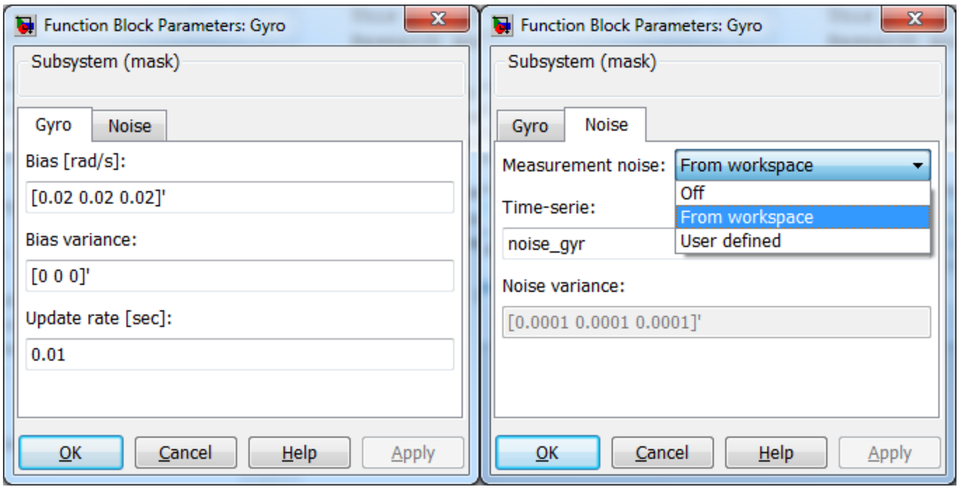


Figure 7.1: Simulator model implemented in Simulink



Bias [rad/s]: [0.02 0.02 0.02]'

Bias variance: [0 0 0]'

Update rate [sec]: 0.01

Measurement noise: From workspace

Time-serie: noise_gyr

Noise variance: [0.0001 0.0001 0.0001]'

Figure 7.2: Gyro parameters dialog box

From workspace Use time-series with only noise from the workspace. Time-series have been recorded with the Xsens MTi-G, without motion or rotation, afterwards the mean value was subtracted.

User defined Use white noise with a user defined variance.

The measurement update rate can also be set in the dialog box. The update rates are chosen similar to the Xsens MTi-G unit, which is 100 Hz for the IMU and 4 Hz for the GPS. Data is acquired at a rate of 100 Hz, consequently there will be 25 equal GPS samples before new data is received. Values used for the magnetic field, gravity and lever arm are:

$$\begin{aligned}
\boldsymbol{m}^n &= [\ 13\,605 \quad 439 \quad 49\,864\ ]^T \text{ nT} \\
\boldsymbol{g}^n &= [\ 0 \quad 0 \quad 9.81\ ]^T \text{ m/s}^2 \\
\boldsymbol{r}^b_{\text{ant}} &= [\ -0.65 \quad -0.08 \quad -0.90\ ]^T \text{ m}
\end{aligned}$$

Chapter 8

Implementation

The nonlinear observers developed in this thesis have been implemented in Matlab. The *Workspace* in Matlab have been used to interface the measurement time-series. This makes it easy to switch between simulated data and experimental data without doing any changes. When it comes to implementation, there are some things that need to be addressed. This chapter highlights some implementation considerations as well as error handling for the nonlinear observers in this thesis.

8.1 Numerical Properties of Different Attitude Representations

The attitude can be represented in several ways, i.e. quaternion or rotation matrix. They have different numerical properties when dicretized. Discretizing the differential equation for the quaternion and the rotation matrix using the Euler method gives

$$\hat{\boldsymbol{q}}(k+1) = \hat{\boldsymbol{q}}(k) + h \cdot \frac{1}{2} T(\hat{\boldsymbol{q}}(k)) \boldsymbol{\omega}^b_{b/n}(k) \tag{8.1}$$

$$\hat{R}^n_b(k+1) = \hat{R}^n_b(k) + h \cdot \hat{R}^n_b(k) S(\boldsymbol{\omega}^b_{b/n}(k)) \tag{8.2}$$

Numerical round-off errors will cause a violation of the unit constraint on the quaternion. To ensure that the constraint is satisfied the following normalization procedure can be used

$$\hat{\boldsymbol{q}}(k+1) = \frac{\hat{\boldsymbol{q}}(k+1)}{\|\hat{\boldsymbol{q}}(k+1)\|} \tag{8.3}$$

Table 8.1: Attitude errors (RMS) for the quaternion and rotation matrix representation due to Euler integration. Perfect data was used in the test.

	roll (deg)	pitch (deg)	yaw (deg)
Quaternion propagation	0.5481	0.3001	0.5183
Rotation matrix propagation	2.8368	1.8804	5.2826

Maintaining the constraints for the rotation matrix (orthogonal and det=1) is more difficult, but the following will help

$$\hat{R}_b^n(k+1) = \frac{\hat{R}_b^n(k+1)}{\left\|\hat{R}_b^n(k+1)\right\|}\sqrt{3} \tag{8.4}$$

where $\|\cdot\|$ in (8.4) is the Frobenius norm. The Frobenius norm of a rotation matrix is always $\sqrt{3}$. Table 8.1 contains the test results from Euler integration of the different attitude representations. The results show that the rotation matrix is drifting away from the true attitude quicker than the quaternion. Thus the quaternion is the preferred choice for implementation.

8.2 Using the q-method

When using the q-method to give a measurement of the attitude, there is an issue that need special attention. Even though it is only the Kalman filter in this thesis that use the q-method, this applies to other observers as well.

Recall from Section 2.3 that the unit quaternion have double coverage of $SO(3)$, hence $\boldsymbol{q}$ and$-\boldsymbol{q}$ represent the same rotation. It is not possible to say which one of these quaternions the q-method returns, which is a problem. This can be solved by selecting the quaternion that is closest to the estimate $\hat{\boldsymbol{q}}$. Consider the following code snippet

```
q = qmethod(...)
if norm(q+q_hat) < norm(q-q_hat) then
    q = -q;
end
```

which will return the appropriate unit quaternion.

8.3 Saturation for $\delta\boldsymbol{\epsilon}$

The multiplicative EKF use the attitude representation in (5.1), and the error quaternion is constructed by

$$\delta \boldsymbol{q}(\delta \boldsymbol{\epsilon}) = \begin{bmatrix} \sqrt{1 - \delta \boldsymbol{\epsilon}^T \delta \boldsymbol{\epsilon}} \\ \delta \boldsymbol{\epsilon} \end{bmatrix}$$

If $\delta \boldsymbol{\epsilon}^T \delta \boldsymbol{\epsilon} > 1$, the error quaternion becomes imaginary. There are two ways that this can happen:

- The change in attitude from one iteration to the next is greater than 180 deg.
- The correction from measurements are so large that the norm of $\delta \boldsymbol{\epsilon}$ exceeds 1.

Considering a filter sampling rate of 100 Hz, it is very unlikely that any of these cases occur. But to be on the safe side a saturation for $\delta \boldsymbol{\epsilon}$ is implemented.

8.4 Dead-reckoning

Dead-reckoning refers to the case of an *unaided* inertial navigation system (INS). Position, velocity and attitude is calculated based on previously estimated position, velocity and attitude. Estimates are advanced using the current acceleration and angular velocity. The term *unaided* means without the help of an external measurement of the position, velocity and attitude.

If the observer is on a corrector-predictor form, dead-reckoning means that the corrector step i skipped. The principle of dead-reckoning can be utilized when the measurements are unhealthy. Each measurement is implemented with its own flag, indicating whether the measurement is valid. If the measurement is invalid it will be discarded, i.e. dead-reckoning (see Figure 8.1). Methods to determine whether signals is healthy will be discussed later.

The GPS and IMU have different sampling rates, 4 Hz and 100 Hz respectively. This means that it only comes new GPS data every 25th sample. Dead-reckoning have been used to account for the different measurement rates. Every sample there is not a new measurement, the corrector step can be skipped for that particular measurement.

8.5 Low-Level Signal Check

Typical sensor faults are signal freeze, drift and wild points. In order to detect wild points it is useful to calculate a moving average. Whenever the signal is too far away from the average it is a wild point (see Figure 8.2). Alternatively the predicted measurements can be used instead of the moving average. The deviation limit can be set to four times the standard deviation, which means that 99,99%

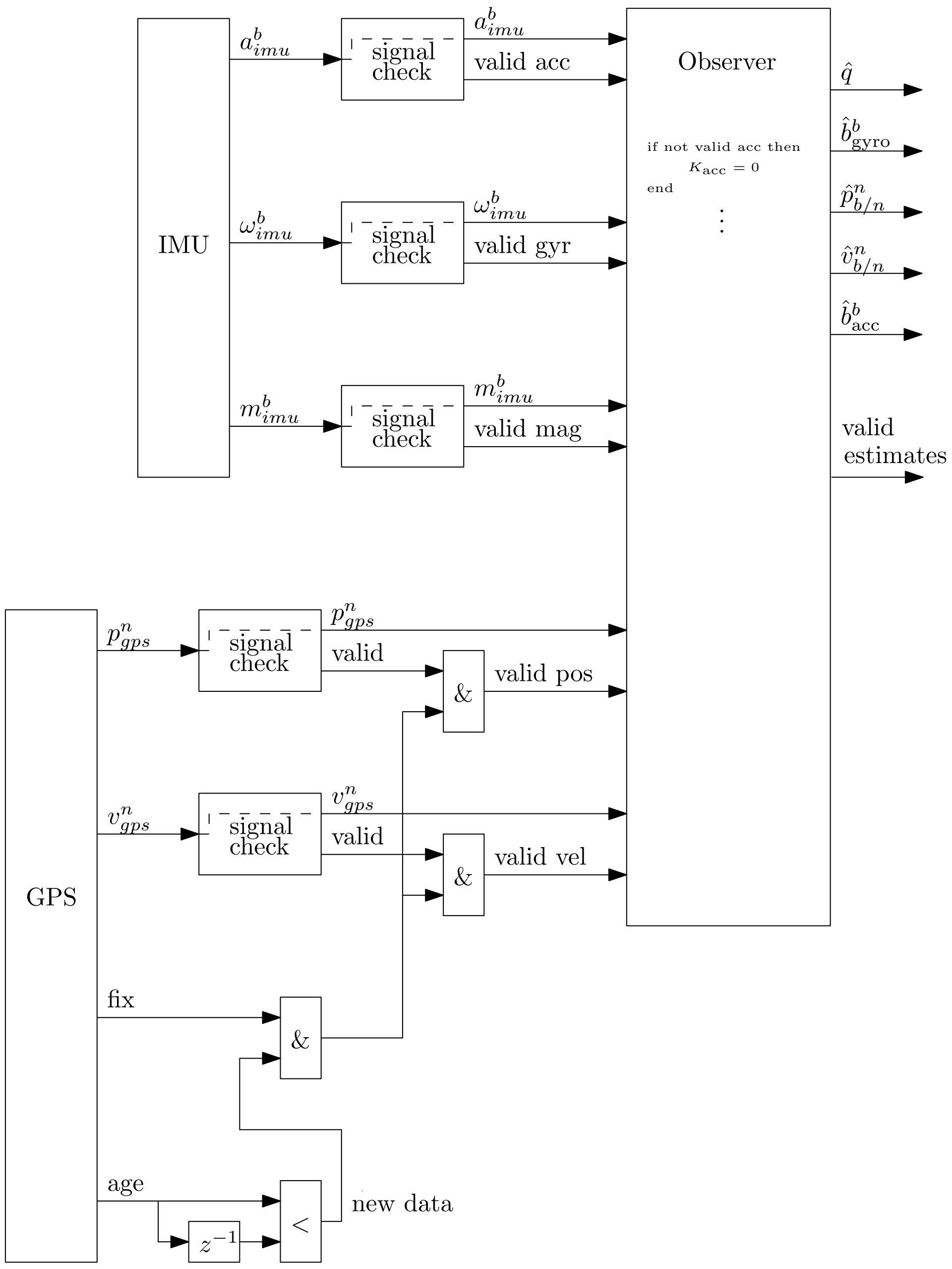


Figure 8.1: Block diagram showing the dead-reckoning strategy. Each measurement has its own low-level signal check and validity flag. Whenever a measurement is invalid, it will be discarded. Different sampling rates are also accounted for as only newly arrived GPS measurements is considered valid.

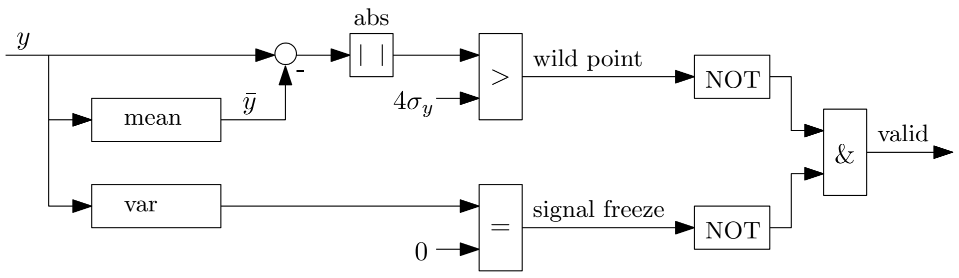


Figure 8.2: Block diagram showing the low-level signal check.

of the measurements will be healthy for a normal distributed signal. Only values outside this limit is abnormal signals.

Signal freeze can be detected by calculating a moving variance. If the variance is zero a signal freeze has occurred (see Figure 8.2). Number of samples included in the variance calculation depends on the signal. For IMU signals, which contains noise, 10 equal samples will most certainly be an abnormal situation.

Slowly time-varying drift of the gyro and accelerometer is accounted for in the observer. Drift is difficult to detect without a complementary measurement. For instance drift in the GPS position measurement is difficult to detect.

If too many of the observer input signals are invalid, i.e magnetometer, accelerometer and gyro at the same time, the output flag *valid estimates* will be reset (see Figure 8.1). Also if a particular measurement is unavailable for a long time, i.e GPS measurements, the the observer output flag is reset.

In addition to the low-level signal checks, a gentle low pass filtering of the raw signals can be applied to reduce the measurement noise. The crossover frequency of the low pass filter should be a decade higher than the observer crossover frequency, in order to avoid a phase lag.

8.6 Magnetic Distortion Compensation

The Earth's magnetic field can be described by three parameters: inclination I, declination D and total field F. Inclination is the angle (positive down) between magnetic north and the horizontal plane. Declination is the angle (positive east) between magnetic north as reported by a compass and true north. Total field is the magnitude of the field.

Measurements of the Earth's magnetic field $\boldsymbol{m}^b_{\text{imu}}$ can easily be distorted by ferromagnetic elements, permanent magnets or very strong currents. Declination errors in the measurement can not be corrected without an additional reference of heading. But the effects of an erroneous inclination and magnitude in the measurement $\boldsymbol{m}^b_{\text{imu}}$, can be reduced by the method of Madgwick (2010). The idea is to calculate

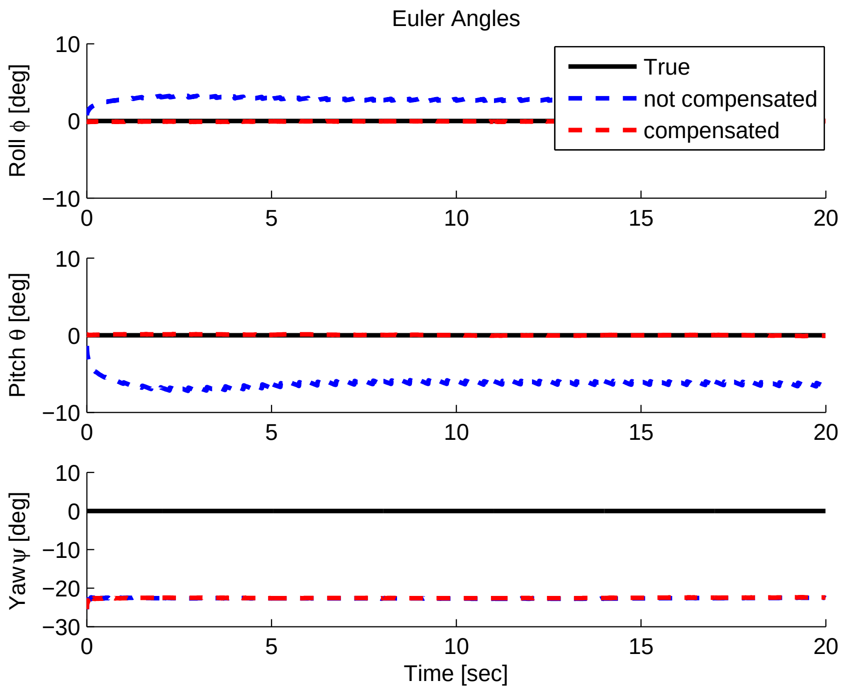


Figure 8.3: The effect of magnetic distortion compensation. The roll and pitch errors are removed, while the yaw error remains the same.

the Earth's magnetic field reference $\boldsymbol{m}^n$, using the same inclination and magnitude as the measurement. The method of Madgwick (2010) has been improved by taking into account the local magnetic declination in the calculation as follows

$$\begin{bmatrix} m_N & m_E & m_D \end{bmatrix}^T = R_b^n(\bar{\boldsymbol{q}}(k))\boldsymbol{m}^b_{\text{imu}} \tag{8.5}$$

$$\boldsymbol{m}^n = \begin{bmatrix} \cos(D)\sqrt{m_N^2 + m_E^2} \\ \sin(D)\sqrt{m_N^2 + m_E^2} \\ m_D \end{bmatrix} \tag{8.6}$$

where $\bar{\boldsymbol{q}}(k)$ is the predicted attitude at $t = k$, for observers on the corrector-predictor form. If the observer is not on this form $\hat{\boldsymbol{q}}(k-1)$ should be used. Attitude errors caused by magnetic field disturbances can be restricted to only affect the yaw estimate, by compensating this way (see Figure 8.3). Another advantage is that the magnetic field reference does not need to be predefined, only the local declination.

Chapter 9

Simulation Results

This chapter contains a case study of the nonlinear observers developed in this thesis, demonstrating how different disturbances affects the estimates. The simulator developed in Chapter 7 have been used to generate IMU and GPS measurements. Test results of the performance are given as Root Mean Square (RMS) errors, which is calculated by the following formula

$$\text{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{x}(i) - x_{\text{true}}(i)\right)^2}$$

where N is the number of samples in the time-serie, and x represent roll, pitch and so on. The first 40 seconds have been omitted in the RMS calculation to remove most of the transient part. Tests have been carried out for the following observers:

MEKF is the multiplicative extended Kalman filter including attitude, position and velocity estimates. The filter was derived with two different measurement equations; one using the q-method and the other using vectorial measurements. The vectorial measurement equation is used. The following Kalman filter tuning matrices have been used

$$\begin{aligned}
Q_k &= \text{diag}(\ \text{5e-7}_{1\times2}\quad \text{5e-6}\quad \text{5e-8}_{1\times2}\quad \text{1e-7}\quad \text{1e-4}_{1\times3}\quad \text{1e-4}_{1\times3}\quad \text{1e-7}_{1\times3}\ )\\
R_k &= \text{diag}(\ \text{2e-5}_{1\times3}\quad \text{2e-3}_{1\times3}\quad 10_{1\times3}\quad \text{1e-4}_{1\times3}\ )\\
P_0 &= \text{diag}(\ \text{1e-5}_{1\times3}\quad \text{1e-9}_{1\times3}\quad 0_{1\times3}\quad \text{1e-4}_{1\times3}\quad \text{1e-8}_{1\times3}\ )
\end{aligned}$$

HuaMahony is the nonlinear observer combination of Hua (2010) and Mahony et al. (2008) including attitude, position and velocity estimates. The discrete-time corrector-predictor formulation of HuaMahony has been implemented. The following observer gains have been used

$$K_{\text{quat}} = \text{diag}(1,1,10)\cdot h, \quad K_{\text{bgyr}} = \text{diag}(0.2,0.2,0.5)\cdot h, \quad k_{\text{mag}} = 1, \quad k_{\text{acc}} = 1$$
$$k_{\text{pos}} = 0.0001\cdot 25h, \quad k_{\text{vel}} = 3\cdot 25h, \quad k_Q = 0.06\cdot 25h$$

where h is the sample time. Values used for the magnetic declination, gravity and lever arm are:

$$\begin{aligned} \text{Declination} &= 0.0323 \text{ rad} \\ \boldsymbol{g}^n &= [\ 0 \quad 0 \quad 9.81\ ]^T \text{ m/s}^2 \\ \boldsymbol{r}^b_{\text{ant}} &= [\ -0.65 \quad -0.08 \quad -0.90\ ]^T \text{ m} \end{aligned}$$

9.1 Description of Case Studies

Six different test cases have been set up, to see how different disturbances affects the estimates:

- Case 1: Perfect data
- Case 2: Noise only
- Case 3: Gyro bias
- Case 4: Local magnetic disturbance
- Case 5: Accelerometer bias
- Case 6: Variable force

The simulator vehicle parameters are chosen as described in Chapter 7. For all cases the vehicle moment input is a sequence of various moments creating different attitudes (see Figure 9.3). The force input is zero for all cases except number 6. Measurement noise recorded from the Xsens MTi-G unit is added to all measurements from the simulator, except case 1. If nothing else is specified for the specific case the measurement biases are zero as default. In addition all initial states are zeros.

In the following each case will be discussed in more detail. The main focus is the attitude estimates as those are the most challenging and interesting. Simulation results are given in Table 9.1 as RMS errors. For visualization, a plot of the attitude errors ($\hat{x} - x_{\text{true}}$) are shown in Figure 9.1 and 9.2.

Table 9.1: Simulation results given as RMS errors

MEKF	Attitude (deg RMS)			Position (m RMS)			Velocity (m/s RMS)		
	Roll	Pitch	Yaw	North	East	Down	North	East	Down
1 Perfect data	0.2316	0.1261	0.2392	0.1669	0.0134	0.2477	0.0110	0.0139	0.0036
2 Noise only	0.2407	0.1547	0.2786	0.2274	0.0377	0.2029	0.0267	0.0165	0.0104
3 Gyro bias	0.2467	0.1769	0.4499	0.3915	0.0496	0.1769	0.0288	0.0224	0.0130
4 Mag bias	0.4452	0.4514	11.9439	0.2151	0.1001	0.3596	0.1022	0.0658	0.0136
5 Acc bias	0.5922	0.6353	1.3616	0.2298	0.0100	0.3033	0.0388	0.0405	0.1437
6 Variable force	0.2493	0.1632	0.3683	0.2079	0.0630	0.1847	0.0346	0.0265	0.0146
Average	0.3342	0.2846	2.4402	0.2397	0.0456	0.2458	0.0403	0.0309	0.0331

HuaMahony	Attitude (deg RMS)			Position (m RMS)			Velocity (m/s RMS)		
	Roll	Pitch	Yaw	North	East	Down	North	East	Down
1 Perfect data	0.2408	0.1374	0.2594	0.1704	0.0129	0.3111	0.0101	0.0088	0.0105
2 Noise only	0.2485	0.1578	0.2962	0.1986	0.0270	0.2427	0.0123	0.0110	0.0178
3 Gyro bias	0.2481	0.1500	0.4396	0.1300	0.2112	0.2178	0.0142	0.0111	0.0176
4 Mag bias	0.8895	1.0532	11.5743	0.2817	0.1993	0.8993	0.1739	0.1123	0.0314
5 Acc bias	0.6179	0.6948	1.1278	0.1720	0.0147	0.2609	0.0258	0.0177	0.0256
6 Variable force	0.2778	0.2178	0.4208	0.2125	0.0247	0.1956	0.0155	0.0149	0.0200
Average	0.4204	0.4018	2.3530	0.1942	0.0816	0.3545	0.0419	0.0293	0.0204

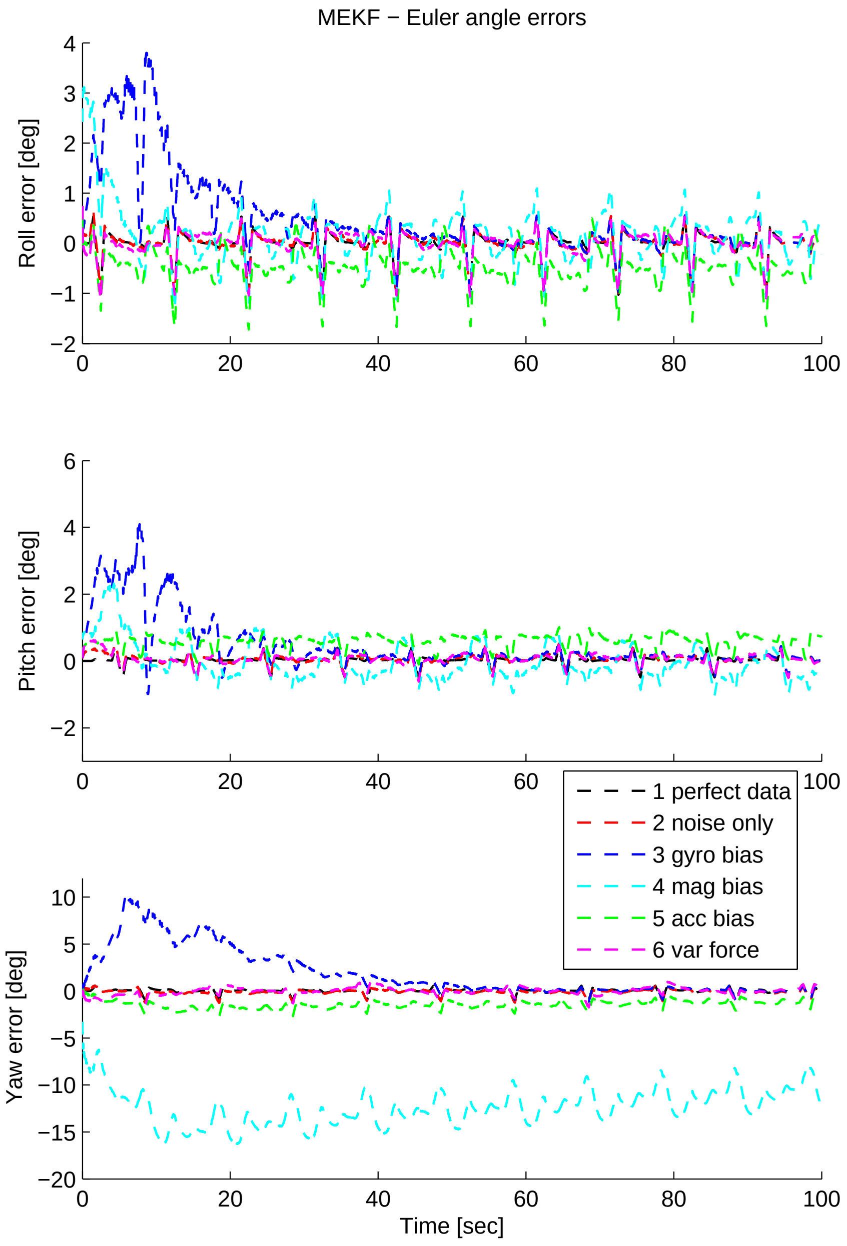


Figure 9.1: Simulation results for MEKF, showing the attitude errors for each case.

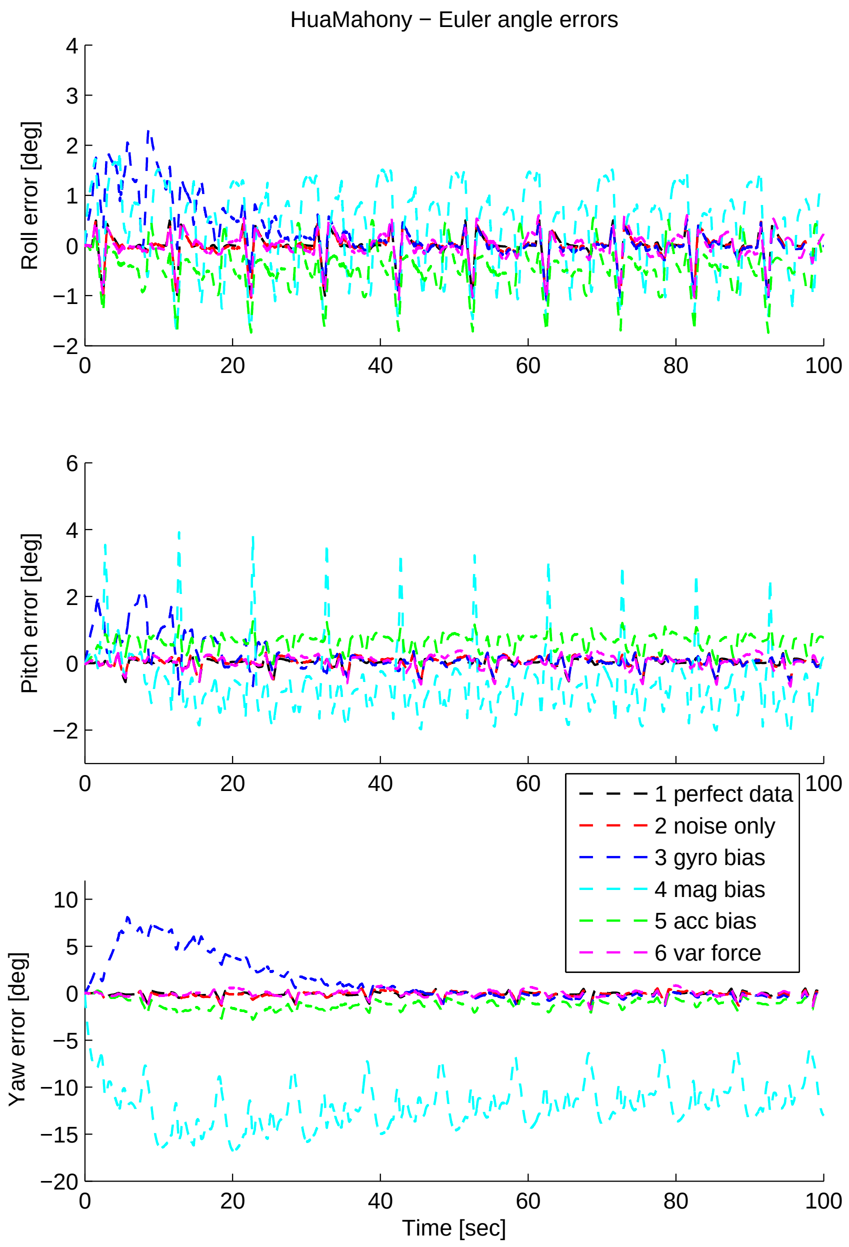


Figure 9.2: Simulation results for HuaMahony, showing the attitude errors for each case.

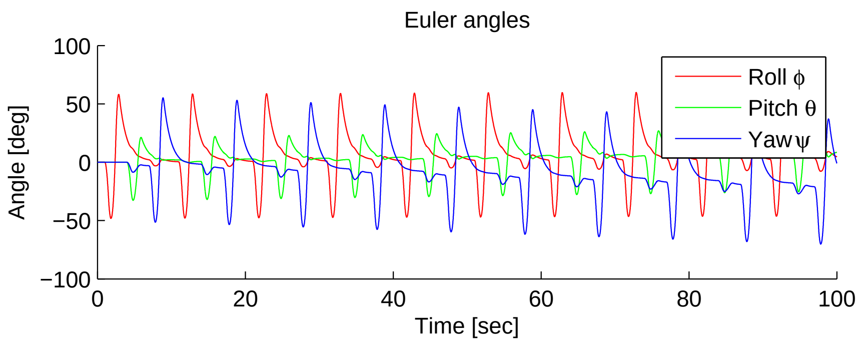


Figure 9.3: True Euler angles for all simulation cases

9.2 Case 1: Perfect Data

Perfect data was generated, with no measurement noise nor bias, to have a reference of the performance without disturbances. One should expect perfect data to give perfect estimates. But the errors due to discretization gives the deviation from zero (see Table 9.1).

9.3 Case 2: Noise Only

Measurement noise recorded from the Xsens MTi-G was added to the simulator output, to see how noise affects the estimates. Results show that zero mean measurement noise has very little impact on the performance.

9.4 Case 3: Gyro Bias

The goal of this case is to study and verify the gyro bias estimation. Simulation is carried out using a constant gyro measurement bias of

$$\boldsymbol{b}^b_{\text{gyro}} = \begin{bmatrix} 0.02 & 0.02 & 0.02 \end{bmatrix}^T \text{ rad/s}$$

The bias makes the attitude errors grow rapidly from the start (see Figure 9.1 and 9.2). But after 40-50 seconds the attitude errors have converged to approximately zero. A look at Figure 9.4 shows that this is the time it takes before the gyro bias estimates have converged to the right value. It is a matter of tuning whether the bias estimates overshoot or not. Good performance is achieved after the gyro bias estimates have converged to the right value (see Table 9.1).

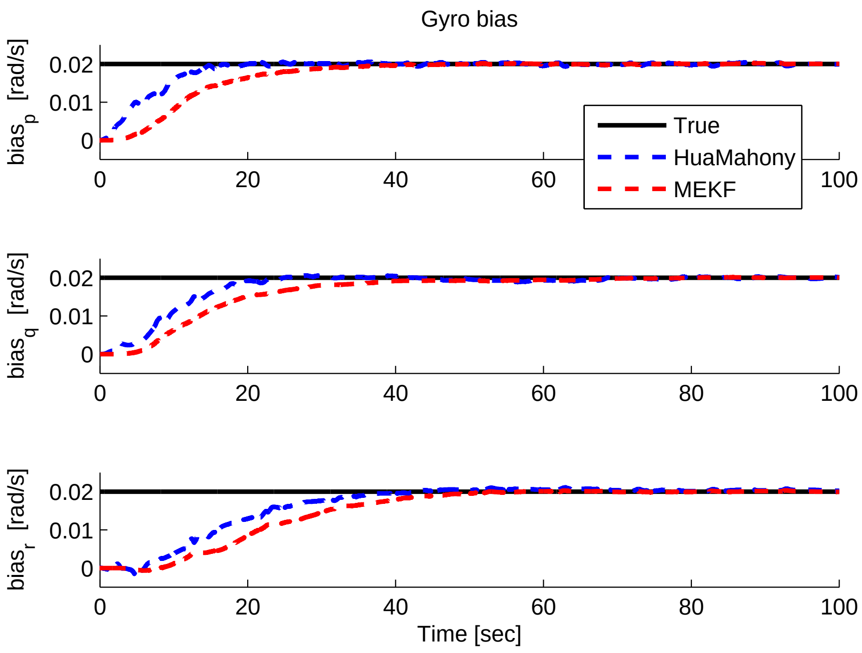


Figure 9.4: Case 3: Plot showing how the gyro bias converge to the right value.

9.5 Case 4: Local Magnetic Disturbance

The goal of this case is to see how a local magnetic disturbance influence the estimates. The simulation is carried out using a disturbance of

$$\boldsymbol{b}^b_{\text{mag}} = \begin{bmatrix} 0.1 & 0.1 & -0.1 \end{bmatrix}^T \text{ a.u.}$$

From Figure 9.1 and 9.2, it is seen that the magnetic disturbance result in a static yaw deviation of about 11 deg. Both the MEKF and HuaMahony is implemented with the magnetic field distortion compensation discussed in Section 8.6. The goal of this method was to limit the disturbance to only affect the yaw estimates. From Table 9.1 and Figure 9.1 and 9.2, it is seen that the MEKF have most success in this task. However there is a big improvement for both observers, which can be seen in Table 9.2 who contains a comparison of the attitude RMS error with and without the magnetic distortion compensation.

Table 9.2: Case 4: A comparison of the attitude RMS errors with and without the magnetic distortion compensation.

	MEKF Attitude (deg RMS)			HuaMahony Attitude (deg RMS)		
	Roll	Pitch	Yaw	Roll	Pitch	Yaw
without compensation	1.9878	3.3871	12.7851	2.4910	3.8794	11.6743
with compensation	0.4452	0.4514	11.9439	0.8895	1.0532	11.5743
improvement	78 %	87 %	7 %	64 %	73 %	1 %

9.6 Case 5: Accelerometer Bias

The goal of this case is to study the accelerometer bias estimation. The MEKF provides estimates of the accelerometer bias, while the HuaMahony observer does not have an explicit accelerometer bias estimate, still it is embedded in the Q-matrix which maps the specific force to the inertial frame. Simulation is carried out using a constant accelerometer measurement bias of

$$\boldsymbol{b}^b_{\text{acc}} = [\ 0.1 \quad 0.1 \quad 0.1\ ]^T\ \text{m/s}^2$$

Test results show that the bias has a negative impact on the attitude estimates (see Table 9.1, Figure 9.1 and 9.2). The events that cause the errors can be explained as follows: First the attitude converges to the wrong value, thinking that the accelerometer measure the direction of the gravity only. After this happens the accelerometer reading will be perpendicular to the estimated body xy-plane, with only a z-component. Then any difference in the measured magnitude compared to the gravity will be interpreted as an accelerometer bias in the z-direction.

Note that this simulation case does not have any translational motion, only rotations. According to Hong et al. (2005) the accelerometer bias is not observable under these conditions, acceleration changes is required.

9.7 Case 6: Variable Force

The goal of this case is to verify that linear accelerations are successfully suppressed from propagating to the attitude estimates. The vehicles motion is generated by the force input

$$\boldsymbol{f}^b = [\ 5\sin(0.2\pi t) \quad 4\sin(0.3\pi t) \quad 3\sin(0.4\pi t)\ ]^T\ \text{N}$$

For this input the vehicle is subject to the linear accelerations shown in Figure 9.5. The plot only shows the linear accelerations, without the gravity component,

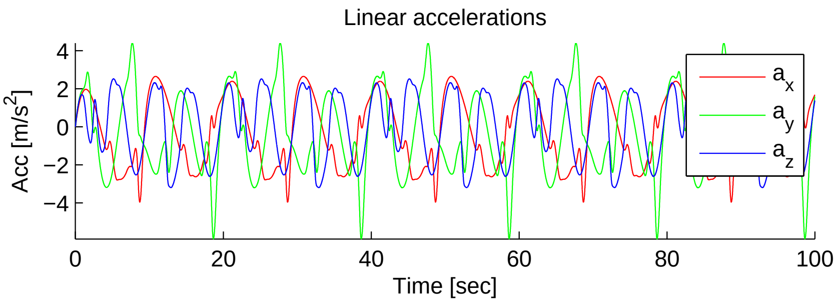


Figure 9.5: Case 6: Plot showing the linear accelerations that the vehicle is subject to.

to give an idea of the magnitude. The results in Table 9.1 shows that the linear accelerations have very little impact on the attitude estimates compared to the case with only noise. When comparing the two observers, the MEKF provides slightly better attitude estimates.

Chapter 10

Experimental Results

This chapter contains real life tests based on experimental data. Several time-series have been recorded with the Xsens MTi-G unit. The Xsens MTi-G unit was mounted in the car as shown in Figure 10.1, and taken for a ride. The map[1] in Figure 10.2 shows the route where the experimental data were recorded. The experimental data contains the following time-series:

Standing still: The car was parked.

City tour 1,2,3,4: Low speed driving in the city with various road conditions including speed dumps

Highway 1,2: A relatively fast and smooth ride along the highway.

Roundabout: Four turns in a roundabout.

Each time-serie contains the IMU measurements, GPS measurements and the built-in EKF attitude, position and velocity estimates (see section 3.3). The built-in estimates are considered to be the true values. The nonlinear observers developed in this thesis have been tested with the experimental data, and the estimates are compared to the estimates provided by the built-in EKF. Test results of the performance are given as Root Mean Square (RMS) errors, calculated from the difference between the estimates and the built-in EKF estimates. That is, the values in Table 10.1 are calculated with the following formula

$$\text{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{x}(i) - x_{\text{xsens}}(i)\right)^2}$$

[1]`http://maps.google.no/maps/ms?ie=UTF8&hl=no&oe=UTF8&msa=0&msid=206709540317555168643.00049c56274f1dc13c01d`

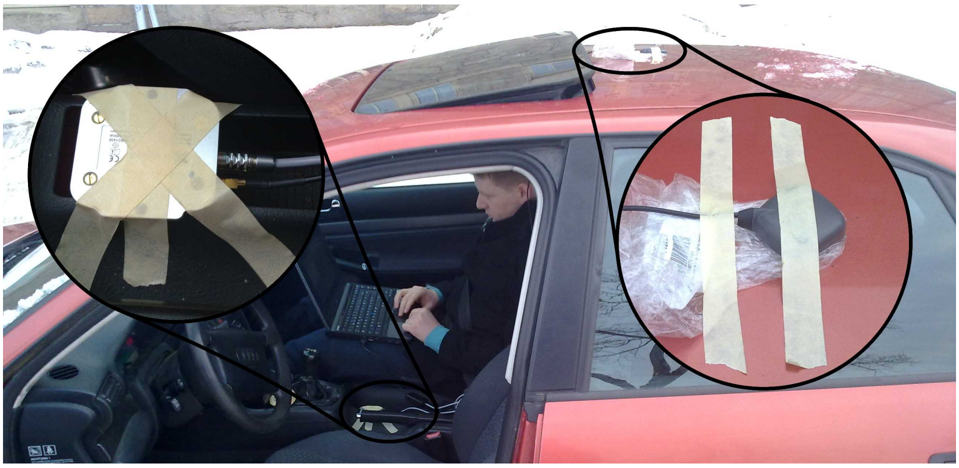


Figure 10.1: The mounting of the Xsens MTi-G in the car

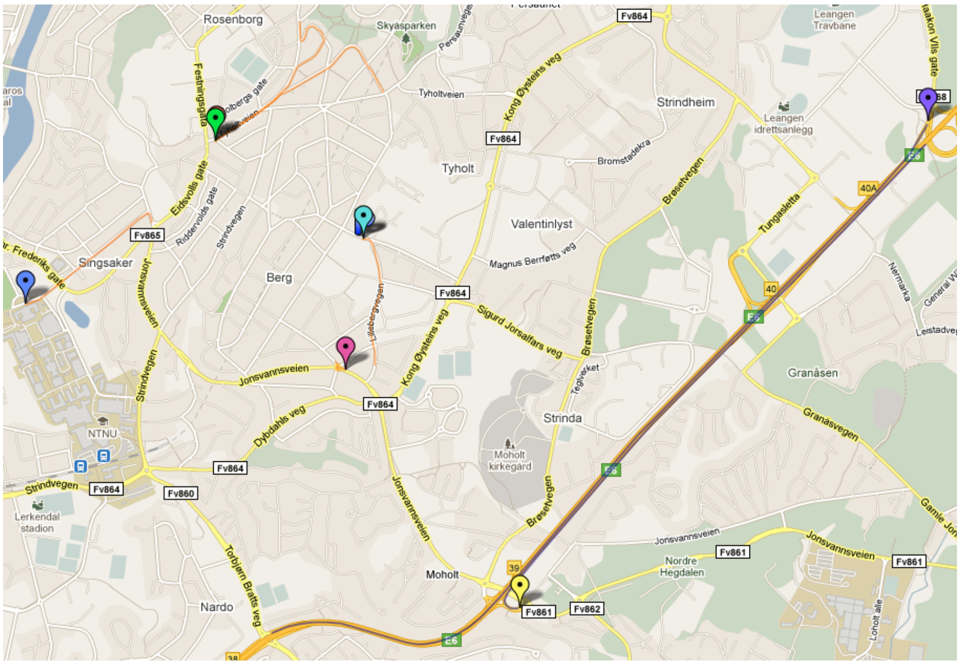


Figure 10.2: Map of the route where the experimental time-series where recorded

Table 10.1: Experimental results given as RMS errors. Calculated using the Xsens built-in EKF as the true values.

MEKF	Attitude (deg RMS)			Position (m RMS)			Velocity (m/s RMS)		
	Roll	Pitch	Yaw	North	East	Down	North	East	Down
Standing still	0.0251	0.1395	0.4914	0.1080	0.0352	6.1020	0.0197	0.0411	0.5296
City tour 1	0.1872	0.2090	2.6295	0.1571	0.0897	5.3643	0.0701	0.0650	0.4288
City tour 2	0.6005	0.4305	2.2433	0.2050	0.1236	4.5144	0.1275	0.2206	0.4098
City tour 3	0.2407	0.2234	1.0988	0.2045	0.1193	4.0288	0.0888	0.1045	0.4000
City tour 4	0.3087	0.2922	1.6391	0.0812	0.0816	3.8299	0.0879	0.1111	0.3952
Highway 1	0.2848	0.4213	1.8131	0.1191	0.2286	6.3888	0.1811	0.1028	0.5052
Highway 2	0.4186	0.4173	2.0521	0.1194	0.2420	7.4600	0.1460	0.1609	0.6070
Roundabout	0.5893	0.4655	3.6662	0.1200	0.1695	3.2498	0.1601	0.2203	0.5027
Average	0.3318	0.3248	1.9541	0.1392	0.1361	5.1172	0.1101	0.1282	0.4722

HuaMahony	Attitude (deg RMS)			Position (m RMS)			Velocity (m/s RMS)		
	Roll	Pitch	Yaw	North	East	Down	North	East	Down
Standing still	0.0479	0.1204	0.9423	0.1033	0.0396	6.1590	0.0549	0.0947	0.5792
City tour 1	0.4639	0.3834	3.4566	0.2081	0.2048	5.4041	0.2761	0.3324	0.4735
City tour 2	1.0336	1.5885	5.1096	0.2780	0.3552	4.5194	0.5360	0.6684	0.4616
City tour 3	0.8218	0.6157	3.9286	0.2592	0.2277	4.0596	0.4932	0.5358	0.4676
City tour 4	1.8859	0.9148	8.1613	0.2187	0.2830	3.8632	0.4674	0.5313	0.4364
Highway 1	1.1608	0.4067	8.2778	0.1707	0.3286	6.4395	0.2143	0.3512	0.5621
Highway 2	0.6750	0.6719	2.9121	0.2932	0.3942	7.4912	0.6143	0.5367	0.6748
Roundabout	0.5982	0.8857	2.2916	0.1659	0.1263	3.3077	0.3147	0.3265	0.5735
Average	0.8358	0.6983	4.3849	0.2121	0.2449	5.1554	0.3713	0.4221	0.5285

where N is the number of samples in the time-serie, and x represent roll, pitch and so on. The first 10 seconds have been omitted in the RMS calculation to remove most of the transient part. Tests have been carried out for the following observers:

MEKF is the multiplicative extended Kalman filter including attitude, position and velocity estimates. The filter was derived with two different measurement equations; one using the q-method and the other using vectorial measurements. If nothing else is specified, the vectorial measurement equation is used. The following Kalman filter tuning matrices have been used

$$\begin{aligned} Q_k &= \mathrm{diag}(\begin{array}{ccccc} \text{5e-7}_{1\times3} & \text{1e-9}_{1\times3} & \text{1e-0}_{1\times3} & \text{7e-4}_{1\times3} & \text{1e-7}_{1\times3} \end{array}) \\ R_k &= \mathrm{diag}(\begin{array}{cccc} \text{1e-4}_{1\times3} & \text{7e-2}_{1\times3} & \text{1e-1}_{1\times3} & \text{5e-4}_{1\times3} \end{array}) \\ P_0 &= \mathrm{diag}(\begin{array}{ccccc} \text{1e-5}_{1\times3} & \text{1e-9}_{1\times3} & \text{1e-0}_{1\times3} & \text{1e-3}_{1\times3} & \text{1e-8}_{1\times3} \end{array}) \end{aligned}$$

HuaMahony is the nonlinear observer combination of Hua (2010) and Mahony et al. (2008) including attitude, position and velocity estimates. The discrete-time corrector-predictor formulation of HuaMahony has been implemented. The following observer gains have been used

$$K_{\text{quat}} = 0.5 \cdot h, \quad K_{\text{bgyr}} = 0.002 \cdot h, \quad k_{\text{mag}} = 0.26, \quad k_{\text{acc}} = 1$$
$$k_{\text{pos}} = 1 \cdot 25h, \quad k_{\text{vel}} = 1 \cdot 25h, \quad k_Q = 0.12 \cdot 25h$$

where h is the sample time. Values used for the magnetic declination, gravity and lever arm are:

$$\begin{aligned} \text{Declination} &= 0.0323 \text{ rad} \\ \boldsymbol{g}^n &= [\begin{array}{ccc} 0 & 0 & 9.81 \end{array}]^T \text{ m/s}^2 \\ \boldsymbol{r}^b_{\text{ant}} &= [\begin{array}{ccc} -0.65 & -0.08 & -0.90 \end{array}]^T \text{ m} \end{aligned}$$

10.1 Main Results

The experimental results for each time-serie are given in Table 10.1. Results show that the MEKF has an average better performance compared to the HuaMahony observer.

The MEKF has an average error in roll, pitch and yaw of 0.3, 0.3 and 1.9 deg respectively. Considering the accuracy of the “true” value (see Table 3.2), this is a very good result. The HuaMahony observer has an average error in roll, pitch and yaw of 0.8, 0.6 and 4.3 deg respectively. Roll an pitch estimates must be said to have

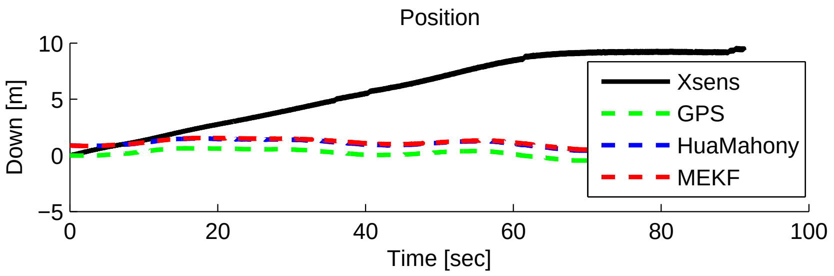


Figure 10.3: Standing still: Plot showing how the Xsens height estimate drifts away from the true value.

good results, while the yaw estimate is a bit weaker. It is worth mentioning that the magnetometer measurements are generally poor, because the measurements contains a lot of magnetic interference from the car. This has a negative effect on the yaw estimates.

Both observers work very well for the position and velocity estimates. Note that there is a big difference in the Down position. This is because the built-in solution is very far away from the GPS measurements. For instance, a look at the time-serie where the car was parked and obviously had the same height all the time, revels that the built-in solution drifts away in the Down position estimate (see Figure 10.3).

10.2 Lever Arm Compensation

When deriving the linearized measurement matrix H_k for the measurement given by (5.23), three terms occurred in H_k because of the lever arm compensation

$$H_k = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \left.\frac{\partial \boldsymbol{p}^n_{\text{gps}}(k)}{\partial \delta\boldsymbol{\epsilon}(k)}\right|_{\delta\boldsymbol{\epsilon}(k)=\delta\bar{\boldsymbol{\epsilon}}(k)} & \cdot & \cdot & \cdot & \cdot \\ \left.\frac{\partial \boldsymbol{v}^n_{\text{gps}}(k)}{\partial \delta\boldsymbol{\epsilon}(k)}\right|_{\delta\boldsymbol{\epsilon}(k)=\delta\bar{\boldsymbol{\epsilon}}(k)} & \left.\frac{\partial \boldsymbol{v}^n_{\text{gps}}(k)}{\partial \boldsymbol{b}^b_{\text{gyro}}(k)}\right|_{\boldsymbol{b}^b_{\text{gyro}}(k)=\bar{\boldsymbol{b}}^b_{\text{gyro}}(k)} & \cdot & \cdot & \cdot \end{bmatrix}$$

which means that there is a feedback from the position and velocity measurement to the attitude and gyro bias estimates. Test results indicate worse attitude estimates with this feedback. Therefore the terms have been omitted. Even though the terms purely mathematically should be there, it can be argued that

- uncertainties in the lever arm $\boldsymbol{r}^b_{\text{ant}}$ will propagate to the attitude estimates.
- GPS measurement errors will propagate to the attitude estimates.

therefore the results may be better without the feedback. The same measurement equation is used, it's only the terms in H_k which is omitted.

10.3 Initial error

The attitude estimates are initialized using the q-method. This cause the initial attitude error to be reasonably small. However big errors can occur if the vehicle is under big linear accelerations during initialization. To study the convergence from initial attitude error, a fixed initial attitude is set. Figure 10.4 shows the convergence of the attitude with an initial error of 45 deg in roll, pitch and yaw. Results show that the Kalman filter converge faster than the HuaMahony observer. Note that the Kalman filter has an advantage her, because the starting gain can be boosted by setting higher values for the initial covariance matrix P_0.

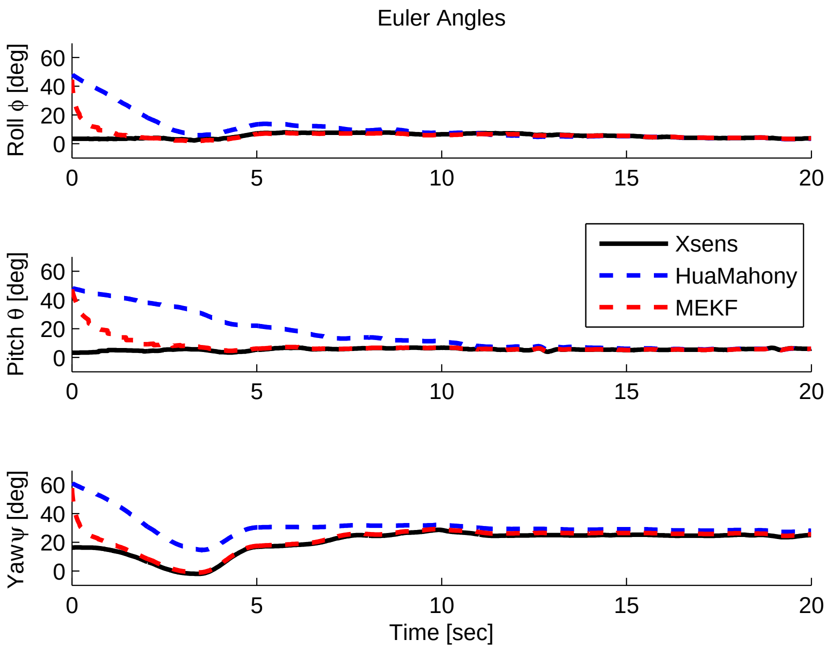


Figure 10.4: City tour 2: Plot showing how the observers converge after an initial attitude error of 45 deg in roll, pitch and yaw.

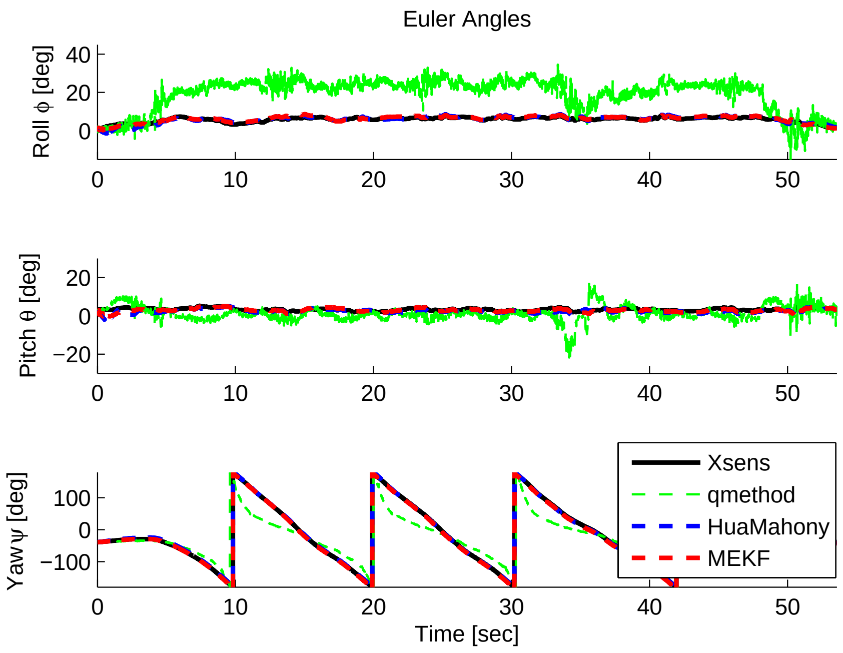


Figure 10.5: Roundabout: Note how the roll estimates are unaffected by the linear accelerations caused by driving in a circle. The q-method shows the error caused by linear accelerations when using the accelerometer and magnetometer directly for attitude determination.

10.4 Linear Accelerations

The time-serie from the roundabout contains an interesting case. Driving in circles cause a linear acceleration pointing towards the center, which is measured by the accelerometer. This gives a corrupted gravity measurement, which cause a roll error if used directly for attitude determination. For the roundabout time-serie the magnitude of the linear acceleration is about 4 m/s^2, corresponding to a roll error of about 20 deg (see qmethod in Figure 10.5). The MEKF and HuaMahony observer does not suffer from this, because the GPS measurements allows the linear acceleration to be estimated. Figure 10.5 shows how the roll estimates are unaffected by the linear acceleration in the roundabout time-serie. The estimates from both observers, including the Xsens solution show a small roll angle different from zero, which is reasonable because the car tilts a bit as the spring dampers gives after when turning hard.

Table 10.2: Comparison of different measurement equation. Errors are given as RMS values.

	MEKF q-method			MEKF vector measurements		
	Roll	Pitch	Yaw	Roll	Pitch	Yaw
Average error (deg)	0.3897	0.3751	4.7041	0.3318	0.3248	1.9541

Table 10.3: Average execution time pr. iteration

	MEKF q-method	MEKF vector measurements	HuaMahony
Execution time (ms)	0.929 ms	0.884 ms	0.323 ms

10.5 Measurement Equation

The extended Kalman filter was derived with two different measurement equations, one using the q-method (5.18) and the other using vector measurements (5.23). Both filters have been tested on the experimental time-series. The average performance is given as RMS errors in Table 10.2. Results show that the filter using vector measurements provide twice as good yaw estimates, while the roll and pitch estimates are approximately the same. Keep in mind that the filter using the q-method have a different measurement covariance matrix R. The matrix was in this test tuned to be

$$R_k = \text{diag}(\ \text{4e-4}_{1\times3} \quad \text{1e-1}_{1\times3} \quad \text{5e-4}_{1\times3}\ )$$

and the accelerometer and magnetometer were weighted equal in the q-method.

10.6 Execution Time

The execution time is an important factor for nonlinear observers to be implemented on embedded hardware. Table 10.3 contains the average execution time per iteration for the MEKF with both measurement equations, and the HuaMahony observer. Results show that the Kalman filter using the q-method has a 0.045 ms increase in execution time, which is about 5%. The winner is the nonlinear observer combination of Hua and Mahony, which is about three times faster than the Kalman filter. Clearly the efficiency depends on the implementation method and operating system. But the observers in this thesis are implemented very similar, so the results should provide a good basis for comparison. The *tic toc* commands in Matlab have been used to measure the execution time

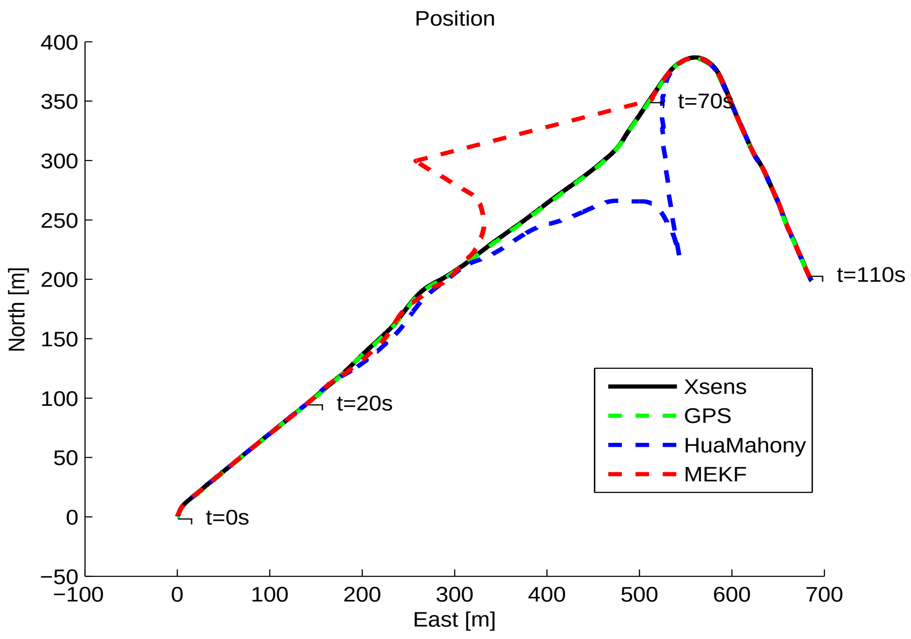


Figure 10.6: City tour 3: Plot showing the effect of a simulated GPS outage from t=20s to 70s.

10.7 GPS Outage

The acceleration is integrated twice to get the position. Small errors in the acceleration (or attitude) will cause exponential growing errors in the position. GPS measurements are used to compensate and estimate these errors.

In Figure 10.6 it is shown how the error grows with time when GPS measurements are unavailable. In just 50 seconds the position error is over 100 meters for both observers. It looks like the MEKF has a bit faster recovery after the dropout.

Chapter 11

Conclusions

Two different nonlinear observers for integration of GNSS and IMU have been implemented and tested:

- a quaternion based multiplicative extended Kalman filter, named MEKF
- a nonlinear observer, named HuaMahony.

Simulations have been carried out as a case study to see how different disturbances affect the estimates of position, velocity and attitude. The most dominating error comes from the local magnetic disturbance, however it has little effect on the roll and pitch estimates thanks to the magnetic distortion compensation. Simulations results reveals that the accelerometer bias is unobservable without acceleration changes. If case 4 is excluded, the average performance is good, showing a dynamic accuracy of < 0.5 deg RMS for the attitude estimates. Both observers have a very similar performance and it is not possible to say anything about which one is better.

Experimental data have been recorded with the Xsens MTi-G unit. Assuming that the Xsens built-in EKF gives the true values, the average performance for the experimental test shows an attitude accuracy of (0.3 0.3 2.0) and (0.8 0.7 4.4) deg RMS for the MEKF and HuaMahony observer respectively. The results indicate that the MEKF has the best performance, but it is difficult to be very conclusive as there is several uncertainties: (i) the performance is depending on the tuning, but an exact equal tuning is difficult to achieve; (ii) there is some uncertainties in the “true” values provided by the Xsens unit.

However some differences between the MEKF and HuaMahony observer can be emphasized:

- the MEKF has a quicker convergence from initial errors, because of the initial covariance matrix which is used in the Riccati equation.

- the HuaMahony observer is three times faster when it comes to execution time. The test was performed in Matlab, which is known to efficiently handle matrix calculations, therefore one may expect an even bigger ratio for implementation on embedded hardware.

Chapter 12

Further Work

Estimation of the yaw angle is the most challenging part. The accuracy of the yaw estimate may be improved by making a no-sideslip assumption. This means that an assumption is made about zero velocity along the body-fixed y-axis. Consequently, the measured velocity will contain information about the yaw angle. However this assumption is most suited for vehicles subject to none or short term sidewards slipping, such as automotive vehicles (Xsens 2009d).

For aerial vehicles there are several actions that might be done to improve redundancy and accuracy. For instance integrating a barometer which measure the altitude, or a dynamic pressure sensor to measure airspeed. Also an application specific dynamic model of the vehicle can be integrated, which can limit the dynamics of the movement (Koifman and Bar-Itzhack 1999). The drawback is the loss of generality.

Adaptive tuning of the observer gains may also be considered for increased performance. For the multiplicative extended Kalman filter this may be achieved by manipulating the measurement covariance matrix R (Cheng et al. 2008).

Bibliography

Abdel-Hafez, M. F. (2010). The autocovariance least-squares technique for gps measurement noise estimation, *IEEE Transactions on Vehicular Technology* **59**(2): 574–588.

Babu, R., Wang, J. and Rao, G. (2008). Analysis of ultra-tight gps/ins integrated system for navigation performance, *IEEE-ICSCN* pp. 234–237.

Bar-Itzhack, I. Y. (1996). Request: A recursive quest algorithm for sequential attitude determination, *Journal of Guidance, Control, and Dynamics* **19**(5): 1034–1038.

Cheng, L., Zhaoying, Z. and Xu, F. (2008). Attitude determination for mavs using a kalman filter, *Tsinghua Science and Technology* **13**(5): 593–597.

Crassidis, J. L., Markley, F. L. and Cheng, Y. (2007). A survey of nonlinear attitude estimation methods, *AIAA Journal of Guidance, Control, and Dynamics* **30**(1): 11–28.

Egeland, O. and Gravdahl, J. T. (2003). *Modeling and Simulation for Automatic Control*, 2nd edn, Marine Cybernetics AS.

Ellingsen, H. (2008). *Development of a low-cost integrated navigation system for usvs*, Master's thesis, Norwegian University of Science and Technology.

Fiorenzani, T., Manes, C., Oriolo, G. and Peliti, P. (2008). Comparative study of unscented kalman filter and extended kalman filter for position/attitude estimation in unmanned aerial vehicles.

Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*, John Wiley & Sons Ltd.

Gade, K. (2009). Introduction to inertial navigation and kalman filtering, *Tutorial for: IAIN World Congress, Stockholm* .

Hong, S., Lee, M. H., Chun, H.-H., Kwon, S.-H. and Speyer, J. L. (2005). Observability of error states in gps/ins integration, *IEEE Transactions on Vehicular Technology* **54**(2): 731–743.

Hua, M.-D. (2010). Attitude estimation for accelerated vehicles using gps/ins measurements, *Control Engineering Practice* **18**: 723–732.

Jouffroy, J. and Fossen, T. (2010). A tutorial on incremental stability analysis using contraction theory, *Modeling, Identi*
cation and Control **31**(3): 93–106.

Keat, J. E. (1977). Analysis of least-squares attitude determination routine doaop, *Computer Sciences Corp.* **Report CSC/TM-77/6034**.

Koifman, M. and Bar-Itzhack, I. (1999). Inertial navigation system aided by aircraft dynamics, *IEEE Transactions on Control System Technology* **7**(4): 487–493.

Lefferts, E. J. and Schuster, M. D. (1982). Kalman filtering for spacecraft attitude estimation, *Journal of Navigation, Control and Dynamics* **JGCD-5**(5): 417–428.

Lerner, G. (1978). Three-axis attitude determination, *Spacecraft Attitude Determination and Control* pp. 420–428.

Madgwick, S. O. (2010). An efficient orientation filter for inertial and inertial/-magnetic sensor arrays.

Mahony, R., Hamel, T. and Pflimlin, J.-M. (2008). Nonlinear complementary filters on the special orthogonal group ., *IEEE Transactions on Automatic Control* **53**(5): 1203–1218.

Markley, F. L. (2003). Attitude error representations for kalman filtering, *Journal of Guidance, Control, and Dynamics* **26**(2): 311–317.

Martin, P. and Salaün, E. (2008). An invariant observer for earth-velocity-aided attitude heading reference systems, *In IFAC World Congress* **17**: 9857–9864.

Maybeck, P. S. (1979). *Stochastic Models, Estimation and Control Vol. 1*, Academic Press.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*, 2nd edn, Springer.

Schmidt, G. and Phillips, R. (2010). Ins/gps integration architectures, *NATO RTO Lecture Series* **RTO-EN-SET-116**.

Shuster, M. D. and Oh, S. (1981). Three-axis attitude determination from vector observations, *Journal of Guidance and Control* **4**(1): 70–77.

Vasconcelos, J. F., Cardeira, B., Silvestre, C., Oliveira, P. and Batista, P. (2011). Discrete-time complementary filters for attitude and position estimation: Design, analysis and experimental validation, *IEEE Transactions on Control System Technology* **19**(1): 181–198.

Vasconcelos, J., Silvestre, C. and Oliveira, P. (2011). Ins/gps aided by frequency contents of vector observations with application to autonomous surface crafts, *IEEE Journal of Oceanic Engineering* **36**(2): 347–363.

Vik, B. (n.d.). Integrated satellite and inertial navigation systems. Lecture notes, Department of Engineering Cybernetics, Norwegian University of Science and Technology.

Vik, B. and Fossen, T. I. (2001). A nonlinear observer for gps and ins integration, *Proceedings of the Conference on Decision and Control, Orlando, Florida* pp. 2956–2961.

Wahba, G. (1965). A least squares estimate of satellite attitude, *SIAM Review* **7**(3): 409.

Wendel, J. and Trommer, G. F. (2004). Tightly coupled gps/ins integration for missile applications, *Aerospace Science and Technology* **8**: 627–634.

Wenstad, P. (2010). *Gps guided r/c car*, Master's thesis, Norwegian University of Science and Technology.

Xsens (2009a). *Magnetic Field Mapper Documentation.* Revision E.

Xsens (2009b). *MT Low-Level Communication Protocol Documentation.* Revision K.

Xsens (2009c). *MT Manager User Manual.* Revision F.

Xsens (2009d). *MTi-G User Manual and Technical Documentation.* Revision G.

Yi, Y. and Grejner-Brzezinska, D. (2006). Tightly-coupled gps/ins integration using unscented kalman filter and particle filter, *Proceedings of the 19th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2006)* pp. 2182–2191.

Appendix A

Matlab Code

This appendix contains Matlab code used in the thesis. Only the main code are shown here (see the CD for the complete set). An overview of the dependencies between the different m-files are given in Figure A.1.

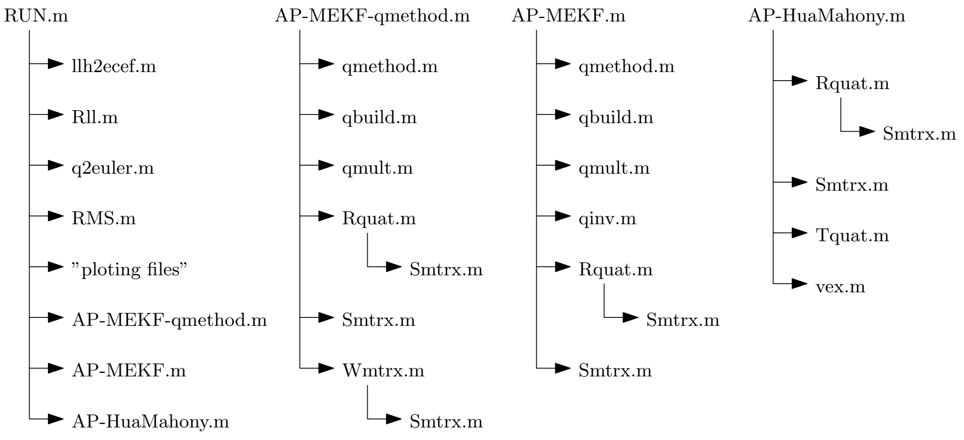


Figure A.1: Call hierarchy showing parent and child functions.

A.1 Multiplicative Extended Kalman Filter

```
function [q deps bgyr pos vel bacc] = AP_MEKF(gyr, mag, acc,...
    gps_pos, gps_vel, init, valid_mag, valid_acc, valid_pos, valid_vel)
% ------------------------------------------------------------------------
% AP_MEKF - Multiplicative Extended Kalman Filter,
% for Position, Velocity and Attitude estimation.
%
% x = [deps bgyr pos vel bacc]
% u = [gyr acc]
% y = [mag acc gps_pos gps_vel]
%
% Input     Description          Unit        Frame
% ..........................................................
% gyr       gyro                 [rad/s]     body
% mag       magnetometer         [a.u.]      body
% acc       accelerometer        [m/s^2]     body
% gps_pos   gps position         [m]         ned
% gps_vel   gps velocity         [m/s]       ned
%
% Output    Description          Unit        Frame
% ..........................................................
% q         quaternion           []          body to ned
% deps      delta epsilon        []          body to body_hat
% bgyr      bias gyro            [rad/s]     body
% pos       position             [m]         ned
% vel       velocity             [m/s]       ned
% bacc      bias accelerometer   [m/s^2]     body
%
% Author:   Harald Nøkland
% Date:     June 2011
% -------------------------------------------------------------------------
h = 0.01;                           % Sampling interval
n = 15;                             % Number of states
decl = 0.0323;                      % Magnetic declination [rad]
g_ned = [0 0 9.81]';                % Gravity [m/s^2]
gps_arm = [-0.65 -0.08 -0.90]';     % GPS lever arm [m]

var_deps = [5e-7 5e-7 5e-6];        % f(Gyro variance)
var_bgyr = [5e-8 5e-8 1e-7];        % f(Gyro bias variance)
var_pos  = [1e-4 1e-4 1e-4];
var_vel  = [1e-4 1e-4 1e-4];        % f(Accelerometer variance)
var_bacc = [1e-7 1e-7 1e-7];        % f(Accelerometer bias variance)
process  = [var_deps var_bgyr var_pos var_vel var_bacc];
Q = diag(process);                  % Process noise

var_mag  = [2e-5 2e-5 2e-5];        % Magnetometer variance
var_acc  = [2e-3 2e-3 2e-3];        % Accelerometer variance
var_pos  = [10   10   10  ];        % GPS position variance
var_vel  = [1e-4 1e-4 1e-4];        % GPS velocity variance
meas     = [var_mag var_acc var_pos var_vel];
R = diag(meas);                     % Measurement noise
% -------------------------------------------------------------------------
persistent q_hat deps_bar bgyr_bar pos_bar vel_bar bacc_bar vel_dot P_bar;
if init
    deps_bar = [0 0 0]';            % Initial delta epsilon
    bgyr_bar = [0 0 0]';            % Initial bias gyro
    pos_bar  = -gps_arm;            % Initial position
    vel_bar  = [0 0 0]';            % Initial velocity
    bacc_bar = [0 0 0]';            % Initial bias accelerometer
    vel_dot  = [0 0 0]';
    q_hat = qmethod(100,1,[0 0 1]',[cos(decl) sin(decl) 0]',...
        -acc/norm(acc),mag/norm(mag));      % Initial attitude
    P_bar = diag([1e-5 1e-5 1e-5   1e-9 1e-9 1e-9   0 0 0 ...
        1e-4 1e-4 1e-4   1e-8 1e-8 1e-8]);  % Initial error covariance
end

% Real measurement:
y = [mag; acc; gps_pos; gps_vel];

% Predicted rotation matrix:
dq_bar = qbuild(deps_bar);
R_bar  = Rquat(q_hat)*Rquat(dq_bar);

% Magnetic field reference vector:
m_ned  = R_bar*mag;
m_ned  = [norm(m_ned(1:2))*cos(decl) norm(m_ned(1:2))*sin(decl) m_ned(3)]';

```

```
% Estimated measurement:
y_bar   = [ R_bar'*m_ned
            R_bar'*(vel_dot-g_ned) + bacc_bar
            pos_bar + R_bar*gps_arm
            vel_bar + R_bar*Smtrx(gyr-bgyr_bar)*gps_arm ];

% Compute Kalman gain:
H1 = [ Wmtrx(deps_bar, Rquat(q_hat)'*m_ned)
      -Wmtrx(deps_bar, Rquat(q_hat)'*g_ned)
       zeros(3,3)
       zeros(3,3) ];
H2 = [ zeros(3,3) zeros(3,3) zeros(3,3) zeros(3,3)
       zeros(3,3) zeros(3,3) zeros(3,3) eye(3,3)
       zeros(3,3) eye(3,3)   zeros(3,3) zeros(3,3)
       zeros(3,3) zeros(3,3) eye(3,3)   zeros(3,3) ];
H = [H1 H2];
K = P_bar*H'/(H*P_bar*H' + R);

% Dead-reckoning:
if valid_mag==0, K(:,1:3)=zeros(n,3); end
if valid_acc==0, K(:,4:6)=zeros(n,3); end
if valid_pos==0, K(:,7:9)=zeros(n,3); end
if valid_vel==0, K(:,10:12)=zeros(n,3); end

% Update estimate with measurement:
deps_hat = deps_bar + K(1:3,:)  *(y - y_bar);
bgyr_hat = bgyr_bar + K(4:6,:)  *(y - y_bar);
pos_hat  = pos_bar  + K(7:9,:)  *(y - y_bar);
vel_hat  = vel_bar  + K(10:12,:)*(y - y_bar);
bacc_hat = bacc_bar + K(13:15,:)*(y - y_bar);
if norm(deps_hat)>1, deps_hat=deps_hat/norm(deps_hat); end %saturation

% Reset:
dq_hat    = qbuild(deps_hat);
q_hat     = qmult(q_hat, dq_hat);
q_hat     = q_hat/norm(q_hat);
deps_hat = [0 0 0]';

% Compute error covariance for updated estimate:
IKH = eye(n) - K*H;
P   = IKH*P_bar*IKH' + K*R*K';

% Project ahead:
deps_dot = 0.5*(gyr - bgyr_hat);
bgyr_dot = [0 0 0]';
pos_dot  = vel_hat;
vel_dot  = Rquat(q_hat)*(acc - bacc_hat) + g_ned;
bacc_dot = [0 0 0]';

deps_bar = deps_hat + h*deps_dot;
bgyr_bar = bgyr_hat + h*bgyr_dot;
pos_bar  = pos_hat  + h*pos_dot;
vel_bar  = vel_hat  + h*vel_dot;
bacc_bar = bacc_hat + h*bacc_dot;
if norm(deps_bar)>1, deps_bar=deps_bar/norm(deps_bar); end %saturation

PHI1 = [-0.5*Smtrx(gyr - bgyr_hat)
         zeros(3,3)
         zeros(3,3)
        -2*Rquat(q_hat)*Smtrx(acc - bacc_hat)
         zeros(3,3) ];
PHI2 = [-0.5*eye(3,3) zeros(3,3) zeros(3,3)  zeros(3,3)
         zeros(3,3)   zeros(3,3) zeros(3,3)  zeros(3,3)
         zeros(3,3)   zeros(3,3) eye(3,3)    zeros(3,3)
         zeros(3,3)   zeros(3,3) zeros(3,3) -Rquat(q_hat)
         zeros(3,3)   zeros(3,3) zeros(3,3)  zeros(3,3) ];

PHI   = eye(n,n) + h*[PHI1 PHI2];
```

```
GAMMA  = h*eye(n,n);
P_bar  = PHI*P*PHI' + GAMMA*Q*GAMMA';

% Output:
q     = q_hat;
deps  = deps_hat;
bgyr  = bgyr_hat;
pos   = pos_hat;
vel   = vel_hat;
bacc  = bacc_hat;
```

A.2 Multiplicative Extended Kalman Filter (q-method)

```
function [q deps bgyr pos vel bacc] = AP_MEKF_qmethod(gyr, mag, acc,...
    gps_pos, gps_vel, init, valid_mag, valid_acc, valid_pos, valid_vel)
% ------------------------------------------------------------------------
% AP_MEKF - Multiplicative Extended Kalman Filter (q-method),
% for Position, Velocity and Attitude estimation.
%
% x = [deps bgyr pos vel bacc]
% u = [gyr acc]
% y = [deps gps_pos gps_vel]
%
% Input     Description          Unit        Frame
% ...........................................................
% gyr       gyro                 [rad/s]     body
% mag       magnetometer         [a.u.]      body
% acc       accelerometer        [m/s^2]     body
% gps_pos   gps position         [m]         ned
% gps_vel   gps velocity         [m/s]       ned
%
% Output    Description          Unit        Frame
% ...........................................................
% q         quaternion           []          body to ned
% deps      delta epsilon        []          body to body_hat
% bgyr      bias gyro            [rad/s]     body
% pos       position             [m]         ned
% vel       velocity             [m/s]       ned
% bacc      bias accelerometer   [m/s^2]     body
%
% Author:   Harald Nøkland
% Date:     June 2011
% ------------------------------------------------------------------------
h = 0.01;                           % Sampling interval
n = 15;                             % Number of states
decl = 0.0323;                      % Magnetic declination [rad]
g_ned = [0 0 9.81]';                % Gravity [m/s^2]
gps_arm = [-0.65 -0.08 -0.90]';     % GPS lever arm [m]

var_deps = [5e-7 5e-7 5e-7];        % f(Gyro variance)
var_bgyr = [1e-9 1e-9 1e-9];        % f(Gyro bias variance)
var_pos  = [1e-0 1e-0 1e-0];
var_vel  = [7e-4 7e-4 7e-4];        % f(Accelerometer variance)
var_bacc = [1e-7 1e-7 1e-7];        % f(Accelerometer bias variance)
process  = [var_deps var_bgyr var_pos var_vel var_bacc];
Q = diag(process);                  % Process noise

var_deps = [4e-4 4e-4 4e-4];        % qmethod variance
var_pos  = [1e-1 1e-1 1e-1];        % GPS position variance
var_vel  = [5e-4 5e-4 5e-4];        % GPS velocity variance
meas     = [var_deps var_pos var_vel];
R = diag(meas);                     % Measurement noise
% ------------------------------------------------------------------------
```

```
persistent q_hat deps_bar bgyr_bar pos_bar vel_bar bacc_bar vel_dot P_bar;
if init
    deps_bar = [0 0 0]';        % Initial delta epsilon
    bgyr_bar = [0 0 0]';        % Initial bias gyro
    pos_bar  = -gps_arm;        % Initial position
    vel_bar  = [0 0 0]';        % Initial velocity
    bacc_bar = [0 0 0]';        % Initial bias accelerometer
    vel_dot  = [0 0 0]';
    q_hat = qmethod(100,1,[0 0 1]',[cos(decl) sin(decl) 0]',...
        -acc/norm(acc),mag/norm(mag));      % Initial attitude
    P_bar = diag([1e-5 1e-5 1e-5   1e-9 1e-9 1e-9   1 1 1 ...
        1e-3 1e-3 1e-3   1e-8 1e-8 1e-8]);  % Initial error covariance
end

% Predicted rotation matrix:
dq_bar = qbuild(deps_bar);
R_bar  = Rquat(q_hat)*Rquat(dq_bar);

% Magnetic field reference vector:
m_ned  = R_bar*mag;
m_ned  = [norm(m_ned(1:2))*cos(decl) norm(m_ned(1:2))*sin(decl) m_ned(3)]';

% Acceleration reference vector:
a_ned = vel_dot - g_ned;
a     = acc - bacc_bar;

% q-method:
q = qmethod(1,1,a_ned/norm(a_ned),m_ned/norm(m_ned),a/norm(a),mag/norm(mag));
if norm(q+q_hat) < norm(q-q_hat)
    q = -q;
end
dq = qmult(qinv(q_hat),q);

% Real measurement:
y = [dq(2:4); gps_pos; gps_vel];

% Estimated measurement:
y_bar   = [ deps_bar
            pos_bar + R_bar*gps_arm
            vel_bar + R_bar*Smtrx(gyr-bgyr_bar)*gps_arm ];

% Compute Kalman gain:
H = [ eye(3,3)      zeros(3,3) zeros(3,3) zeros(3,3) zeros(3,3)
      zeros(3,3)    zeros(3,3) eye(3,3)   zeros(3,3) zeros(3,3)
      zeros(3,3)    zeros(3,3) zeros(3,3) eye(3,3)   zeros(3,3) ];
K = P_bar*H'/(H*P_bar*H' + R);

% Dead-reckoning:
valid_deps = valid_mag & valid_acc;
if valid_deps==0, K(:,1:3)=zeros(n,3); end
if valid_pos==0,  K(:,4:6)=zeros(n,3); end
if valid_vel==0,  K(:,7:9)=zeros(n,3); end

% Update estimate with measurement:
deps_hat = deps_bar + K(1:3,:)   *(y - y_bar);
bgyr_hat = bgyr_bar + K(4:6,:)   *(y - y_bar);
pos_hat  = pos_bar  + K(7:9,:)   *(y - y_bar);
vel_hat  = vel_bar  + K(10:12,:)*(y - y_bar);
bacc_hat = bacc_bar + K(13:15,:)*(y - y_bar);
if norm(deps_hat)>1, deps_hat=deps_hat/norm(deps_hat); end %saturation

% Reset:
dq_hat   = qbuild(deps_hat);
q_hat    = qmult(q_hat, dq_hat);
q_hat    = q_hat/norm(q_hat);
deps_hat = [0 0 0]';

% Compute error covariance for updated estimate:
```

```
IKH = eye(n) - K*H;
P   = IKH*P_bar*IKH' + K*R*K';

% Project ahead:
deps_dot = 0.5*(gyr - bgyr_hat);
bgyr_dot = [0 0 0]';
pos_dot  = vel_hat;
vel_dot  = Rquat(q_hat)*(acc - bacc_hat) + g_ned;
bacc_dot = [0 0 0]';

deps_bar = deps_hat + h*deps_dot;
bgyr_bar = bgyr_hat + h*bgyr_dot;
pos_bar  = pos_hat  + h*pos_dot;
vel_bar  = vel_hat  + h*vel_dot;
bacc_bar = bacc_hat + h*bacc_dot;
if norm(deps_bar)>1, deps_bar=deps_bar/norm(deps_bar); end %saturation

PHI1 = [-0.5*Smtrx(gyr - bgyr_hat)
         zeros(3,3)
         zeros(3,3)
        -2*Rquat(q_hat)*Smtrx(acc - bacc_hat)
         zeros(3,3) ];
PHI2 = [-0.5*eye(3,3) zeros(3,3) zeros(3,3)  zeros(3,3)
         zeros(3,3)   zeros(3,3) zeros(3,3)  zeros(3,3)
         zeros(3,3)   zeros(3,3) eye(3,3)    zeros(3,3)
         zeros(3,3)   zeros(3,3) zeros(3,3) -Rquat(q_hat)
         zeros(3,3)   zeros(3,3) zeros(3,3)  zeros(3,3) ];

PHI   = eye(n,n) + h*[PHI1 PHI2];
GAMMA = h*eye(n,n);
P_bar = PHI*P*PHI' + GAMMA*Q*GAMMA';

% Output:
q    = q_hat;
deps = deps_hat;
bgyr = bgyr_hat;
pos  = pos_hat;
vel  = vel_hat;
bacc = bacc_hat;
```

A.3 Nonlinear Observer HuaMahony

```
function [q bgyr pos vel] = AP_HuaMahony(gyr, mag, acc,...
    gps_pos, gps_vel, init, valid_mag, valid_acc, valid_pos, valid_vel)
% ------------------------------------------------------------------------
% AP_HuaMahony - Nonlinear observer,
% for Position, Velocity and Attitude estimation.
%
% Input     Description         Unit        Frame
% ...........................................................
% gyr       gyro                [rad/s]     body
% mag       magnetometer        [a.u.]      body
% acc       accelerometer       [m/s^2]     body
% gps_pos   gps position        [m]         ned
% gps_vel   gps velocity        [m/s]       ned
%
% Output    Description         Unit        Frame
% ...........................................................
% q         quaternion          []          body to ned
% bgyr      bias gyro           [rad/s]     body
% pos       position            [m]         ned
% vel       velocity            [m/s]       ned
%
```

```matlab
% Author:   Harald Nøkland
% Date:     June 2011
% ----------------------------------------------------------------------
h = 0.01;                               % Sampling interval
decl = 0.0323;                          % Magnetic declination [rad]
g_ned = [0 0 9.81]';                    % Gravity [m/s^2]
gps_arm = [-0.65 -0.08 -0.90]';         % GPS lever arm [m]

Kquat = diag([1 1 10])*h;
Kbgyr = diag([0.2 0.2 0.5])*h;
kmag = 1; kacc = 1;

kpos = 0.0001*h*25;
kvel = 3*h*25;
kQ = 0.06*h*25;
% ----------------------------------------------------------------------
persistent q_bar bgyr_bar pos_bar vel_bar Q_bar;
if init
    bgyr_bar = [0 0 0]';                % Initial bias gyro
    pos_bar  = -gps_arm;                % Initial position
    vel_bar  = [0 0 0]';                % Initial velocity
    Q_bar    = eye(3);                  % Initial Q
    q_bar = qmethod(100,1,[0 0 1]',[cos(decl) sin(decl) 0]',...
                    -acc/norm(acc),mag/norm(mag)); % Initial attitude
end

% Dead-reckoning:
if valid_mag==0, kmag=0; end
if valid_acc==0, kacc=0; end
if valid_pos==0, kpos=0; end
if valid_vel==0, kvel=0; kQ=0; end

% Real measurement:
m = mag/norm(mag);
a = acc/norm(acc);

% Predicted rotation matrix:
R_bar = Rquat(q_bar);

% Lever arm compensation:
gps_pos = gps_pos - R_bar*gps_arm;
gps_vel = gps_vel - R_bar*Smtrx(gyr-bgyr_bar)*gps_arm;

% Magnetic field reference vector:
m_ned = R_bar*mag;
m_ned = [norm(m_ned(1:2))*cos(decl) norm(m_ned(1:2))*sin(decl) m_ned(3)]';

% Acceleration reference vector:
a_ned = Q_bar*acc + kvel*(gps_vel - vel_bar);

% Estimated measurement:
m_bar = R_bar'*m_ned/norm(m_ned);
a_bar = R_bar'*a_ned/norm(a_ned);

% Update estimate with measurement:
m_err = 0.5*(m*m_bar' - m_bar*m');
a_err = 0.5*(a*a_bar' - a_bar*a');
w_mes = -vex(kmag*m_err + kacc*a_err);

q_hat    = q_bar    + 0.5*Tquat(q_bar)*Kquat*w_mes;
bgyr_hat = bgyr_bar - Kbgyr*w_mes;
pos_hat  = pos_bar  + kpos*(gps_pos - pos_bar);
vel_hat  = vel_bar  + kvel*(gps_vel - vel_bar);
Q_hat    = Q_bar    + kQ*(gps_vel - vel_bar)*acc';

% Normalize:
q_hat = q_hat/norm(q_hat);
Q_hat = Q_hat/norm(Q_hat,'fro')*sqrt(3);
```

```

% Project ahead:
q_dot     = 0.5*Tquat(q_hat)*(gyr - bgyr_hat);
bgyr_dot = [0 0 0]';
pos_dot  = vel_hat;
vel_dot  = Q_hat*acc + g_ned;
Q_dot    = Q_hat*Smtrx(gyr - bgyr_hat);

q_bar    = q_hat    + h*q_dot;
bgyr_bar = bgyr_hat + h*bgyr_dot;
pos_bar  = pos_hat  + h*pos_dot;
vel_bar  = vel_hat  + h*vel_dot;
Q_bar    = Q_hat    + h*Q_dot;

% Normalize:
q_bar = q_bar/norm(q_bar);
Q_bar = Q_bar/norm(Q_bar,'fro')*sqrt(3);

% Output:
q    = q_hat;
bgyr = bgyr_hat;
pos  = pos_hat;
vel  = vel_hat;
```

A.4 q-method

```
function q = qmethod(a1, a2, r1, r2, b1, b2)
% -------------------------------------------------------------------------
% The q-method computes the unit quaternion q = [eta eps1 eps2 eps3]'
% from a set of vector measurements {b1 b2} in the body frame. The
% set {r1 r2} is the corresponding reference vectors in the reference
% frame, and {a1 a2} are weights. It is the solution to an optimization
% problem of minimizing the objective function:
%
% f(q) = a1*norm(r1-R(q)*b1) + a2*norm(r2-R(q)*b2)
%
% Input:
% a - weight
% r - reference vector   (NED)
% b - observation vector (BODY)
%
% Author:   Harald Nøkland
% Date:     June 2011
% -------------------------------------------------------------------------
Z = a1*cross(r1,b1) + a2*cross(r2,b2);
B = a1*r1*b1' + a2*r2*b2';
S = B + B';
sigma = trace(B);

% Compute the K matrix
K = [ -sigma     Z'
       Z        -S+sigma*eye(3) ];

% Find the eigenvector for the smallest eigenvalue of K
[V,E] = eig(K);
[m,i] = min([E(1,1) E(2,2) E(3,3) E(4,4)]);
q = V(:,i);
q = q/norm(q);
```

Appendix B

Simulink Diagrams

This appendix contains the Simulink diagrams for the simulator. Only the main diagrams are shown here (see the CD for the complete set).

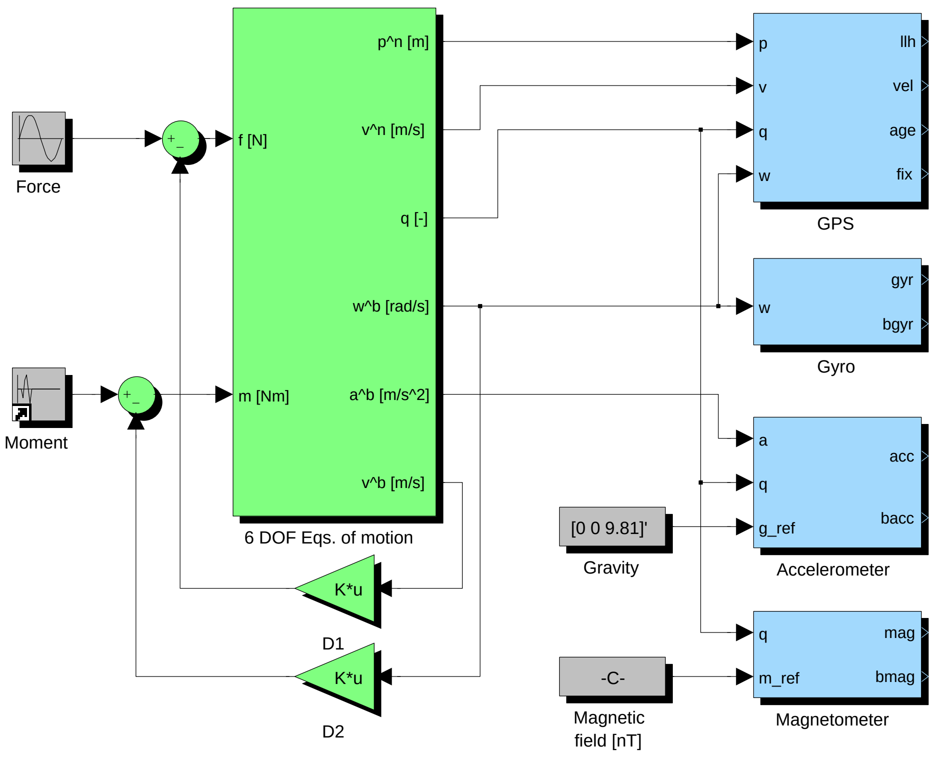


Figure B.1: Simulator model

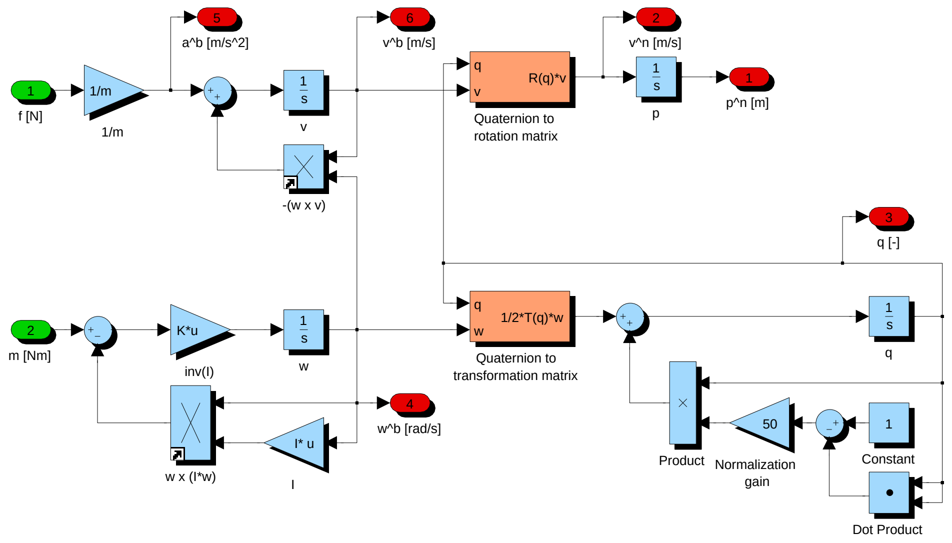


Figure B.2: 6 DOF Equations of motion

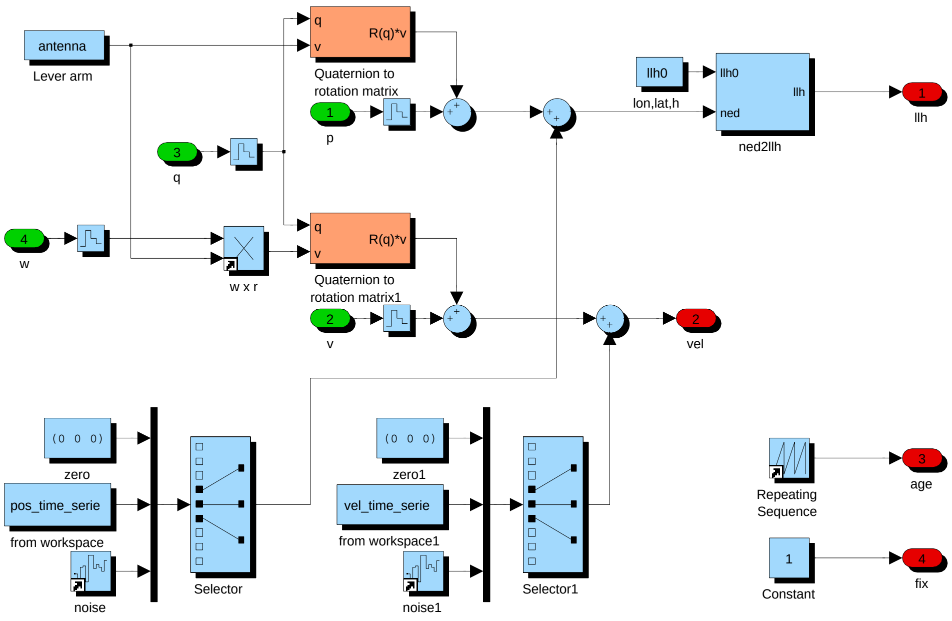


Figure B.3: GPS model

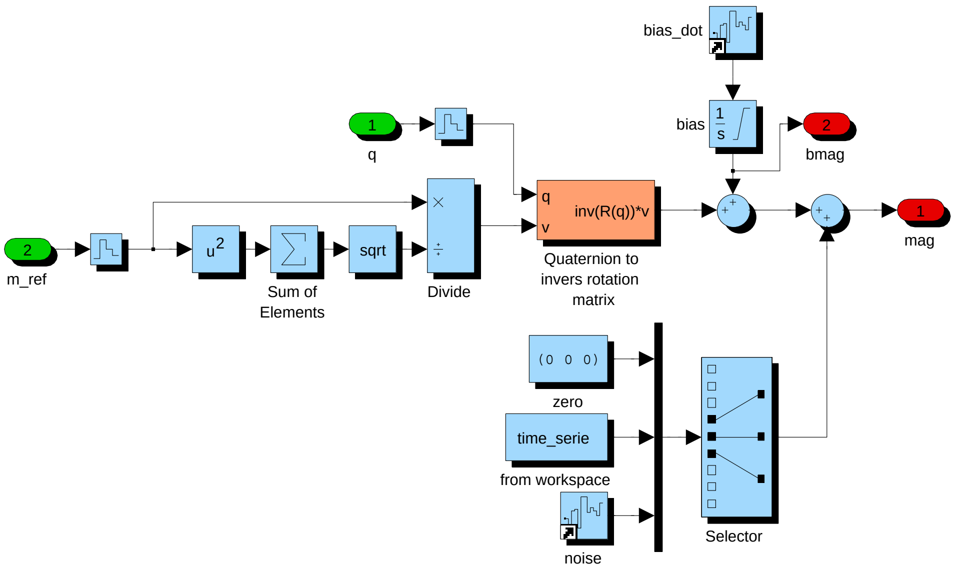


Figure B.4: Magnetometer model

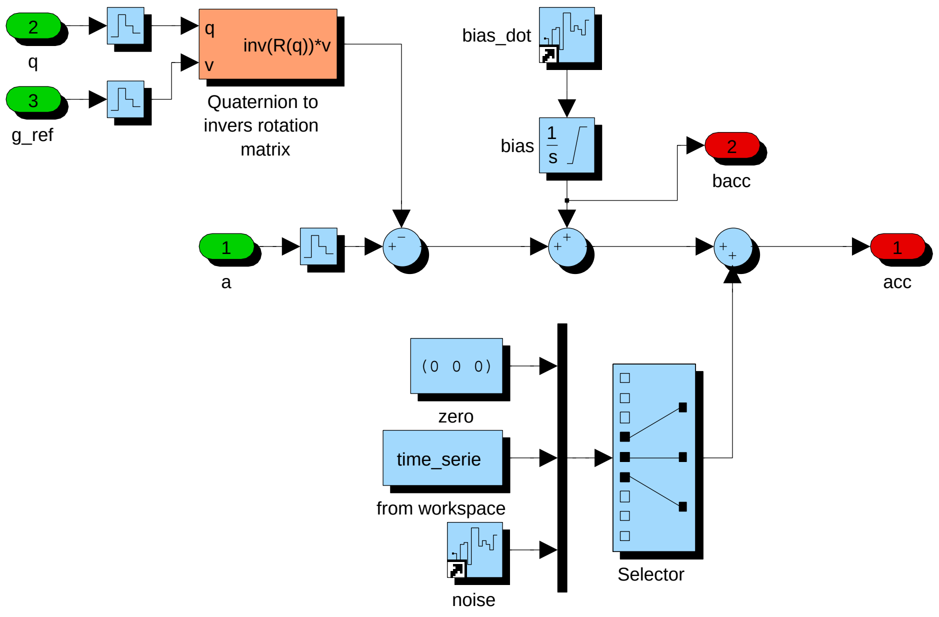


Figure B.5: Accelerometer model

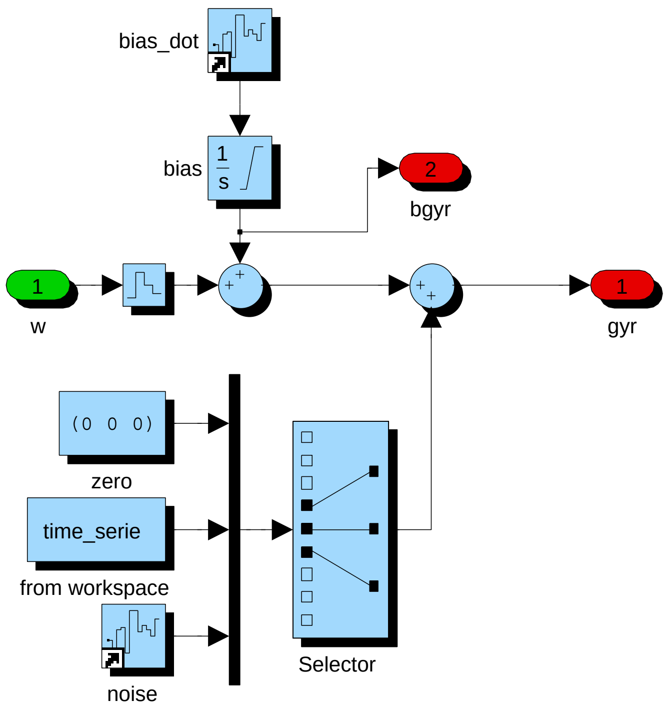


Figure B.6: Gyro model

Appendix C

CD Content

All files needed to run the nonlinear observers developed in this thesis are included on the CD. Some of the m-files are from the *MSS GNC toolbox*[1], which is marked in the column named “T” for toolbox.

Table C.1: Matlab files included on CD

File	T	Description
A_EKF.m		Attitude estimation, extended Kalman filter
A_MEKF.m		Attitude estimation, multiplicative extended Kalman filter
AP_HuaMahony.m		Attitude, position and velocity estimation, nonlinear observer HuaMahony
AP_MEKF.m		Attitude, position and velocity estimation, multiplicative extended Kalman filter
AP_MEKF_qmethod.m		Attitude, position and velocity estimation, multiplicative extended Kalman filter using q-method
dllh2dned.m		Transformation from longitude, latitude and height to NED coordinates
ecef2llh.m	x	Transformation from Cartesian to ellipsoidal ECEF coordinates
euler2q.m	x	Transformation for Euler angles to quaternion
llh2ecef.m	x	Transformation form ellipsoidal to Cartesian ECEF coordinates
Omega.m		Matrix in the quaternion differential equation

[1]MSS GNC is a Matlab toolbox for guidance, navigation and control. The toolbox is part of the Marine Systems Simulator (MSS). URL: <http://www.marinecontrol.org>

Table C.2: Matlab files included on CD

File	T	Description
q2euler.m	x	Transformation from quaternion to Euler angles
qbuild.m		Construct the quaternion from the unit constraint
qinv.m		Inverse quaternion
qmethod.m		q-method, attitude determination
qmult.m		quaternion multiplication
R2euler.m	x	Transformation from rotation matrix to Euler angles
Rll.m	x	Rotation matrix from NED to ECEF
RMS.m		Root mean square calculation
Rquat.m	x	Rotation matrix from BODY to NED
RUN.m		Framework, main file
Rzyx.m	x	Rotation matrix from BODY to NED
Smtrx.m	x	The skew-symmetric matrix
Tquat.m		Matrix in the quaternion differential equation
vex.m		Vex operator
Vmtrx.m		Jacobian of rotation matrix
Wmtrx.m		Jacobian of Transposed rotation matrix

Table C.3: Data files (.mat) included on CD

File	Description
citytour1.mat	Experimental data
citytour2.mat	Experimental data
citytour3.mat	Experimental data
citytour4.mat	Experimental data
highway1.mat	Experimental data
highway2.mat	Experimental data
roundabout.mat	Experimental data
withoutmotion.mat	Experimental data
xsens_noise.mat	Experimental data, for use in the simulator
01_perfect_data.mat	Simulation data
02_noise_only.mat	Simulation data
03_gyro_bias.mat	Simulation data
04_mag_bias.mat	Simulation data
05_acc_bias.mat	Simulation data
06_const_force.mat	Simulation data

Table C.4: Other files included on CD

File	Description
extractPVT.c	Decodes the MTB log files from Xsens unit
Instrument_Simulation_thesis.mdl	Simulink simulator
MSc_Thesis_Nøkland(2011).pdf	This Report