DEGREE PROJECT IN VEHICLE ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2018

Design and Implementation of an Exogenous Kalman Filter for UAVs

PATRIK SJÖBERG

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Abstract

State estimations for nonlinear systems are often done using the Extended Kalman Filter (EKF), which has good performance in terms of minimum variance for the estimation error. However, global stability is not guaranteed, and the filter has a risk of becoming unstable. A recent paper proposes the eXogenous Kalman Filter (XKF), which combines the stability properties of a nonlinear observer and the performance of the Kalman filter. The combination of the two observers leads to better stability while the accuracy of the estimations is maintained. The topic of this thesis is the design and implementation of an XKF for the control system of a UAV. The filter was simulated, and the performance was compared to the existing EKF. Once the algorithm for the filter was constructed, it was implemented in a real avionics system for UAVs and tested using real sensors. It was shown through simulations that the XKF indeed provides better stability properties for the state estimations compared to the EKF, without loss of accuracy.

Sammanfattning

Skattningen av tillståndsvariabler för olinjära system görs ofta med det så kallade Extended Kalman Filter (EKF), som har bra noggrannhet när det kommer till kovariansmatrisen för rekonstruktionsfelet. Problemet är att stabiliteten i skattningarna inte kan garanteras så att filtret riskerar att bli instabilt. En nyligen publicerad artikel föreslår det så kallade eXogenous Kalman Filter (XKF), som kombinerar stabiliteten från en olinjär observatör med noggrannheten från ett Kalmanfilter. Kombinationen av de två observatörerna leder till bättre stabilitet utan att noggrannheten försämras. Det här examensarbetet går ut på att utveckla och implementera ett XKF i ett styrsystem för drönare. Filtret har simulerats och utvärderats i jämförelse med ett existerande EKF. När algoritmen för filtret var skapat så implementerades filtret i ett riktigt avioniksystem för drönare och testades med hjälp av riktiga sensorer. Det har visats genom simuleringar att XKF har bättre stabilitetsegenskaper jämfört med EKF, utan förlust av noggrannhet.

Acknowledgements

I would like to thank Philip Nyströmer for allowing me to do my master thesis at his company and for his continuous support from day one. I am also grateful for being introduced to the business and for the valuable insights that have come from it.

I would also like to thank Sami Gustafsson at Nystromer Avionics for his tireless effort during the programming and debugging of the code.

Finally, I would like to express my gratitude to my supervisors Linnea Persson and Bo Wahlberg at KTH for their support regarding the technical work and the project overall.

Contents

Abbreviations **vi**

1 Introduction **1**
1.1 Unmanned Aerial Vehicles . . . 1
1.2 State Estimation . . . 2
1.3 Problem Formulation . . . 3
1.4 Outline of Thesis . . . 3

2 Reference Frames **4**
2.1 Definitions . . . 4
2.2 Converting Between Reference Frames . . . 4
2.2.1 Rotation Matrix . . . 5
2.2.2 Euler Angles . . . 6
2.2.3 Direction Cosine Matrix . . . 6
2.2.4 Geodetic Coordinates . . . 7
2.2.5 Quaternion . . . 8
2.2.6 Finding the Quaternion . . . 9
2.2.7 Quaternion as State Variable . . . 10

3 Governing Equations **12**
3.1 Equations of Motion . . . 12
3.1.1 State Space Form . . . 13
3.1.2 Gravity Model . . . 14
3.2 Measurements . . . 14
3.2.1 Noise and Bias . . . 14
3.2.2 Measurement Model . . . 15

4 The Kalman Filter **17**
4.1 Optimal Kalman Filter . . . 17
4.2 Linearized Kalman Filter . . . 18
4.3 Extended Kalman Filter . . . 20
4.4 Nonlinear Observers . . . 21
4.5 eXogenous Kalman Filter . . . 21
4.6 Summary of Differences Between the Filters . . . 22

5 Practical Aspects **24**
5.1 Nonlinear Observer for UAV Application 24
5.1.1 Discretization of the Attitude Observer 26
5.1.2 Discretization of the TMO 27
5.1.3 Gain Selection 28
5.2 Linearized Kalman Filter 29
5.3 Algorithm 29
5.3.1 Initializing the Filter 29
5.3.2 Kalman Filter Loop 30
5.3.3 Managing the Measurements 31
5.4 Wind Estimation 32

6 Stratos Pilot System **35**
6.1 Stratos Pilot Core Loop 35
6.2 Sensors 36
6.2.1 Covariance Matrices 36

7 Simulations **38**
7.1 Simulation Environment 38
7.1.1 Generation of Input Data 39
7.1.2 Generating Measurements 41
7.2 Gains 41
7.3 Simulation Results 42
7.3.1 Perfect Initialization 42
7.3.2 Unstable EKF 44
7.3.3 Unstable XKF 45
7.4 State Estimation Covariance Matrix 46
7.5 Improvement of Bias Estimation Method 47
7.6 Wind Estimation 50

8 Implementation and Testing **52**
8.1 Test Rig 52
8.2 Implementation Process 52
8.3 Ground Testing 54

9 Conclusions **55**
9.1 Comparison Between XKF and EKF 55
9.2 Future of the XKF 56
9.3 Future Work 56

Appendices **58**

A XKF Algorithm **59**

B Jacobi Matrices **62**

C Covariance Matrices **67**

Abbreviations

ECEF	Earth-Centered-Earth-Fixed
EKF	Extended Kalman Filter
ENU	East-North-Up
FSS	Full Scale Span
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
IMU	Inertial Measurement Unit
KF	Kalman Filter
LKF	Linearized Kalman Filter
NED	North-East-Down
NLO	NonLinear Observer
RMSE	Root-Mean-Square Error
TMO	Translational-Motion Observer
UAV	Unmanned Aerial Vehicle
WMM	World Magnetic Model
XKF	eXogenous Kalman Filter

Chapter 1

Introduction

This master thesis was conducted at Nystromer Avionics AB, which is a Swedish company that specializes in control systems for unmanned aerial vehicles (UAVs). The task was to improve the method of state estimation (compared to current solution) by implementing an eXogenous Kalman Filter (XKF) in the control system. The performance of the filter was verified through simulations and experiments. The thesis was performed in cooperation with the department of Automatic Control at KTH Royal Institute of Technology in Stockholm.

This chapter will give a brief introduction to the concept of UAVs and state estimation in order to put the thesis in context. The problem formulation will be specified based on the background of the thesis. The outline of the thesis report will be presented at the end of this chapter.

1.1 Unmanned Aerial Vehicles

An unmanned aerial vehicle (UAV) is often used where the mission either requires no pilot or where the potential risk for the pilot is deemed too high. Today, UAVs are used in many applications, both military and civilian. UAVs are often controlled remotely by a pilot on the ground, but completely autonomous systems are being introduced more and more. Fixed wing UAVs and so called quadcopters are getting more popular on the market so the demand for cheaper and robust solutions is increasing.

It is possible to design the UAV to efficiently conduct a mission due to the fact that there is no need for a pilot and hence the size of the aircraft can be significantly reduced. Lighter aircrafts have generally more freedom in the choice of propellant and in the design, which makes it possible to adapt the UAV to its mission more efficiently. Even complete electrical powered solutions, with for example solar panels, are possible in a much larger scale compared to regular passenger aircraft. Creative ideas for future applications of UAVs could improve efficiency, safety and reduce environmental impact of many tasks performed by different means today.

Regardless of the mission and design of the UAV, the control system needs to be robust and provide good performance. In order for this control system

to give efficient control inputs it is required to have good estimations of the current state of the UAV (for example position and attitude). Sensors attached to the structure provide information about this in terms of measurements, but some post processing is needed to improve the quality of the final estimates.

1.2 State Estimation

Noise is almost always present in any type of measurement from a sensor. It makes it challenging to determine what the actual value is for what is intended to estimate. It is usually assumed that the measurement noise is Gaussian white noise and the standard deviation of this random signal is depending on the quality of the sensor. Designing the sensors and systems so that the generation of noise is minimized is a first step to improve the quality of any measurements, but it is impossible to do so perfectly.

While accepting that noise will be part of the measurements the question is rather how to reduce the impact of the noise to the accuracy of the estimates. Many people have tried to solve this problem by filtering the signals, i.e. removing the noise from the measurements. Before Kalman presented his result in 1960 [1], the Wiener filter was commonly used. By using the mean-square error the Wiener filter weighting functions could be optimized and the filter could be seen as optimal. However, the problem with this filter was that it was not suitable for multiple-input/output time-variable problems [2]. What Kalman instead presented was a way to formulate the problem of mean-square error by using the state space methods. This allowed for vector modeling and also recursive processing of the measurements [2]. Since then Kalman filtering has become a standard method of processing noisy measurements and gaining better state estimations.

The Kalman Filter (KF) is hence an observer with the principle that a state space model is used to predict the states in the next time step. This prediction is improved by a correction term, which is the difference between the real measurements and the expected measurements from the model, weighted by the so called Kalman gain. In the linear case, the KF provides the optimal estimation i.e. minimum estimation error under some assumptions [2]. For the nonlinear case, a linearized Kalman filter (LKF) can be applied, which involves linearization of the nonlinearities that leads to the loss of optimality. If the linearization is made every time step it is often referred to as the Extended Kalman Filter (EKF), which is a commonly used method for state estimation in nonlinear systems. The strength with this approach is that the filter provides sub-optimal performance in terms of estimation error, but since linearization is made there is a chance for the filter to become unstable.

A nonlinear observer (NLO) preserves the nonlinearities of the system for state estimation and is designed differently for each system. By not linearizing the system it can be proven (usually with Lyapunov stability theory) that the designed NLO is globally stable. However, the performance of the observer, in terms of estimation error, is rarely considered in the design [3].

A new approach to estimating states in nonlinear systems has recently been presented in [3] that combines the advantages of the LKF and NLO while excluding their respective drawbacks. This eXogenous Kalman filter (XKF) uses a stable estimation from an NLO as linearization point in an LKF, which improves the quality of the estimate. It is shown in [3] that the two-step observer inherits the stability properties of the NLO without loss of accuracy compared to the EKF.

1.3 Problem Formulation

The main objective with this thesis is to design and implement an XKF in the Stratos Pilot control system for UAVs, developed by Nystromer Avionics AB. The performance of the filter will be evaluated through simulations prior to the implementation phase and verified with flight tests. The desired goal is that the XKF should have better stability properties compared to the implemented EKF without losing accuracy in the estimations. Meeting the design goal means that the Stratos Pilot control system will be more robust than it would have been with an EKF.

1.4 Outline of Thesis

Chapter 2 presents the reference frames that will be used throughout this report. The different ways of transforming a vector between these frames is an important topic, also discussed in this chapter. The quaternion will be defined, and it will be explained how it can be used as a state variable to represent the attitude of the UAV.

Chapter 3 will focus on the equations of motion for the UAV. Two forms of these equations will be presented as they are useful in different applications. The measurement model will be presented together with a discussion about the disturbances present in the measurements.

Chapter 4 presents the theory regarding the Kalman filters with the main focus on the XKF.

Chapter 5 is about the practical aspects of the XKF. The nonlinear observer and the discretized equations will be presented. Also, the designed algorithm for the whole filter is described. In addition, an observer for the wind velocity and direction will be proposed as a compliment to the XKF.

Chapter 6 is about the Stratos Pilot System in which the XKF will be implemented. The overall structure of the system and the sensors used will be presented.

Chapter 7 presents the simulation model built for this project and the results from multiple simulations.

Chapter 8 contains a description about the implementation process and the result from the tests performed using the XKF in the real system.

Chapter 9 concludes this report and the final remarks about the filters is given. Also, the suggested future work related to this project will be stated.

Chapter 2

Reference Frames

This chapter will present the reference frames used in the presentation of the governing equations. The chapter starts with brief definitions of the frames, followed by the methods for transforming a vector between multiple frames. The definition of the quaternion will be a major part of this chapter and it is concluded by a description of the quaternion when used as a state variable.

2.1 Definitions

Multiple reference frames are needed to express the position and attitude of the UAV. The attitude of an aircraft is most commonly expressed as the Euler angles (defined later) between the body fixed frame of reference and the North-East-Down (NED) frame of reference. These coordinate systems will be referred to as the body- and NED-frame respectively. The body-frame for an aircraft is defined by having the first axis positive in the nose direction, the second axis positive out of the right wing and the third is defined with the right-hand rule to point below the aircraft.

The position and velocity of a UAV are expressed in a third system, which is the Earth-centered-Earth-fixed system (ECEF). This is a common reference frame for aerospace applications and it is suitable to express the equations of motion in. This system is referred to as the earth-frame and can be seen in Figure 2.1 together with the East-North-Up (ENU) system. The rotation rates are measured in relation to an inertial-frame. The inertial-frame is defined as the frame in which Newton's laws of motion hold, [4].

2.2 Converting Between Reference Frames

When working with multiple reference frames it is necessary to be able to transform any vector quantity between these frames. The methods used in this report to will be presented here.

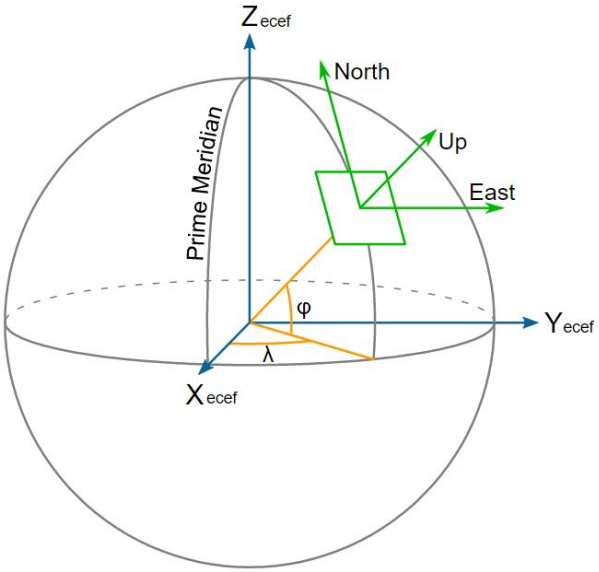


Figure 2.1: The ECEF and ENU frame of reference. NED can be obtained by just replacing the Up-axis to a Down-axis. [5]

2.2.1 Rotation Matrix

A convenient method of converting between reference frames is to use rotation matrices. For example, a velocity vector expressed in the body-frame can be converted to the earth-frame by a simple matrix multiplication

$$\boldsymbol{v}^e = \boldsymbol{C}_b^e \boldsymbol{v}^b, \tag{2.1}$$

where $\boldsymbol{C}_b^e$ is the rotation matrix between the body- and the earth-frame of reference. The inverse rotation is given by

$$\boldsymbol{v}^b = \boldsymbol{C}_e^b \boldsymbol{v}^e, \tag{2.2}$$

where the relation between the two rotation matrices are, according to [6],

$$\boldsymbol{C}_b^e = (\boldsymbol{C}_e^b)^{-1} = (\boldsymbol{C}_e^b)^T. \tag{2.3}$$

Rotation through an intermediate frame is also done by

$$\boldsymbol{v}^b = \boldsymbol{C}_n^b \boldsymbol{C}_e^n \boldsymbol{v}^e, \tag{2.4}$$

where we now transform a velocity vector from earth- to body-frame while using the NED-frame as an intermediate frame of reference. Rotation matrices are very simple to use when they are known. However, the problem with the rotation matrix is that it is not always trivial how these matrices should be obtained.

2.2.2 Euler Angles

In aeronautics, it is very common to express the attitude of the aircraft by the Euler angles between the body- and the NED-frame. The Euler angles correspond to a sequence of rotations about coordinate axes. The order of rotation can be chosen arbitrarily leading to twelve possible sequences. It should be noted that every rotation made is around the system created by the previous rotation. The Euler angle definition between the NED- and body-frame can be seen in Figure 2.2 where the direction of each rotation is defined.

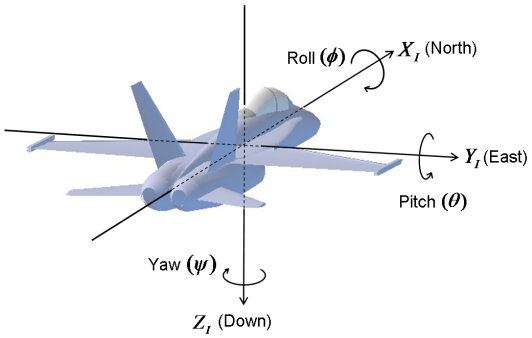


Figure 2.2: The Euler angles [7].

The order of rotation used in the aerospace community (sometimes referred to as the aerospace sequence, [8]) will be used here when transforming between the NED-frame to the body-frame. The aerospace sequence is specified by an initial rotation around the z-axis (ψ, yaw/heading). This rotation is followed by a rotation about the y-axis of the new system (θ, pitch/elevation) and the last rotation is around the x-axis in the previous system (ϕ, roll/bank), [8]. The Euler angles can be seen in Figure 2.3 where the order of rotation follows this particular order.

2.2.3 Direction Cosine Matrix

With the above definition of the Euler angels, the rotation matrix which performs the rotation from the NED-frame to the body-frame is, according to [6] and [8], given by

$$\boldsymbol{C}_n^b = \begin{pmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\theta\cos\psi\sin\phi - \sin\psi\cos\phi & \sin\theta\sin\psi\sin\phi + \cos\psi\cos\phi & \cos\theta\sin\phi \\ \sin\theta\cos\psi\cos\phi + \sin\psi\sin\phi & \sin\theta\sin\psi\cos\phi - \cos\psi\sin\phi & \cos\theta\cos\phi \end{pmatrix}. \tag{2.5}$$

This method of computing the rotation matrix is from a computational perspective not ideal since there are many cosine and sine operations which requires more computational power than other operations. A better way to calculate the rotation matrix is by using quaternions, which will be discussed later.

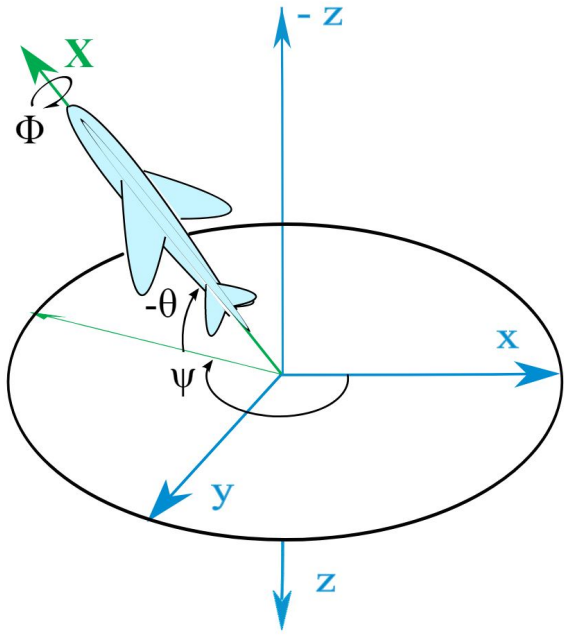


Figure 2.3: The Euler angles following the aerospace sequence. The blue system is the ENU frame where the x-axis is north, [9].

2.2.4 Geodetic Coordinates

In Global Positioning System (GPS) navigation it is standard to use the geodetic coordinates, longitude (λ), latitude (Φ) and height (h) as information about the position. The World Geodetic System 1984 (WGS84) is the earth fixed frame that is used in all geospatial intelligence, [10]. The parameters that define this ellipsoid model of the earth is given in Table 2.1. The GPS height, h,

Description	**Parameter**	**Value**
Major radius	a	6378137 [m]
Minor radius	b	6356752.3142 [m]
Flattening	f	1/298.257223563
First eccentricity	e_1	$\sqrt{1-\frac{b^2}{a^2}}$
Second eccentricity	e_2	$\sqrt{\frac{a^2}{b^2}-1}$
IERS reference meridian	-	102.5 [m] east of the Greenwich Prime

Table 2.1: Parameters defining the WGS84 ellipsoid [11].

is the perpendicular distance above this reference ellipsoid and not the height above the sea level which is more commonly used [11].

The transformation from these coordinates to the earth-frame is done, according to [4], with the equations

$$\begin{aligned} x_1^e &= (N+h)\cos(\Phi)\cos(\lambda), \\ x_2^e &= (N+h)\cos(\Phi)\sin(\lambda), \\ x_3^e &= \left(\frac{b^2}{a^2}N + h\right)\sin(\Phi), \end{aligned} \tag{2.6}$$

where

$$N = \frac{a}{\sqrt{1 - e_1^2\sin^2(\Phi)}}.$$

The inverse transformation can according to [11] be performed using the closed form

$$\begin{aligned} \Phi &= \arctan\left(\frac{x_3^e + e_2^2 b\sin^3(\theta)}{p - e_1^2 a\cos^3(\theta)}\right), \\ \lambda &= \arctan\left(\frac{x_2^e}{x_1^e}\right), \\ h &= \frac{p}{\cos(\Phi)} - N, \end{aligned} \tag{2.7}$$

where N is the same as above and

$$\begin{aligned} p &= \sqrt{(x_1^e)^2 + (x_2^e)^2}, \\ \theta &= \arctan\left(\frac{x_3^e a}{pb}\right). \end{aligned}$$

With the ability to transform between these two sets of coordinates one can define the transformation matrix from the earth-frame to the NED-frame as

$$\boldsymbol{C}_{\boldsymbol{e}}^{\boldsymbol{n}} = \begin{pmatrix} -\sin(\Phi)\cos(\lambda) & -\sin(\Phi)\sin(\lambda) & \cos(\Phi) \\ -\sin(\lambda) & \cos(\lambda) & 0 \\ -\cos(\Phi)\cos(\lambda) & -\cos(\Phi)\sin(\lambda) & -\sin(\Phi) \end{pmatrix}, \tag{2.8}$$

which will be used later when initializing the Kalman filter.

2.2.5 Quaternion

Converting between reference frames can also be done using quaternions. A quaternion can be seen as a vector in R^4

$$\boldsymbol{q} = (q_0, q_1, q_2, q_3), \tag{2.9}$$

with four real numbers. It consists of a scalar part (first number, q_0) and a vector part ($[q_1, q_2, q_3]$). The definition of the quaternion and the mathematical

operations that exist for it (except for product) will not be discussed here but can be found in for example [8]. Each quaternion defined represents a rotation between two frames, for example the body- and earth-frame. In this report the quaternion that represents this specific rotation will be denoted $\boldsymbol{q}_b^e = (q_0, q_1, q_2, q_3)$. A quaternion representing the rotation between the body- and NED-frame will in this report be denoted as $\boldsymbol{s}_b^n = (s_0, s_1, s_2, s_3)$. The rotation matrix (coordinate frame rotation) between the earth- and body-frame can from [6] and [8] be defined as

$$\boldsymbol{C}_e^b = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix} \tag{2.10}$$

which is a robust method of computing the transformation matrix. This rotation is the quaternion rotation operator that rotates the **reference frame** (i.e. keeping the vector constant in space and just rotating the coordinate system). It should be noted that a quaternion rotation operator can also be defined as a point/vector rotation, which can be visualized as rotating the **vector** itself to a new coordinate (i.e. rotating the vector in space). The rotation matrices have different definitions depending on what operation is intended, with them being the transpose of each other [8]. In this report we are only interested in the coordinate frame rotation as our vectors are representing physical properties, which does not depend on the coordinate system they are presented in. As an example, the velocity of the UAV remains the same regardless of the coordinate system it is expressed in.

The result of the quaternion product between two quaternions $\boldsymbol{p}$ and $\boldsymbol{q}$ (notation $\boldsymbol{p} \otimes \boldsymbol{q}$) with scalar parts p_0, q_0 and vector parts $\boldsymbol{p}_v$, $\boldsymbol{q}_v$ is another quaternion, which according to [8] has the scalar part

$$p_0q_0 - \boldsymbol{p}_v \cdot \boldsymbol{q}_v, \tag{2.11}$$

and vector part

$$p_0\boldsymbol{q}_v + q_0\boldsymbol{p}_v + \boldsymbol{p}_v \times \boldsymbol{q}_v. \tag{2.12}$$

This quaternion product will be used later when expressing the equations of motion.

2.2.6 Finding the Quaternion

The quaternion itself is challenging to explicitly define as the elements represent different properties of the rotation. Hence, the quaternion needs to be computed based on some known property of the attitude which is sometimes not known.

By knowing the Euler angles defined by the aerospace sequence, i.e. the attitude of the UAV, the corresponding quaternion can from [6] and [8] be

computed using the equations

$$\begin{aligned} s_0 &= \cos\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2}, \\ s_1 &= \cos\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} - \sin\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2}, \\ s_2 &= \cos\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2}, \\ s_3 &= \sin\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} - \cos\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2}. \end{aligned} \tag{2.13}$$

A second approach to finding the quaternion is to use a known rotation matrix. If given a rotation matrix, $\boldsymbol{C}_e^b$ with elements $[m_{ij}]$, then the corresponding quaternion can according to [8] be calculated with the equations

$$\begin{aligned} q_0 &= \frac{1}{2}\sqrt{1 + m_{11} + m_{22} + m_{33}}, \\ q_1 &= \frac{1}{4q_0}(m_{23} - m_{32}), \\ q_2 &= \frac{1}{4q_0}(m_{31} - m_{13}), \\ q_3 &= \frac{1}{4q_0}(m_{12} - m_{21}). \end{aligned} \tag{2.14}$$

Both methods will be used later in the simulation and implementation of the Kalman filter, depending on the known information.

It is also possible to calculate the corresponding Euler angles, for the rotation specified by the rotation matrix, with the equations

$$\begin{aligned} \psi &= \arctan2(\frac{m_{12}}{m_{11}}), \\ \theta &= \arcsin(-m_{13}), \\ \phi &= \arctan2(\frac{m_{23}}{m_{33}}). \end{aligned} \tag{2.15}$$

This makes it possible to illustrate the attitude with the information from the quaternion, for example in the UAV ground station. Remember that different rotation matrices correspond to different quaternions, so to find the Euler angles defined by the aerospace sequence, then the matrix $\boldsymbol{C}_n^b$ should be used.

2.2.7 Quaternion as State Variable

The quaternion serves as a good state variable to account for the attitude of the UAV. The phenomena of Gimbal lock (see for example [12]), which results in a singularity for the Euler angles does not give a singularity with the use of quaternions. This is an advantage for the use of quaternions. However, the main reason that the quaternion is a good choice is because the derivative of the quaternion can, according to [6], be expressed as a matrix multiplication

$$\dot{\boldsymbol{q}}_b^e = \frac{1}{2}\boldsymbol{A}\boldsymbol{q}_b^e. \tag{2.16}$$

Here, $\boldsymbol{A}$ is the skew- symmetric matrix

$$\boldsymbol{A} = \begin{pmatrix} 0 & -\omega_1 & -\omega_2 & -\omega_3 \\ \omega_1 & 0 & \omega_3 & -\omega_2 \\ \omega_2 & -\omega_3 & 0 & \omega_1 \\ \omega_3 & \omega_2 & -\omega_1 & 0 \end{pmatrix}, \tag{2.17}$$

where $\boldsymbol{\omega}_{eb}^{b} = (\omega_1, \omega_2, \omega_3)^T$ is the vector containing the angular rates of the UAV in relation to the earth-frame, expressed in the body-frame. How these angular rates are obtained will be presented later.

Chapter 3

Governing Equations

The navigation equations that govern the motion of the UAV will be presented in this chapter. Two representations of these equations will be given as they are both needed in the XKF implementation. The measurements with the disturbances in terms of noise and bias will be included in the equations of motion to give a more accurate model. The chapter ends with a description of the measurement model that relates the sensor measurements to the states in the UAV model.

For the Kalman filter application, there is no need to have a model of the UAV that consider the aerodynamics, but rather a pure kinematic model. As will be seen, the reason for this is that the measurements from the accelerometer and gyro will contain all the forces acting on the UAV and there is no reason to distinguish between for example aerodynamic and engine forces.

3.1 Equations of Motion

The equations of motion describe how the velocity $\boldsymbol{v}$, position $\boldsymbol{p}$ and quaternion $\boldsymbol{q}$ change with time. The equations are conveniently expressed in the earth-frame of reference. The motivation is that there are fewer nonlinearities in this frame of reference if quaternions are used to describe the attitude compared to the other frames. For example, if the equations of motion are expressed in the NED-frame with the Euler angles, there are many cosine and sine operators, see [4]. The specific force (acceleration) $\boldsymbol{f}^b$ and angular rates $\boldsymbol{\omega}_{ib}^b$, measured by the accelerometer and gyro respectively, are considered as input signals in the equations. All forces acting on the UAV are included in the vector $\boldsymbol{f}^b$ which is why we are not interested in modeling for example the aerodynamics separately. The rotation of earth, $\boldsymbol{\omega}_{ie}^e = (0, 0, \omega_e)^T$, and gravity vector $\boldsymbol{g}^e$ are used in the model. The gravity model will be presented later.

A compact representation of the equations of motion is given as

$$\begin{aligned}
\dot{\boldsymbol{p}}^e &= \boldsymbol{v}^e, \\
\dot{\boldsymbol{v}}^e &= -2\boldsymbol{S}(\boldsymbol{\omega}_{ie}^e)\boldsymbol{v}^e + \boldsymbol{f}^e + \boldsymbol{g}^e(\boldsymbol{p}^e), \\
\dot{\boldsymbol{q}}_b^e &= \frac{1}{2}\boldsymbol{q}_b^e \otimes \bar{\boldsymbol{\omega}}_{ib}^b - \frac{1}{2}\bar{\boldsymbol{\omega}}_{ie}^e \otimes \boldsymbol{q}_b^e,
\end{aligned} \tag{3.1}$$

where $\boldsymbol{S}$ is the skew- symmetric matrix

$$\boldsymbol{S}(\boldsymbol{x}) = \begin{pmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{pmatrix}. \tag{3.2}$$

Remember here that the notation $\otimes$ is the quaternion product defined earlier. The bar subscript on the terms used in the quaternion product in (3.1) means that it is a quaternion with real part zero and vector part as the vector with the subscript, i.e. $\bar{\boldsymbol{x}} = [0; \boldsymbol{x}]$.

3.1.1 State Space Form

It is useful to write the equations of motion on an explicit state space form as they will be easier to linearize. The state space representation from [13], with p_1^e, p_2^e and p_3^e representing the position in the earth-frame, is

$$\begin{aligned}
\dot{v}_1^e &= 2\omega_e v_2^e + \omega_e^2 x_1^e + f_1^e + g_1^e, \\
\dot{v}_2^e &= -2\omega_e v_1^e + \omega_e^2 x_2^e + f_2^e + g_2^e, \\
\dot{v}_3^e &= f_3^e + g_3^e, \\
\dot{p}_1^e &= v_1^e, \\
\dot{p}_2^e &= v_2^e, \\
\dot{p}_3^e &= v_3^e, \\
\dot{\boldsymbol{q}}_b^e &= \frac{1}{2}\boldsymbol{A}\boldsymbol{q}_b^e,
\end{aligned} \tag{3.3}$$

where the last equation is the same as (2.16). The state vector is defined as, $\boldsymbol{x} = (v_1^e, v_2^e, v_3^e, p_1^e, p_2^e, p_3^e, q_1, q_2, q_3, q_4)^T$ so that these equations can be expressed as

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}). \tag{3.4}$$

The vector $\boldsymbol{u} = (f_1^b, f_2^b, f_3^b, \omega_1, \omega_2, \omega_3)$ is the input vector containing measurements from the accelerometer and gyro. Note that the gyro terms are modified to compensate for bias and the rotation of earth, which will be discussed later in this chapter. Also, it is important to remember that the specific force terms are given from

$$\boldsymbol{f}^e = \boldsymbol{C}_b^e \boldsymbol{f}^b, \tag{3.5}$$

which introduces nonlinearities in the model.

3.1.2 Gravity Model

The gravity terms (acceleration due to gravity) used in the equations of motion are defined by

$$
\begin{aligned}
g_1^e &= -\frac{GMp_1^e}{((p_1^e)^2 + (p_2^e)^2 + (p_e^e)^2)^{3/2}}, \\
g_2^e &= -\frac{GMp_2^e}{((p_1^e)^2 + (p_2^e)^2 + (p_e^e)^2)^{3/2}}, \\
g_3^e &= -\frac{GMp_3^e}{((p_1^e)^2 + (p_2^e)^2 + (p_e^e)^2)^{3/2}},
\end{aligned} \tag{3.6}
$$

where G is the gravitational constant and M is the mass of the earth. Since the change in position is very small between each sampling instance, the derivatives of these terms will be neglected i.e. considered to be zero. The properties of the earth used in these equations are presented in Table 3.1.

Description	**Parameter**	**Value**
Gravitational constant	G	$6.67408 \cdot 10^{-11}$ [$\mathrm{m}^3/(\mathrm{kg} \cdot \mathrm{s}^2)$]
Mass of earth	M	$5.972 \cdot 10^{24}$ [kg]
Rotation of earth	ω_e	$7.292115 \cdot 10^{-}5$[rad/s]

Table 3.1: Properties of the earth.

3.2 Measurements

The states can be measured with the use of different sensors. Multiple options are available when it comes to estimating the position and attitude. A common combination of sensors is to have a global navigation satellite system (GNSS) together with an inertial measurement unit (IMU). The GNSS sensor is typically a GPS receiver, which is commonly used in many applications today. The IMU consists of an accelerometer and a gyro. The accelerometer measures the specific forces (accelerations) and the gyro measures the rotation rates acting on the body. In addition to theses sensors, it is also possible to use a magnetometer to gain information regarding the attitude of the body.

More advanced options, not discussed further in this report, can be to implement a downward facing camera in order to measure the body velocity through optical flow. One example of where this type of sensor is used can be found in [14].

3.2.1 Noise and Bias

Noise is, as discussed earlier, always present in any measurement. There are also process noise, which in this application can be for example gusts i.e. distur-

bances in the accelerometer, gyro and magnetometer measurements. However, in this case the process noise will also include the measurement noise in these sensors.

All noise will be considered as random variables added to the measured quantity. They are assumed to be Gaussian white noise with standard deviations determined experimentally by letting the sensor gather measurements undisturbed during a long period of time. This has already been done by Nystromer Avionics for the sensors used and the result will be presented later. Process noise in terms of gusts and other outer disturbances is not modeled as it is very difficult to get good estimations of.

Bias is a small, almost static, error that exist for all sensor. It can vary slowly with for example temperature but does not have the same variations as the noise. Most biases are of interest to estimate as it affects the accuracy of the result. As a first step the gyro bias noise will be estimated, meaning that three new states are introduced as $\boldsymbol{b}_g^b = (b_{g1}^b, b_{g2}^b, b_{g3}^b)^T$ in the state vector. The final state vector is hence given as

$$\boldsymbol{x} = (v_1^e, v_2^e, v_3^e, p_1^e, p_2^e, p_3^e, q_1, q_2, q_3, q_4, b_{g1}^b, b_{g2}^b, b_{g3}^b)^T. \tag{3.7}$$

There exist many models for how the bias changes with time, for example as a random constant or random harmonic [15]. Since the bias varies very slowly, it will be modeled to be constant in time with the exception of a small random deviation as

$$\dot{\boldsymbol{b}}_g = 0 + \boldsymbol{w}_{b,g}^b, \tag{3.8}$$

where $\boldsymbol{w}_{b,g}$ can be considered to be process noise for the gyro bias. This type of model is often referred to as a random walk, [15]. The bias is added to the gyro terms as

$$\boldsymbol{\omega}_{ib}^b = \boldsymbol{\omega}_{IMU}^b - \boldsymbol{b}_g^b, \tag{3.9}$$

where $\boldsymbol{\omega}_{IMU}^b$ is the actual measurement vector from the gyro. This is the angular rotation with the inertial frame as reference so these parameters need to be corrected with the rotation of the earth according to [4] so that

$$\boldsymbol{\omega}_{eb}^b = \boldsymbol{\omega}_{ib}^b - \boldsymbol{C}_e^b \boldsymbol{\omega}_{ie}^e. \tag{3.10}$$

These gyro terms in $\boldsymbol{\omega}_{eb}^b$ are the ones to be used in (2.16).

3.2.2 Measurement Model

With the sensors used, the resulting measurement vector is given as $\boldsymbol{y} = (v_1^e, v_2^e, v_3^e, p_1^e, p_2^e, p_3^e, m_1^b, m_2^b, m_3^b)^T$, where the velocity and position is from the GNSS and the magnetic field is from the magnetometer. The general measurement model can be written as

$$\boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}), \tag{3.11}$$

which for the first six measurements are simply the values of the first six states. The model used for the magnetometer measurements is the World Magnetic

Model (WMM), [16], which gives the earth magnetic field components for a certain position on earth in the NED-frame, $\boldsymbol{B}^n_{earth} = (B^n_1, B^n_2, B^n_3)$. The measurement model using the states is then

$$\boldsymbol{m}^b = \boldsymbol{C}^b_e(q)\boldsymbol{C}^e_n(\Phi, \lambda)\boldsymbol{B}^n_{earth} + \boldsymbol{v}^b_m, \tag{3.12}$$

where $\boldsymbol{v}^b_m$ is the noise in the measurements. Note that noise is also added to the measurements of the velocity and position in the same way as for the magnetometer.

Chapter 4

The Kalman Filter

This chapter will cover the theory behind the Kalman filter and what distinguishes the different types of filters used. The chapter starts with a description of the basic linear filter followed by the linearized and extended Kalman filter for nonlinear system. Nonlinear estimators will be presented, and it will be described how the nonlinear observer and LKF can be combined into the exogenous Kalman Filter. The chapter ends with a summary of the most important differences between the filters.

4.1 Optimal Kalman Filter

The linear Kalman filter presented in the introduction can be shown to have optimal performance in terms of minimum mean-square estimation error (under some assumptions). The exact derivation of the filter equations, which involves the Riccati equation, and assumptions will not be presented here but can be found in for example [2] and [17].

We consider a general discrete linear system

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_k \boldsymbol{x}_k + \boldsymbol{B}_k \boldsymbol{u}_k + \boldsymbol{w}_k, \tag{4.1}$$

with measurement model

$$\boldsymbol{y}_k = \boldsymbol{C}_k \boldsymbol{x}_k + \boldsymbol{D}_k \boldsymbol{u}_k + \boldsymbol{v}_k, \tag{4.2}$$

where $\boldsymbol{w}$ is the process noise and $\boldsymbol{v}$ is measurement noise, both assumed to be Gaussian white noise. The gain which then minimizes the mean-square estimation error is according to [2]

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{C}_k^T (\boldsymbol{C}_k \boldsymbol{P}_k^- \boldsymbol{C}_k^T + \boldsymbol{R})^{-1}, \tag{4.3}$$

where $\boldsymbol{P}_k$ is the estimation error covariance matrix and $\boldsymbol{R}$ is the measurement noise covariance matrix. The $\boldsymbol{P}_k$ matrix can be seen as a measure of how accurate the estimates are compared to the real system. For practical use the $\boldsymbol{P}_k$ matrix is computed in two steps. First a prior estimate denoted $\boldsymbol{P}_k^-$, which

is used in the Kalman gain followed by a corrected matrix denoted without the subscript. The a priori estimate is calculated from the previous estimate

$$\boldsymbol{P}_{k+1}^{-} = \boldsymbol{A}_k \boldsymbol{P}_k \boldsymbol{A}_k^T + \boldsymbol{Q}, \tag{4.4}$$

where $\boldsymbol{Q}$ is the process noise covariance matrix. After using this estimate to calculate the Kalman gain, an updated estimate is calculated from

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{C}_k)\boldsymbol{P}_k^{-}. \tag{4.5}$$

The estimation of the states, $\hat{\boldsymbol{x}}$, is also computed in two steps. The prior estimate is solely from the model itself i.e.

$$\hat{\boldsymbol{x}}_{k+1}^{-} = \boldsymbol{A}_k \hat{\boldsymbol{x}}_k + \boldsymbol{B}_k \boldsymbol{u}_k. \tag{4.6}$$

This estimation is then corrected by using the measurements and Kalman gain

$$\hat{\boldsymbol{x}}_{k+1} = \hat{\boldsymbol{x}}_{k+1}^{-} + \boldsymbol{K}_{k+1}(\boldsymbol{y}_{k+1} - \boldsymbol{C}_{k+1}\hat{\boldsymbol{x}}_{k+1}^{-}), \tag{4.7}$$

to achieve the optimal estimation.

To conceptually understand how the filter works, one can see the a priori estimate as how the model predicts where the system will be in the next time step. The difference $\boldsymbol{y}_{k+1} - \boldsymbol{C}_{k+1}\hat{\boldsymbol{x}}_{k+1}^{-}$ is how well the measurements match the measurement model, i.e. how much the real measurements differ from the predicted measurements. This is an indication for how good the model has predicted the real states. For example, if this difference is zero it means that the measurements equals the predicted measurements so then the a priori estimate is assumed to be the best estimate of the real system. The Kalman gain is simply weighting the impact from the model compared to the measurements. If the gain is zero it means that only the model is trusted and as the gain is increased it can be seen as more trust being put on the measurements.

4.2 Linearized Kalman Filter

In most cases the dynamics of a system is not linear so that it can be written on the form (4.1) and hence the equations for the optimal filter is not feasible. However, by linearizing the system it is possible to use the same equations for the linear filter, but optimality is no longer guaranteed. Instead sub-optimality is achieved [3]. The choice of linearization point is an important problem. If a reference trajectory $\boldsymbol{x}_0(t)$ is used, then the nonlinear process and measurement equations

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{w}, \tag{4.8}$$

$$\boldsymbol{y} = h(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{v}, \tag{4.9}$$

can be linearized with a first order Taylor approximation

$$\dot{\boldsymbol{x}} \approx f(\boldsymbol{x}_0(t), \boldsymbol{u}) + \boldsymbol{F}(\boldsymbol{x} - \boldsymbol{x}_0(t)) + \boldsymbol{w}, \tag{4.10}$$

$$\boldsymbol{y} \approx h(\boldsymbol{x}_0(t), \boldsymbol{u}) + \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}_0(t)) + \boldsymbol{v}. \tag{4.11}$$

Here

$$\boldsymbol{F} = \frac{\partial f}{\partial x_i}(\boldsymbol{x}_0(t), \boldsymbol{u}),$$

is the Jacobi matrix of the system dynamics with respect to the states, evaluated at the linearization point and

$$\boldsymbol{H} = \frac{\partial h}{\partial x_i}(\boldsymbol{x}_0(t), \boldsymbol{u}),$$

is the Jacobi matrix of the measurement model with respect to the states, evaluated at the linearization point. The equation for the Kalman gain is then almost the same as for the optimal KF

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}_k^T (\boldsymbol{H}_k \boldsymbol{P}_k^- \boldsymbol{H}_k^T + \boldsymbol{V}\boldsymbol{R}\boldsymbol{V}^T)^{-1}, \tag{4.12}$$

where

$$\boldsymbol{V}_k = \frac{\partial h}{\partial v_i}(\boldsymbol{x}_0(t), \boldsymbol{u}),$$

is the Jacobi matrix for the measurement model with respect to noise evaluated at the linearization point. The meaning of a reference trajectory is that it is assumed to be known where the system is headed or in what region it operates. For example, this could be valid in the process industry where the estimated quantity follows a known pattern.

The state estimation covariance matrix will be slightly different compared to the optimal KF since it is chosen to be on the Joseph form. The Joseph form increases the numerical stability of the estimations according to [4]. The prediction step of the matrix is given as

$$\boldsymbol{P}_k^- = (\boldsymbol{I} + T\boldsymbol{F}_k)\boldsymbol{P}_{k-1}(\boldsymbol{I} + T\boldsymbol{F}_k)^T + T^2 \boldsymbol{W}_k \boldsymbol{Q} \boldsymbol{W}_k^T, \tag{4.13}$$

and the correction step is

$$\boldsymbol{P}_k = (\boldsymbol{I} + \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_k^-(\boldsymbol{I} + \boldsymbol{K}_k\boldsymbol{H}_k)^T + \boldsymbol{K}_k\boldsymbol{R}\boldsymbol{K}_k^T. \tag{4.14}$$

Here

$$\boldsymbol{W}_k = \frac{\partial f}{\partial w_i}(\boldsymbol{x}_0(t), \boldsymbol{u}),$$

is the Jacobi matrix of the system dynamics with respect to process noise evaluated at the linearization point. The constant T is the sampling time for the discrete equations. The algorithm for the LKF is the same as for the linear filter with a prediction and correction step. The linearized system equations

$$\dot{\hat{\boldsymbol{x}}} = f(\boldsymbol{x}_0(t), \boldsymbol{u}) + \boldsymbol{F}(\boldsymbol{x} - \boldsymbol{x}_0(t)), \tag{4.15}$$

is integrated to give the prediction estimate, $\hat{\boldsymbol{x}}_{k+1}^-$. The correction step is then applied as

$$\hat{\boldsymbol{x}}_{k+1} = \hat{\boldsymbol{x}}_{k+1}^- + \boldsymbol{K}_{k+1}(\boldsymbol{y}_{k+1} - h(\boldsymbol{x}_{0,k+1}, \boldsymbol{u}_{k+1}) - \boldsymbol{H}(\hat{\boldsymbol{x}}_{k+1}^- - \boldsymbol{x}_{0,k+1})), \tag{4.16}$$

and the LKF estimation is complete.

4.3 Extended Kalman Filter

For most systems, including the UAV, it is not feasible to use a reference trajectory (as it is not known how the system will behave) so the approach is then to linearize around the last known state estimate. If the system is linearized around the last estimate at each time instant, then the filter is called Extended Kalman Filter (EKF) [17].

For systems with nonlinear dynamics, the EKF is a very common type of filter used in the industry today. However, the weakness of the EKF is that if the initial guess of the state vector is far off the real system or if the filter for some reason would lose information about the state vector, it is possible for the filter to become unstable. This is due to the fact that if the linearization point deviates too much from the real values, then the linearization error is fed back into the estimation since the EKF relies on a feedback loop. This feedback loop is illustrated in Figure 4.1.

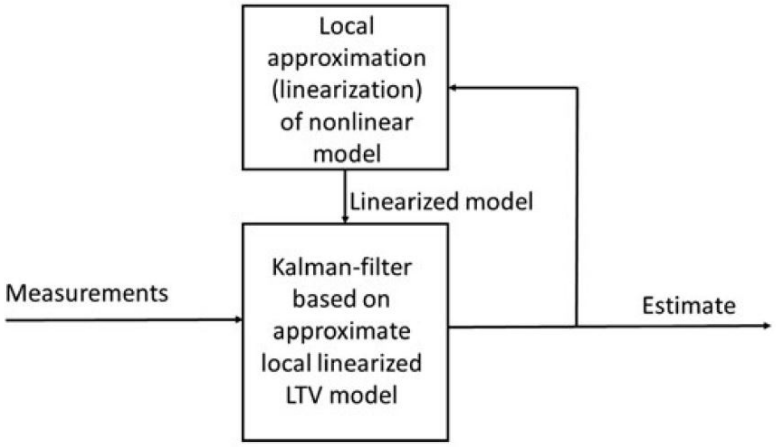


Figure 4.1: Schematic of the EKF, [3].

The algorithm for the discrete time EKF is similar to the one for the optimal KF and LKF with a prior estimate followed by a correction step. The prediction step considers both a prediction for the states and the state estimate covariance matrix in the next time step. The equations for the state estimation covariance matrix and Kalman gain are the same as for the LKF. An important difference in the equations of the a priori estimate is that the nonlinear dynamics are used so that the prediction estimate, $\boldsymbol{x}_{k+1}^{-}$, is calculated by integrating

$$\dot{\hat{\boldsymbol{x}}} = f(\boldsymbol{x}_k(t), \boldsymbol{u}). \tag{4.17}$$

This is because the second term of (4.15) is zero since the point of evaluation is the same as the linearization point. This is also applied to the correction step and (4.16) so that the correction is

$$\hat{\boldsymbol{x}}_{k+1} = \hat{\boldsymbol{x}}_{k+1}^{-} + \boldsymbol{K}_{k+1}(\boldsymbol{y}_{k+1} - h(\boldsymbol{x}_{k+1}^{-}, \boldsymbol{u}_{k+1})). \tag{4.18}$$

4.4 Nonlinear Observers

A NonLinear Observer (NLO) is an alternative to the KFs when dealing with nonlinear dynamics or measurement models. These are not as general as a KF and is designed based on the system it is applied to. The advantage with an NLO is that it can be designed using nonlinear stability theory, [18], so that it achieves global asymptotic or exponential stability, [3]. This is possible since no linearization is required. Also, there is no need for assumptions regarding the characteristics of noise and bias. The computational footprint of the NLO is generally smaller than the EKF [19] making it a more efficient observer, which could be important for many applications. Drawbacks with these observers are that they can be challenging to discretize and to include the effects of different sampling times for different sensors, [18]. It is also generally not possible to design or evaluate the observer from a minimum variance perspective, [3] so the accuracy of the NLO is typically worse compared to the KFs.

4.5 eXogenous Kalman Filter

An observer that combines an NLO and KF is the eXogenous Kalman Filter (XKF) presented recently in [3]. The main idea with this filter is to use an exogenous signal as linearization point for an LKF. It is similar to the EKF, but with the important difference that the estimation from the KF is not used as linearization point in a feedback loop. The XKF is instead a two-stage filter consisting of an NLO and an LKF. An NLO can as stated earlier be designed to be globally stable which is a desirable property for an observer, but the performance is generally not considered. The LKF is on the other hand sub-optimal but can have stability problems due to the linearization.

The idea with the XKF is then to utilize the advantages of both of these types of observers while avoiding their drawbacks. The globally stable estimation from the NLO is used as linearization point in the LKF. It can then be shown according to [3] that the stability of the NLO is inherited to the final estimation which is made more accurate by the LKF. A drawback that can be considered with the XKF is that essentially two separate observers are needed to provide one estimation. This increases the computational effort needed compared to for example the EKF. The cascade of the XKF can be seen in Figure 4.2 where it can be seen more clearly how the estimation from the NLO is used in the LKF.

The algorithm for the XKF is similar to both of the previous ones presented but with some slight differences. The nonlinear system and measurement model considered is on the same form as (4.8) and (4.9). If we first assume that we have an estimation from the NLO, $\tilde{\boldsymbol{x}}$, and only consider the LKF part, then the continuous time XKF is

$$\dot{\hat{\boldsymbol{x}}} = f(\tilde{\boldsymbol{x}}, \boldsymbol{u}) + \boldsymbol{F}(\tilde{\boldsymbol{x}}, \boldsymbol{u})(\hat{\boldsymbol{x}} - \tilde{\boldsymbol{x}}) + \boldsymbol{K}(\boldsymbol{y} - h(\tilde{\boldsymbol{x}}, \boldsymbol{u}) - \boldsymbol{H}(\tilde{\boldsymbol{x}}, \boldsymbol{u})(\hat{\boldsymbol{x}} - \tilde{\boldsymbol{x}})). \quad (4.19)$$

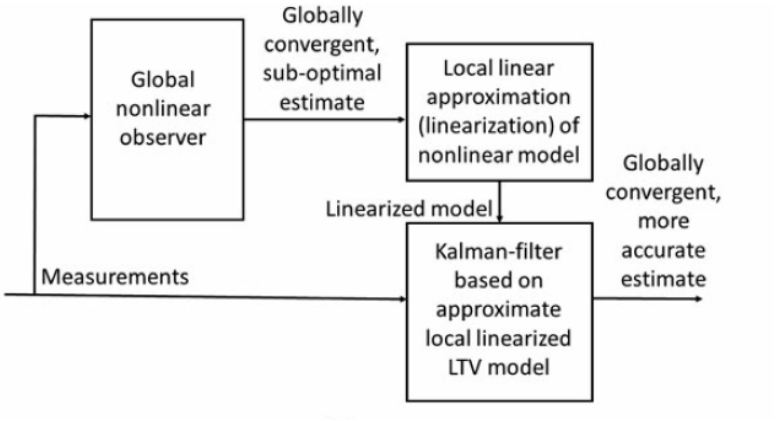


Figure 4.2: Schematic of the XKF, [3].

Here

$$\boldsymbol{F}(\tilde{\boldsymbol{x}}, \boldsymbol{u}) = \frac{\partial f}{\partial x_i}(\tilde{\boldsymbol{x}}, \boldsymbol{u}),$$

is the Jacobian of the system dynamics,

$$\boldsymbol{H}(\tilde{\boldsymbol{x}}, \boldsymbol{u}) = \frac{\partial h}{\partial x_i}(\tilde{\boldsymbol{x}}, \boldsymbol{u}),$$

is the Jacobian of the measurement model and $\boldsymbol{K}$ is the Kalman gain. Note that the evaluation points for the Jacobi matrices are the estimation from the NLO. If we break this equation down, we can see that the first two terms on the right hand side is the first order Taylor approximation of the nonlinear system. This can be seen as the prediction step. Next, we have the correction step where we now use the linearized measurement model (first order Taylor approximation). Note also that the NLO estimation acts as linearization point in the Taylor approximations.

The prediction step for the discrete time version is

$$\dot{\hat{\boldsymbol{x}}} = f(\tilde{\boldsymbol{x}}_k, \boldsymbol{u}_k) + \boldsymbol{F}(\tilde{\boldsymbol{x}}_k, \boldsymbol{u}_k)(\hat{\boldsymbol{x}}_k - \tilde{\boldsymbol{x}}_k), \tag{4.20}$$

which is integrated to give the a priori estimation $\hat{\boldsymbol{x}}_{k+1}^-$. The state estimation covariance matrix and the Kalman gain are calculated in the same way as for the LKF but with the NLO estimation as evaluation point in the Jacobi matrices. The discrete time correction step for the XKF is then

$$\hat{\boldsymbol{x}}_{k+1} = \hat{\boldsymbol{x}}_{k+1}^- + \boldsymbol{K}_{k+1}(\boldsymbol{y}_{k+1} - h(\tilde{\boldsymbol{x}}_{k+1}, \boldsymbol{u}_{k+1}) - \boldsymbol{H}_{k+1}(\tilde{\boldsymbol{x}}_{k+1}, \boldsymbol{u}_{k+1})(\hat{\boldsymbol{x}}_{k+1} - \tilde{\boldsymbol{x}}_{k+1})). \tag{4.21}$$

4.6 Summary of Differences Between the Filters

The choice of linearization point is the main difference between the filters that are suitable for nonlinear dynamics. The strategy for the linearization for each observer is

- For the LKF it is a predefined trajectory from knowledge about the system.
- The EKF uses its own previous estimate which is also the root to its stability problems.
- The XKF uses an exogenous estimate from a globally stable NLO as linearization point in an LKF so that the stability properties are inherited from the NLO.

None of the KFs are optimal in the same way as for the linear case due to the linearization but they can all be considered to be sub-optimal. The NLO uses no linearization, which gives good stability properties, but it is often not possible to design this observer from a performance perspective leading to a less accurate estimation.

Chapter 5

Practical Aspects

This chapter will consider the practical aspects of the filter implementation. The NLO for UAV applications will be presented and discretized so that it can be implemented in the Stratos Pilot system. The algorithm for the whole XKF implementation with the filter initialization and loop will be presented. Also, a wind estimator will be suggested to be added to the XKF in order to improve the control performance of the UAV.

5.1 Nonlinear Observer for UAV Application

The NLO that serves as an intermediate observer, exist to ensure that the XKF is globally stable. The performance of the observer is of minor interest as the LKF will improve the estimation from the NLO. A proposed NLO for UAV navigation given in [19] and [18] will be used in this report to provide the intermediate estimation for the state variables. This NLO itself can be seen as two separate observers, one attitude observer and one translational-motion observer (TMO). The equations of motion used for the NLO is on the form (3.1). The NLO works in a similar fashion as for the KF or any other type of observer in the sense that a model is used to predict the states which are then corrected with measurements. Difference here is that no linearization is made which is also why it can be shown to be globally stable. The cascade of the NLO can be seen in Figure 5.1.

The attitude observer in [19] provides an estimation of the quaternion. It requires two vector measurements in the body-frame and two reference vectors in the earth-frame. The measurements used are the specific force from the accelerometer and magnetic field measurements from the magnetometer, both in the body-frame. The magnetic field in the earth-frame is assumed to be known but the accelerations in the earth-frame is unknown. The solution is to estimate that vector as $\hat{\boldsymbol{f}}^e$ and use this vector as reference. The reason that these vectors are necessary is because the correction will be done with a so called injection term defined as

$$\hat{\boldsymbol{\sigma}} = k_1 \boldsymbol{v}_1^b \times \boldsymbol{C}_e^b \boldsymbol{v}_1^e + k_2 \boldsymbol{v}_2^b \times \boldsymbol{C}_e^b \boldsymbol{v}_2^e, \tag{5.1}$$

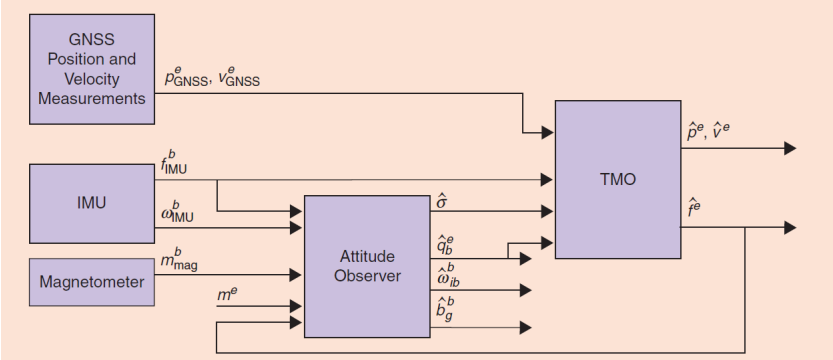


Figure 5.1: Schematic of the NLO, [18].

where $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are the two vector measurements with corresponding reference vectors. It is further motivated in [18] that these vectors can be chosen as

$$\begin{aligned}
\boldsymbol{v}_1^b &= \frac{\boldsymbol{f}_{IMU}^b}{\|\boldsymbol{f}_{IMU}^b\|_2}, \\
\boldsymbol{v}_1^e &= \frac{\hat{\boldsymbol{f}}^e}{\|\hat{\boldsymbol{f}}^e\|_2}, \\
\boldsymbol{v}_2^b &= \frac{\boldsymbol{f}_{IMU}^b}{\|\boldsymbol{f}_{IMU}^b\|_2} \times \frac{\boldsymbol{m}_{mag}^b}{\|\boldsymbol{m}_{mag}^b\|_2}, \\
\boldsymbol{v}_2^e &= \frac{\hat{\boldsymbol{f}}^e}{\|\hat{\boldsymbol{f}}^e\|_2} \times \frac{\boldsymbol{m}^e}{\|\boldsymbol{m}^e\|_2},
\end{aligned} \tag{5.2}$$

which can increase the accuracy of the attitude estimation as well as giving the gains k_1 and k_2 the properties of being seen as cutoff frequencies for a complementary filter. The two gains k_1 and k_2 are chosen arbitrarily so that the observer is stable. It can be seen that if the measurements match the reference vectors perfectly, then the injection term is zero and no correction will be applied to the estimations. A numerical issue can occur if the norm of the specific force vector is equal to zero, which could happen in free fall. It must be programmed so that if the norm is equal to zero, then the vector should not be normalized and instead used as it is. The same can be applied to the magnetometer for safety reasons, but it is unlikely that the norm of that vector is zero.

The attitude observer presented in [19] has the form

$$\begin{aligned}
\dot{\hat{\boldsymbol{q}}}_b^e &= \frac{1}{2}\hat{\boldsymbol{q}}_b^e \otimes (\bar{\boldsymbol{\omega}}_{ib}^b - \bar{\hat{\boldsymbol{b}}}_g^b + \bar{\hat{\boldsymbol{\sigma}}}) - \frac{1}{2}\bar{\boldsymbol{\omega}}_{ie}^e \otimes \hat{\boldsymbol{q}}_b^e, \\
\dot{\hat{\boldsymbol{b}}}_g^b &= -k_I \hat{\boldsymbol{\sigma}},
\end{aligned} \tag{5.3}$$

where the injection term is used to correct both the quaternion and bias.

The TMO is according to [19] given as

$$\begin{aligned}
\dot{\hat{\boldsymbol{p}}}^e &= \hat{\boldsymbol{v}}^e + \theta \boldsymbol{K}_{pp}(\boldsymbol{p}^e_{GNSS} - \hat{\boldsymbol{p}}^e) + \boldsymbol{K}_{vp}(\boldsymbol{v}^e_{GNSS} - \hat{\boldsymbol{v}}^e), \\
\dot{\hat{\boldsymbol{v}}}^e &= -2\boldsymbol{S}(\boldsymbol{\omega}^e_{\boldsymbol{ie}})\hat{\boldsymbol{v}}^e + \hat{\boldsymbol{f}}^e + \boldsymbol{g}^e(\hat{\boldsymbol{p}}^e) + \theta^2 \boldsymbol{K}_{pv}(\boldsymbol{p}^e_{GNSS} - \hat{\boldsymbol{p}}^e) + \\
&\quad + \theta \boldsymbol{K}_{vv}(\boldsymbol{v}^e_{GNSS} - \hat{\boldsymbol{v}}^e), \\
\dot{\boldsymbol{\xi}} &= \boldsymbol{C}^e_b \boldsymbol{S}(\hat{\sigma}) \boldsymbol{f}^b_{IMU} + \theta^3 \boldsymbol{K}_{p\xi}(\boldsymbol{p}^e_{GNSS} - \hat{\boldsymbol{p}}^e) + \theta^2 \boldsymbol{K}_{v\xi}(\boldsymbol{v}^e_{GNSS} - \hat{\boldsymbol{v}}^e), \\
\hat{\boldsymbol{f}}^e &= \boldsymbol{C}^e_b \boldsymbol{f}^b_{IMU} + \boldsymbol{\xi},
\end{aligned} \tag{5.4}$$

where θ and all $\boldsymbol{K}$ matrices are gains chosen to ensure stability. A proof of stability is conducted in [19] where it is shown, using Lyapunov functions, that stability is achieved if the tuning parameters are chosen "sufficiently high". It is clearly then not trivial to make this choice, which is why they in [18] suggest other options for the gain matrices for implementation purposes, which will be discussed later. Also, one probably notices an extra state introduced, which is the vector $\boldsymbol{\xi}$. The idea behind the introduction of an extra state is to create an extended system as motivated in [20] to estimate some of the inputs to the system. In this case, the specific force is estimated using this auxiliary state. Its physical interpretation is in [18] that it couples the rotational and translational motions.

The complete NLO is then the combination of the attitude observer and the TMO. The equations presented are in continuous time but need to be discretized for implementation.

5.1.1 Discretization of the Attitude Observer

The discretization of (5.3) is done in [18] and the resulting equations are

$$\begin{aligned}
\hat{\boldsymbol{q}}^e_b[k+1] &= e^{\left(\frac{T}{2}\boldsymbol{\Omega}(\hat{\boldsymbol{\omega}}[k])\right)} e^{\left(-\frac{T}{2}\bar{\boldsymbol{\Omega}}(\boldsymbol{\omega}^e_{ie})\right)} \hat{\boldsymbol{q}}^e_b[k], \\
\hat{\boldsymbol{b}}^b_g[k+1] &= \hat{\boldsymbol{b}}^b_g[k] - Tk_I[k]\hat{\boldsymbol{\sigma}}.
\end{aligned} \tag{5.5}$$

Starting with the quaternion, the matrices needed to perform this calculation are

$$\begin{aligned}
e^{\left(\frac{T}{2}\boldsymbol{\Omega}(\hat{\boldsymbol{\omega}}[k])\right)} &= \cos\left(\frac{T}{2}\|\hat{\boldsymbol{\omega}}\|_2\right)\boldsymbol{I}_4 + \frac{T}{2}\operatorname{sinc}\left(\frac{T}{2}\|\hat{\boldsymbol{\omega}}\|_2\right)\boldsymbol{\Omega}(\hat{\boldsymbol{\omega}}), \\
e^{\left(-\frac{T}{2}\bar{\boldsymbol{\Omega}}(\boldsymbol{\omega}^e_{ie})\right)} &= \left(\cos\left(\frac{T}{2}\|\boldsymbol{\omega}^e_{ie}\|_2\right)\boldsymbol{I}_4 + \frac{T}{2}\operatorname{sinc}\left(\frac{T}{2}\|\boldsymbol{\omega}^e_{ie}\|_2\right)\bar{\boldsymbol{\Omega}}(\boldsymbol{\omega}^e_{ie})\right)^{-1},
\end{aligned} \tag{5.6}$$

where

$$\hat{\boldsymbol{\omega}} = \boldsymbol{\omega}^b_{IMU} - \hat{\boldsymbol{b}}^b_g + \hat{\boldsymbol{\sigma}}, \tag{5.7}$$

is a corrected angular rotation that takes bias and the injection term into account. The final two matrices used in the discretization are given as

$$\boldsymbol{\Omega}(\hat{\boldsymbol{\omega}}) = \begin{pmatrix} 0 & -\hat{\boldsymbol{\omega}}^T \\ \hat{\boldsymbol{\omega}} & -\boldsymbol{S}(\hat{\boldsymbol{\omega}}) \end{pmatrix}, \tag{5.8}$$

$$\bar{\boldsymbol{\Omega}}(\boldsymbol{\omega}_{ie}^{e}) = \begin{pmatrix} 0 & -(\boldsymbol{\omega}_{ie}^{e})^{T} \\ \boldsymbol{\omega}_{ie}^{e} & \boldsymbol{S}(\boldsymbol{\omega}_{ie}^{e}) \end{pmatrix}. \tag{5.9}$$

The matrices denoted as $\boldsymbol{S}(\boldsymbol{\omega}_{ie}^{e})$ and $\boldsymbol{S}(\hat{\boldsymbol{\omega}})$ are the skew symmetric matrices of the vectors $\boldsymbol{\omega}_{ie}^{e}$ and $\hat{\boldsymbol{\omega}}$.

Sensors deliver measurements at different frequencies. The consequence of this is for example that the magnetometer measurements are not available at all time steps. In order to account for this, the injection term will use two time constants T_{mag} and T_{acc}. These time constants represent the sampling time of each sensor. The definition of the injection term is then modified so that

$$\hat{\boldsymbol{\sigma}} = \frac{T_{acc}}{T} k_1 \boldsymbol{v}_1^{b} \times \boldsymbol{C}_{e}^{b} \boldsymbol{v}_1^{e} + \frac{T_{mag}}{T} k_2 \boldsymbol{v}_2^{b} \times \boldsymbol{C}_{e}^{b} \boldsymbol{v}_2^{e}, \tag{5.10}$$

which ensures that the gains k_1 and k_2 are the only factors affecting the vector measurements.

The variable T is again the general sampling time and k_I is a gain which is used in the correction of the bias estimation. It is assumed that the sampling time is small enough so that the angular rotations, $\hat{\boldsymbol{\omega}}$, can be considered to be constant between sampling instances [18].

The gyro bias estimation in (5.5) has a simple interpretation. If there exist a deviation between the measurements and the estimated vectors, then the injection term will gain a nonzero value. Depending on the sampling time and gain chosen, a change in the bias will then be conducted. The advantage with this method is that the performance can be easily controlled with the gain k_I. However, the drawback is that the bias estimation will be updated regardless of the cause of the error. This could force the bias estimation away from the real value instead of converging towards it if for example there are large errors in some other sensor.

5.1.2 Discretization of the TMO

The practical implementation of the TMO is very similar to a Kalman filter. It consists of a correction step and a prediction step. Focusing on the prediction step here, the equations in (5.4) need to be discretized. The first thing to note is that these equations are linear and hence the TMO can be written on a simple matrix form

$$\begin{aligned} \hat{\boldsymbol{x}}_{n}^{*} &= \boldsymbol{A}^{*} \hat{\boldsymbol{x}}^{*} + \boldsymbol{B}^{*} \boldsymbol{u}^{*}, \\ \boldsymbol{y}^{*} &= \boldsymbol{C}^{*} \hat{\boldsymbol{x}}_{n}^{*} + \boldsymbol{D}^{*}. \end{aligned} \tag{5.11}$$

The matrices for the continuous system, denoted with a star, will not be presented explicitly here but can be found in [18]. It should be noted that the state vector, $\hat{\boldsymbol{x}}_{n}^{*} = (\hat{\boldsymbol{p}}^{e}, \hat{\boldsymbol{v}}^{e}, \boldsymbol{\xi})^{T}$, is defined differently compared to the LKF and that the TMO has an input vector defined as $\boldsymbol{u}^{*} = (\boldsymbol{f}_{IMU}^{b}, -\boldsymbol{S}(\hat{\boldsymbol{\sigma}})\boldsymbol{f}_{IMU}^{b})^{T}$ [18].

The linear equations are then discretized so that the prediction step is

$$\hat{\boldsymbol{x}}^{-}[k+1] = \boldsymbol{A}_d \hat{\boldsymbol{x}}^{+}[k] + \boldsymbol{B}_d[k] \boldsymbol{u}^{*}[k] + \boldsymbol{D}_d, \tag{5.12}$$

where

$$\boldsymbol{A}_d = \begin{pmatrix} \boldsymbol{I}_3 & T\boldsymbol{I}_3 & \frac{T^2}{2}\boldsymbol{I}_3 \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_3 & T\boldsymbol{I}_3 \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{I}_3 \end{pmatrix}, \tag{5.13}$$

$$\boldsymbol{B}_d[k] = \begin{pmatrix} \frac{T^2}{2}\boldsymbol{C}_b^e[k] & \frac{T^3}{6}\boldsymbol{C}_b^e[k] \\ T\boldsymbol{C}_b^e[k] & \frac{T^2}{2}\boldsymbol{C}_b^e[k] \\ \boldsymbol{0}_{3\times3} & T\boldsymbol{C}_b^e[k] \end{pmatrix}, \tag{5.14}$$

$$\boldsymbol{D}_d[k] = \begin{pmatrix} \frac{T^2}{2}(\boldsymbol{g}^e(\hat{\boldsymbol{p}}[k]) - 2\boldsymbol{S}(\boldsymbol{\omega}_{ie}^e)\boldsymbol{v}^e[k]) \\ T(\boldsymbol{g}^e(\hat{\boldsymbol{p}}[k]) - 2\boldsymbol{S}(\boldsymbol{\omega}_{ie}^e)\boldsymbol{v}^e[k]) \\ \boldsymbol{0}_{3\times3} \end{pmatrix}. \tag{5.15}$$

It can be noted that $\boldsymbol{A}_d$ is constant while the other two changes each iteration. In the discretization it is assumed that the specific force vector, rotation matrix and gravity vector are constant between sampling instances [18].

5.1.3 Gain Selection

There are multiple gains to be chosen for the NLO, particularly in the TMO (5.4). The choices of these parameters are crucial for the stability to be guaranteed. However, this choice is not trivial and it is hence argued in [18] that it is more practical to set

$$\theta = 1,$$

and then choosing the remaining gains in a minimum- variance sense, namely as the Kalman gain. By introducing a state vector for the NLO as before, $\boldsymbol{x}_n = (\hat{\boldsymbol{p}}^e, \hat{\boldsymbol{v}}^e, \boldsymbol{\xi})$, the correction step for the TMO can be performed in a similar sense as for a regular Kalman filter. The equation for this is

$$\hat{\boldsymbol{x}}_n^+[k] = \hat{\boldsymbol{x}}_n^-[k] + \boldsymbol{K}_d[k](\boldsymbol{y}[k] - \boldsymbol{C}\hat{\boldsymbol{x}}_n^-[k]), \tag{5.16}$$

where

$$\boldsymbol{K}_d[k] = \boldsymbol{P}_n^-[k]\boldsymbol{C}^T(\boldsymbol{C}\boldsymbol{P}_n^-[k]\boldsymbol{C}^T + \boldsymbol{R}_d)^{-1} \tag{5.17}$$

Here, $\boldsymbol{R}_d$ is the measurement noise covariance matrix for the measurement vector $\boldsymbol{y}[k] = (\hat{\boldsymbol{p}}_{GNSS}^e, \hat{\boldsymbol{v}}_{GNSS}^e)^T$. The matrix $\boldsymbol{C}$ is given as

$$\boldsymbol{C} = \begin{pmatrix} \boldsymbol{I}_3 & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_3 & \boldsymbol{0}_{3\times3} \end{pmatrix}, \tag{5.18}$$

which is the measurement model matrix for the linear system.

The state estimation covariance matrix, $\boldsymbol{P}_n$, for the NLO is needed to calculate the gain and is updated each iteration with

$$\boldsymbol{P}_n^-[k+1] = \boldsymbol{A}_d\boldsymbol{P}_n^+[k]\boldsymbol{A}_d^T + \boldsymbol{B}_d[k]\boldsymbol{Q}_d\boldsymbol{B}_d[k]^T, \tag{5.19}$$

where $\boldsymbol{Q}_d$ is the process noise covariance matrix for the input vector $\boldsymbol{u}^*$. The matrix is prior to this corrected just as for an optimal Kalman filter with the equation

$$\boldsymbol{P}_n^+[k] = (\boldsymbol{I}_9 - \boldsymbol{K}_d[k]\boldsymbol{C})\boldsymbol{P}_n^-[k]. \tag{5.20}$$

Symmetry is then enforced on the state estimation covariance matrix by calculating

$$\boldsymbol{P}_n^-[k+1] = \frac{1}{2}(\boldsymbol{P}_n^-[k+1] + (\boldsymbol{P}_n^-)^T[k+1]). \tag{5.21}$$

The remaining gains that have to be chosen are then k_1, k_2 and k_I. These are chosen arbitrarily so that desired performance is achieved.

5.2 Linearized Kalman Filter

The equations presented in chapter 4.2 can be used without modifications but they require linearized equations of motion. The nonlinear equations of motion presented earlier must be linearized in order to apply the Kalman filter. The state space form (3.3) is most suitable for linearization. The core of the linearization of the equations are the Jacobi matrices $\boldsymbol{F}$ and $\boldsymbol{H}$ so that the linearized equations become

$$\dot{\boldsymbol{x}} \approx f(\boldsymbol{x}_0(t)) + \boldsymbol{F}(\boldsymbol{x} - \boldsymbol{x}_0(t)) + \boldsymbol{w}, \tag{5.22}$$

$$\boldsymbol{y} \approx h(\boldsymbol{x}_0(t)) + \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}_0(t)) + \boldsymbol{v}. \tag{5.23}$$

In addition, the Jacobi matrices $\boldsymbol{W}$ and $\boldsymbol{V}$ are required in the Kalman Filter equations. All Jacobi matrices are gained by partially deriving the equations of motion and the measurement model. The expressions of the Jacobi matrices are given explicitly in appendix B.

The a priori estimate for the LKF is obtained by integrating the linearized equations of motion with the fourth order Runge-Kutta method, see for example [21]. The correction of the predicted state vector is done as described in section 4.2.

5.3 Algorithm

The algorithm for the filter implementation can be designed using the equations presented in this chapter, so that all calculations are done in a correct order. The full algorithm, with all steps and more details, is given in appendix A.

5.3.1 Initializing the Filter

Before the filter loop can be started, the filter needs to be initialized. The main objective with this part of the system is to define constant matrices and get initial values for the states variables. This process is already implemented in the Stratos system (see [22]), but extra steps for the NLO are added in this implementation. In addition to the normal state vector introduced for the LKF, the NLO requires an intermediate state vector that is used in the prediction step and to calculate the Jacobi matrices for the LKF.

A few steps are made in order to find the initial state vector. The first step is to take measurements from all sensors and populate the state vector with

those (where appropriate). The position and speed are taken directly from the GPS, while the quaternion requires more steps.

The first step in finding the quaternion is to calculate the initial Euler angles. These must be obtained when the acceleration measured by the accelerometer is only the gravity, i.e. when the UAV is standing still on the ground or flying with constant speed. The accelerometer and magnetometer measurement vectors, $\boldsymbol{f^b}$ and $\boldsymbol{m^b}$, are normalized and the Euler angles in the aerospace sequence can then, according to [23], be calculated as

$$\begin{aligned} \theta &= -\arcsin(f_1^b), \\ \phi &= \arcsin(f_2^b/\cos\theta), \\ \psi &= \arccos([m_1^b + \sin\theta\sin\alpha]/[\cos\theta\cos\alpha]), \end{aligned} \tag{5.24}$$

where α is the magnetic field inclination angle given by the WMM, [16].

The quaternion, $\boldsymbol{s}_b^n$, is then calculated with (2.13) so that the rotation matrix $\boldsymbol{C}_n^b$ can be calculated as

$$\boldsymbol{C}_n^b = \begin{pmatrix} s_0^2+s_1^2-s_2^2-s_3^2 & 2(s_1s_2+s_0s_3) & 2(s_1s_3-s_0s_2) \\ 2(s_1s_2-s_0s_3) & s_0^2-s_1^2+s_2^2-s_3^2 & 2(s_2s_3+s_0s_1) \\ 2(s_1s_3+s_0s_2 & 2(s_2s_3-s_0s_1) & s_0^2-s_1^2-s_2^2+s_3^2 \end{pmatrix}. \tag{5.25}$$

With the knowledge from the GPS about the position, the rotation matrix $\boldsymbol{C}_e^n$ can be calculated with (2.8). The next step is to calculate the rotation matrix between the earth-frame and body-frame with the equation

$$\boldsymbol{C}_e^b = \boldsymbol{C}_n^b \boldsymbol{C}_e^n. \tag{5.26}$$

With (2.14), the initial $\boldsymbol{q}_b^e$ quaternion can be calculated using the rotation matrix above.

The initial value for the auxiliary state, $\boldsymbol{\xi}$, for the NLO is defined to be zero according to [18]. Once the state vectors are populated and the constant matrices are defined, the filter is ready to go into the loop.

5.3.2 Kalman Filter Loop

The Kalman filter loop is divided into six sections, each containing multiple operations. The TMO and attitude observer of the NLO are separated and follow the schematic in Figure 5.1. In practical implementation, the operation of each of these observers overlap and hence they both need to be divided into two parts. The TMO follow the same principle as the Kalman filter, with a correction step and a prediction step. The attitude observer does not follow the same principle but is divided when the information from the TMO is needed. The details of this algorithm can be found in [18].

Since the LKF uses the estimation from the NLO as linearization point, it is important to understand when each of these estimations are used and in what order the operations are performed. This order is visualized in Figure 5.2. The time step begins by performing the correction step of the NLO estimate

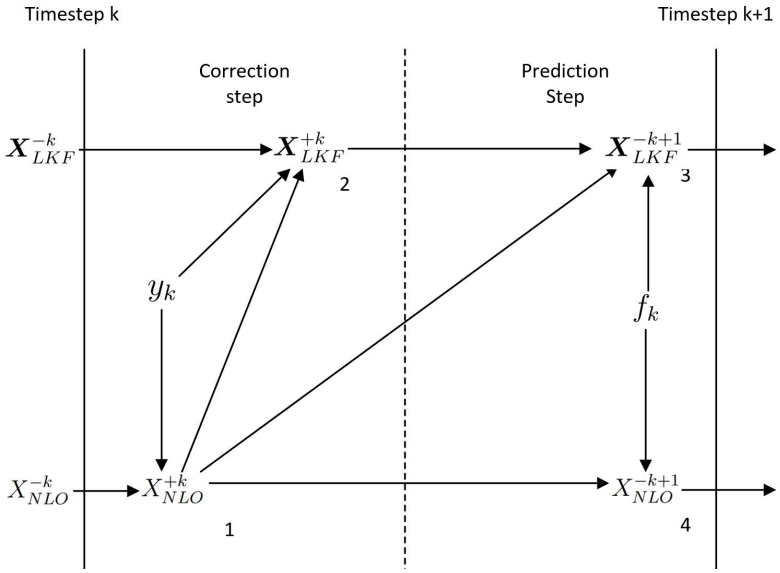


Figure 5.2: Computational order in the Kalman filter loop.

followed by a correction in the LKF. The time step is ended with the prediction of the next time step for both observers. The corrected NLO estimation is used as linearization point when correcting the LKF estimation and also in the prediction step. Since this corrected estimate should be slightly more accurate, this makes the linearization errors smaller. Since the Kalman filter is a loop, it is also possible to define the start of the time step to be when the prediction occurs. This choice depends on the system and when the measurements are available. The same result can be obtained regardless of when the start of the time step is defined.

5.3.3 Managing the Measurements

Depending on the sensors used, the frequency of which measurements are gathered may vary. This is an important issue in the implementation of the filter. In the NLO this is handled through the time constants defined earlier and it is simpler since data from each sensor is used separately meaning that if data from one sensor is not present, that step can be bypassed. However, for the LKF it is required that all measurements are available at the same time in the correction step. This means that before a correction can be performed, all measurements must be gathered. This may result in that it is not possible to do a correction at all time steps. In the implementation of the filter there needs to be a logical variable that is true when all measurements are gathered which allows for the correction to be made. If this variable is false, then the Kalman gain is set to zero. In the simulation this can easily be tested by for example, letting the Kalman gain be calculated at every third time step and otherwise set it to zero. Most sensors gather data with high frequencies, so

the correction should in most cases be performed at a sufficiently high rate resulting in good performance.

5.4 Wind Estimation

In order to make optimal flight paths it is of interest to have updated knowledge about the wind velocity (both magnitude and direction). One option is to implement a separate observer rather than including the wind in the XKF. This is possible since the states estimated in the XKF do not depend on the wind velocity. The velocity estimated in the XKF is the velocity relative to the ground and the equations of motion does not depend explicitly on the wind (the wind affects the accelerations and rotations of the UAV). An observer presented in [24] estimates the wind by using only an airspeed sensor together with the standard sensors GNSS, IMU and Magnetometer. It is also possible to estimate the angle-of-attack and sideslip angle from the wind velocity estimation giving more useful information to the control system. In this observer, the GNSS measurements are used directly in the wind observer. However, since the XKF is already implemented it should be possible to use the more accurate estimations of the velocity from the XKF instead. The proposed cascade is seen in Figure 5.3 where the XKF provides the ground velocity to the wind velocity observer. One advantage with this observer is that there is no need for aircraft data or more advanced models than already presented in this report. The observer uses a standard structure with a prediction and correction step meaning that the Kalman theories can be applied.

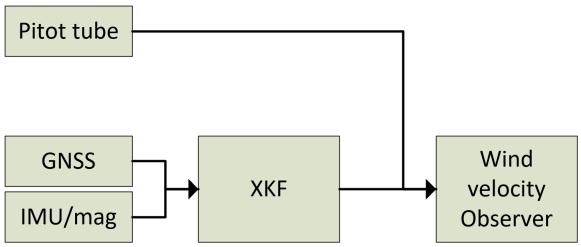


Figure 5.3: The cascade of the wind observer.

We start by defining the notations for the velocity of the UAV relative to the ground and air as $\boldsymbol{v}_g^b = (u, v, w)^T$ and $\boldsymbol{v}_r^b = (u_r, v_r, w_r)^T$ respectively. The wind velocity is easiest represented in the NED frame and denoted $\boldsymbol{v}_w^n = (u_w, v_w, w_w)^T$. It is important to remember that the velocity estimated by the XKF is the velocity relative to the ground. The pitot tube gives measurements of the pressure difference between the total and static pressure, i.e. the dynamic pressure, which is assumed to be proportional to the square of the airspeed (velocity relative to the air) by a factor K_p. As the pitot tube is mounted in the longitudinal degree of freedom in the body fixed frame of reference, the

measured velocity component is

$$(v_p)^2 = K_p \Delta P. \tag{5.27}$$

Here ΔP is the dynamic pressure from the pitot tube and v_p is the airspeed related to the pitot tube measurement. When flying, the UAV usually has an angle-of-attack, α, and sideslip angle, β. These angles are not known unless an angle-of-attack and sideslip angle indicator is installed on the UAV. The measured longitudinal velocity component relative to the wind is then given as

$$(u_r)^2 = \left(\frac{v_p}{\cos(\alpha)\cos(\beta)} \right)^2 = \frac{K_p \Delta P}{(\cos(\alpha)\cos(\beta))^2}. \tag{5.28}$$

It is suggested in this report to introduce the artificial measurement $u_r^m = \sqrt{\Delta P}$ and scale factor $\gamma = \sqrt{K_p}/(\cos(\alpha)\cos(\beta))$ so that the velocity component can be expressed on the form presented in [24],

$$u_r = \gamma u_r^m. \tag{5.29}$$

As stated in [24], the state vector for the wind velocity observer can now be expressed as $\boldsymbol{x}_w = ((\boldsymbol{v}_w^n)^T, \gamma)^T$ where the scale factor will be estimated together with the wind velocity vector. It is assumed in [24] that the wind is steady and that the scale factor is slowly time-varying so that the model

$$\begin{aligned} \dot{\boldsymbol{v}}_w^n &= \boldsymbol{0}_{3\times 1}, \\ \dot{\gamma} &= 0, \end{aligned} \tag{5.30}$$

is valid. This means that there will be no change in the states in the prediction step but only in the correction step of the observer.

With the estimation of the velocity relative to the ground from the XKF it is possible to create a measurement model since the ground velocity is the sum of the velocity relative to the air and the wind velocity. The longitudinal component of the ground velocity is then modeled as

$$u^m = \boldsymbol{d}_1^T \boldsymbol{C}_n^b \boldsymbol{v}_w^n + \gamma u_r^m, \tag{5.31}$$

where $\boldsymbol{d}_1 = (1, 0, 0)^T$. We can see that the measurement model is linear with the artificial sensor data and defined states meaning that it is possible to design a linear Kalman filter for optimal estimation. By defining the measurement model matrix as

$$\boldsymbol{C}_w = (\boldsymbol{d}_1^T \boldsymbol{C}_n^b, u_r^m), \tag{5.32}$$

and system matrix $\boldsymbol{A} = \boldsymbol{0}$, the observer can be written as

$$(\boldsymbol{x}_w)_{k+1} = \boldsymbol{K}_w(u - u^m) = \boldsymbol{K}_w(u - \boldsymbol{C}_w(\boldsymbol{x}_w)_k), \tag{5.33}$$

where u is the estimation of the longitudinal velocity component relative to the ground from the XKF. The Kalman gain is calculated in a standard way as

$$\boldsymbol{K}_w = \boldsymbol{P}_w \boldsymbol{C}_w^T R_w, \tag{5.34}$$

where $\boldsymbol{P}_w$ is the state estimation covariance matrix for the state vector $\boldsymbol{x}_w$ and R_w is the noise covariance of the pitot tube. The state estimation covariance matrix cannot be calculated in the same way as presented previously since $\boldsymbol{A} = \boldsymbol{0}$. In order to achieve a time-varying gain, the continuous equation

$$\dot{\boldsymbol{P}}_w = \boldsymbol{Q}_w - \boldsymbol{P}_w \boldsymbol{C}_w^T \boldsymbol{R}_w^{-1} \boldsymbol{C}_w \boldsymbol{P}_w, \tag{5.35}$$

is solved numerically with Euler forward method. The matrix $\boldsymbol{Q}_w$ is the covariance matrix for the process noise in the state model.

An observability analysis was conducted in [24] which showed that the wind velocity is only observable if the UAV is performing some maneuver that changes the attitude. This result means that the UAV must perform some maneuver in order to update the wind estimation and that the observation becomes unreliable if the UAV is flying straight for a long period. The limitation due to the observability could be removed by including a sensor that measures the angle-of-attack and sideslip angle, but this has not been investigated further.

With the wind velocity estimation it is then possible to calculate the angle-of-attack and sideslip angle with the equations

$$\begin{aligned} \alpha &= \arctan(w_r/u_r), \\ \beta &= \arcsin(v_r/V_a), \end{aligned} \tag{5.36}$$

where $V_a = \sqrt{u_r^2 + v_r^2 + w_r^2}$ is the magnitude of the velocity relative to the air.

Chapter 6

Stratos Pilot System

The Stratos Pilot System is a guidance and control system for autonomous or remotely controlled land,sea and air vehicles. The system is developed by Nystromer Avionics AB, which is a Swedish company located in the Stockholm region. This chapter will present the basic structure of the system. Also, the sensors that are used in the system will be described and its noise properties will be discussed at the end of this chapter.

6.1 Stratos Pilot Core Loop

The Stratos Pilot system is programmed in the language C and uses no operating system. Instead it relies on a synchronous "core loop" triggered by one CPU interrupt. This core loop is divided into 80 cases, each taking a maximum of 250 μs meaning the cases are changed with a frequency of 4 kHz. The entire loop then takes 0.02 s and the frequency is 50 Hz. Every operation performed within the control system is done in one of these 80 cases and they are divided so that the total time for one case does not exceed the limit. The first and last cases are reserved for managing the loop and to write to memory meaning that the XKF is implemented in the middle of the loop. Figure 6.1 shows the overview structure of the loop and examples of what the cases in the beginning and end of the loop do. At the end of the loop sequence (the last 16 cases), the output signals to the servos are generated. In addition to the XKF, the control loops and all operations that are necessary for the entire system to function is done in these cases.

The sensors used gather data at different frequencies. If the sensor gather data at a frequency which is higher than the frequency of the loop, then the strategy in the Stratos system is that this sensor is averaging its own measurements until all other sensors have provided their data. Once all the measurements are available, then a logical variable allows for the Kalman gain to be calculated.

A major difference between the implementation in the language C compared to the simulations in MATLAB is that each operation must be broken down into smaller pieces. For example, in MATLAB it is possible to calculate

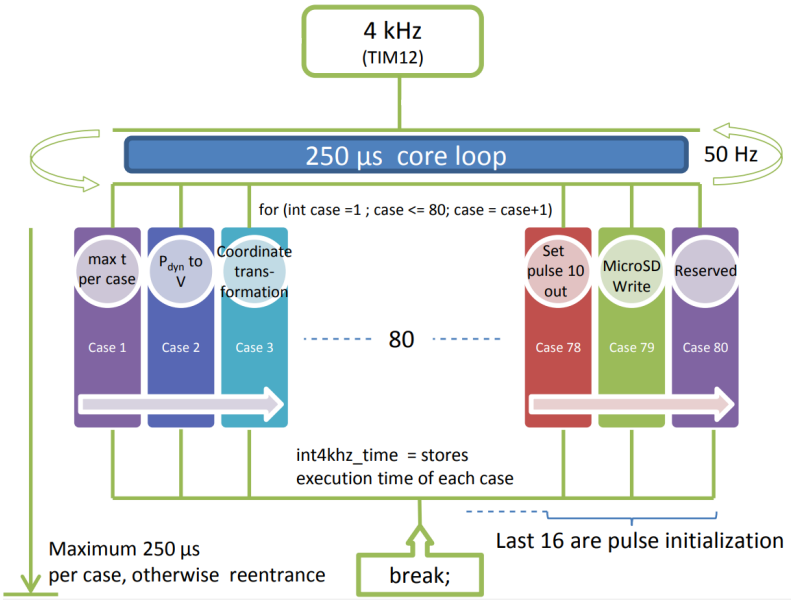


Figure 6.1: Structure overview of the Stratos Pilot core loop.

the Kalman gain in one operation with all matrix multiplication. In C it is only possible to do one matrix multiplication at a time which leads to more lines of code. The equations themselves remain the same but it requires more code to perform the same calculations.

6.2 Sensors

The sensors used in the Stratos Pilot system is currently an accelerometer, a gyro and a magnetometer. Also, a pitot tube is installed which provides airspeed measurements and a pressure sensor for the static pressure. The product name and properties of each sensor can be seen in Table 6.1. It should be noted that the accuracy for the pressure sensors are given as a percentage of the Full Scale Span (FSS) i.e. the difference between the output voltage at maximum and minimum pressure (within range).

6.2.1 Covariance Matrices

The covariance matrices used in the Kalman filter are assumed to be constant in time, meaning that the noise variances does not change with time. The values of the variances is estimated by letting the sensors gather data while remaining stationary. The variation in the data is hence caused by the noise and the variances can be calculated. Nystromer Avionics has performed this experiment and the result is presented in appendix C. The experiment was performed for 15 hours for all sensors. Both of the matrices, $\boldsymbol{R}$ and $\boldsymbol{Q}$ are diagonal meaning that it is assumed to be no cross-correlation between the

Sensor	**Product name**	**Range (typical)**	**Accuracy/Noise Density**
Single axis gyro (yaw)	LY330ALH	±300 dps	0.014 dps/$\sqrt{}$Hz
Dual axis gyro (pitch and roll)	LPR430AL	±1200 dps	0.018 dps/$\sqrt{}$Hz
Accelerometer	LIS344ALH	±6 g	50 µg/$\sqrt{}$Hz
Pitot tube	MP3V5004G	3.92 kPa	±2.5 %V_{FSS}
Static pressure sensor	MP3H6115A	115 kPa	±1.5 %V_{FSS}
Magnetometer	LSM303D	±12 gauss	5 mgauss/RMS
GPS receiver	NEO-7	N.A.	±2.5 m

Table 6.1: Properties of the sensors used (from the data sheet for each sensor)

sensor noise.

For the NLO, the covariance matrices $\boldsymbol{R}_d$ and $\boldsymbol{Q}_d$ are defined differently compared to the Kalman filter. This is because of the different definitions of the state and measurement vector. The measurement noise covariance matrix, $\boldsymbol{R}_d$ is almost the same as $\boldsymbol{R}$ but without the magnetometer data and with the elements corresponding to the velocity being swapped with the ones for position.

The process noise covariance matrix, $\boldsymbol{Q}_d$, is defined as

$$\boldsymbol{Q}_d = \text{blockdiag}(\boldsymbol{S}_f, \boldsymbol{S}_{\hat{\sigma}f}). \tag{6.1}$$

The matrices $\boldsymbol{S}_f$ and $\boldsymbol{S}_{\hat{\sigma}f}$ are the covariance matrices associated with the accelerometer and auxiliary state noise. It is stated in [18] that the last matrix is not trivial to determine and hence it is argued that stability is guaranteed if this matrix is always chosen to be larger than zero. The values for the accelerometer are the same as for $\boldsymbol{Q}$ and can be found in appendix C.

Due to lack of information regarding the pitot tube in the Stratos system, the same values for the covariance matrices used in [24] is used in the simulations here as well.

Chapter 7

Simulations

Simulations are a convenient tool to use in order to evaluate and compare the filters as parameters can be easily changed and the initial guesses for the states can be chosen arbitrarily. It is the only situation where the filters can be compared to a known trajectory to see how well they actually perform. Simulations make it possible to modify the filter in any way and safely see the result. In this chapter the basic idea behind the simulation model will be presented as well as the results from the analysis performed with the simulations. The filter will be tuned using the simulations and some modifications to the filter will be tested.

7.1 Simulation Environment

The simulation model used to evaluate the performance of the filter has been created in MATLAB. The first part is to generate some realistic flight data for a typical UAV. This is done by solving the equations of motion with artificial (known) input data for a specific maneuver. With the "real" flight trajectory and attitude of the UAV it is then possible to apply the filters and let them estimate this trajectory based on corrupted measurement data. By adding noise and bias to the real input data they serve as accelerometer and gyro measurements. The real flight data (for example position) is then used to create measurements for the GNSS and magnetometer by adding noise to the corresponding values. The Kalman filters are then simulated to estimate the real flight data using the noisy measurements.

The XKF is implemented in the same way as presented earlier in this report and an EKF is also implemented in a standard fashion for comparison. The main difference between the real implementation in the control system and the simulation is that the measurements are artificially made and that the initial states are defined arbitrarily (compared to being estimated at startup).

The trajectory generated for the simulation will remain the same throughout the entire analysis. It is visualized in Figure 7.1 and contains multiple maneuvers of different types, for example climb and turn. The total simulation time is 85 seconds. For simplicity each maneuver is isolated and performed

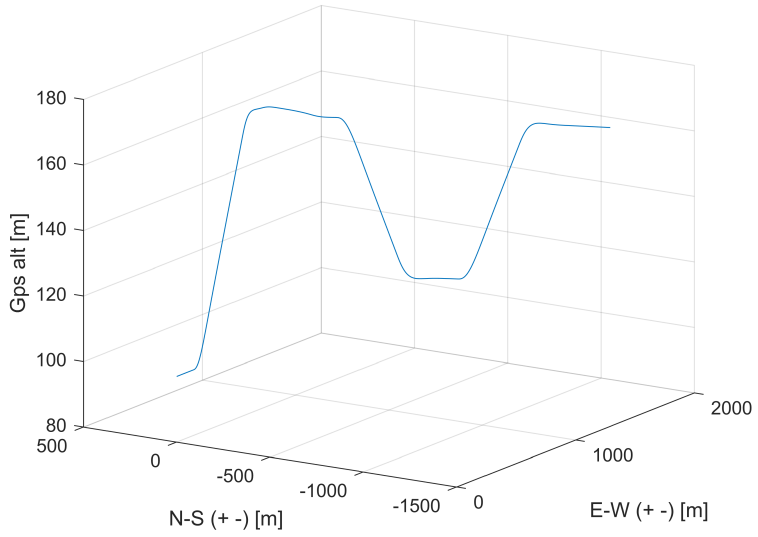


Figure 7.1: Reference trajectory of the simulated UAV.

separately in order to make the generation of input data easier. As this report aims to only consider the state estimation problem rather than control of the UAV, the trajectory only serves as reference for the filters and can be chosen arbitrarily. It should be noted that both the position and attitude of the UAV is simulated so that each maneuver is performed with the corresponding attitude with smooth transitions. For example a climb maneuver is performed at an elevation angle of 10 degrees.

7.1.1 Generation of Input Data

Input data is generated in the body fixed frame of reference. The simplest of maneuvers is to fly straight which requires no angular rotations but only a force that is equal to the gravity (lift force). By knowing the position of the UAV, the gravity vector in the earth-frame $\boldsymbol{g}^e$ can be considered to be known through the gravitational model. The specific force vector in the body-frame (measured by the accelerometer) is then

$$\boldsymbol{f}^b = -\boldsymbol{C}_e^b \boldsymbol{g}^e. \tag{7.1}$$

Note that due to the definition of the body fixed frame, negative specific force means upward force (if UAV is not inverted). The forces acting on the UAV while climbing or descending are the same as flying straight since no acceleration is made, hence (7.1) gives the specific force vector for these maneuvers as well.

When the UAV also experiences angular rotations, for example while turning, then the specific force vector is given as

$$\boldsymbol{f}^b = \boldsymbol{\omega}_{ib}^b \times \boldsymbol{v}^b - \boldsymbol{C}_e^b \boldsymbol{g}^e. \tag{7.2}$$

The method of finding realistic input data for a flight trajectory with smooth transition between maneuvers is to implement a very basic PD- controller in the simulation model. In order to make the transition between flying straight and climbing smooth, the first derivative of the angular rotation needs to be decided based on the current elevation angle and the rate of change in elevation angle. With the current elevation angle denoted as θ_0, the desired elevation angle denoted as θ_{des} and pitch rate denoted as $\omega_{ib_2}^b$ (often denoted q in aerospace literature), the basic PD- controller can be written as

$$\dot{\omega}_{ib_2}^b = K_p(\theta_{des} - \theta_0) - K_d\omega_{ib_2}^b, \tag{7.3}$$

where K_p and K_d are parameters that can be tuned to give realistic performance. The pitch rate can now be considered as a state variable and integrated together with the equations of motion to give a smooth transition into the climb. The forces acting on the UAV while performing this maneuver is calculated with (7.2).

A turn is performed in a similar way but instead with a desired bank angle or radius of the turn. A turn is more complicated since we can have two rotations simultaneously. These are the turn rate and roll rate. If we again assume that we want to go from flying straight but now into a right turn, then the roll rate $\omega_{ib_1}^b$ (often denoted p in aerospace literature) is calculated with

$$\dot{\omega}_{ib_1}^b = K_p(\phi_{des} - \phi_0) - K_d\omega_{ib_1}^b, \tag{7.4}$$

where ϕ_{des} is the desired bank angle and ϕ_0 is the current bank angle. This is the same as for going into a climb. However, in addition to this, the yaw rate (often denoted r in aerospace literature) is also changing with time depending on the bank angle. By knowing that the lift force should counteract the gravity and also provide the centripetal acceleration of the turn it is possible to calculate the turn radius as

$$R = \frac{\|\boldsymbol{v}^b\|^2}{\|\boldsymbol{g}^e\| \tan\phi_0}, \tag{7.5}$$

and then the turn rate, $\omega_{ib_3}^n$, as

$$\omega_{ib_3}^n = \frac{\|\boldsymbol{v}^b\|}{R} = \frac{\|\boldsymbol{g}^e\| \tan\phi_0}{\|\boldsymbol{v}^b\|}. \tag{7.6}$$

Here it is assumed that it is a clean turn, i.e. only velocity in the nose direction and zero elevation angle. The turn rate is then given in the NED-frame and has to be transformed to the body-frame with the rotation matrix $\boldsymbol{C}_n^b$. The total specific force vector is then again calculated with (7.2) and the roll rate is integrated with the equations of motion to give a smooth turn.

For simplicity it is assumed that the UAV always perform one maneuver at the time and that each maneuver is separated by a short time of flying straight. This is of course not true in all flying conditions but for the purpose of the generated input data, this method is considered to be sufficient.

7.1.2 Generating Measurements

Once the reference trajectory is made, it is necessary to add noise and bias to the relevant variables so that they can serve as measurements to the XKF. In the standard case, all noise is Gaussian white noise with variances as describes in chapter 6.2 and appendix C. This is an ideal case where the "real" variances perfectly match the estimations in the covariance matrices and is unlikely the case in reality. It was hence tested to both increase and decrease the values in the covariance matrices by a factor 100 while keeping the "real" variances constant. There was no significant change in the estimations between the sets of covariance matrices meaning that the choice of parameters does not have a large impact. In general, if the $\boldsymbol{R}$ and $\boldsymbol{Q}$ matrices are chosen with large values, then it is assumed that the measurements contain a lot of noise, so the correction will not affect the estimations as much. This may result in slow convergence if initial errors are present. However, low values may result in worse accuracy as more noise is introduced into the estimations. The tuning (estimations) of these variances should be done once the filter is implemented in the real system, as that is the only time when all disturbances are present with their real values.

7.2 Gains

The gains that have to be chosen arbitrarily are k_1, k_2 and k_I in the NLO. A reasonable magnitude of the first two gains were found to be between 0.5-10 so different combinations of these where simulated and the result evaluated. A parameter study was conducted with perfect initial values to ensure convergence. Also, the first 2 seconds of the simulation data was discarded as the filters need time to settle. The root mean square errors (RMSEs) of the altitude and roll estimations from the NLO were calculated for multiple runs and the mean value is presented in Table 7.1. The RMSE is a quantity which indicates the accuracy of the estimation, so any performance difference from the choices of the gains should be seen in these values. The accuracy of the

k_1	k_2	T	Number of runs	RMSE (Altitude)	RMSE (Roll)
0.5	0.5	0.1	20	0.25	0.17
3	3	0.1	20	0.24	0.15
10	10	0.1	20	0.28	0.17
0.5	10	0.1	20	0.26	0.16
10	0.5	0.1	20	0.25	0.17
0.5	3	0.1	20	0.23	0.18
3	0.5	0.1	20	0.24	0.16

Table 7.1: Parameter study for gain selection showing the RMSE for different combinations of gains.

observer does not seem to be largely affected by variation of the gains. The

best combination seems to be $k_1 = 3$ and $k_2 = 3$, which were the values chosen in the other simulation runs. The gain k_I affect the bias estimation. The larger the gain is, the quicker the estimation will converge but at the same time it might cause the estimation to oscillate heavily. After simulating different values, it was found that a value of 0.005 gives reasonable performance. However, the method is very sensitive to other disturbances so suggestions for improvements will be made later.

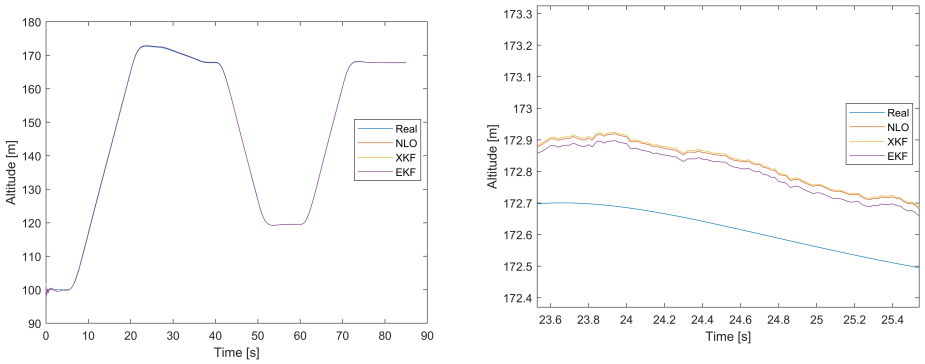
It is discussed in [18] whether a time-varying gain could improve the convergence properties of the estimates. The authors have made simulations where they test this and found that no improvement was made with a time-varying gain. Hence, the gains will remain constant in this implementation.

7.3 Simulation Results

The most important information to find using the simulations are the difference in performance between the filters. Also, the stability of the filters is of interest as that should be the main advantage of the XKF compared to the EKF. All states will not be considered in the comparison as the general performance of the filters remains the same for all states. However, since the NLO has both the attitude observer and the TMO, it is important to compare at least the attitude and one of the states from the TMO as performance might differ there. In this case, that state was chosen to be the altitude.

7.3.1 Perfect Initialization

When the initial guesses of the states are exactly the same as the real values, then all filters will converge and follow the trajectory very closely, which can be seen in Figure 7.2. All three observers experience some oscillations during



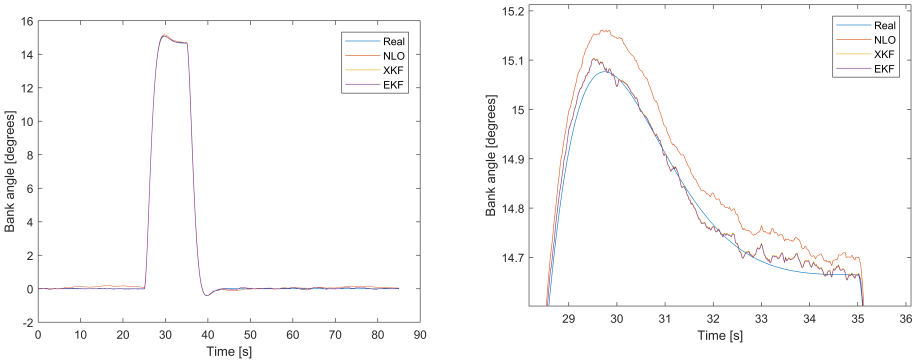
(a) Altitude estimations

(b) Enlarged image of the altitude estimation.

Figure 7.2: Altitude estimation when the initial state guesses perfectly match the real values.

the first couple of seconds. This is due to the fact that the state estimation covariance matrix needs to converge from the initial guess to a better estimation. The reason that the NLO has relatively good estimations is because it uses the Kalman gain in the TMO and hence the performance of all three observers are almost identical. The small differences are due to the linearization and general design of the NLO, but also the choice of linearization points for the EKF and XKF. It is interesting to note that in this case, the XKF has the worst performance in altitude estimation out of the three. This can be explained by considering that the XKF uses a linearization point that is worse than the one for the EKF giving larger linearization errors.

The bank angle (or roll angle) estimation is given in Figure 7.3, where it can be seen that the NLO has considerably worse performance compared to both the EKF and XKF. The difference between the EKF and XKF is almost none in this case.



(a) Estimation of the bank angle

(b) Enlarged image of the bank angle estimation.

Figure 7.3: Estimation of bank angle when the initial state guesses perfectly match the real values.

Since there are a lot of random factors in this type of simulation, the result is most likely different for different runs. Hence an analysis of the RMSE was performed for 50 runs in order to compare the observers in a more general way. The gains were chosen as discussed earlier and the sampling time was 0.02 seconds. The result presented in Table 7.2 shows that the accuracy of the XKF is almost identical to the EKF when the states have perfect initial values. The NLO is close when it comes to altitude estimation, which was also seen in Figure 7.2. However, the performance is significantly worse for attitude estimation, which was seen in Figure 7.3. Also, note that the RMSEs for the NLO presented in Table 7.2 are not comparable to the ones presented in Table 7.1 since the sampling time was chosen differently. In addition to the RMSE, the average computation time for one iteration (evaluated with 15000 iterations) of the EKF and XKF is also presented in table 7.2. Note that the NLO is part of the XKF so its computational time is included in the time

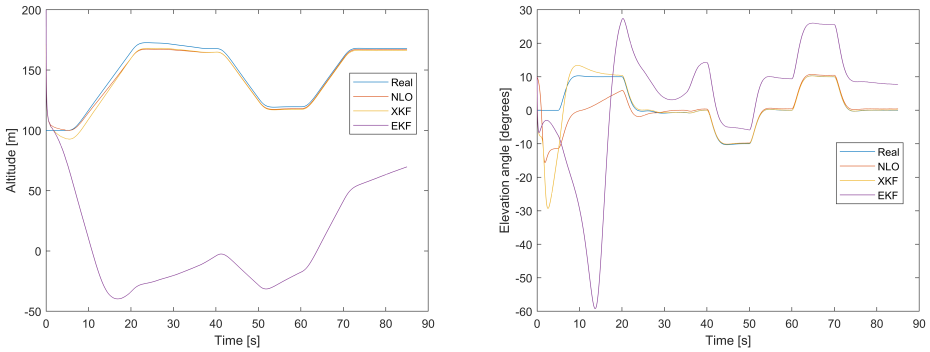
Observer	RMSE (Altitude)	RMSE (Roll)	Comp. time [s]
EKF	0.1180	0.0302	0.0027
NLO	0.1182	0.108	-
XKF	0.1182	0.0302	0.0046

Table 7.2: Performance difference of the three observers by comparing the RMSE.

presented for the XKF. The exact values for the time are of minor interest as it depends on the hardware used, but it shows that the XKF takes approximately 70% more time than the EKF on the same hardware. This result is expected since the XKF uses two observers and it should be considered when choosing what observer to use.

7.3.2 Unstable EKF

The main advantage of the XKF compared to the EKF presented in [3] is its stability properties. In order to find out if the XKF really provides better stability, the initial values of the states are pushed further away from the real values. As the initial error increases the convergence time for all filters are increased, but they all converge eventually. When the difference between the real initial values and the guesses are large enough, the EKF becomes unstable. As random factors are present in the simulation, it is not possible to



(a) Altitude estimations

(b) Elevation angle estimation

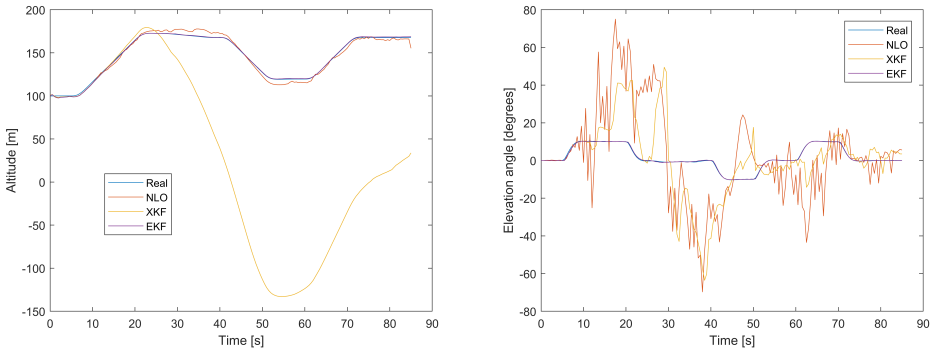
Figure 7.4: Altitude and elevation angle estimation when the initial values have large errors which causes the EKF to become unstable.

state exactly when the EKF goes unstable, but it was observed multiple times. However, the XKF has never been seen to diverge from the real trajectory in the same way. One example of when this happens can be seen in Figure 7.4. Here the EKF clearly does not estimate the states correctly while the XKF remains close to the real trajectory. In the same figures one can also see that the XKF takes approximately a minute to converge properly and the

performance is worse compared to the NLO for altitude estimation during the first seconds. The XKF is able to estimate the attitude better than any of the other two observers and converges after 20 seconds.

7.3.3 Unstable XKF

The XKF was shown to be more robust towards initial errors in the state vector than the EKF. However, during the simulations it was discovered that if the sampling time is not small enough, the XKF becomes unstable while the EKF remains stable. One example of this can be seen in Figure 7.5. The reason for



(a) Altitude estimations

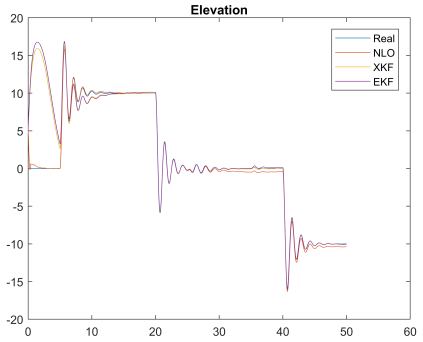
(b) Elevation angle estimation

Figure 7.5: Altitude and elevation angle estimation when the sampling time is chosen to 0.5s.

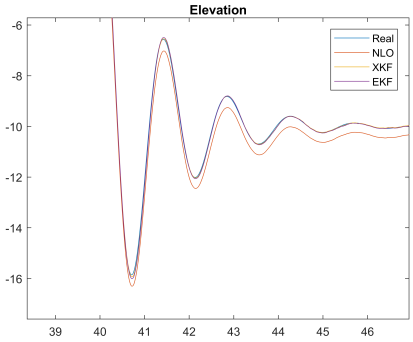
the XKF to become unstable is that the sampling time is part of the integration of the quaternion in the NLO (see (5.6)). The assumption that some quantities remains constant between the sampling instances is not valid as the sampling time grows too large and might be the reason for the instabilities to occur. It can be seen in Figure 7.5 that the attitude estimation from the NLO is the first thing to deviate from the real values. This is followed by a deviation of the whole XKF estimation in both attitude and altitude. The choice of sampling time clearly affects the stability of the NLO and must be chosen carefully.

An experiment where the flight was performed with more aggressive maneuvers was conducted to see if instability could occur due to the fact that the sampling time is not sufficient to cover the fast changes of the states. The factor K_p in the PD- controller that generates input signals was increased by a factor of 20 to make the input signals much larger. The resulting trajectory and estimations can be seen in Figure 7.6. It seems that the sampling frequency is sufficiently large and that the estimations are stable even when there are rapid changes of the states.

The topic of sampling time is not discussed in [19] nor [18] and it seems like the only method of determining a valid sampling time is through trial and error. In simulations this type of instability has never occurred for a sampling



(a) Trajectory and estimations



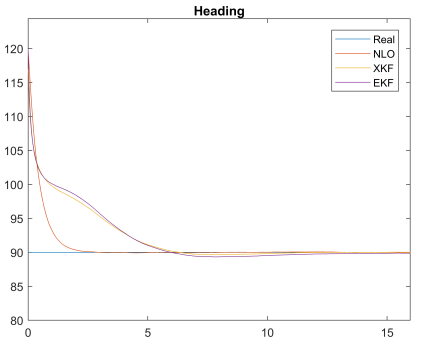
(b) Enlarged image

Figure 7.6: Elevation angle estimation with more aggressive maneuvers.

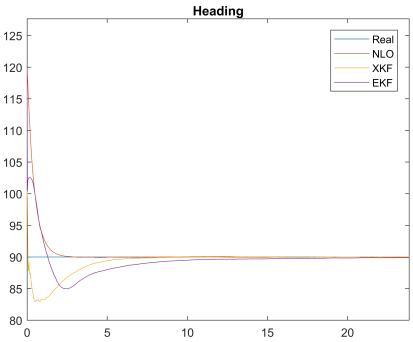
time of less than 0.1s. The sampling time of 0.02s used in the Stratos system should then be sufficiently small.

7.4 State Estimation Covariance Matrix

The error covariance matrix, $\boldsymbol{P}$, for the state estimate is a measure of how large the error is in the estimations compared to the real system. If the initial guesses of the values in this matrix are relatively high, it means that the initial state vector is assumed to contain large errors. Opposite, if the initial guesses for the variances are relatively low, it means that it is assumed that the initial state vector is more accurate. The transient performance of the filter when it is started is affected by how this initial matrix is chosen.



(a) Set 1

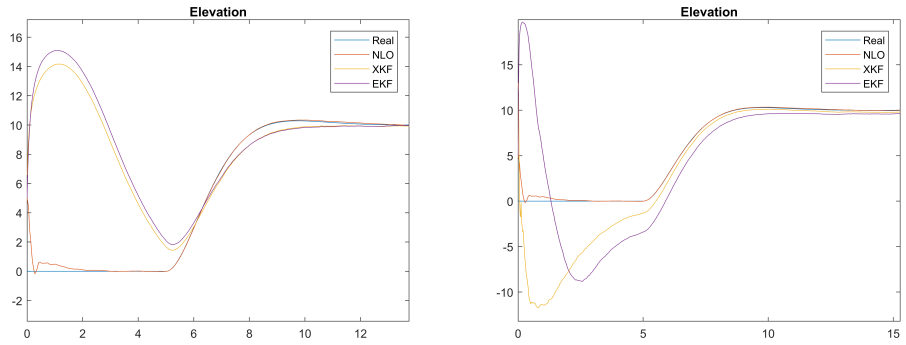


(b) Set 2

Figure 7.7: Heading estimation with the two sets of initial values for the $\boldsymbol{P}$ matrix.

An experiment was conducted in the simulation to see how the performance is affected by the choice of the initial $\boldsymbol{P}$ matrix. An initial error in the attitude

was set and the filters were tested with two sets of values in the $\boldsymbol{P}$ matrix. One of the sets was with low values (called set 1) and the other was with much higher values (called set 2). See appendix C for numerical values that are used by both the XKF and EKF simultaneously. The initial $\boldsymbol{P}$ matrix for the NLO is set to the identity matrix but can be tuned with the same parameters obtained for the KFs. The heading estimation with the two sets can be seen in Figure 7.7. It is clear that the response is quicker with set two, but it gets an overshoot while set one does not give an overshoot. The elevation angle estimation can be seen in Figure 7.8, where the response is similar to the heading estimation.



(a) Set 1 (b) Set 2

Figure 7.8: Elevation angle estimation with the two sets of initial values for the $\boldsymbol{P}$ matrix.

Both of the sets have similar performance in terms of convergence time, but set two has much larger overshoots. Hence, set one is the better choice in this case. However, the choice depends on how good the first state estimations are believed to be. If there are small initial errors then set 1 is better but, if there are even more errors than those simulated, then set 2 might give better convergence properties. This tuning of the parameters must be done when the filter is implemented in the real system. What should be observed when doing the tuning is how quickly the states changes and at what time the states seem to converge to a fixed value.

7.5 Improvement of Bias Estimation Method

As discussed in chapter 5.1.1, the bias estimation in the NLO is very simple. As long as there is a deviation in the states compared to measurements, the bias is corrected regardless where the origin of the error actually is. This could lead to large deviations and oscillations in the bias estimation. The Kalman filter estimation of the bias term is generally very slow and takes long time to converge, but it does not oscillate as much. It would be interesting to see if some modification to the filters could improve the bias estimation. In

Figure 7.9 one can see a typical estimation of the bias when the other states have perfect initial values. It can be seen that both Kalman filters converge very slowly and is almost identical while the NLO is faster, but with more oscillations. The gain k_I, affects the performance of the bias estimation from the NLO. Here it is chosen to the standard value of 0.005 The same seed of random noise has been used in the following analysis so that any changes of the quality of the estimate can be discovered. One suggestion in order to

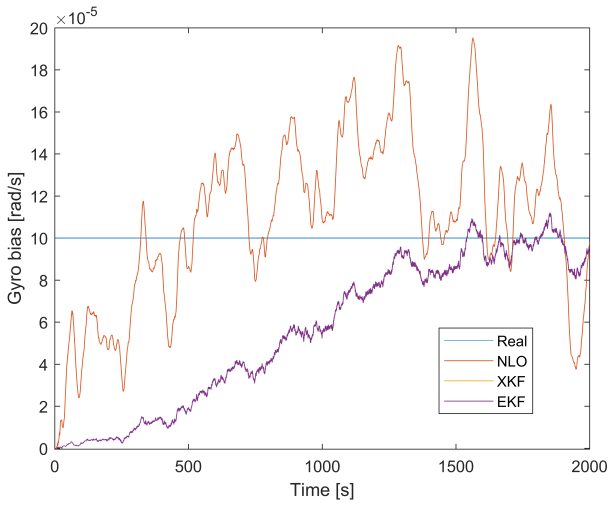
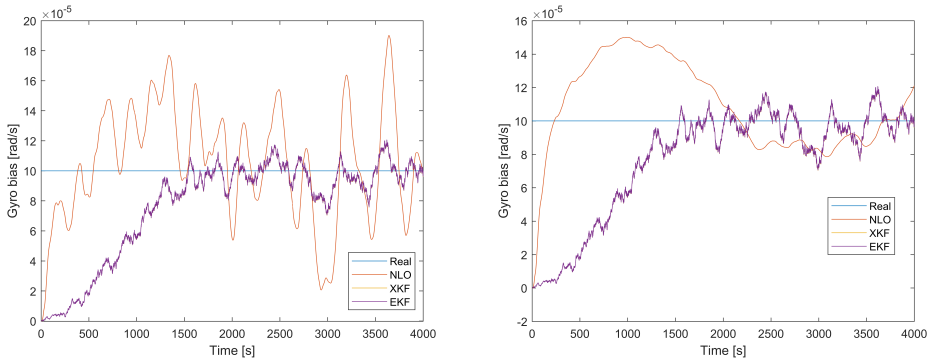


Figure 7.9: Bias estimation using the method presented in [18]

improve the bias estimate is to save a finite number of the injection term, $\hat{\boldsymbol{\sigma}}$, and instead use the moving average of these in equation 5.5. The goal with this is to make the bias estimation less sensitive to other disturbances (for example measurement noise) and eliminate some of the oscillations. It can be



(a) Average of 1000 terms

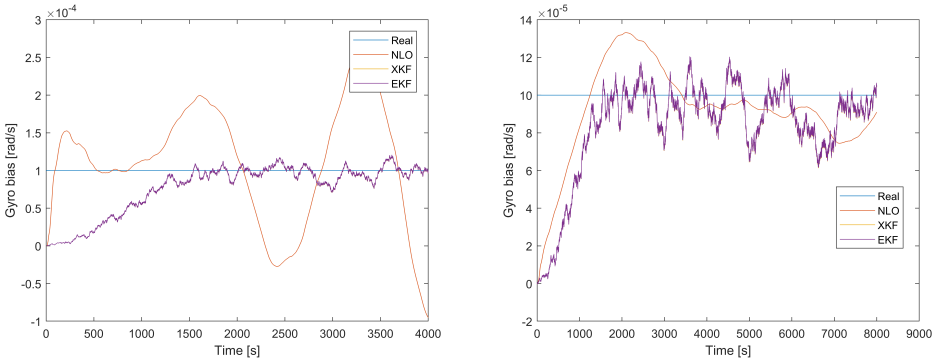
(b) Average of 10000 terms

Figure 7.10: Bias estimation when using a moving average of the injection term

seen in Figure 7.10 that the oscillations are reduced by taking a moving average of 1000 terms while the estimation seem to be much more stable when using

10000 terms. The oscillations when using 10000 terms in the moving average is of similar magnitude as for the Kalman Filter. Even though the rise time for the NLO estimate in Figure 7.10b is faster compared to the Kalman filters, there is an overshoot which causes the estimate to return to a stable estimate at approximately the same time as for the Kalman filter.

It is now reasonable to ask if there exist a better choice of the gain k_I, which can improve the performance. Figure 7.11 show the bias estimation for two other choices of the gain k_I. As expected, the larger gain gives faster convergence but very large oscillations, while the lower gain gives opposite behavior. What is interesting is that the NLO gives better bias estimation compared to the Kalman Filters for the case seen in Figure 7.11b.



(a) $k_I = 0.01$ (b) $k_I = 0.001$

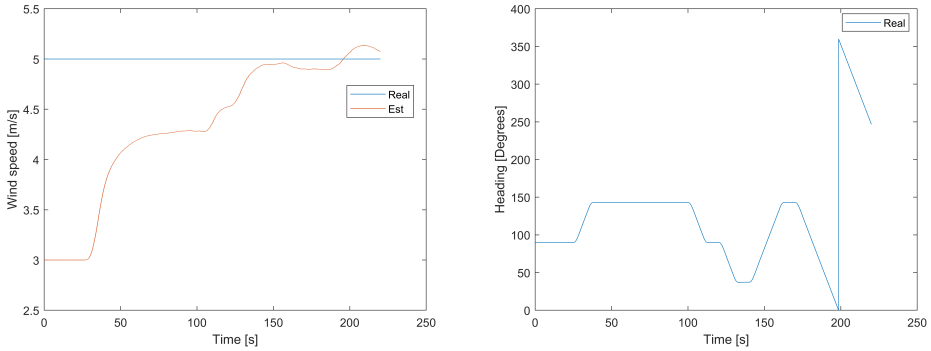
Figure 7.11: Bias estimation when using a moving average with 10000 terms for two choices of k_I.

It is seen that the oscillations can be removed, and it is possible to achieve a more reliable bias estimation with the NLO compared to the EKF and XKF by modifying the model used. A remaining problem is the extremely slow convergence of the bias estimation. This is a problem for two reasons. The first one is that it takes a long time for the estimations to reach the real values and secondly, if the bias varies with time it is likely that the estimations will not follow sufficiently.

The Kalman gain is calculated with carefully derived equations and for the nonlinear case it provides sub-optimal performance. However, it might be possible to modify this gain manually so that faster convergence for the bias is achieved. By multiplying the rows of the Kalman gain matrix corresponding to the bias terms with a constant, the correction of the bias can be faster. When testing this in the simulations it was shown that if the constant was equal to 2 the bias estimation had faster convergence, but more oscillations as expected. When the constant was increased to 3 the observer showed unstable results with estimations growing to infinity. It was then decided to not modify the Kalman gain manually as it is possible to introduce instabilities in the estimations.

7.6 Wind Estimation

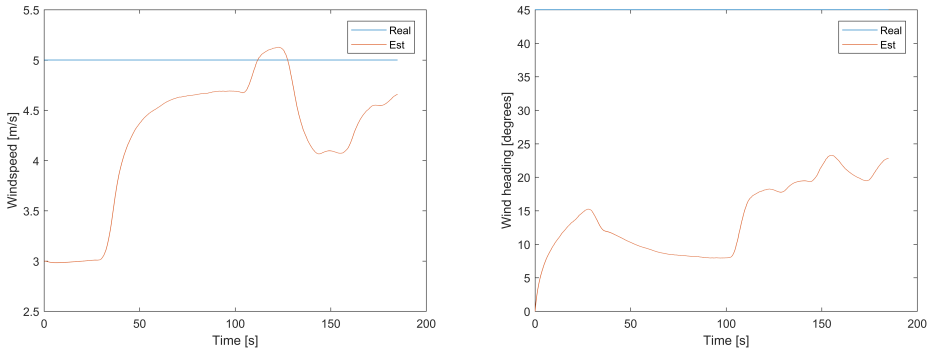
The wind velocity observer presented earlier is simulated together with the XKF. A simple model where it is assumed that a constant wind of 5 m/s to the north is present. The reference trajectory is extended with more maneuvers in order to fully understand the behavior of the observer. The observer is given a starting guess of 3 m/s to the north. The estimated magnitude of the wind velocity is presented in Figure 7.12 together with the heading of the UAV. What is interesting to see is that the wind estimation is only updated when



(a) Magnitude of wind velocity (b) Heading of the simulated UAV

Figure 7.12: Simulated wind velocity observer.

there is a change in the attitude of the UAV just as shown in [24]. It is clear that the wind velocity is only observable with this observer when the attitude is changed. However, the performance of the observer seems to be good with relatively fast convergence.



(a) Magnitude of wind velocity (b) Heading of the wind

Figure 7.13: Simulated wind velocity observer with error in the initial wind direction.

A second run was tested where the initial guess for the wind direction is to the north-east. The estimations from the observer can be seen in Figure 7.13

where the estimations have not converged after 190 seconds. It seems that the wind direction takes more time to estimate than the speed so a good initial guess for the heading is important for fast convergence.

If the intention is to fly the UAV with constant attitude, then the wind estimation will not be reliable (over time) with this observer and other more complex observers/sensors might be needed to provide accurate estimates. Otherwise a strategy where the UAV regularly performs a set of attitude changing maneuvers until the wind estimate seems to have converged can be implemented.

Chapter 8

Implementation and Testing

The practical implementation of the XKF in the Stratos Pilot system was done mainly by software engineers at Nystromer Avionics since the programming is written in C and they have more experience of the system. The programming was done based on the algorithm in appendix A. This chapter will present the process of implementing the XKF in the Stratos system and the insights gained will be discussed.

8.1 Test Rig

To be able to evaluate the XKF during the implementation process, it is necessary to have a real test rig that provides the same data as a real UAV would. This experimental test rig has all the sensors mounted on a beam together with all the hardware for the Stratos Pilot System. The interface of the software enables live presentation of any quantity (scalars,vectors and matrices) while the system is running which is very useful when trying to debug or understand the filter. The test rig that was used can be seen in Figure 8.1, where the blue box contains the hardware for the Stratos Pilot system. The magnetometer and GPS are mounted on the right side of the beam while the left side has the servos for the control surfaces.

8.2 Implementation Process

As mentioned earlier, the actual programming of the filter in the Stratos system was done by a software engineer at Nystromer Avionics. The strategy was that the algorithm created and simulated in MATLAB should be copied and translated to the C language. The challenge here was that the software engineer has little knowledge about the background of the filter while I have minor knowledge about the programming language. This leads to confusions and ultimately bugs in the implementation. So once the code was written in C there was a long process of debugging and understanding the behavior of the filter in the real system. The main issue initially was that the state vectors from both

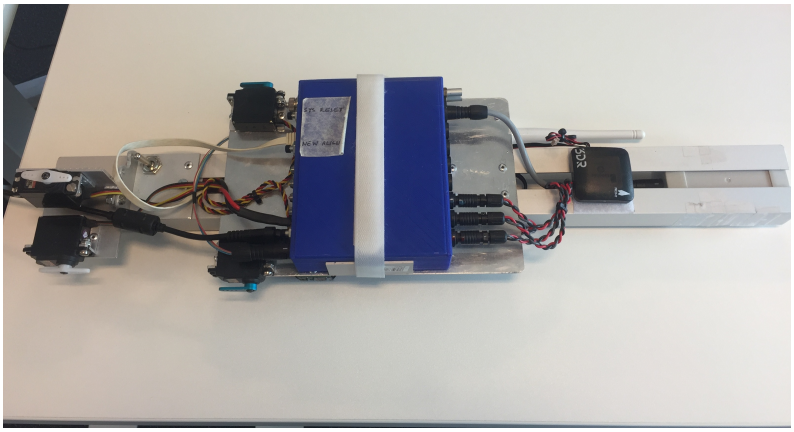


Figure 8.1: The test rig used when implementing the filter.

the NLO and final XKF was unstable and the values was increasing rapidly even when the test rig remained at rest. It is important to realize that when simulating the system, it is done in an ideal situation where all disturbances are intentional and known. In reality there are many more disturbances that are not modeled and hence they will have an unknown effect on the results. For example, it was noticed that the magnetic field measurements were highly disturbed by objects close to the test rig. Also, being inside a building affected the GPS signal which causes uncertainties in the measurements. It is difficult to distinguish between an error in the code to some unknown disturbance. The process of debugging the code was affected by these factors.

The strategy of debugging was to isolate each part of the filter so that they could be evaluated separately. For example, disabling the correction step in the NLO so that only the prediction step was providing the result. If there were large deviations of the states while the test rig was at rest, then there must be errors in that part of the code. This is an iterative process where all parts of the code are isolated to the highest extent.

It was suggested to begin the debugging process with the NLO since it is an independent observer. The NLO should be able to give accurate estimate regardless of the state of the LKF, which is not true in the opposite case. Once the errors in the code were found and corrected it was discovered that the attitude estimation was oscillating with an amplitude of about 20 degrees. It is believed that the noise in the magnetometer and uncertainty in the magnetic model is the cause of this since the same behavior was not observed when k_2 in (5.10) was set to zero. It was then decided to reduce the gain k_2 to 0.5 so that it has less impact on the estimations. The drawback with this is that the convergence of the attitude observer is slower, but it is necessary in order to avoid the oscillations. However, the final values of these parameters can only be chosen after testing the system more thoroughly.

Once the estimations were stable and provided reasonable values it was decided that the filter should be tested outside for further evaluation.

8.3 Ground Testing

Due to lack of time, there was not possible to perform real flight tests as intended. Instead the test rig was mounted on the roof of a car and connected to the software through a computer. Even though this was not real flight tests, the filter performance could still be observed and evaluated since the filter should estimate the states of the UAV regardless of the motion.

During the testing it was observed that the estimations remained stable based on the initial values, but the heading was not correct. It was then discovered that the magnetic field was highly disturbed by the car itself, so the test rig was removed from the roof and instead moved by hand. The filter then provided accurate estimations of the attitude and velocity while remaining stable when the test rig was moved in slow motions. If the rotations of the test rig were increased to more quicker motions, both the NLO and final XKF estimations started to diverge. The motions where similar to those tested in section 7.3.3 (which did not give the same result), so the cause of this is still unknown. The process of debugging the code had to be resumed. Due to lack of time, no more tests could be performed so that they can be included in this report.

Chapter 9

Conclusions

The objective with this thesis was to design and implement and XKF in the Stratos Pilot system. It was found that the XKF improves the stability of the state estimations while providing the same accuracy. This chapter will present the final remarks and conclusions regarding this project and suggestions for future work will be given.

9.1 Comparison Between XKF and EKF

The XKF has been simulated with the result that it provides better robustness and stability towards estimation errors than the EKF. It was also shown through the calculation of the RMSE that the accuracy of the XKF is identical to the EKF meaning that the main goal of this project is achieved with the use of the XKF.

It was proven in simulations that the EKF indeed can diverge from the real trajectory if the initial guess for the state vector contain large errors, while the XKF remains stable. The XKF does not always provide perfect stability as it is sensitive to the choice of tuning parameters of the NLO but if the NLO is designed so that it is globally stable then the final estimation of the XKF will inherit the same property.

It is clear that also the XKF has its disadvantages when it comes to stability. Just as the stability properties for the NLO are inherited to the final estimation, any instabilities in the NLO will also affect the final estimation. It is crucial that the NLO used as intermediate estimation is globally stable, which is a property affected by many variables. It is not trivial to guarantee the stability of the NLO and that introduces an uncertainty in the implementation.

It has been found throughout this project that the XKF has some drawbacks compared to the more established EKF. The complexity of the XKF that improves the stability properties can also introduce more errors in the implementation. The EKF has a well known algorithm that is independent of the type of system it is used for while the NLO is designed differently for each system. The process of implementing an XKF is hence more challenging compared to the EKF so that the simplicity of the EKF might be of higher

value for some applications. The complexity of the XKF leads also to higher requirements on the hardware as it was discovered that one iteration of the XKF takes 70% longer time compared to the EKF on the same hardware.

Overall, the XKF is considered to be a better choice of observer for UAV application compared to the EKF as the stability and safety of a UAV is of highest value.

9.2 Future of the XKF

The advantage of the XKF provides new possibilities for the use of UAVs that is not achievable with an EKF. With an EKF it is necessary that the initial state vector is as accurate as possible since the filter might be unstable otherwise. This means that most UAVs must be started on the ground while remaining at rest. With an XKF it can be possible to completely restart a UAV mid-air or have quicker take offs since the initialization process is not as important. The flexibility and safety of the UAV can hence be improved with an XKF compared to an EKF. In practice, this can open new applications for the UAV or make existing applications more efficient. This is a topic that should be investigated further.

The XKF can also improve the stability of the state estimations for other systems. A natural extension could be to look at the space industry where systems remain at rest for long periods of time. When these are restarted, the last known state vector might have changed drastically, which can make an EKF (if used) unstable. The XKF could then solve potential problems that can occur in those situations. A major challenge for the implementation of an XKF in other systems is the design of the NLO. Since there is no general method for this, it might not always be possible to design an NLO that is globally stable. However, if it is possible, it would be advantageous for systems with high uncertainty in the initial state estimations.

9.3 Future Work

The future work for Nystromer Avionics is to:

- Continue the debugging and verification of the XKF in order to see that it provides stable and accurate estimations. Tuning of the gains and initial state estimation covariance matrix can affect the performance and must be done through testing in real flight.
- The wind observer should be implemented and tested. Possible improvements with a sensor that measures the angle-of-attack and sideslip angle could be interesting to investigate.
- See if it is possible to implement the improved methods for bias estimation and use the bias estimation from the NLO in the state vector used

in the XKF. It might also be of interest to continue the development of the bias estimation to see if faster convergence can be achieved.

In the process of testing the filter in real flight, there are some interesting aspect that should be investigated. The suggested experiments that should be conducted is:

- Basic flight test to see that the states do not visually deviate from the reality or oscillate too much.
- Restart the filter mid-air to see that the states return to reasonable values. This can be done in multiple ways, but it would be interesting to change the heading significantly before the filter is restarted with the previous state vector. The XKF should return to more stable values, and if the same experiments are conducted with the EKF, then it might be possible to observe the same instabilities that was seen in the simulations.
- Perform "extreme" maneuvers to verify that the sampling frequency is sufficiently high. States should not start to diverge after this maneuver.

Other future research that could be interesting is to investigate if it is possible to combine the estimations from all three observers (EKF, NLO and XKF) to create better safety towards instability in the estimations. A simulation result where all three observers become unstable at the same time has never been observed. It could then be possible to have an "active safety" system that is activated when the estimations from the three observers start to deviate from each other. The control system could then receive a warning that there are problems with the estimations so that the control inputs are not made based on the information from the observer, but maybe from the sensors directly. The observers could then be restarted without affecting the UAV significantly. However, the computational power needed would be increased and it is questionable if such a system would be necessary since the XKF itself provides good stability properties.

Appendices

Appendix A

XKF Algorithm

This appendix gives the complete XKF algorithm for implementation. The details of the operation are either given in the comment column or can be found in the report.

Initialization

Step	Operation	Comment
1	Define constant matrices $\boldsymbol{Q}$,$\boldsymbol{R}$,$\boldsymbol{Q}_d$, $\boldsymbol{R}_d$, $\boldsymbol{S}(\omega_e)$, $\bar{\boldsymbol{\Omega}}(\omega_{ie}^e)$ $\boldsymbol{C}$ and $\boldsymbol{A}_d$	
2	Define initial $\boldsymbol{P}_x$ matrix for XKF and $\boldsymbol{P}_n$ for the NLO	
3	Estimate initial position, velocity and Euler angles	
4	Define initial $\boldsymbol{s}$ quaternion	See (2.13)
5	Define initial rotation matrix from NED-frame to body-frame, $\boldsymbol{C}_n^b$	See (5.25)
6	Calculate rotation matrix from earth-frame to NED-frame, $\boldsymbol{C}_e^n$	See (2.8)
7	Calculate rotation matrix from earth-frame to body-frame	$\boldsymbol{C}_e^b = \boldsymbol{C}_n^b \boldsymbol{C}_e^n$
8	Calculate initial quaternion with $\boldsymbol{C}_e^b$	See (2.14)
9	Populate initial state vector and set $\boldsymbol{\xi} = (0, 0, 0)$	
10	Define gains and time constants for the NLO	

Kalman filter loop

Correction step TMO

Step	Operation	Comment
1	Define intermediate state vector and measurement vector for the NLO	$\hat{\boldsymbol{x}}_n = (\hat{\boldsymbol{p}}_n^e, \hat{\boldsymbol{v}}_n^e, \boldsymbol{\xi})^T$, $\boldsymbol{y}_n = (\boldsymbol{p}_{GNSS}, \boldsymbol{v}_{GNSS})^T$
2	Calculate Kalman gain matrix	$\boldsymbol{K}_d = \boldsymbol{P}_n^- \boldsymbol{C}^T (\boldsymbol{C}\boldsymbol{P}_n^- \boldsymbol{C}^T + \boldsymbol{R}_d)^{-1}$
3	Correct state vector	$\hat{\boldsymbol{x}}_n^+ = \hat{\boldsymbol{x}}_n^- + \boldsymbol{K}_d(\boldsymbol{y}_n - \boldsymbol{C}\hat{\boldsymbol{x}}_n^-)$
4	Correct estimate covariance matrix	$\boldsymbol{P}_n^+ = (\boldsymbol{I} - \boldsymbol{K}_d\boldsymbol{C})\boldsymbol{P}_n^-$
5	Estimate specific force in earth-frame	$\hat{\boldsymbol{f}}^e = \boldsymbol{C}_b^e \boldsymbol{f}_{IMU}^b + \boldsymbol{\xi}$

Correction step XKF

Step	Operation	Comment
1	Create measurement model vector and measurement vector for XKF	$\boldsymbol{h}_x = (\hat{\boldsymbol{v}}_n, \hat{\boldsymbol{p}}_n, \hat{\boldsymbol{m}}_n)^T$, $\boldsymbol{y}_x = (\boldsymbol{v}_{GNSS}^e, \boldsymbol{p}_{GNSS}^e, \boldsymbol{m}_{mag}^b)^T$
2	Calculate Jacobi matrices $\boldsymbol{F}$, $\boldsymbol{H}$, $\boldsymbol{W}$ and $\boldsymbol{V}$ (using NLO estimates)	See appendix B
3	Calculate Kalman gain	$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}_k^T (\boldsymbol{H}_k \boldsymbol{P}_k^- \boldsymbol{H}_k^T + \boldsymbol{V}\boldsymbol{R}\boldsymbol{V}^T)^{-1}$
4	Correct state vector	$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(\boldsymbol{y}_k - \boldsymbol{h}_x - \boldsymbol{H}(\hat{\boldsymbol{x}}_k^- - \tilde{\boldsymbol{x}}_n))$
5	Update state estimation covariance matrix	$\boldsymbol{P}_k = (\boldsymbol{I} + \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_k^- (\boldsymbol{I} + \boldsymbol{K}_k\boldsymbol{H}_k)^T + \boldsymbol{K}_k\boldsymbol{R}\boldsymbol{K}_k^T$.

Prediction step XKF

Step	Operation	Comment
1	Integrate linearized model using Runge-Kutta method	See for example [21]
2	Predict state estimation covariance matrix	$\boldsymbol{P}_k^- = (\boldsymbol{I} + \Delta t \boldsymbol{F}_k)\boldsymbol{P}_{k-1}(\boldsymbol{I} + \Delta t \boldsymbol{F}_k)^T + (\Delta t)^2 \boldsymbol{W}_k \boldsymbol{Q} \boldsymbol{W}_k^T$

Attitude observer part 1

Step	Operation	Comment
1	Normalize specific force vectors	$\boldsymbol{f}^b$ and $\boldsymbol{f}^e$
2	Calculate first term of the injection term	$\hat{\boldsymbol{\sigma}}_1 = \frac{T_{acc}}{T} k_1 \boldsymbol{v}_1^b \times \boldsymbol{C}_e^b \boldsymbol{v}_1^e$
3	Calculate magnet field vector in the e- frame and normalize	
4	Normalize magnetometer measurement vector	
5	Calculate reference vectors v_2^b and v_2^e	See (5.2)
6	Calculate second term of the injection term	$\hat{\boldsymbol{\sigma}}_2 = \frac{T_{mag}}{T} k_2 \boldsymbol{v}_2^b \times \boldsymbol{C}_e^b \boldsymbol{v}_2^e$
7	Calculate the injection term	$\hat{\boldsymbol{\sigma}} = \hat{\boldsymbol{\sigma}}_1 + \hat{\boldsymbol{\sigma}}_2$
8	Calculate $\boldsymbol{S}(\hat{\boldsymbol{\sigma}})$	

Prediction step TMO

Step	Operation	Comment
1	Create input vector	$\boldsymbol{u}_d = (\boldsymbol{f}_{IMU}, -\boldsymbol{S}(\hat{\boldsymbol{\sigma}}) \boldsymbol{f}_{IMU})^T$
2	Calculate gravity force vector	See chapter 3.1.2
3	Update matrices $\boldsymbol{B}_d$, $\boldsymbol{D}_d$	See (5.14) and (5.15)
4	Propagate state vector in time	See (5.12)
5	Predict state estimation covariance matrix	See (5.19)
6	Enforce symmetry of the state estimation covariance matrix	See (5.21)

Attitude observer part 2

Step	Operation	Comment
1	Calculate the angular rotations	$\hat{\omega} = \omega_{IMU}^b - \hat{b}_g^b + \hat{\sigma}$
2	Calculate skew- symmetric matrix of the angular rotations	$\boldsymbol{S}(\hat{\omega})$
3	Calculate the matrix $\boldsymbol{\Omega}(\hat{\omega})$	See (5.8)
4	Calculate the matrix $e^{(\frac{T}{2}\Omega(\hat{\omega}))}$	See (5.6)
5	Calculate the matrix $e^{(-\frac{T}{2}\Omega(\hat{\omega_e}))}$	See (5.6)
6	Update quaternion estimation	See (5.5)
7	Normalize quaternion	
8	Update bias estimation	See (5.5)
9	Calculate rotation matrix from body-frame to earth-frame, $\boldsymbol{C}_b^e$	See (2.10) (Transpose!)

Appendix B

Jacobi Matrices

The Jacobi matrix $\mathbf{F}_k$

The matrix $\mathbf{F}_k$ is the Jacobi matrix of $f(\hat{\mathbf{x}}_k, \mathbf{u}_k)$, i.e the partial derivatives of the continuous system dynamics function, f (see eqn. 3.3) with respect to state variables x_i.

$$\mathbf{F}_k = \frac{\partial f}{\partial x_i}(\hat{\mathbf{x}}_k, \mathbf{u}_k, 0) \tag{B.1}$$

In matrix form $\mathbf{F}_k$ is written (with control input, $\mathbf{u}_k = \mathbf{0}$)

$$\mathbf{F}_k = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots \\ \vdots & \vdots & \end{pmatrix}_{13x13} = \tag{B.2}$$

$$\begin{pmatrix}
0 & 2\omega_e & 0 & \omega_e^2 & 0 & 0 & \frac{\partial f_1}{\partial q_0} & \frac{\partial f_1}{\partial q_1} & \frac{\partial f_1}{\partial q_2} & \frac{\partial f_1}{\partial q_3} & 0 & 0 & 0 \\
-2\omega_e & 0 & 0 & 0 & \omega_e^2 & 0 & \frac{\partial f_2}{\partial q_0} & \frac{\partial f_2}{\partial q_1} & \frac{\partial f_2}{\partial q_2} & \frac{\partial f_2}{\partial q_3} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial f_3}{\partial q_0} & \frac{\partial f_3}{\partial q_1} & \frac{\partial f_3}{\partial q_2} & \frac{\partial f_3}{\partial q_3} & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2}\omega_1 & -\frac{1}{2}\omega_2 & -\frac{1}{2}\omega_3 & \frac{1}{2}q_1 & \frac{1}{2}q_2 & \frac{1}{2}q_3 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2}\omega_1 & 0 & \frac{1}{2}\omega_3 & -\frac{1}{2}\omega_2 & -\frac{1}{2}q_0 & \frac{1}{2}q_3 & -\frac{1}{2}q_2 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2}\omega_2 & -\frac{1}{2}\omega_3 & 0 & \frac{1}{2}\omega_1 & -\frac{1}{2}q_3 & -\frac{1}{2}q_0 & \frac{1}{2}q_1 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2}\omega_3 & \frac{1}{2}\omega_2 & -\frac{1}{2}\omega_1 & 0 & \frac{1}{2}q_2 & -\frac{1}{2}q_1 & -\frac{1}{2}q_0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

where from 3.3 ($f_1, f_2 \ldots\ldots f_n$ refers to the n system equations respectively)

$$\frac{\partial f_1}{\partial q_0} = 2(f_1^b q_0 - f_2^b q_3 + f_3^b q_2), \qquad \frac{\partial f_1}{\partial q_1} = 2(f_1^b q_1 + f_2^b q_2 + f_3^b q_3)$$
$$\frac{\partial f_1}{\partial q_2} = 2(-f_1^b q_2 + f_2^b q_1 + f_3^b q_0), \qquad \frac{\partial f_1}{\partial q_3} = 2(-f_1^b q_3 - f_2^b q_0 + f_3^b q_1)$$
$$\frac{\partial f_2}{\partial q_0} = 2(f_1^b q_3 + f_2^b q_0 - f_3^b q_1), \qquad \frac{\partial f_2}{\partial q_1} = 2(f_1^b q_2 - f_2^b q_1 - f_3^b q_0)$$
$$\frac{\partial f_2}{\partial q_2} = 2(f_1^b q_1 + f_2^b q_2 + f_3^b q_3), \qquad \frac{\partial f_2}{\partial q_3} = 2(f_1^b q_0 - f_2^b q_3 + f_3^b q_2)$$
$$\frac{\partial f_3}{\partial q_0} = 2(-f_1^b q_2 + f_2^b q_1 + f_3^b q_0), \qquad \frac{\partial f_3}{\partial q_1} = 2(f_1^b q_3 + f_2^b q_0 - f_3^b q_1)$$
$$\frac{\partial f_3}{\partial q_2} = 2(-f_1^b q_0 + f_2^b q_3 - f_3^b q_2), \qquad \frac{\partial f_3}{\partial q_3} = 2(f_1^b q_1 + f_2^b q_2 + f_3^b q_3)$$

The Jacobi matrix $\mathbf{H}_k$

The matrix $\mathbf{H}_k$ is the Jacobi matrix of $h(\hat{\mathbf{x}}_k, 0)$, i.e the partial derivatives of the continuous measurement model matrix, h (see chapter 3.2.2) with respect to state variables x_i.

$$\mathbf{H}_k = \frac{\partial h}{\partial x_i}(\hat{\mathbf{x}}_k, 0) \tag{B.3}$$

In matrix form $\mathbf{H}_k$ is written

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial h_7}{\partial q_0} & \frac{\partial h_7}{\partial q_1} & \frac{\partial h_7}{\partial q_2} & \frac{\partial h_7}{\partial q_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial h_8}{\partial q_0} & \frac{\partial h_8}{\partial q_1} & \frac{\partial h_8}{\partial q_2} & \frac{\partial h_8}{\partial q_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial h_9}{\partial q_0} & \frac{\partial h_9}{\partial q_1} & \frac{\partial h_9}{\partial q_2} & \frac{\partial h_9}{\partial q_3} & 0 & 0 & 0 \end{pmatrix}_{9x13} \tag{B.4}$$

where

$$
\begin{aligned}
\frac{\partial h_7}{\partial q_0} &= 2q_0B^e_{x_1} + 2q_3B^e_{x_2} - 2q_2B^e_{x_3} &&= 2(q_0B^e_{x_1} + q_3B^e_{x_2} - q_2B^e_{x_3})\\
\frac{\partial h_7}{\partial q_1} &= 2q_1B^e_{x_1} + 2q_2B^e_{x_2} + 2q_3B^e_{x_3} &&= 2(q_1B^e_{x_1} + q_2B^e_{x_2} + q_3B^e_{x_3})\\
\frac{\partial h_7}{\partial q_2} &= -2q_2B^e_{x_1} + 2q_1B^e_{x_2} - 2q_0B^e_{x_3} &&= 2(-q_2B^e_{x_1} + q_1B^e_{x_2} - q_0B^e_{x_3})\\
\frac{\partial h_7}{\partial q_3} &= -2q_3B^e_{x_1} + 2q_0B^e_{x_2} + 2q_1B^e_{x_3} &&= 2(-q_3B^e_{x_1} + q_0B^e_{x_2} + q_1B^e_{x_3})\\
\frac{\partial h_8}{\partial q_0} &= -2q_3B^e_{x_1} + 2q_0B^e_{x_2} + 2q_1B^e_{x_3} &&= 2(-q_3B^e_{x_1} + q_0B^e_{x_2} + q_1B^e_{x_3})\\
\frac{\partial h_8}{\partial q_1} &= 2q_2B^e_{x_1} - 2q_1B^e_{x_2} + 2q_0B^e_{x_3} &&= 2(q_2B^e_{x_1} - q_1B^e_{x_2} + q_0B^e_{x_3})\\
\frac{\partial h_8}{\partial q_2} &= 2q_1B^e_{x_1} + 2q_2B^e_{x_2} + 2q_3B^e_{x_3} &&= 2(q_1B^e_{x_1} + q_2B^e_{x_2} + q_3B^e_{x_3})\\
\frac{\partial h_8}{\partial q_3} &= -2q_0B^e_{x_1} - 2q_3B^e_{x_2} + 2q_2B^e_{x_3} &&= 2(-q_0B^e_{x_1} - q_3B^e_{x_2} + q_2B^e_{x_3})\\
\frac{\partial h_9}{\partial q_0} &= 2q_2B^e_{x_1} - 2q_1B^e_{x_2} + 2q_0B^e_{x_3} &&= 2(q_2B^e_{x_1} - q_1B^e_{x_2} + q_0B^e_{x_3})\\
\frac{\partial h_9}{\partial q_1} &= 2q_3B^e_{x_1} - 2q_0B^e_{x_2} - 2q_1B^e_{x_3} &&= 2(q_3B^e_{x_1} - q_0B^e_{x_2} - q_1B^e_{x_3})\\
\frac{\partial h_9}{\partial q_2} &= 2q_0B^e_{x_1} + 2q_3B^e_{x_2} - 2q_2B^e_{x_3} &&= 2(q_0B^e_{x_1} + q_3B^e_{x_2} - q_2B^e_{x_3})\\
\frac{\partial h_9}{\partial q_3} &= 2q_1B^e_{x_1} + 2q_2B^e_{x_2} + 2q_3B^e_{x_3} &&= 2(q_1B^e_{x_1} + q_2B^e_{x_2} + q_3B^e_{x_3})
\end{aligned}
\tag{B.5}
$$

and

$$
\begin{aligned}
B^e_{x_1} &= -B_N sin\varphi cos\lambda \quad -B_E sin\lambda - B_D cos\varphi cos\lambda\\
B^e_{x_2} &= -B_N sin\varphi sin\lambda \quad +B_E cos\lambda - B_D cos\varphi sin\lambda\\
B^e_{x_3} &= +B_N cos\varphi \qquad\qquad\qquad\quad - B_D sin\varphi
\end{aligned}
$$

The Jacobi matrix $\mathbf{W}_k$

The matrix $\mathbf{W}_k$ is the Jacobi matrix of $f(\hat{\mathbf{x}}_k, \mathbf{u}_k, w_i)$, i.e the partial derivatives of the continuous system dynamics function,f with respect to system noise w_i

$$
\mathbf{W}_k = \frac{\partial f}{\partial w_i}(\hat{\mathbf{x}}_k, \mathbf{u}_k, w_i) \tag{B.6}
$$

In matrix form $\mathbf{W}_k$ is written

$$\mathbf{W}_k = \begin{pmatrix} \frac{\partial f_1}{\partial w_{f_1}} & \frac{\partial f_1}{\partial w_{f_2}} & \cdots \\ \frac{\partial f_2}{\partial w_{f_1}} & \frac{\partial f_2}{\partial w_{f_2}} & \cdots \\ \vdots & \vdots & \end{pmatrix}_{13x9}$$

$$= \begin{pmatrix} -Q_{11} & -Q_{12} & -Q_{13} & 0 & 0 & 0 & 0 & 0 & 0 \\ -Q_{21} & -Q_{22} & -Q_{23} & 0 & 0 & 0 & 0 & 0 & 0 \\ -Q_{31} & -Q_{32} & -Q_{33} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2}q_1 & \frac{1}{2}q_2 & \frac{1}{2}q_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{2}q_0 & \frac{1}{2}q_3 & -\frac{1}{2}q_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{2}q_3 & -\frac{1}{2}q_0 & \frac{1}{2}q_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2}q_2 & -\frac{1}{2}q_1 & -\frac{1}{2}q_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{B.7}$$

where Q_{11} through Q_{33} have expressions according to transformation matrix $\mathbf{C}_b^e$ (see transpose of eqn: 2.10), which here are the partial derivatives of the accelerometer bias noise terms.

The Jacobi matrix $\mathbf{V}_k$

The matrix $\mathbf{V}_k$ is the Jacobi matrix of $h(\hat{\mathbf{x}}, v_i)$, (see chapter 3.2.2) i.e the partial derivatives of the continuous measurement model matrix, h with respect to measurement noise v_i

$$\mathbf{V}_k = \frac{\partial h}{\partial v_i}(\hat{\mathbf{x}}_k, v_i) \tag{B.8}$$

In matrix form $\mathbf{V}_k$ is written

$$\mathbf{V}_k = \begin{pmatrix} \frac{\partial h_1}{\partial v_1} & \frac{\partial h_1}{\partial v_2} & \cdots & \cdots & \frac{\partial h_1}{\partial v_{10}} \\ \frac{\partial h_2}{\partial v_1} & \frac{\partial h_2}{\partial v_2} & \cdots & & \\ \vdots & \vdots & \ddots & & \\ \vdots & \vdots & & & \\ \frac{\partial h_9}{\partial v_1} & \cdots & & & \frac{\partial h_9}{\partial v_9} \end{pmatrix}_{9x9}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{B.9}$$

Appendix C

Covariance Matrices

Measurement noise covariance matrix

The measurement covariance matrix, $\mathbf{R}$ consists of the measurement noise variances along the matrix diagonal, under the assumption that there is no cross-correlation between sensor noise.

$$\mathbf{R} = \begin{pmatrix} \sigma^2_{v^e_{1GPS} v^e_{1GPS}} & 0 & \cdots & 0 \\ 0 & \sigma^2_{v^e_{2GPS} v^e_{2GPS}} & & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & & \sigma^2_{m^b_3 m^b_3} \end{pmatrix}_{9x9} \tag{C.1}$$

where

$$\begin{aligned}
\sigma^2_{v^e_{1GPS} v^e_{1GPS}} &= 0.012\ (m/s)^2 \\
\sigma^2_{v^e_{2GPS} v^e_{2GPS}} &= 0.005\ (m/s)^2 \\
\sigma^2_{v^e_{3GPS} v^e_{3GPS}} &= 0.024\ (m/s)^2 \\
\sigma^2_{x^e_{1GPS} x^e_{1GPS}} &= 22.8\ \ (m)^2 \\
\sigma^2_{x^e_{2GPS} x^e_{2GPS}} &= 9.2\ \ (m)^2 \\
\sigma^2_{x^e_{3GPS} x^e_{3GPS}} &= 103.8\ \ (m)^2 \\
\sigma^2_{m^b_1 m^b_1} &= 9.147 \cdot 10^{-16}\ \ (T)^2 \\
\sigma^2_{m^b_2 m^b_2} &= 1.622 \cdot 10^{-15}\ \ (T)^2 \\
\sigma^2_{m^b_3 m^b_3} &= 4.030 \cdot 10^{-15}\ \ (T)^2
\end{aligned}$$

Data from [22].

The corresponding matrix for the NLO is without the magnetometer data and having the velocity components swap place with the position components. The size of this matrix is 6×6.

Process noise covariance matrix

The process noise driving the system is the noise from the accelerometer, the gyros and the bias of the gyro.

$$\mathbf{Q} = \begin{pmatrix} \sigma^2_{w_{f_1 f_1}} & 0 & \cdots & 0 \\ 0 & \sigma^2_{w_{f_2 f_2}} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & \cdots & \sigma^2_{w_{b_{\omega 3} b_{\omega 3}}} \end{pmatrix}_{9x9} \tag{C.2}$$

where

$$\begin{aligned} Q_{1,1} &= \sigma^2_{w_{f_1 f_1}} = 7.20 \cdot 10^{-5}\ (m/s^2)^2 \\ Q_{2,2} &= \sigma^2_{w_{f_2 f_2}} = 7.65 \cdot 10^{-5}\ (m/s^2)^2 \\ Q_{3,3} &= \sigma^2_{w_{f_3 f_3}} = 4.91 \cdot 10^{-5}\ (m/s^2)^2 \\ Q_{4,4} &= \sigma^2_{w_{\omega_1 \omega_1}} = 7.44 \cdot 10^{-6}\ (rad/s)^2 \\ Q_{5,5} &= \sigma^2_{w_{\omega_2 \omega_2}} = 7.65 \cdot 10^{-6}\ (rad/s)^2 \\ Q_{6,6} &= \sigma^2_{w_{\omega_3 \omega_3}} = 4.91 \cdot 10^{-6}\ (rad/s)^2 \\ Q_{7,7} &= \sigma^2_{w_{b_{w1} b_{w1}}} = 1.0 \cdot 10^{-11} (rad/s)^2 \\ Q_{8,8} &= \sigma^2_{w_{b_{w2} b_{w2}}} = 1.0 \cdot 10^{-11} (rad/s)^2 \\ Q_{9,9} &= \sigma^2_{w_{b_{w3} b_{w3}}} = 1.0 \cdot 10^{-11} (rad/s)^2 \end{aligned}$$

Data from [22].
The corresponding NLO matrix is defined as

$$\boldsymbol{Q}_d = \text{blockdiag}(\boldsymbol{S}_f, \boldsymbol{S}_{\hat{\sigma} f}). \tag{C.3}$$

where the first matrix, $\boldsymbol{S}_f$, uses the data for the accelerometer. $\boldsymbol{S}_{\hat{\sigma} f}$ is chosen to $T\boldsymbol{S}_f$ but must be tuned during testing. The size of this matrix is 6×6.

Initial state estimate covariance matrix

The initial guess for the state estimate covariance matrix for the XKF and NLO is given here. For the XKF the initial matrix is given as

$$\mathbf{P}_0 = \begin{pmatrix} \sigma^2_{v_1^e v_1^e} & 0 & \cdots & 0 \\ 0 & \sigma^2_{v_2^e v_2^e} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & \cdots & \sigma^2_{b_{\omega 3} b_{\omega 3}} \end{pmatrix}_{13x13} \tag{C.4}$$

Set number 1 of the estimates of the initial state variables variances are

$$\begin{aligned} \{\sigma^2_{v_1^e v_1^e},\, \sigma^2_{v_2^e v_2^e},\, \sigma^2_{v_3^e v_3^e}\} &= \{\ 0.5,\, 0.5,\, 0.5\}\ (m/s)^2 \\ \{\sigma^2_{x_1^e x_1^e},\, \sigma^2_{x_2^e x_2^e},\, \sigma^2_{x_3^e x_3^e}\} &= \{9.0,\, 9.0,\, 9.0\ \}\ (m)^2 \\ \{\sigma^2_{q_0 q_0},\, \sigma^2_{q_1 q_1},\, \sigma^2_{q_2 q_2},\, \sigma^2_{q_3 q_3}\} &= \{5.0,\, 5.0,\, 5.0,\, 5.0\ \}\times 10^{-8}\ (dimensionless)^2 \\ \{\sigma^2_{b_{\omega 1} b_{\omega 1}},\, \sigma^2_{b_{\omega 2} b_{\omega 2}},\, \sigma^2_{b_{\omega 3} b_{\omega 3}}\} &= \{1.0,\, 1.0,\, 1.0\ \}\times 10^{-10}\ (rad/s)^2 \end{aligned}$$

Set number 2 of the estimates of the initial state variables variances are

$$\begin{aligned} \{\sigma^2_{v_1^e v_1^e},\, \sigma^2_{v_2^e v_2^e},\, \sigma^2_{v_3^e v_3^e}\} &= \{\ 1,\, 1,\, 1\}\ (m/s)^2 \\ \{\sigma^2_{x_1^e x_1^e},\, \sigma^2_{x_2^e x_2^e},\, \sigma^2_{x_3^e x_3^e}\} &= \{100,\, 100,\, 100\ \}\ (m)^2 \\ \{\sigma^2_{q_0 q_0},\, \sigma^2_{q_1 q_1},\, \sigma^2_{q_2 q_2},\, \sigma^2_{q_3 q_3}\} &= \{0.01,\, 0.01,\, 0.01,\, 0.01\ \}\ (dimensionless)^2 \\ \{\sigma^2_{b_{\omega 1} b_{\omega 1}},\, \sigma^2_{b_{\omega 2} b_{\omega 2}},\, \sigma^2_{b_{\omega 3} b_{\omega 3}}\} &= \{1.0,\, 1.0,\, 1.0\ \}\times 10^{-4}\ (rad/s)^2 \end{aligned}$$

The chosen NLO matrix is defined as

$$\mathbf{P}_{0,nlo} = \begin{pmatrix} \sigma^2_{x_1^e x_1^e} & 0 & \cdots & 0 \\ 0 & \sigma^2_{x_2^e x_2^e} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & \cdots & \sigma^2_{b_{\xi 3} b_{\xi 3}} \end{pmatrix}_{9x9} \tag{C.5}$$

where

$$\begin{aligned} \{\sigma^2_{x_1^e x_1^e},\, \sigma^2_{x_2^e x_2^e},\, \sigma^2_{x_3^e x_3^e}\} &= \{1.0,\, 1.0,\, 1.0\ \}\ (m)^2 \\ \{\sigma^2_{v_1^e v_1^e},\, \sigma^2_{v_2^e v_2^e},\, \sigma^2_{v_3^e v_3^e}\} &= \{\ 1,\, 1,\, 1\}\ (m/s)^2 \\ \{\sigma^2_{b_{\xi 1} b_{\xi 1}},\, \sigma^2_{b_{\xi 2} b_{\xi 2}},\, \sigma^2_{b_{\xi 3} b_{\xi 3}}\} &= \{1.0,\, 1.0,\, 1.0\ \}\ (m/s^2)^2 \end{aligned}$$

Bibliography

[1] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960.

[2] R. G. Brown and P. Y. C. Hwang, *Introduction to random signals and applied Kalman filtering.* New York: John Wiley & Sons, Inc., 1997.

[3] T. A. Johansen and T. I. Fossen, "The exogenous kalman filter (XKF)," *International Journal of Control*, vol. 90, no. 2, pp. 161–167, 2017.

[4] C. Jekeli, *Inertial Navigation Systems with Geodetic Applications.* Berlin: Walter de Gruyter, 2001.

[5] Wikipedia, the free encyclopedia, "North east down," 2017, [Accessed May 7, 2018]. [Online]. Available: https://en.wikipedia.org/wiki/North_east_down#/media/File:ECEF_ENU_Longitude_Latitude_relationships.svg

[6] B. L. Stevens and F. L. Lewis, *Aircraft Control and Simulation.* Hoboken: John Wiley & Sons, Inc., 2003.

[7] CH Robotics, "Understanding Euler angles," [Accessed May 7, 2018]. [Online]. Available: http://www.chrobotics.com/wp-content/uploads/2012/11/Inertial-Frame.png

[8] J. B. Kuipers, *Quaternions and Rotation Sequences.* Princeton: Princeton University Press, 2002.

[9] Wikipedia, the free encyclopedia, "Axes conventions," 2018, [Accessed May 7, 2018]. [Online]. Available: https://en.wikipedia.org/wiki/Axes_conventions#/media/File:Plane.svg

[10] NGA, "World geodetic system," https://www.nga.mil/ProductsServices/GeodesyandGeophysics/Pages/WorldGeodeticSystem.aspx, accessed May 16, 2018.

[11] "Datum transformations of GPS positions," https://microem.ru/files/2012/08/GPS.G1-X-00006.pdf, u-blox, Tech. Rep., 1999.

[12] J. Strickland, "What is a gimbal – and what does it have to do with NASA?" https://science.howstuffworks.com/gimbal.htm, accessed April 23, 2018.

[13] B. Boberg and S.-L. Wirkander, "Robust navigation using GPS and INS: Comparing the Kalman estimator and the particle estimator," Swedish Defence Research agency, Tech. Rep., 2002.

[14] L. Fusini, T. I. Fossen, and T. A. Johansen, "Nonlinear camera- and GNSS-aided INS for fixed-wing UAV using the exogenous kalman filter," in *Sensing and Control for Autonomous Vehicles: Applications to Land, Water and Air Vehicles*, T. I. Fossen, K. Y. Pettersen, and H. Nijmeijer, Eds. Cham: Springer International Publishing, 2017, vol. 474, ch. Lecture Notes in Control and Information Sciences, pp. 25–50.

[15] A. Kelly, *A 3D State Space Formulation of a Navigation Kalman Filter for Autonomous Vehicles*. Pittsburgh: The Robotics Institute Carnegie Mellon University, 1994.

[16] NOAA, "The world magnetic model," https://www.ngdc.noaa.gov/geomag/WMM/DoDWMM.shtml, accessed April 3, 2018.

[17] T. Glad and L. Ljung, *Control Theory: Multivariable and Nonlinear Methods*. London: CRC Press, 2000.

[18] T. H. Bryne, J. M. Hansen, R. H. Rogne, N. Sokolova, T. I. Fossen, and T. A. Johansen, "Nonlinear observers for integrated INS/GNSS navigation: Implementation aspects," *IEEE Control Systems*, vol. 37, no. 3, pp. 59–86, June 2017.

[19] H. F. Grip, T. I. Fossen, T. A. Johansen, and A. Saberi, "Nonlinear observer for GNSS-aided inertial navigation with quaternion-based attitude estimation," *American Control Conference*, pp. 272–279, June 2013.

[20] H. F. Grip, A. Saberi, and T. A. Johansen, "Observers for interconnected nonlinear and linear systems," *Automatica*, vol. 48, no. 7, pp. 1339–1346, July 2012.

[21] P. Pohl, *Grundkurs i numeriska metoder*. Stockholm: Liber, 2005.

[22] P. Nyströmer, "Stratos pilot kalman filter implementation," Nystromer Avionics AB, Tech. Rep., 2018, version 0.90.

[23] C. Jagadish and B.-C. Chang, "Diversified redundancy in the measurement of Euler angles using accelerometer and magnetometers," *IEEE Conference on Decision and Control*, vol. 46, pp. 2669–2674, December 2007.

[24] T. A. Johansen, A. Cristofaro, K. Sorensen, J. M. Hansen, and T. I. Fossen, "On estimation of wind velocity, angle-of-attack and sideslip angle of small UAVs using standard sensors," *International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 510–519, June 2015.

TRITA EECS-EX-2018:123
ISSN 1653-5146