An Investigation into Underwater Navigation Accuracy with regard to Sensor Combinations and Quality

by

Leo Herselman

Thesis presented at the University of Stellenbosch in partial fulfilment of the requirements for the degree of

Master of Science in Engineering

Department of Electrical Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Study leader: Mr J. Treurnicht

March 2008

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature: . . . .
L. Herselman

Date: . . . .

Abstract

A navigation performance sensitivity study with respect to the quality variation of sensors and sensor combinations is presented in this thesis.

Navigation simulation software capable of using real-time and simulated sensor data is developed in this project. The simulation software is used to simulate different sensor combinations and therefore evaluate the best configuration for each AUV mission. A sensor module is also developed to capture real-time sensor data. The sensor module includes a low-cost 6-degree-of-freedom inertial measurement module (rate gyroscopes and accelerometers), a three-axes magnetometer and other sensor interfaces. The real-time sensor data are used to test and calibrate the navigation simulation software. Different sensor combinations are evaluated by using the navigation simulation software with simulated sensor data as input. The quality of each sensor is varied by changing its noise characteristics.

The performance study, together with the developed simulation tools, simplifies the process of selecting a sensor combination to fulfill a specific navigation accuracy requirement.

Opsomming

'n Navigasie werksverrigtingstudie met betrekking tot sensor kwaliteit en kombinasies is in hierdie tesis voorgedra.

Navigasie simulasie sagteware, wat daartoe instaat is om intydse en gesimuleerde sensor data as intree te ontvang, word in hierdie projek ontwikkel. Die simulasie sagteware word gebruik om verskillende sensor kombinasies te evalueer om sodoende die mees optimale konfigurasie vir 'n selfbesturende onderwater voertuig te kies. 'n Sensor module is ook ontwikkel om intydse data te versamel. Die sensor module bevat 'n lae-koste 6-grade-van-vryheid inersiële metingseenheid (giroskope en versnellingsmeters), 'n drie-as magnetometer asook ander sensor koppelvlakke. Die intydse sensor data word gebruik om die simulasie sagteware te toets en te kalibreer. Verskillende sensor kombinasies is geëvalueer deur gebruik te maak van die simulasie sagteware en gesimuleerde sensor data. Die kwaliteit van elke sensor is gevarieer deur die sensor se ruis eienskappe te verander.

Die werksverrigtingstudie, saam met die ontwikkelde simulasie gereedskap, vergemaklik die proses waar 'n sensor kombinasie gekies moet word om 'n spesifieke navigasie akkuraatheidsvereiste te bevredig.

Acknowledgements

This project would not have been possible without the support, guidance and investment of numerous people. I would like to thank everybody who assisted me. In particular I would like to extend my gratitude to:

- Jesus Christ for giving me the strength and wisdom to complete this project.
- Mr Treurnicht for all your guidance and suggestions.
- My lab partners for all your help and discussions: Keith Browne, Izak Marais, Johan Schoonwinkel, Helgard van Rensburg and Emile Rossouw.
- Willie van Rooyen, Johan Arendse and other staff members from the University of Stellenbosch. Your support is greatly appreciated.
- Laurence Hill for proofreading this thesis.
- My parents and sister for their continued love and support.
- And most importantly, my wonderful fiancée, Cecile Kingma. Thank you for your love, patience, understanding and support throughout this project.

Contents

List of Tables

List of Figures

Abbreviations

6-DOF	Six Degrees of Freedom
A/D	Analog-to-Digital
AUV	Autonomous Underwater Vehicle
CAN	Controller Area Network
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
DCM	Direction Cosine Matrix
DSP	Digital Signal Processor
ECEF	Earth Centred Earth Fixed
EKF	Extended Kalman Filter
ESL	Electronic Systems Laboratory
FIR	Finite Impulse Response
GPS	Global Positioning System
GUI	Graphical User Interface
HMO	Hermanus Magnetic Observatory
I^2C	Inter-Integrated Circuit
IC	Integrated Circuit
IEEE	Institute for Electrical and Electronic Engineers
IIR	Infinite Impulse Response
IMT	Institute for Maritime Technology
IMU	Inertial Measurement Unit
ISR	Interrupt Service Routine
I/O	Input/Output
KB	Kilobytes

LSB	Least Significant Bit
LLH	Latitude Longitude Height
MCM	Mine-Countermeasures
NED	North East Down
MEKF	Multiplicative Extended Kalman Filter
MEMS	Micro Electromechanical Sensor
MIPS	Million Instructions Per Second
OPAMP	Operational Amplifier
PC	Personal Computer
PCB	Printed Circuit Board
PSD	Power Spectral Density
PVEKF	Position and Velocity EKF
RMS	Root Mean Square
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver Transmitter
UKF	Unscented Kalman Filter
US	University of Stellenbosch
WGS-84	World Geodetic System 1984

Symbols / Nomenclature

Symbols

Ω	Earth turn rate
λ, φ, h	Latitude, Longitude, Height
R	Earth radius
ϕ, θ, ψ	Roll, Pitch, Yaw
$q_1 - q_4$	Quaternion parameters
u, v, w	Body velocity vector
F_x, F_y, F_z	Applied forces
M_x, M_y, M_z	Applied moments
I_x, I_y, I_z	Moment of inertia in roll, pitch and yaw
V_N, V_E, V_D	Velocity along the NED axes
X_b, Y_b, Z_b	Vector in body axes
X_r, Y_r, Z_r	Vector in reference axes

Mathematical operations

$\otimes$	Quaternion multiplication
$\times$	Vector cross product

Matrices

$\mathbf{I}$	Identity matrix
$\mathbf{F}$	State transition matrix in the continuous domain
$\mathbf{G}$	Input matrix in the continuous domain

$\mathbf{L}_k$	Estimator feedback gain
$\mathbf{M}_k$	Error covariance matrix before a measurement update
$\mathbf{P}_k$	Error covariance matrix after a measurement update
$\mathbf{Q}_k$	Equivalent process noise covariance matrix in the discrete domain
$\mathbf{R}_k$	Equivalent measurement noise covariance matrix in the discrete domain
$\mathbf{T}$	Direction Cosine Matrix
$\mathbf{\Phi}_k$	State transition matrix in the discrete domain

Chapter 1

Introduction and Overview

1.1 Background

An AUV (Autonomous Underwater Vehicle) is a submersible vessel capable of operating or remaining underwater without the need for human occupancy or control.

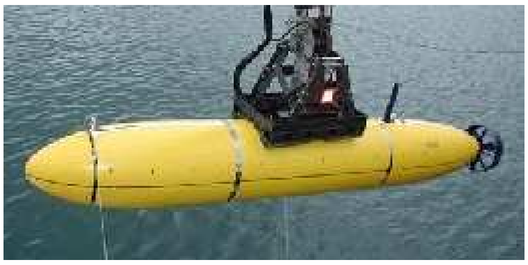


Figure 1.1: An AUV

AUVs can perform a multitude of tasks. These tasks include: search and rescue missions, military operations and the mapping of underwater terrain. The most important advantage of AUVs is their ability to perform hazardous missions. Entering unknown, deep ocean waters or crossing enemy lines to gather critical information can be extremely dangerous for the occupants of a manned underwater vehicle.

The threat of international terrorism makes harbor protection increasingly important [1]. Enemy mines in a harbor prevent supply ships from delivering their merchandise and can cripple a country's economy. AUVs can patrol the harbor floor, detect terrorist mines and, if deemed necessary, perform a MCM (Mine-Countermeasures) role. AUVs functioning in a harbor environment is considered as the main theatre of operation throughout this project. The requirements and constraints of an AUV inspecting the harbor floor are discussed in the next section.

1.2 AUV requirements and constraints

In order to classify an underwater vehicle as an AUV certain requirements need to be satisfied. The submerged vessel must be unmanned and must be able to calculate its current position, velocity and orientation from sensor data in order to navigate autonomously in its designated area. Obstacle avoidance technology, like sonar, is critical for detecting harbor walls or any other obstruction that may be encountered. An AUV must also be able to complete missions several hours long.

The most significant constraint on a submerged AUV is the absence of a GPS (Global Positioning System) signal. The AUV will only receive a GPS signal when it surfaces. Surfacing is not always a possibility, especially if the AUV needs to operate undetected. A lack of GPS signal will force the AUV to rely on its remaining inertial sensors. The inertial sensors will cause significant route drift, depending on sensor quality, if they are not corrected by the GPS signal. Surface buoys transmitting a GPS signal underwater can alleviate the drift problem, but it will decrease the AUV's chances of operating undetected and only a limited amount of area will be covered by the buoys' signal.

1.3 Project requirements

When a decision is made to design and manufacture a new AUV, a variety of aspects regarding navigation need to be taken into consideration. Particularly to what degree the navigation accuracy is required and what inertial sensor combination is required to deliver the accuracy. Also whether the GPS signal can be acquired periodically. These are some of the questions that need to be answered during the AUV design process. The most significant factor of any project is the budget. The budget factor raises the question: Will the available money be enough to satisfy the required specifications?

The stated questions create a need for a process by which sensor combinations and other navigation aspects can be simulated beforehand to attain the maximum navigation precision possible given the project budget and other relevant information. The demand for such a process led to the US (University of Stellenbosch) and IMT (Institute for Maritime Technology) creating the following requirements for this project:

1. Develop a sensor module. The module must allow the capturing of data from a variety of sensors and must include a low-cost 6-degree-of-freedom inertial measurement module (rate gyroscopes and accelerometers), a three-axes magnetometer and other sensor interfaces.

2. Develop a navigation algorithm simulation capable of using the batch measurement data in (1) to simulate the sensor combinations and therefore evaluate the best configuration for each mission.

3. Calibrate simulation inputs using measured and simulated sensor inputs.

4. Provide a GUI (Graphical User Interface) to allow the use of the simulation module for performance prediction.

5. Provide a sensitivity analysis per sensor option and sensor combination (fusion).

1.4 Outline

The chapters in this thesis are outlined as follows:

Chapter 2 discusses the theory behind inertial navigation. Chapter 3 focuses on the development of the sensor module and its embedded software, while Chapter 4 focuses on the development of the navigation simulation program and its GUI. Chapter 5 compares the different simulation results and Chapter 6 delivers the summary and recommendations for this project.

Chapter 2

Navigation Concepts

2.1 Overview

Chapter 2 focuses on the definition of navigation concepts and algorithms. Section 2.2 defines axial systems and rotations, while section 2.3 focuses on the equations of motion. Section 2.4 discusses the TRIAD algorithm. Section 2.5 concentrates on the Allan variance and section 2.6 explains state estimation concepts. Chapter 2 concludes with section 2.7 which gives a short summary of the chapter.

2.2 Axes definitions and rotations

Navigation is the process of determining the position, velocity and attitude of a physical body relative to a specified reference frame or axial system [2], [3]. For navigation over the earth, axial systems enable inertial measurements to be related to the primary directions of the earth [4]. The axial systems selected for this project are the inertial, Earth, navigation and body co-ordinate frames. The chosen co-ordinate frames have also been implemented by [5], [6] and [7] in the ESL (Electronic Systems Laboratory).

The axial systems are discussed in the following sections.

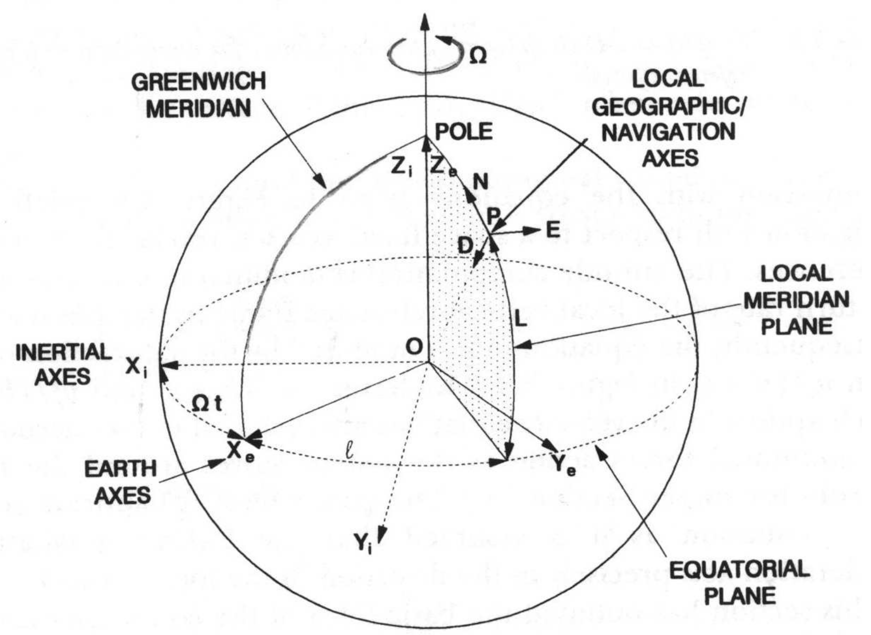


Figure 2.1: Axial systems [4]

2.2.1 Inertial axial system

The inertial axial system is defined by the orthogonal axes OX_i, OY_i and OZ_i illustrated in figure 2.1. The axial system is non-rotating with respect to the fixed stars. Point O represents the centre of Earth and the axis OZ_i is aligned with Earth's polar axis. Since the inertial axial system is non-rotating, it serves as a reference for rotating axial systems that change over time.

2.2.2 Earth axial system

The ECEF (Earth Centred Earth Fixed) axial system is the Earth axial system of choice for this project. Point O in figure 2.1 also represents the origin of the ECEF axial system. The axial system rotates, with respect to the inertial axial system, at a rate Ω (15 deg/h) about the axis OZ_i. The rectangular and geodetic ECEF systems are discussed below.

2.2.2.1 ECEF rectangular axial system

The ECEF rectangular axial system is defined by the orthogonal axes OX_e, OY_e and OZ_e illustrated in figure 2.1. The axis OZ_e is aligned with Earth's polar axis, while the axis OX_e lies along the intersection of the Greenwich meridian and Earth equatorial plane.

2.2.2.2 ECEF geocentric axial system

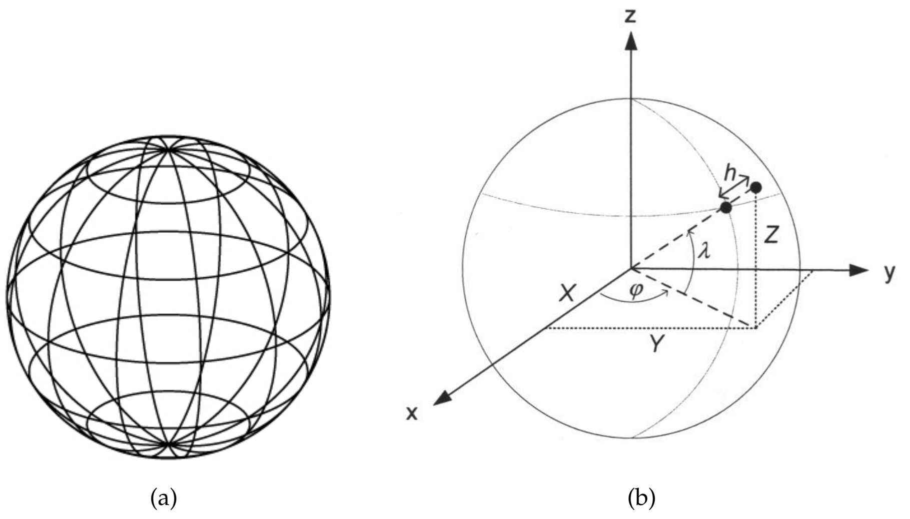


Figure 2.2: (a) Earth grids [8], (b) ECEF geocentric axial system [5]

Figure 2.2(a) illustrates the earth's surface divided into grids. The grids are known as latitude, λ, and longitude, φ. Figure 2.2(b) shows that latitude is measured from the equator with positive angles in a northern direction and negative angles in a southern direction, while longitude is measured from the Greenwich meridian with positive angles in an eastern direction and negative angles in a western direction. Latitude is given between -90° and 90°, while longitude is given between -180° and 180°. The height above the earth's radius, R, represents the height, h, of an object.

The use of the ECEF geocentric axial system indicates that a round earth model is used. However, the earth is not round and there are numerous models to describe the earth more accurately. A popular model is the WGS-84 (World Geodetic System 1984). The round earth model is accurate enough for this project considering that an AUV in a harbor environment will not travel at a high speed whereas the earth's radius will change significantly over time.

To convert from the ECEF rectangular axial system to the ECEF geocentric axial system and vice versa the following formulas can be used [5] :

$$\lambda = \tan^{-1}(\frac{Y}{X}) \tag{2.2.1}$$

$$\varphi = \tan^{-1}(\frac{Z}{\sqrt{X^2+Y^2}}) \tag{2.2.2}$$

$$h = \sqrt{X^2+Y^2+Z^2} - R \tag{2.2.3}$$

$$X = (R+h)\cos\lambda\cos\varphi \tag{2.2.4}$$

$$Y = (R+h)\cos\lambda\sin\varphi \tag{2.2.5}$$

$$Z = (R+h)\sin\lambda \tag{2.2.6}$$

where:

R = 6378137m = spherical radius of the earth

2.2.3 Navigation axial system

The navigation axial system, also called the NED (North East Down) axial system, is an axial system used for local navigation at a specific point on Earth. The three orthogonal axes, depicted in figure 2.1, are aligned with the directions of north, east and local vertical (pointing to the centre of the earth). The motion of the NED axial system with respect to the Earth axial system is referred to as the transport rate.

The NED axial system is adopted as the reference axial system for this project. The motion of the NED axial system with respect to the Earth axial system causes the NED axial system to deviate from the requirements of an inertial axis system. However, as AUVs usually maneuver at a low speed and do not travel far, the NED axial system can be considered to be an inertial axial system.

The position differential equations using the NED velocity states are useful when working with GPS measurements. Most GPS models provide velocity in the NED axial system and position in terms of latitude, longitude and height. The position differential equations are expressed as follows [9], [5] :

$$\dot{\lambda} = \frac{V_N}{R+h} \tag{2.2.7}$$

$$\dot{\varphi} = \frac{V_E}{(R+h)\cos\lambda} \tag{2.2.8}$$

$$\dot{h} = -V_D \tag{2.2.9}$$

where:

V = velocity North, East, Down

2.2.4 Body axial system

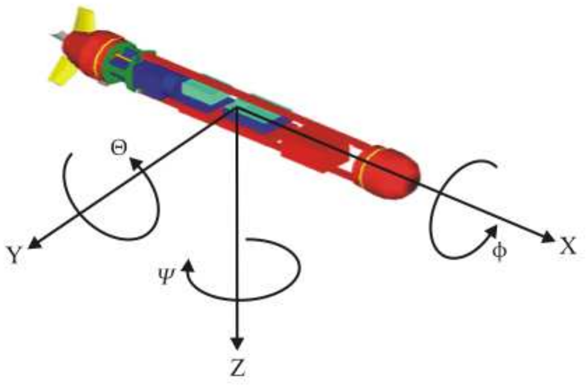


Figure 2.3: Body axes definition

The body axial system is a right handed orthogonal axial system, fixed to the body of a vehicle and constrained to move with it. Navigation sensors that are fixed to the body (known as a strap down configuration) will output sensor data in terms of the body axial system.

Figure 2.3 illustrates the body axial system. The X-axis points forward through the nose of the AUV, the Y-axis is directed to the right and the Z-axis points downward. The origin of the body axial system is chosen to coincide with the AUV's centre of gravity. A clockwise rotation around the X-axis, looking into the direction of the X-axis, generates a positive roll angle, ϕ, while a clockwise rotation around the Y-axis, looking into the direction of the Y-axis, produces a positive pitch angle, θ. Finally, a clockwise rotation around the Z-axis, looking into the direction of the Z-axis, causes a positive yaw angle, ψ. Roll, pitch and yaw angles usually define the attitude of a body.

2.2.5 Euler angles

The three Euler angles: roll, pitch and yaw, define the attitude of an object with respect to the reference axial system - i.e. to rotate the reference axial system through the Euler angles it will coincide with the body axial system. The order of rotations is important since the angles do not obey the commutative law [4], [10]. The Euler 3-2-1 system is adopted as the rotation order for this project and allows the transformation from one axial system to another by executing the rotations in the sequence: yaw, pitch and roll. Figure 2.4 illustrates the Euler 3-2-1 angle rotations.

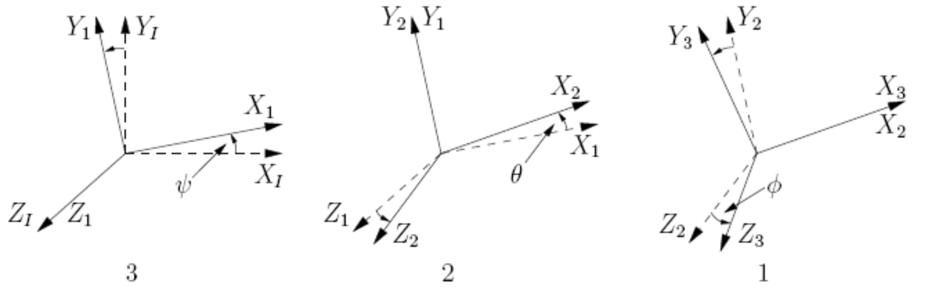


Figure 2.4: Definition of Euler 3-2-1 angle rotation [7]

The DCM (Direction Cosine Matrix) is a transformation matrix that transforms a vector in inertial axes to a vector in body axes. The DCM is denoted by $\boldsymbol{T}$ [9]:

$$\boldsymbol{T} = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix} \tag{2.2.10}$$

Now given the three Euler angles and equation (2.2.10) it is possible to calculate the DCM and perform the required axes transformation. Converting a vector in body axes to a vector in inertial axes is performed by calculating the inverse of the DCM. The DCM is orthogonal which implies that its inverse is simply its transpose.

$$\boldsymbol{T}^{-1} = \boldsymbol{T}^{T} =$$
$$\begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{2.2.11}$$

The following shows how the conversion from reference axes coordinates to body axes coordinates and vice versa is applied:

$$\begin{bmatrix} X_b \\ Y_b \\ Z_b \end{bmatrix} = \boldsymbol{T} \begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix} \tag{2.2.12}$$

$$\begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix} = \boldsymbol{T}^{T} \begin{bmatrix} X_b \\ Y_b \\ Z_b \end{bmatrix} \tag{2.2.13}$$

where:

$[\, X_b\ Y_b\ Z_b \,]^T$ = vector in body axes
$[\, X_r\ Y_r\ Z_r \,]^T$ = vector in reference axes

It is important to be cognizant of the relationship between the body angular rates p, q and r and the Euler rotation rates $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$. The relationship allows the Euler angles and thus the attitude of an object to be determined dynamically from the object-mounted gyros which provide body angular rates. The following equation depicts the relationship [9], [5], [3]:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.2.14}$$

The tangent and secant terms in equation (2.2.14) result in singularities at pitch angles of +90° and -90°. The singularities are an undeniable shortcoming of the Euler attitude representation method. An alternative method called the quaternion or "four parameter" method eliminates the mathematical singularities. The quaternion method is discussed in the following section.

2.2.6 Quaternions

The quaternion attitude representation is based on the concept that a conversion from one axial system to another may be accomplished by a single rotation about a vector $\boldsymbol{\mu}$ defined with respect to the reference axial system. The four element quaternion vector consists of functions of the vector, $\boldsymbol{\mu}$, and the magnitude of the rotation [4], [9] :

$$\boldsymbol{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} \frac{\mu_x}{\mu}\sin\frac{\mu}{2} \\ \frac{\mu_y}{\mu}\sin\frac{\mu}{2} \\ \frac{\mu_z}{\mu}\sin\frac{\mu}{2} \\ \cos\frac{\mu}{2} \end{bmatrix} \tag{2.2.15}$$

where:

μ_x, μ_y, μ_z = components of angle vector $\boldsymbol{\mu}$

μ = magnitude of $\boldsymbol{\mu}$

The reference axial system is rotated into coincidence with the body axial system by rotating about $\boldsymbol{\mu}$ through an angle μ.

The parameters of the quaternion vector must satisfy the following equation at all points in time [9] :

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1 \tag{2.2.16}$$

The Euler 3-2-1 DCM from equation (2.2.10) can now be converted to a quaternion form [5], [4] :

$$\boldsymbol{T} = \begin{bmatrix} q_4^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_4 q_3 + q_1 q_2) & 2(q_1 q_3 - q_4 q_2) \\ 2(q_1 q_2 - q_4 q_3) & q_4^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_4 q_1 + q_2 q_3) \\ 2(q_4 q_2 + q_1 q_3) & 2(q_2 q_3 - q_4 q_1) & q_4^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{2.2.17}$$

with the inverse rotation:

$$T^T = \begin{bmatrix} q_4^2+q_1^2-q_2^2-q_3^2 & 2(q_1\,q_2-q_4\,q_3) & 2(q_4\,q_2+q_1\,q_3) \\ 2(q_4\,q_3+q_1\,q_2) & q_4^2-q_1^2+q_2^2-q_3^2 & 2(q_2\,q_3-q_4\,q_1) \\ 2(q_1\,q_3-q_4\,q_2) & 2(q_4\,q_1+q_2\,q_3) & q_4^2-q_1^2-q_2^2+q_3^2 \end{bmatrix} \quad (2.2.18)$$

The conversion from Euler angles to quaternions and vice versa is shown in the following equations [4], [9] :

$$\begin{aligned} q_1 &= \cos\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2}-\sin\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} && (2.2.19)\\ q_2 &= \cos\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2}+\sin\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} && (2.2.20)\\ q_3 &= -\cos\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2}+\sin\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} && (2.2.21)\\ q_4 &= \cos\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2}+\sin\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2} && (2.2.22) \end{aligned}$$

$$\begin{aligned} \phi &= \tan^{-1}\left(\frac{2(q_4\,q_1+q_2\,q_3)}{q_4^2-q_1^2-q_2^2+q_3^2}\right) && (2.2.23)\\ \theta &= \sin^{-1}\left(-2(q_1\,q_3-q_4\,q_2)\right) && (2.2.24)\\ \psi &= \tan^{-1}\left(\frac{2(q_4\,q_3+q_1\,q_2)}{q_4^2+q_1^2-q_2^2-q_3^2}\right) && (2.2.25) \end{aligned}$$

where:

$q_4^2-q_1^2-q_2^2+q_3^2 \neq 0$ and $q_4^2+q_1^2-q_2^2-q_3^2 \neq 0$

In the event that only the DCM from equation (2.2.10) is available, the following equations can be utilized to calculate the four quaternions :

Define:

$$\text{DCM} = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \quad (2.2.26)$$

where:

$e_{[1..3][1..3]}$ = elements of DCM

Then, for small angular displacements:

$$q_4 = \frac{\sqrt{1+e_{11}+e_{22}+e_{33}}}{2} \tag{2.2.27}$$

$$q_1 = \frac{e_{23}-e_{32}}{4\,q_4} \tag{2.2.28}$$

$$q_2 = \frac{e_{31}-e_{13}}{4\,q_4} \tag{2.2.29}$$

$$q_3 = \frac{e_{12}-e_{21}}{4\,q_4} \tag{2.2.30}$$

where:

$q_4 \neq 0$

In equation (2.2.14) the relationship between the body angular rates p, q and r and the Euler rotation rates $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$ is established. However, by making use of the quaternion method the relationship is defined in terms of the body angular rates p, q and r and the quaternion rotation rates $\dot{q_1}$, $\dot{q_2}$, $\dot{q_3}$ and $\dot{q_4}$. The following equation depicts the relationship [5], [4] :

$$\begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \\ \dot{q_4} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} q_4 & -q_3 & q_2 & q_1 \\ q_3 & q_4 & -q_1 & q_2 \\ -q_2 & q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix}\begin{bmatrix} p \\ q \\ r \\ 0 \end{bmatrix} \tag{2.2.31}$$

Quaternion multiplication is defined as:

$$\boldsymbol{a} \otimes \boldsymbol{b} = \begin{bmatrix} b_4 & -b_3 & b_2 & b_1 \\ b_3 & b_4 & -b_1 & b_2 \\ -b_2 & b_1 & b_4 & b_3 \\ -b_1 & -b_2 & -b_3 & b_4 \end{bmatrix}\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} \tag{2.2.32}$$

Using equation (2.2.32), equation (2.2.31) can be written in terms of quaternion multiplication as:

$$\dot{\boldsymbol{q}} = \frac{1}{2}\,\boldsymbol{\omega} \otimes \boldsymbol{q} \tag{2.2.33}$$

where:

$\boldsymbol{\omega} = [\,\omega_1\,\omega_2\,\omega_3\,0\,]^T = [\,p\,q\,r\,0\,]^T$

Equation (2.2.33) is used for quaternion prediction - i.e. to keep track of the quaternion parameters which define body orientation. The quaternion parameters can then be utilized to calculate an equivalent DCM.

2.3 Equations of motion

The equations of motion of an AUV are identical to those of an aircraft and can be deduced from Newton's first and second law. The generalized 6-DOF (Six Degrees of Freedom) equations of motion in body coordinates are defined as follows [9], [11], [12] :

Translation:

$$\begin{aligned}
F_x &= m[\dot{u} - r\,v + q\,w - x(q^2 + r^2) + y(p\,q - \dot{r}) + z(p\,r + \dot{q})] && (2.3.1)\\
F_y &= m[\dot{v} - p\,w + r\,u + x(p\,q + \dot{r}) - y(p^2 + r^2) + z(q\,r - \dot{p})] && (2.3.2)\\
F_z &= m[\dot{w} - q\,u + p\,v + x(p\,r - \dot{q}) + y(q\,r + \dot{p}) - z(p^2 + q^2)] && (2.3.3)
\end{aligned}$$

Rotation:

$$\begin{aligned}
&M_x + F_y z - F_z y = \\
&\quad I_x\dot{p} - (I_y - I_z)q\,r + I_{xy}(p\,r - \dot{q}) - I_{xz}(p\,q + \dot{r}) + I_{yz}(r^2 - q^2) && (2.3.4)\\
&M_y + F_z x - F_x z = \\
&\quad I_y\dot{q} + (I_x - I_z)p\,r + I_{yz}(p\,q - \dot{r}) + I_{xz}(p^2 - r^2) - I_{xy}(q\,r + \dot{p}) && (2.3.5)\\
&M_z + F_x y - F_y x = \\
&\quad I_z\dot{r} - (I_x - I_y)p\,q - I_{yz}(p\,r - \dot{q}) + I_{xz}(q\,r - \dot{p}) + I_{xy}(q^2 + p^2) && (2.3.6)
\end{aligned}$$

where:

$[\,p\,q\,r\,]^T$ = body angular rotation vector

$[\,u\,v\,w\,]^T$ = body velocity vector

$[\,x\,y\,z\,]^T$ = offset between AUV centre and body axial system centre

$[\,F_x\,F_y\,F_z\,]$ = applied forces

$[\,M_x\,M_y\,M_z\,]$ = applied moments

$\begin{bmatrix} I_x & I_{xy} & I_{xz} \\ I_{xy} & I_y & I_{yz} \\ I_{xz} & I_{yz} & I_z \end{bmatrix}$ = body inertia tensor

Equation (2.3.1) to (2.3.6) can be simplified by making the following three assumptions:

- The centre of the AUV coincides with the centre of the body axial system.
  $\Rightarrow x = y = z = 0$
- The AUV is symmetric about the xz plane with the mass equally distributed.
  $\Rightarrow I_{xy} = I_{yz} = 0$
- The AUV body axes are aligned with the principle inertia axes.
  $\Rightarrow I_{xz} = 0$

With the three assumptions taken into consideration, equations (2.3.1) to (2.3.6) simplify to:

$$F_x = m[\dot{u} - r\,v + q\,w] \tag{2.3.7}$$
$$F_y = m[\dot{v} - p\,w + r\,u] \tag{2.3.8}$$
$$F_z = m[\dot{w} - q\,u + p\,v] \tag{2.3.9}$$

$$M_x = I_x\dot{p} - (I_y - I_z)q\,r \tag{2.3.10}$$
$$M_y = I_y\dot{q} + (I_x - I_z)p\,r \tag{2.3.11}$$
$$M_z = I_z\dot{r} - (I_x - I_y)p\,q \tag{2.3.12}$$

2.4 The TRIAD algorithm

The TRIAD algorithm is a deterministic method for calculating the DCM using vector pairs [13]. Each vector pair consists of a reference vector in reference axes, $\boldsymbol{v}^r$, and a measurement vector in body axes, $\boldsymbol{v}^b$. These two vectors are related through the DCM, $\boldsymbol{T}$, as follows [5] :

$$\boldsymbol{v}^b = \boldsymbol{T}\,\boldsymbol{v}^r \tag{2.4.1}$$

The reference vectors are known directions (e.g., the direction of the earth, the sun, the geomagnetic field or gravity), while the body vectors represent the same directions as measured by the vehicle's onboard navigation sensors.

The TRIAD algorithm uses two vector pairs, which means that there are four independent parameters. Only three parameters are needed for attitude determination, creating the need for one parameter to be discarded. The extra parameter or vector is eliminated by creating two triads: one for the two reference vectors and one for the two measurement vectors. A triad consists of three orthonormal vectors. The most accurate of the two input vectors forms the base vector of the triad. The second triad vector is constructed orthonormal to the two input vectors, and the third triad vector is generated orthonormal to the first and second triad vectors. The triad vectors must be unit vectors. The equations for the two triads are as follows [5], [13] :

Reference vector triad:

$$\boldsymbol{u}_1^r = \frac{\boldsymbol{v}_1^r}{|\boldsymbol{v}_1^r|} \tag{2.4.2}$$

$$\boldsymbol{u}_2^r = \frac{\boldsymbol{v}_1^r \times \boldsymbol{v}_2^r}{|\boldsymbol{v}_1^r \times \boldsymbol{v}_2^r|} \tag{2.4.3}$$

$$\boldsymbol{u}_3^r = \boldsymbol{u}_1^r \times \boldsymbol{u}_2^r \tag{2.4.4}$$

Measurement (body) vector triad:

$$\boldsymbol{u}_1^b = \frac{\boldsymbol{v}_1^b}{|\boldsymbol{v}_1^b|} \tag{2.4.5}$$

$$\boldsymbol{u}_2^b = \frac{\boldsymbol{v}_1^b \times \boldsymbol{v}_2^b}{|\boldsymbol{v}_1^b \times \boldsymbol{v}_2^b|} \tag{2.4.6}$$

$$\boldsymbol{u}_3^b = \boldsymbol{u}_1^b \times \boldsymbol{u}_2^b \tag{2.4.7}$$

In effect, one vector of the second reference-measurement pair is discarded, and therefore the first vector pair must be chosen to be more accurate than the second pair in order to maximize the TRIAD algorithm accuracy. The DCM can now be calculated as follows:

$$\boldsymbol{T} = \boldsymbol{M}_{bt}\,\boldsymbol{M}_{rt}^T \tag{2.4.8}$$

where:

$\boldsymbol{M}_{bt} = [\,\boldsymbol{u}_1^b\,\boldsymbol{u}_2^b\,\boldsymbol{u}_3^b\,]$ and $\boldsymbol{M}_{rt} = [\,\boldsymbol{u}_1^r\,\boldsymbol{u}_2^r\,\boldsymbol{u}_3^r\,]$

The DCM, calculated from the TRIAD algorithm, is used in this project to initialize the quaternions utilizing equations (2.2.27) to (2.2.30).

The QUEST algorithm is another attitude determination method that attempts to minimize a loss function in order to find the optimal DCM. The QUEST approach can be used when two or more vector pairs are available. However, only two vector pairs are available for this project and the added computational complexity of the QUEST algorithm is not justified by the small increase in accuracy [5]. The QUEST method performs well in situations where there are more than two vector pairs available.

2.5 Allan Variance

Different types of noise cause sensor data to be of questionable accuracy. Allan Variance is a method for examining recorded sensor data in order to obtain a characteristic curve that provides a systematic characterization of various random errors contained in the inertial sensor output data [14]. The types and magnitude of various noise terms can be provided by the measurable Allan variance quantity. Identifying the aspects of various noise terms enables the user to model and simulate the noise, resulting in more precise sensors. The increase in sensor accuracy will also contribute to the decrease in navigation errors. There are many methods available for the identification and stochastic modeling of inertial sensor noise, [14]. However, the simple implementation and ease of interpretation of the Allan variance method makes it the noise identification method of choice for this project.

There are five basic noise terms [14], [5] :

1. Angle / velocity random walk
2. Rate random walk
3. Bias instability
4. Quantisation
5. Drift rate ramp

The first two sources dominate the noise on MEMS (Micro Electromechanical Sensor) devices as well as many other sensors, [5], and are the only two noise sources modeled and simulated in this project.

The Allan variance method is based upon variance calculation as a function of average time. Sensor data, sampled for at least a hour, are utilized for the variance calculation process. It is important to keep the selected sensor in a static position, to prevent the sampled values from being influenced by movement. During the variance determination process the use of short average times will result in the variance being overshadowed by high-frequency measurement noise. Long average times will result in the variance being dominated by low-frequency bias drift.

The Allan variance process and equations are defined as follows [5] :

Assume the sensor output is recorded as $x(k)$ and has a sampling time of T_s seconds. Select a bin size of T seconds where:

$$T = n\, T_s \tag{2.5.1}$$

Divide the sensor data, $x(k)$, into consecutive bins. See figure 2.5.

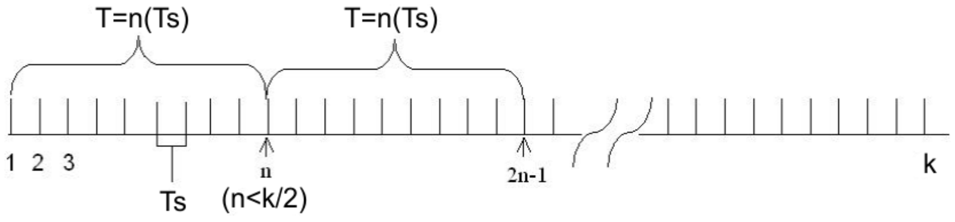


Figure 2.5: Allan Variance calculation process [14]

Next, calculate the average value, $y(k)$, of each bin:

$$y(k) = \frac{1}{n} \sum_{i=(k-1)n+1}^{kn} x(i) \tag{2.5.2}$$

Now, calculate the Allan variance by summing the squared difference between two successive bins over the whole range and normalizing it:

$$\sigma_A^2(T) = \frac{1}{2(N-1)} \sum_{k=1}^{N-1} (y(k+1) - y(k))^2 \tag{2.5.3}$$

where:

N = number of bins in the data

It is important to note that the Allan variance, equation (2.5.3), is a function of the bin size, T, and not a scalar value as in the case of normal deviation computation.

There exists a special relationship between the Allan variance and the PSD (Power Spectral Density) of the random noise processes [14]. The IEEE (Institute for Electrical and Electronic Engineers) defines the relationship as follows:

$$\sigma_A^2(T) = 4\int_0^\infty S(f)\frac{\sin^4(\pi f T)}{(\pi f T)^2}\ df \tag{2.5.4}$$

where:

$S(f)$ = PSD of the random noise process

2.5.1 Angle / velocity random walk

The integration of the measurement noise on the output of a sensor results in a random walk. Integrating the output noise of a rate gyroscope, given in rad/s, produces an angle random walk, while integrating the output noise of a accelerometer, given in m/s^2, causes a velocity random walk. The IEEE standard characterizes the angle / velocity random walk as a white noise spectrum defined by the following PSD [14], [5] :

$$S_{ARW}(f) = S_{VRW}(f) = Q^2 \tag{2.5.5}$$

where:

Q = angle / velocity random walk coefficient

It is important to be aware of the fact that equation (2.5.5) implies that the noise signal will have unlimited energy. Infinite energy is impossible in the real world. However, (2.5.5) still serves as an acceptable approximation of real noise when the bandwidth is restricted to a certain range.

The Allan deviation (square root of Allan variance) for angle / velocity random walk can be formulated by using equation (2.5.5) in equation (2.5.4) and results in the following [14] :

$$\sigma_{ARW}(T) = \frac{Q}{\sqrt{T}} \tag{2.5.6}$$

Angle / velocity random walk	SI-units
Rate gyroscope	$rad/s/\sqrt{Hz}$
Accelerometer	$m/s^2/\sqrt{Hz}$

Table 2.1: The SI-units of Q

There exists a relationship between the angle / velocity random walk coefficient, Q, and the variance used in Kalman filters (discussed in section 2.6). The relationship is as follows [15] :

$$\sigma^2_{ARW\ KF} = 2\int_{f1}^{f2} Q^2\ df \tag{2.5.7}$$

2.5.2 Rate random walk

The drifting of the sensor bias is called a rate random walk. The sensor bias instability can be due to temperature dependency or any other drift-causing factor.

The rate random walk can be characterized by the following IEEE standard PSD [14], [5] :

$$S_{RRW}(f) = \left(\frac{K}{2\pi f}\right)^2 \tag{2.5.8}$$

where:

K = rate random walk coefficient

Equation (2.5.8) has more power in lower frequencies, verifying that bias drifts are usually slow processes. The rate random walk PSD can be interpreted as the integration of white noise [5].

The Allan deviation for rate random walk can be formulated by using equation (2.5.8) in equation (2.5.4) and results in the following [14] :

$$\sigma_{RRW}(T) = K\sqrt{\frac{T}{3}} \tag{2.5.9}$$

Rate random walk	SI-units
Rate gyroscope	$rad/s^2/\sqrt{Hz}$
Accelerometer	$m/s^3/\sqrt{Hz}$

Table 2.2: The SI-units of K

By substituting Q with K in equation (2.5.7) the rate random walk coefficient can also be converted to a variance for use in Kalman filters.

2.5.3 Log-log plot

Figure 2.6 illustrates a straight line with a slope of $-\frac{1}{2}$. The line is the result of plotting equation (2.5.6) as a function of averaging time, T, on a log-log scale. The value of Q can be obtained by reading the slope line at $T = 1$.

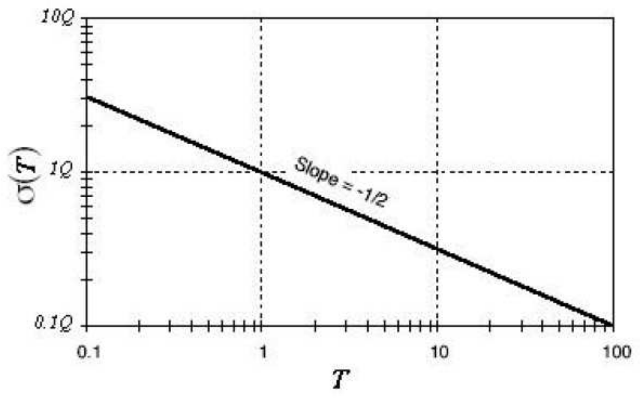


Figure 2.6: $\sigma(T)$ plot for angle / velocity random walk [14]

Figure 2.7 illustrates a straight line with a slope of $\frac{1}{2}$. The line is the result of plotting equation (2.5.9) as a function of averaging time, T, on a log-log scale. The value of K can be obtained by reading the slope line at $T = 3$.

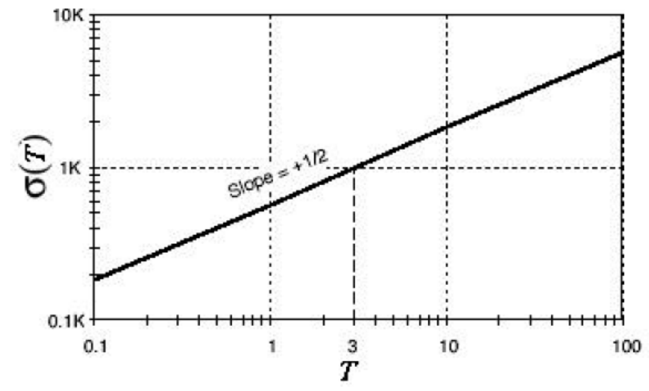


Figure 2.7: $\sigma(T)$ plot for rate random walk [14]

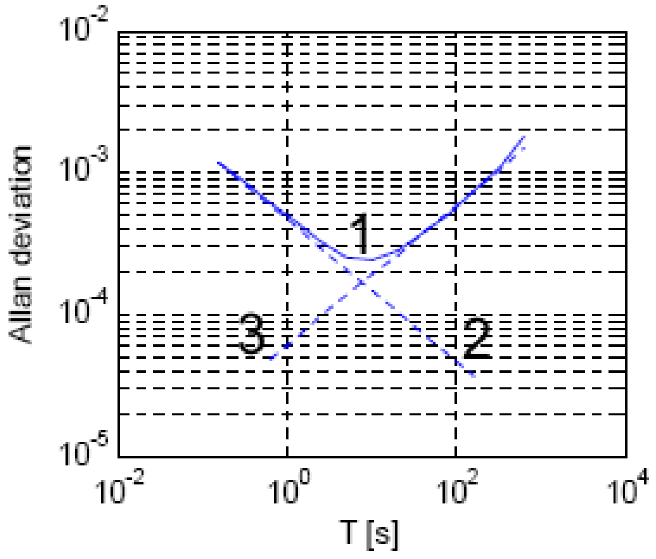


Figure 2.8: Allan Deviation results [5]

Figure 2.8 summarizes the process for calculating the coefficients Q and K:

- Plot the square root of equation (2.5.3) as a function of averaging time, T, on a log-log scale (1).
- Plot equation (2.5.6) on the same log-log scale as (1). Shift the line plot up or down until it roughly coincides with a part of the negative slope of (1). Obtain Q by reading the slope line (2) at $T = 1$.
- Plot equation (2.5.9) on the same log-log scale as (1). Shift the line plot up or down until it roughly coincides with a part of the positive slope of (1). Obtain K by reading the slope line (3) at $T = 3$.

2.6 State estimation

"The states of a system are those variables that provide a complete representation of the internal condition or status of the system at a given instant of time [16]." Position, velocity and angular orientation are the selected AUV states for this project. The translation and rotation of an AUV cause the states to change with respect to time. Keeping track of the changing states requires estimation algorithms. There are a multitude of estimation algorithms, each with its own advantages and disadvantages [16], [17]. A study by [5] concludes that the best approach for state estimation of a vehicle with slow dynamics and small angular rotations requires the use of two estimation algorithms, the MEKF (Multiplicative Extended Kalman Filter) and the PVEKF (Position and Velocity Extended Kalman Filter). The MEKF estimates the attitude of the vehicle, while the PVEKF estimates the position and velocity of the vehicle. The study by [5] states that the use of two algorithms decreases the computational load considerably, while sacrificing minimal accuracy compared to an approach that implements one algorithm to estimate all the states. Another advantage of using two estimation algorithms is that the MEKF will still operate optimally even when there are no GPS updates available, while the one-algorithm approach will be significantly affected by the absence of GPS updates. GPS signals are unavailable underwater and an AUV in a harbor environment usually demonstrates slow dynamics. The previous two facts make the MEKF and the PVEKF the adopted estimation algorithms for this project. Section 2.6.1 briefly describes the basic EKF (Extended Kalman Filter) and its equations. Section 2.6.2 discusses the attitude MEKF algorithm, while section 2.6.3 focuses on the PVEKF algorithm.

2.6.1 Basic EKF

The Kalman filter is an effective and versatile algorithm for combining noise polluted sensor data to estimate the states of a system with uncertain dynamics. "A Kalman filter is essentially a set of mathematical equations that implement an estimator that is optimal in minimizing estimated error covariances between the prediction of parameters from a previous time constant and the external observations at a present time (measurements) [9]." An extended Kalman filter is utilized when the system dynamics or measurement equations are non-linear. In this project the system dynamics are non-linear and thus require the use of an EKF algorithm.

The EKF process and its equations are stated concisely in this section. For a detailed explanation and derivation of the Kalman filter equations see [16], [17], [19] and [18].

The dynamics of a non-linear system can be defined by the following continuous difference equations:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}, t) \tag{2.6.1}$$

with a measurement that is:

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{v}, t) \tag{2.6.2}$$

where:

$\boldsymbol{x}$ = system states
$\boldsymbol{u}$ = control inputs
$\boldsymbol{w}$ = process noise
$\boldsymbol{v}$ = measurement noise

The random zero-mean processes, $\boldsymbol{w}$ and $\boldsymbol{v}$, are assumed to be independent of one another, white and with normal probability distributions.

$$p(\boldsymbol{w}) \sim (0, \boldsymbol{Q}) \tag{2.6.3}$$

$$p(\boldsymbol{v}) \sim (0, \boldsymbol{R}) \tag{2.6.4}$$

where:

$\boldsymbol{Q}$ = process noise covariance matrix
$\boldsymbol{R}$ = measurement noise covariance matrix

Assuming that the noise can be decoupled from the non-linear system dynamics, equation (2.6.1) and equation (2.6.2) can be written as follows:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{w}(t) \tag{2.6.5}$$

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}, t) + \boldsymbol{v}(t) \tag{2.6.6}$$

Equation (2.6.5) is used for state propagation between measurement updates. When update measurements are available, the Riccati equations (see equations 2.6.12 to 2.6.14) need to be calculated to determine the Kalman gains and propagate the error covariance matrix. The Riccati equations require linear plant dynamics, thus a first order approximation around the current best estimate is used to calculate the continuous linearized plant dynamics:

$$\boldsymbol{F} = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}} \tag{2.6.7}$$

$$\boldsymbol{G} = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)}{\partial \boldsymbol{u}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}} \tag{2.6.8}$$

$$\boldsymbol{H} = \left.\frac{\partial \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}, t)}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}} \tag{2.6.9}$$

The EKF is implemented as a discrete algorithm, resulting in the use of the discrete Riccati equations which require discrete versions of $\boldsymbol{F}$, $\boldsymbol{Q}$ and $\boldsymbol{R}$. The matrix $\boldsymbol{F}$ can be approximated by a Taylor-series expansion and is defined as follows [19] :

$$\begin{aligned} \boldsymbol{\Phi}_k &= \boldsymbol{I} + \boldsymbol{F}T_s + \frac{\boldsymbol{F}^2 T_s^2}{2!} + \frac{\boldsymbol{F}^3 T_s^3}{3!} + \dots \\ &\approx \boldsymbol{I} + \boldsymbol{F}T_s \end{aligned} \tag{2.6.10}$$

where:

$\boldsymbol{\Phi}_k$ = fundamental matrix (discrete $\boldsymbol{F}$)
T_s = sampling time of system
$\boldsymbol{I}$ = identity matrix

The discrete process-noise matrix, $\boldsymbol{Q}_k$, is calculated by using the following equation:

$$\boldsymbol{Q}_k = \boldsymbol{G}\boldsymbol{Q}\boldsymbol{G}^T T_s \tag{2.6.11}$$

where:

$\boldsymbol{G}$ = linearized process-noise coupling matrix (equation 2.6.8)

The discrete measurement noise matrix, $\boldsymbol{R}_k$, simply consists of each measurement noise source's variance.

Now, the discrete Riccati equations are calculated as follows:

$$\begin{aligned} \boldsymbol{M}_k &= \boldsymbol{\Phi}_k \boldsymbol{P}_{k-1} \boldsymbol{\Phi}_k^T + \boldsymbol{Q}_k & (2.6.12) \\ \boldsymbol{L}_k &= \boldsymbol{M}_k \boldsymbol{H}^T \left(\boldsymbol{H}\boldsymbol{M}_k \boldsymbol{H}^T + \boldsymbol{R}_k \right)^{-1} & (2.6.13) \\ \boldsymbol{P}_k &= \left(\boldsymbol{I} - \boldsymbol{L}_k \boldsymbol{H} \right) \boldsymbol{M}_k \left(\boldsymbol{I} - \boldsymbol{L}_k \boldsymbol{H} \right)^T + \boldsymbol{L}_k \boldsymbol{R}_k \boldsymbol{L}_k^T & (2.6.14) \end{aligned}$$

where:

$\boldsymbol{M}_k$ = covariance matrix representing errors in the state estimates before a measurement update
$\boldsymbol{L}_k$ = optimal EKF feedback gain
$\boldsymbol{P}_k$ = covariance matrix representing errors in the state estimates after a measurement update

Equation (2.6.14) can be simplified, but its current form (called the Joseph form) is less susceptible to round off errors and keeps $\boldsymbol{P}_k$ positive semidefinite [5].

The EKF process can now be summarised as follows:

- Perform state propagation:

$$\bar{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_{k-1} + \hat{\dot{\boldsymbol{x}}}_{k-1} T_s \tag{2.6.15}$$

where:

$\bar{\boldsymbol{x}}_k$ = new propagated system states
$\hat{\boldsymbol{x}}_{k-1}$ = estimated system states from previous time step
$\hat{\dot{\boldsymbol{x}}}_{k-1} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}_k) \mid_{\boldsymbol{x} = \hat{\boldsymbol{x}}_{k-1}}$

$\hat{\boldsymbol{x}}_{k-1}T_s$ = Euler integration of the non-linear system differential equation

Euler integration is effective if the sampling time, T_s, is small enough. Integration methods like Runge-Kutta can be used to improve accuracy.

- Calculate the matrices $\boldsymbol{F}$, $\boldsymbol{G}$ and $\boldsymbol{H}$ by utilizing equations (2.6.7) to (2.6.9).
- Calculate the discrete matrices $\boldsymbol{\Phi}_k$ and $\boldsymbol{Q}_k$ by using equations (2.6.10) to (2.6.11).
- Calculate the error covariance matrix, $\boldsymbol{M}_k$, before any update measurements are used by implementing equation (2.6.12).

When update measurements are available the following steps must also be executed:

- Calculate the EKF optimal feedback gain by utilizing equation (2.6.13).
- Update the noise figures by calculating the error covariance matrix, $\boldsymbol{P}_k$, using equation (2.6.12).
- Update the system states by implementing the following equation:

$$\hat{\boldsymbol{x}}_k = \bar{\boldsymbol{x}}_k + \boldsymbol{L}_k \left[\boldsymbol{y}_k - \boldsymbol{h}(\bar{\boldsymbol{x}}_k, \boldsymbol{u}_k)\right] \tag{2.6.16}$$

The EKF does not perform strictly optimally, since the error covariance matrix is only a linear approximation of the real error figures. There exist algorithms that track the error more accurately like the UKF (Unscented Kalman Filter), but a study from [23] shows that the UKF does not improve estimates in the case of quaternion motion. The UKF also requires more computations than the EKF.

2.6.2 MEKF

The MEKF is an estimation algorithm that estimates the four quaternion states and the three rate gyroscope bias states. Using quaternions to represent attitude eliminates singularity problems (sections 2.2.5 and 2.2.6) and causes the prediction equations to be treated linearly [20]. Quaternion dynamics are described as a function of the rate gyroscope inputs and serve as the plant model. The quaternion estimate is used to transform known reference vectors (e.g., the geomagnetic field or gravity) to vectors in body axes (equation 2.2.17). The difference between the transformed reference vectors and the measured body vectors is utilized to keep the quaternion error within bounds.

2.6.2.1 Quaternion vector perturbations

The error quaternion, $\delta \boldsymbol{q}$, is defined as the difference between the real, $\boldsymbol{q}$, and estimated, $\hat{\boldsymbol{q}}$, quaternion. Composing the error quaternion with the estimated quaternion forms the true quaternion [22] and can be expressed as [5] :

$$\boldsymbol{q} = \delta \boldsymbol{q} \otimes \hat{\boldsymbol{q}} \tag{2.6.17}$$

The error quaternion is very useful, since the unity constraint of the quaternion will be satisfied when quaternion multiplication is utilized.

The time derivative of equation (2.6.17) gives:

$$\dot{\boldsymbol{q}} = \delta \dot{\boldsymbol{q}} \otimes \hat{\boldsymbol{q}} + \delta \boldsymbol{q} \otimes \dot{\hat{\boldsymbol{q}}} \tag{2.6.18}$$

Equation (2.2.33) in terms of the real quaternion, $\boldsymbol{q}$, and real body angular rates, $\boldsymbol{\omega}$, gives:

$$f(\boldsymbol{x}, \boldsymbol{u}, t) = \dot{\boldsymbol{q}} = \frac{1}{2}\, \boldsymbol{\omega} \otimes \boldsymbol{q} \tag{2.6.19}$$

Equation (2.2.33) can also be expressed in terms of the estimated quaternion, $\hat{\boldsymbol{q}}$, and estimated angular body rates, $\hat{\boldsymbol{\omega}}$:

$$\dot{\hat{\boldsymbol{q}}} = \frac{1}{2}\, \hat{\boldsymbol{\omega}} \otimes \hat{\boldsymbol{q}} \tag{2.6.20}$$

Substituting equation (2.6.19) and equation (2.6.20) into equation (2.6.18) and rearranging the terms, the following can be obtained:

$$\delta \dot{\boldsymbol{q}} \otimes \hat{\boldsymbol{q}} = \frac{1}{2} \boldsymbol{\omega} \otimes \boldsymbol{q} - \delta \boldsymbol{q} \otimes \left(\frac{1}{2} \hat{\boldsymbol{\omega}} \otimes \hat{\boldsymbol{q}} \right) \tag{2.6.21}$$

Right multiplying equation (2.6.21) by the inverse of $\hat{\boldsymbol{q}}$, produces the following:

$$\delta\dot{\boldsymbol{q}} = \frac{1}{2}\left(\boldsymbol{\omega} \otimes \delta\boldsymbol{q} - \delta\boldsymbol{q} \otimes \hat{\boldsymbol{\omega}}\right) \tag{2.6.22}$$

The rate gyroscope measurements play an important role in the quaternion dynamics (equation 2.6.19). However, measurement noise and bias drift (sections 2.5.1 and 2.5.2) can corrupt the rate gyroscope measurements. Estimating the rate gyroscope bias results in more accurate attitude estimation [20], [22].

The real rate gyroscope measurements vector, $\boldsymbol{\omega}$, is defined as follows:

$$\boldsymbol{\omega} = \boldsymbol{u} - \boldsymbol{b} \tag{2.6.23}$$

where:

$\boldsymbol{u}$ = real rate gyroscope output vector
$\boldsymbol{b}$ = real rate gyroscope bias vector

The estimated rate gyroscope measurements vector, $\hat{\boldsymbol{\omega}}$, is defined as follows:

$$\hat{\boldsymbol{\omega}} = \boldsymbol{u} - \hat{\boldsymbol{b}} + \boldsymbol{\eta}_{gm} \tag{2.6.24}$$

where:

$\hat{\boldsymbol{b}}$ = estimated rate gyroscope bias vector
$\boldsymbol{\eta}_{gm}$ = rate gyroscope measurement noise vector

Now, define the body angular rates perturbation as:

$$\delta\boldsymbol{\omega} = \begin{bmatrix} \boldsymbol{\omega} - \hat{\boldsymbol{\omega}} \\ 0 \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{b}} - \boldsymbol{b} - \boldsymbol{\eta}_{gm} \\ 0 \end{bmatrix} = \begin{bmatrix} -\delta\boldsymbol{b} - \boldsymbol{\eta}_{gm} \\ 0 \end{bmatrix} \tag{2.6.25}$$

Equation (2.6.22) can now be rewritten by using equation (2.6.25):

$$\begin{aligned} \delta\dot{\boldsymbol{q}} &= \frac{1}{2}\left((\hat{\boldsymbol{\omega}} + \delta\boldsymbol{\omega}) \otimes \delta\boldsymbol{q} - \delta\boldsymbol{q} \otimes \hat{\boldsymbol{\omega}}\right) \\ &= \frac{1}{2}\left(\hat{\boldsymbol{\omega}} \otimes \delta\boldsymbol{q} - \delta\boldsymbol{q} \otimes \hat{\boldsymbol{\omega}}\right) + \frac{1}{2}\delta\boldsymbol{\omega} \otimes \delta\boldsymbol{q} \end{aligned} \tag{2.6.26}$$

Using the quaternion multiplication definition from equation (2.2.32) and the following quaternion multiplication definition,

$$\boldsymbol{a} \otimes \boldsymbol{b} = \begin{bmatrix} a_4 & a_3 & -a_2 & a_1 \\ -a_3 & a_4 & a_1 & a_2 \\ a_2 & -a_1 & a_4 & a_3 \\ -a_1 & -a_2 & -a_3 & a_4 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \tag{2.6.27}$$

equation (2.6.26) can be rewritten in matrix form as:

$$\begin{aligned} \begin{bmatrix} \delta\dot{q}_1 \\ \delta\dot{q}_2 \\ \delta\dot{q}_3 \\ \delta\dot{q}_4 \end{bmatrix} &= \begin{bmatrix} 0 & \hat{\omega}_3 & -\hat{\omega}_2 & 0 \\ -\hat{\omega}_3 & 0 & \hat{\omega}_1 & 0 \\ \hat{\omega}_2 & -\hat{\omega}_1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta q_1 \\ \delta q_2 \\ \delta q_3 \\ \delta q_4 \end{bmatrix} \\ &+ \frac{1}{2} \begin{bmatrix} 0 & \delta\omega_3 & -\delta\omega_2 & \delta\omega_1 \\ -\delta\omega_3 & 0 & \delta\omega_1 & \delta\omega_2 \\ \delta\omega_2 & -\delta\omega_1 & 0 & \delta\omega_3 \\ -\delta\omega_1 & -\delta\omega_2 & -\delta\omega_3 & 0 \end{bmatrix} \begin{bmatrix} \delta q_1 \\ \delta q_2 \\ \delta q_3 \\ \delta q_4 \end{bmatrix} \end{aligned} \tag{2.6.28}$$

Assuming that a small rotation will be implied by the error quaternion, δq_4 will be close to unity, and equation (2.6.28) can be simplified to:

$$\begin{bmatrix} \delta\dot{q}_1 \\ \delta\dot{q}_2 \\ \delta\dot{q}_3 \\ \delta\dot{q}_4 \end{bmatrix} = \begin{bmatrix} 0 & \hat{\omega}_3 & -\hat{\omega}_2 & 0 \\ -\hat{\omega}_3 & 0 & \hat{\omega}_1 & 0 \\ \hat{\omega}_2 & -\hat{\omega}_1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta q_1 \\ \delta q_2 \\ \delta q_3 \\ \delta q_4 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \delta\omega_1 \\ \delta\omega_2 \\ \delta\omega_3 \\ 0 \end{bmatrix} \tag{2.6.29}$$

Vector multiplication can be defined as:

$$\boldsymbol{a} \times \boldsymbol{b} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{2.6.30}$$

Now, using equation (2.6.25) and equation (2.6.30), equation (2.6.29) can be formulated as follows:

$$\delta\dot{\vec{\boldsymbol{q}}} = -\boldsymbol{\omega} \times \delta\vec{\boldsymbol{q}} + \frac{1}{2}\left(-\delta\boldsymbol{b} - \boldsymbol{\eta}_{gm}\right) \tag{2.6.31}$$

$$\delta\dot{q}_4 = 0 \tag{2.6.32}$$

where:

$$\delta\vec{\boldsymbol{q}} = [\,\delta q_1\, \delta q_2\, \delta q_3\,]^T$$

Equation (2.6.31) and equation (2.6.32) show that the dynamics of the system are contained in the vector part of the perturbation quaternion. The previous fact implies that only the vector perturbations need to be used.

2.6.2.2 Quaternion covariance matrix

Utilizing the quaternion vector perturbations will result in the MEKF calculating a 3×3 covariance matrix. However, a full 4×4 covariance matrix is needed to examine the noise statistics on all four quaternion elements.

Using equation (2.2.32), equation (2.6.17) can be rewritten in matrix form:

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} \hat{q}_4 & -\hat{q}_3 & \hat{q}_2 & \hat{q}_1 \\ \hat{q}_3 & \hat{q}_4 & -\hat{q}_1 & \hat{q}_2 \\ -\hat{q}_2 & \hat{q}_1 & \hat{q}_4 & \hat{q}_3 \\ -\hat{q}_1 & -\hat{q}_2 & -\hat{q}_3 & \hat{q}_4 \end{bmatrix} \begin{bmatrix} \delta q_1 \\ \delta q_2 \\ \delta q_3 \\ \delta q_4 \end{bmatrix} \tag{2.6.33}$$

Assuming that a small rotation will be implied by the error quaternion, δq_4 will be close to unity, and equation (2.6.33) can be simplified to:

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} \hat{q}_4 & -\hat{q}_3 & \hat{q}_2 \\ \hat{q}_3 & \hat{q}_4 & -\hat{q}_1 \\ -\hat{q}_2 & \hat{q}_1 & \hat{q}_4 \\ -\hat{q}_1 & -\hat{q}_2 & -\hat{q}_3 \end{bmatrix} \begin{bmatrix} \delta q_1 \\ \delta q_2 \\ \delta q_3 \end{bmatrix} + \begin{bmatrix} \hat{q}_1 \\ \hat{q}_2 \\ \hat{q}_3 \\ \hat{q}_4 \end{bmatrix} \tag{2.6.34}$$

Equation (2.6.34) clearly shows that all four quaternion elements can be determined when the quaternion vector perturbations are used. The full 4×4 quaternion covariance matrix can be calculated from the 3×3 quaternion vector perturbations covariance matrix by using the standard covariance transformation method [5] :

$$\boldsymbol{P}_{4\times 4} = \begin{bmatrix} \hat{q}_4 & -\hat{q}_3 & \hat{q}_2 \\ \hat{q}_3 & \hat{q}_4 & -\hat{q}_1 \\ -\hat{q}_2 & \hat{q}_1 & \hat{q}_4 \\ -\hat{q}_1 & -\hat{q}_2 & -\hat{q}_3 \end{bmatrix} \boldsymbol{P}_{3\times 3} \begin{bmatrix} \hat{q}_4 & \hat{q}_3 & -\hat{q}_2 & -\hat{q}_1 \\ -\hat{q}_3 & \hat{q}_4 & \hat{q}_1 & -\hat{q}_2 \\ \hat{q}_2 & -\hat{q}_1 & \hat{q}_4 & -\hat{q}_3 \end{bmatrix} \tag{2.6.35}$$

2.6.2.3 Bias drift

Section 2.5.2 shows that the bias dynamics of the rate gyroscope can be interpreted as integrated zero mean Gaussian white noise [20]. Band limiting the noise before integration, creates a good approximation of bias drift and is used throughout this project. The real bias vector is defined as follows [5] :

$$\dot{\boldsymbol{b}} = \boldsymbol{\eta}_{gb} \tag{2.6.36}$$

where:

$\boldsymbol{\eta}_{gb}$ = rate gyroscope bias noise vector

As stated by [5], there is no model of analysis for the bias vector in terms of some state or input. Thus, the estimated bias vector is defined as follows:

$$\dot{\hat{\boldsymbol{b}}} = 0 \tag{2.6.37}$$

Using equation (2.6.36) and equation (2.6.37) the bias perturbation (error) dynamics can be written as follows:

$$\begin{aligned} \delta\dot{\boldsymbol{b}} &= \dot{\boldsymbol{b}} - \dot{\hat{\boldsymbol{b}}} \\ &= \boldsymbol{\eta}_{gb} \end{aligned} \tag{2.6.38}$$

2.6.2.4 State space model

The combination of equation (2.6.31) and equation (2.6.38) forms the following continuous MEKF state space perturbation model:

$$\overbrace{\begin{bmatrix} \delta\dot{q}_1 \\ \delta\dot{q}_2 \\ \delta\dot{q}_3 \\ \delta\dot{b}_1 \\ \delta\dot{b}_2 \\ \delta\dot{b}_3 \end{bmatrix}}^{\dot{\boldsymbol{x}}} = \overbrace{\begin{bmatrix} 0 & \omega_3 & -\omega_2 & -0.5 & 0 & 0 \\ -\omega_3 & 0 & \omega_1 & 0 & -0.5 & 0 \\ \omega_2 & -\omega_1 & 0 & 0 & 0 & -0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}^{\boldsymbol{F}} \overbrace{\begin{bmatrix} \delta q_1 \\ \delta q_2 \\ \delta q_3 \\ \delta b_1 \\ \delta b_2 \\ \delta b_3 \end{bmatrix}}^{\boldsymbol{x}}$$

$$+\overbrace{\begin{bmatrix} -0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}^{\boldsymbol{G}} \overbrace{\begin{bmatrix} \eta_{gmx} \\ \eta_{gmy} \\ \eta_{gmz} \\ \eta_{gbx} \\ \eta_{gby} \\ \eta_{gbz} \end{bmatrix}}^{\boldsymbol{u}} \qquad (2.6.39)$$

The matrix, $\boldsymbol{F}$, has three rate gyroscope inputs, while the three rate gyroscope measurement and bias noise sources are coupled into the system by the matrix, $\boldsymbol{G}$.

In the event of a rotating Earth model the rate gyroscope measurements will be affected by the earth's rotation rate and the geographical platform velocity. The two effects must be taken into consideration in order to produce accurate angular rate measurements.

Define the turn rate of the earth in NED axes [4] :

$$\boldsymbol{\omega}_{earth} = \begin{bmatrix} \omega_{N_{earth}} \\ \omega_{E_{earth}} \\ \omega_{D_{earth}} \end{bmatrix} = \begin{bmatrix} \Omega \cos \lambda \\ 0 \\ -\Omega \sin \lambda \end{bmatrix} \qquad (2.6.40)$$

where:

$\boldsymbol{\omega}_{earth}$ = Earth's turn rate vector
Ω = Earth's rotation rate

Compensation must also be made for the transport rate (section 2.2.3) and is defined as follows [4] :

$$\boldsymbol{\omega}_{tr} = \begin{bmatrix} \omega_{N_{tr}} \\ \omega_{E_{tr}} \\ \omega_{D_{tr}} \end{bmatrix} = \begin{bmatrix} \dot{\varphi} \cos \lambda \\ -\dot{\lambda} \\ -\dot{\varphi} \sin \lambda \end{bmatrix} = \begin{bmatrix} \frac{V_E}{R+h} \\ \frac{-V_N}{R+h} \\ \frac{-V_E \tan \lambda}{R+h} \end{bmatrix} \qquad (2.6.41)$$

where:

$\boldsymbol{\omega}_{tr}$ = transport rate vector

Equation (2.6.40) and equation (2.6.41) are with reference to the NED axial system and need to be transformed to the body axial system, using the quaternion derived DCM, to allow the rate gyroscope measurements to be adjusted. The adjustments are made as follows:

$$\begin{aligned}\begin{bmatrix}\omega 1_{adj}\\ \omega 2_{adj}\\ \omega 3_{adj}\end{bmatrix} &= \begin{bmatrix}\omega 1\\ \omega 2\\ \omega 3\end{bmatrix} - \boldsymbol{T}\left[\boldsymbol{\omega}_{earth} + \boldsymbol{\omega}_{tr}\right]\\ &= \begin{bmatrix}\omega 1\\ \omega 2\\ \omega 3\end{bmatrix} - \boldsymbol{T}\begin{bmatrix}\Omega\cos\lambda + \frac{V_E}{R+h}\\ 0 - \frac{V_N}{R+h}\\ -\Omega\sin\lambda - \frac{V_E\tan\lambda}{R+h}\end{bmatrix}\end{aligned} \quad (2.6.42)$$

where:

$[\,\omega 1_{adj}\,\omega 2_{adj}\,\omega 3_{adj}\,]^T$ = adjusted rate gyroscope body measurements
$[\,\omega 1\,\omega 2\,\omega 3\,]^T$ = actual rate gyroscope body measurements

The adjusted rate gyroscope measurements can now be used as the inputs for the matrix, $\boldsymbol{F}$, of equation (2.6.39).

2.6.2.5 Measurement updates

After state propagation, the update measurement vectors need to be characterized as a function of the quaternion vector perturbations. The DCM in quaternion form (equation 2.2.17) can be written as a function of the perturbation quaternion and estimated quaternion [21], [22] :

$$\boldsymbol{T}(\boldsymbol{q}) = \boldsymbol{T}(\delta\boldsymbol{q}\otimes\hat{\boldsymbol{q}}) = \boldsymbol{T}(\delta\boldsymbol{q})\boldsymbol{T}(\hat{\boldsymbol{q}}) \quad (2.6.43)$$

Define the update measurement error (or innovation), $\boldsymbol{i}_m$, as follows [21] :

$$\begin{aligned}\boldsymbol{i}_m &= \boldsymbol{v}^{bm} - \boldsymbol{T}(\hat{\boldsymbol{q}})\boldsymbol{v}^{r}\\ &= \boldsymbol{T}(\delta\boldsymbol{q})\boldsymbol{T}(\hat{\boldsymbol{q}})\boldsymbol{v}^{rm} - \boldsymbol{T}(\hat{\boldsymbol{q}})\boldsymbol{v}^{r}\end{aligned} \quad (2.6.44)$$

where:

$\boldsymbol{v}^{bm}$ = body measurement vector
$\boldsymbol{v}^{rm}$ = body measurement vector transformed to a vector in reference axes
$\boldsymbol{v}^{r}$ = known reference vector (e.g. the direction of the earth, the sun, the geomagnetic field or gravity)

Rewriting equation (2.6.44) in terms of body axes gives:

$$\begin{aligned} \boldsymbol{i}_m &= \boldsymbol{T}(\delta\boldsymbol{q})\hat{\boldsymbol{v}}^{\,bm} - \hat{\boldsymbol{v}}^{\,b} \\ &\approx [\boldsymbol{T}(\delta\boldsymbol{q}) - \boldsymbol{I}]\hat{\boldsymbol{v}}^{\,b} \end{aligned} \tag{2.6.45}$$

where:

$\hat{\boldsymbol{v}}^{\,b}$ = known reference vector, $\boldsymbol{v}^{\,r}$, transformed to a vector in body axes

A small error quaternion results in the following approximation:

$$\boldsymbol{T}(\delta\boldsymbol{q}) \approx \begin{bmatrix} 1 & 2\delta q_3 & -2\delta q_2 \\ -2\delta q_3 & 1 & 2\delta q_1 \\ 2\delta q_2 & -2\delta q_1 & 1 \end{bmatrix} \tag{2.6.46}$$

Now, expanding the right-hand side of equation (2.6.45) gives:

$$\begin{aligned} [\boldsymbol{T}(\delta\boldsymbol{q}) - \boldsymbol{I}]\hat{\boldsymbol{v}}^{\,b} &= \begin{bmatrix} 0 & 2\delta q_3 & -2\delta q_2 \\ -2\delta q_3 & 0 & 2\delta q_1 \\ 2\delta q_2 & -2\delta q_1 & 0 \end{bmatrix} \begin{bmatrix} \hat{v}^b_x \\ \hat{v}^b_y \\ \hat{v}^b_z \end{bmatrix} \\ &= \begin{bmatrix} 2\delta q_3 \hat{v}^b_y - 2\delta q_2 \hat{v}^b_z \\ -2\delta q_3 \hat{v}^b_x + 2\delta q 1 \hat{v}^b_z \\ 2\delta q_2 \hat{v}^b_x - 2\delta q_1 \hat{v}^b_y \end{bmatrix} \\ &= 2 \begin{bmatrix} 0 & -\hat{v}^b_z & \hat{v}^b_y \\ \hat{v}^b_z & 0 & -\hat{v}^b_x \\ -\hat{v}^b_y & \hat{v}^b_x & 0 \end{bmatrix} \begin{bmatrix} \delta q_1 \\ \delta q_2 \\ \delta q_3 \end{bmatrix} \qquad (2.6.47) \\ &= 2 \begin{bmatrix} 0 & -\hat{v}^b_z & \hat{v}^b_y & 0 & 0 & 0 \\ \hat{v}^b_z & 0 & -\hat{v}^b_x & 0 & 0 & 0 \\ -\hat{v}^b_y & \hat{v}^b_x & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta q_1 \\ \delta q_2 \\ \delta q_3 \\ \delta b_1 \\ \delta b_2 \\ \delta b_3 \end{bmatrix} \end{aligned}$$

Thus, the measurement matrix, $\boldsymbol{H}$, can be defined as follows:

$$\boldsymbol{H} = 2 \begin{bmatrix} 0 & -\hat{v}^b_z & \hat{v}^b_y & 0 & 0 & 0 \\ \hat{v}^b_z & 0 & -\hat{v}^b_x & 0 & 0 & 0 \\ -\hat{v}^b_y & \hat{v}^b_x & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.6.48}$$

The measurement matrix, $\boldsymbol{H}$, is suitable for one vector update. In the event that two vector updates are available, two of the matrices in equation (2.6.48) can be combined to form a 6×6 measurement matrix.

2.6.3 Position and Velocity EKF

The PVEKF is an estimation algorithm that estimates the position and velocity of an object. Accelerometers measure the vehicle's static (gravity) and dynamic accelerations. Subtracting the static accelerations from the measurements allows the PVEKF to use the dynamic accelerations to determine the position and velocity of the vehicle. Integrating the dynamic accelerations produces velocity and integrating the velocity produces position. The double integrating system is highly affected by noise and bias drift, resulting in growing errors over time. However, the growing errors are kept within bounds by update measurements from other available sensors (e.g. GPS).

This project follows the strap down platform approach where sensors are "strapped down" to the body of the vehicle. A gimbaled platform can also be implemented where the sensors are kept in a fixed orientation with respect to the inertial axial system, independent of the body orientation. However, the strap down approach has more advantages than the gimbal approach [25], [9]. The accelerometer body measurements need to be transformed to measurements in NED axes, since the integration of the dynamic accelerations occurs in NED axes. The estimated quaternions from the MEKF (section 2.6.2) are used to calculate the quaternion DCM. The DCM is then used to perform the necessary axes transformations.

2.6.3.1 Non-linear dynamics

From section 2.2.3 the three non-linear position equations in terms of NED velocities are as follows:

$$\dot{\lambda} = \frac{V_N}{R+h} \tag{2.6.49}$$

$$\dot{\varphi} = \frac{V_E}{(R+h)\cos\lambda} \tag{2.6.50}$$

$$\dot{h} = -V_D \tag{2.6.51}$$

The three non-linear velocity equations are derived below [9] :

Define the accelerations in NED axes as follows:

$$\begin{bmatrix} \alpha_N \\ \alpha_E \\ \alpha_D \end{bmatrix} = \boldsymbol{T}^T \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \tag{2.6.52}$$

where:

$[\, \alpha_N \; \alpha_E \; \alpha_D \,]^T$ = accelerations in NED axes
$[\, a_x \; a_y \; a_z \,]^T$ = measured accelerations in body axes
$\boldsymbol{T}^T$ = quaternion defined inverse DCM

A rotating Earth model requires the earth's rotation rate and gravity anomalies to be taken into account in order to obtain high accuracy inertial navigation [26]. Coriolis accelerations need to be considered:

$$\begin{bmatrix} c_N \\ c_E \\ c_D \end{bmatrix} = \begin{bmatrix} -2V_E\,\Omega\sin\lambda + \frac{V_D V_N - V_E^2 \tan\lambda}{R+h} \\ 2V_N\,\Omega\sin\lambda + 2V_D\,\Omega\cos\lambda + \frac{V_D V_N \tan\lambda + V_E V_D}{R+h} \\ -2V_E\,\Omega\cos\lambda + \frac{-V_E^2 - V_N^2}{R+h} \end{bmatrix} \tag{2.6.53}$$

where:

$[\, c_N \; c_E \; c_D \,]^T$ = Coriolis accelerations
Ω = Earth's rotation rate

The earth's gravity and centripetal accelerations are also considered:

$$\begin{aligned} \begin{bmatrix} g'_N \\ g'_E \\ g'_D \end{bmatrix} &= \frac{\Omega^2(R+h)}{2} \begin{bmatrix} \sin 2\lambda \\ 0 \\ 1+\cos 2\lambda \end{bmatrix} + \begin{bmatrix} g_N \\ g_E \\ g_D \end{bmatrix} \\ &= \begin{bmatrix} -\Omega^2(R+h)\cos\lambda\sin\lambda \\ 0 \\ -\Omega^2(R+h)\cos^2\lambda + g_D \end{bmatrix} \end{aligned} \tag{2.6.54}$$

where:

$[\, g'_N \; g'_E \; g'_D \,]^T$ = Earth's gravity and centripetal acceleration
$[\, g_N \; g_E \; g_D \,]^T$ = Earth's gravity constant

It is assumed that there is no change in the earth's gravitational field with variations in the position of the navigation system. Thus $g_N = g_E = 0$ and $g_D = 9.81 m/s^2$.

Now, the equations for $\dot{V}_N$, $\dot{V}_E$ and $\dot{V}_D$ can be defined as follows:

$$\begin{aligned}
\begin{bmatrix} \dot{V}_N \\ \dot{V}_E \\ \dot{V}_D \end{bmatrix} &= \begin{bmatrix} \alpha_N \\ \alpha_E \\ \alpha_D \end{bmatrix} + \begin{bmatrix} c_N \\ c_E \\ c_D \end{bmatrix} + \begin{bmatrix} g'_N \\ g'_E \\ g'_D \end{bmatrix} \\
&= \boldsymbol{T}^T \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} + \begin{bmatrix} -2V_E\,\Omega \sin\lambda + \frac{V_D V_N - V_E^2 \tan\lambda}{R+h} \\ 2V_N\,\Omega\sin\lambda + 2V_D\,\Omega\cos\lambda + \frac{V_D V_N \tan\lambda + V_E V_D}{R+h} \\ -2V_E\,\Omega\cos\lambda + \frac{-V_E^2 - V_N^2}{R+h} \end{bmatrix} \\
&+ \begin{bmatrix} -\Omega^2(R+h)\cos\lambda\sin\lambda \\ 0 \\ -\Omega^2(R+h)\cos^2\lambda + g_D \end{bmatrix}
\end{aligned} \tag{2.6.55}$$

The non-linear system kinematic equations can now be written in the following format:

$$f(x,u,t) = \begin{bmatrix} \dot{\lambda} & \dot{\varphi} & \dot{h} & \dot{V}_N & \dot{V}_E & \dot{V}_D \end{bmatrix}^T \tag{2.6.56}$$

2.6.3.2 State space model

Using equation (2.6.7) and equation (2.6.8) together with equation (2.6.56) results in the following continuous linearized PVEKF state space model:

$$\overbrace{\begin{bmatrix} \dot{\lambda} \\ \dot{\varphi} \\ \dot{h} \\ \dot{V}_N \\ \dot{V}_E \\ \dot{V}_D \end{bmatrix}}^{\dot{\boldsymbol{x}}} = \overbrace{\begin{bmatrix}
\frac{\partial\dot{\lambda}}{\partial\lambda} & \frac{\partial\dot{\lambda}}{\partial\varphi} & \frac{\partial\dot{\lambda}}{\partial h} & \frac{\partial\dot{\lambda}}{\partial V_N} & \frac{\partial\dot{\lambda}}{\partial V_E} & \frac{\partial\dot{\lambda}}{\partial V_D} \\
\frac{\partial\dot{\varphi}}{\partial\lambda} & \frac{\partial\dot{\varphi}}{\partial\varphi} & \frac{\partial\dot{\varphi}}{\partial h} & \frac{\partial\dot{\varphi}}{\partial V_N} & \frac{\partial\dot{\varphi}}{\partial V_E} & \frac{\partial\dot{\varphi}}{\partial V_D} \\
\frac{\partial\dot{h}}{\partial\lambda} & \frac{\partial\dot{h}}{\partial\varphi} & \frac{\partial\dot{h}}{\partial h} & \frac{\partial\dot{h}}{\partial V_N} & \frac{\partial\dot{h}}{\partial V_E} & \frac{\partial\dot{h}}{\partial V_D} \\
\frac{\partial\dot{V}_N}{\partial\lambda} & \frac{\partial\dot{V}_N}{\partial\varphi} & \frac{\partial\dot{V}_N}{\partial h} & \frac{\partial\dot{V}_N}{\partial V_N} & \frac{\partial\dot{V}_N}{\partial V_E} & \frac{\partial\dot{V}_N}{\partial V_D} \\
\frac{\partial\dot{V}_E}{\partial\lambda} & \frac{\partial\dot{V}_E}{\partial\varphi} & \frac{\partial\dot{V}_E}{\partial h} & \frac{\partial\dot{V}_E}{\partial V_N} & \frac{\partial\dot{V}_E}{\partial V_E} & \frac{\partial\dot{V}_E}{\partial V_D} \\
\frac{\partial\dot{V}_D}{\partial\lambda} & \frac{\partial\dot{V}_D}{\partial\varphi} & \frac{\partial\dot{V}_D}{\partial h} & \frac{\partial\dot{V}_D}{\partial V_N} & \frac{\partial\dot{V}_D}{\partial V_E} & \frac{\partial\dot{V}_D}{\partial V_D}
\end{bmatrix}}^{\boldsymbol{F}} \overbrace{\begin{bmatrix} \lambda \\ \varphi \\ h \\ V_N \\ V_E \\ V_D \end{bmatrix}}^{\boldsymbol{x}}$$

$$+\overbrace{\begin{bmatrix} \frac{\partial\dot{\lambda}}{\partial a_x} & \frac{\partial\dot{\lambda}}{\partial a_y} & \frac{\partial\dot{\lambda}}{\partial a_z} & \frac{\partial\dot{\lambda}}{\partial q_1} & \frac{\partial\dot{\lambda}}{\partial q_2} & \frac{\partial\dot{\lambda}}{\partial q_3} & \frac{\partial\dot{\lambda}}{\partial q_4} \\ \frac{\partial\dot{\varphi}}{\partial a_x} & \frac{\partial\dot{\varphi}}{\partial a_y} & \frac{\partial\dot{\varphi}}{\partial a_z} & \frac{\partial\dot{\varphi}}{\partial q_1} & \frac{\partial\dot{\varphi}}{\partial q_2} & \frac{\partial\dot{\varphi}}{\partial q_3} & \frac{\partial\dot{\varphi}}{\partial q_4} \\ \frac{\partial\dot{h}}{\partial a_x} & \frac{\partial\dot{h}}{\partial a_y} & \frac{\partial\dot{h}}{\partial a_z} & \frac{\partial\dot{h}}{\partial q_1} & \frac{\partial\dot{h}}{\partial q_2} & \frac{\partial\dot{h}}{\partial q_3} & \frac{\partial\dot{h}}{\partial q_4} \\ \frac{\partial\dot{V}_N}{\partial a_x} & \frac{\partial\dot{V}_N}{\partial a_y} & \frac{\partial\dot{V}_N}{\partial a_z} & \frac{\partial\dot{V}_N}{\partial q_1} & \frac{\partial\dot{V}_N}{\partial q_2} & \frac{\partial\dot{V}_N}{\partial q_3} & \frac{\partial\dot{V}_N}{\partial q_4} \\ \frac{\partial\dot{V}_E}{\partial a_x} & \frac{\partial\dot{V}_E}{\partial a_y} & \frac{\partial\dot{V}_E}{\partial a_z} & \frac{\partial\dot{V}_E}{\partial q_1} & \frac{\partial\dot{V}_E}{\partial q_2} & \frac{\partial\dot{V}_E}{\partial q_3} & \frac{\partial\dot{V}_E}{\partial q_4} \\ \frac{\partial\dot{V}_D}{\partial a_x} & \frac{\partial\dot{V}_D}{\partial a_y} & \frac{\partial\dot{V}_D}{\partial a_z} & \frac{\partial\dot{V}_D}{\partial q_1} & \frac{\partial\dot{V}_D}{\partial q_2} & \frac{\partial\dot{V}_D}{\partial q_3} & \frac{\partial\dot{V}_D}{\partial q_4} \end{bmatrix}}^{\boldsymbol{G}} \overbrace{\begin{bmatrix} a_x \\ a_y \\ a_z \\ q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}}^{\boldsymbol{u}} \tag{2.6.57}$$

The six state PVEKF has seven inputs: three for the accelerometers and four for the quaternion (as estimated by the MEKF).

2.6.3.3 Measurement updates

A multitude of update sensors are utilized in this project. Each sensor has its own measurement matrix, $\boldsymbol{H}$, that couples the sensor data into the system through the six PVEKF states. The measurement matrix is calculated by using equation (2.6.9). Each sensor and its measurement matrix are discussed in more detail in chapter 4.

2.7 Summary

Chapter 2 discusses the relevant theory used in this project. Section 2.2 focuses on axial systems, Euler angles and quaternions, while the equations of motion and the TRIAD algorithm are discussed in sections 2.3 and 2.4 respectively. Section 2.5 focuses on the Allan variance and section 2.6 discusses the EKF, MEKF and PVEKF.

Chapter 3

Hardware Test Bed and Embedded Software

3.1 Overview

Chapter 3 focuses on the design and implementation of a sensor module that captures selected sensor data. The data is processed and sent to a PC (Personal Computer) for storage and analysis. Sensor data captured in real time serve as a test input for the navigation simulation software discussed in Chapter 4. Section 3.2 focuses on the hardware aspects of the sensor module, while section 3.3 discusses the embedded software developed for the module. Section 3.4 discusses the GUI (Graphical User Interface) developed for the sensor module. Sensor calibration is the focus of section 3.5 and finally, section 3.6 gives a brief summary of Chapter 3.

3.2 Hardware

The sensor module is divided into four main parts:

- A capturing and processing unit, called *NodeSense* (section 3.2.1).
- A 6-DOF IMU (Inertial Measurement Unit) (section 3.2.2).
- Other inertial sensors (section 3.2.3).
- A power supply unit (section 3.2.4).

3.2.1 NodeSense

The *NodeSense* unit has to fulfill the following requirements:

- A CPU (Central Processing Unit) is required to process the captured sensor data, execute the necessary communication protocols and maintain the timing of all the different processes. The CPU must have onboard A/D (Analog-to-Digital) converters, enough I/O (Input/Output) pins and must be able to perform digital signal processing.

- Communication transceiver units are necessary to implement the physical layer of the required communication protocols.

- High resolution A/D converters are required for certain analog sensors.

- Low-pass anti-aliasing filters are required for analog sensors to prevent the high frequency signals from corrupting the sampled data.

- Power distribution and filtering are required to supply each component on the *NodeSense* unit with the necessary voltage and current.

The fulfillment of the *NodeSense* requirements is discussed in the following sections.

3.2.1.1 Processor

Processors from different manufacturers, like *Analog*, *Microchip*, *ST*, *Atmel*, *Freescale* and *Renesas*, were compared to one another. The *dsPIC30F6014A* from *Microchip* is chosen as the most suitable CPU for the *NodeSense* unit due to its low-cost, availability and combination of a microcontroller and DSP (Digital Signal Processor) engine. The *dsPIC30F6014A* has also been successfully used in the ESL.

The *dsPIC30F6014A* is a low-power 16-bit high-performance digital signal controller capable of executing up to 30 MIPS (Million Instructions Per Second). Further features of the CPU include: 144 KB (kilobytes) of embedded program space, 8 KB of on-chip data RAM, interrupt sources, 16-bit timers and 12-bit A/D converters. UART (Universal Asynchronous Receiver Transmitter), SPI™(Serial Peripheral Interface), I^{2}C™(Inter-Integrated Circuit) and CAN (Controller Area Network) modules are also part of the CPU's peripheral features. The main features of the *dsPIC30F6014A* are summarized in the following table:

Pins	**Program Memory Bytes**	**SRAM Bytes**	**EEPROM Bytes**	**Timer 16-bit**	**ADC 12-bit 100ksps**
80	144K	8192	4096	5	16 ch
Max MIPS	**Voltage**	**UART**	**SPI**	**I^{2}C**	**CAN**
30	5V	2	2	1	2

Table 3.1: *dsPIC30F6014A* features

The CPU's DSP engine consists of a high-speed 17-bit $\times$ 17-bit multiplier, a barrel shifter and a 40-bit adder/subtracter (with two target accumulators, round and saturation logic). Digital filtering and other signal processing tasks can easily be implemented by making use of the DSP instruction set. The DSP engine performs fixed-point calculations, with software implemented floating-point calculations if necessary. A dedicated floating-point unit is preferred when intense floating-point calculations and high accuracy are required. However, only small DSP operations are required for this project, thus the fixed-point architecture is accepted as accurate enough. For a complete description of the *dsPIC30F6014A* and all its features see [27].

3.2.1.2 Communication hardware

To accommodate a variety of sensors and external devices, the following four communication protocols are selected:

- RS232
- SPI
- I^{2}C
- CAN

Each communication protocol requires hardware and software. The hardware component generates the correct signals and signal levels to enable communication between the devices using the protocol. The software implements the rules of the protocol and is discussed in section 3.3.2.

The *dsPIC30F6014A* already implements the physical layer of the SPI and I^2C protocol. However, external components are required for the RS232 and CAN protocol. The *MAX3238* from *Texas Instruments* is chosen for the RS232 protocol due to its low cost and availability. For a complete description of the *MAX3238* and all its features see [28]. The *SN65HVD1050* from *Texas Instruments* is chosen for the CAN protocol also due to its low cost and availability. For a complete description of the *SN65HVD1050* and all its features see [29].

3.2.1.3 A/D conversion

Many sensors have analog outputs and need to be digitized for CPU processing. An A/D converter is required to digitize analog sensor data at a fixed rate. The A/D resolution must first be calculated before an A/D converter can be selected and this is done by using the following equation [30] :

$$bits = \mathrm{ceil}\left[\log_2\left(\frac{V_{range}}{V_{noise}}\right)\right] \tag{3.2.1}$$

where:

$bits$ = A/D resolution required
V_{range} = range of the sensor voltage
V_{noise} = rms (root mean square) noise of sensor
ceil = function to round argument to nearest integer

Accelerometers, rate gyroscopes, a magnetometer and a pressure sensor are used in this project and are discussed in more detail in section 3.2.2 and section 3.2.3. All of the sensors produce analog outputs between 0V and 5V. Therefore, to facilitate the analog interface, the A/D resolution is distributed between the 0V - 5V range. The alternative is to make use of the full dynamic range of the A/D converter (assumed to be 5V) by scaling the sensor output. Scaling the sensor output would require more complicated analog circuitry, but allow for lower resolution A/D converters in some cases [30]. The sensors together with their characteristics and minimum A/D resolution (equation 3.2.1) are shown in table 3.2.

Sensor Type	**Model Number**	**V$_{min}$ [V]**	**V$_{max}$ [V]**	**V$_{noise}$ [mV]**	**Min A/D Resolution**	**True resolution with 16-bit A/D**
Accelerometer	LIS3L02AS4	0.33	2.97	0.33	13	15
Rate Gyroscope	ADXRS401	1.38	3.63	3	10	14
Absolute Pressure	MPX4115A	0.20	4.79	1	13	15

Table 3.2: Sensor characteristics with noise calculated at 100Hz bandwidth where required and A/D resolution unscaled over 5V.

Note that the magnetometer has been omitted from table 3.2. The magnetometer was given for use in this project, but there is no available data sheet specifying its noise statistics. However, a previous implementation of the magnetometer showed that 12-bit A/D converters are sufficient for digitizing the analog magnetometer signals [31]. Thus, the 12-bit A/D converters of the *dsPIC30F6014A* are used for the magnetometer output.

Table 3.2 shows that a 13-bit A/D converter satisfies the minimum requirement for all the analog sensors. However, quantization noise can be introduced by the A/D converter, so a 16-bit device is adopted as a better choice. The last column of table 3.2 shows that the true resolution achieved with a 16-bit A/D converter also satisfies the minimum A/D requirement. The *dsPIC30F6014A* only provides 12-bit A/D converters, creating the need for an external 16-bit A/D device. Simultaneous sampling A/D IC's (Integrated Circuit) with parallel data interfaces were considered, but proved to be slow, expensive and difficult to obtain. The *ADS8344* form *Texas Instruments* is a high speed, serial A/D converter with 8 channels, 16-bit resolution and a reference voltage input. Due to the *ADS8344*'s low cost, availability and attractive features, it is utilized in this project. For a complete description of the *ADS8344* and all its features see [32].

3.2.1.4 Analog anti-aliasing filters

As discussed in section 3.2.1.3, analog signals are digitized at a fixed sampling rate by an A/D converter. Using a sampling frequency, f_s, all frequency components below $f_s/2$ are reliably digitized. However, frequency components above $f_s/2$ will fold back into the $0 - f_s/2$ bandwidth section when digitized [33]. Thus, to correctly represent an analog signal with samples requires that the highest frequency components be less than $f_s/2$. The previous statement is known as Nyquist's Theorem and $f_s/2$ is known as the Nyquist frequency [35]. The aliasing or fold back phenomenon is depicted as follows:

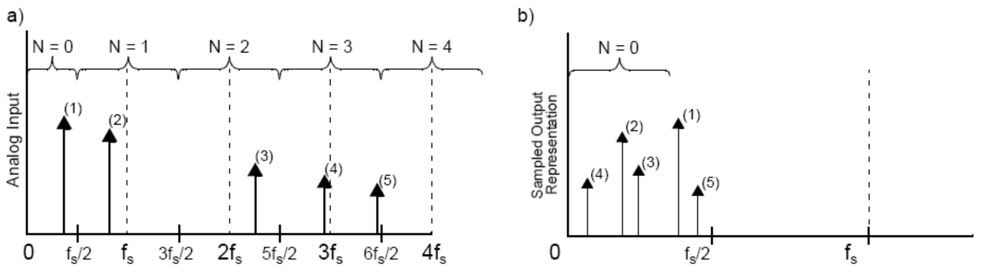


Figure 3.1: (a) Frequency components of analog signal, (b) Sampling at f_s, the frequency components below $f_s/2$ are reliably digitized while the frequency components above $f_s/2$ are folded back and appear as lower frequencies in the digital output [33]

The folded back frequencies in figure 3.1b, $f_{aliased}$, are calculated by using the information from figure 3.1a and the following equation [33] :

$$f_{aliases} = |f_{in} - N f_s| \tag{3.2.2}$$

where:

f_{in} = frequency of input signal component
N = segment where input signal component appears

The aliasing effect can be eliminated or considerably reduced by implementing an analog low-pass filter prior to the A/D converter [33]. The low-pass filter allows lower frequencies to pass through up to the cut-off frequency ($f_{cut-off} <= f_s/2$) and attenuates high frequency noise above the cut-off frequency, preventing undesirable aliased harmonic information from corrupting the digital output code. The effect of the low-pass anti-aliasing filter is illustrated in the following figure:

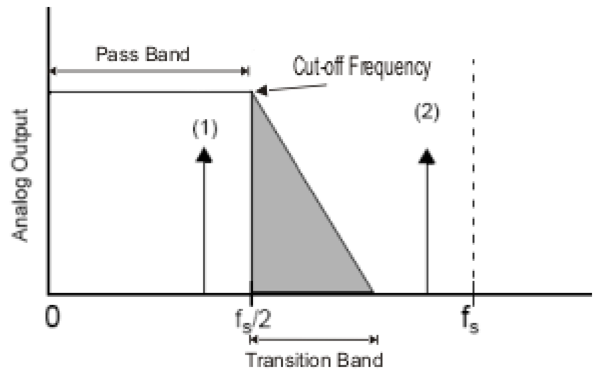


Figure 3.2: Effect of low-pass anti-alias filter [33]

Figure 3.2 shows how the low-pass filter attenuates the input signal component at frequency (2). The gray area in figure 3.2, known as the transition band, is generally determined by the amount of poles that are used to implement the filter design. An increase in filter poles will result in a decrease in the transition bandwidth and vice versa. Since the transition band in figure 3.2 is greater than $f_s/2$, signal components existing in this band will be aliased into the output of the A/D converter. By selecting a cut-off frequency smaller than $f_s/2$ or by increasing the order (add more poles) of the filter, the transition band aliasing effect is minimized. Increasing the order of the low-pass filter until the transition band is 0 may seem like a good approach, but practically, it may not be the best approach for an anti-aliasing solution. Every two filter poles require a minimum of one OPAMP (Operational Amplifier), two capacitors and two resistors. High order filters will take up too much PCB (Printed Circuit Board) space and additionally, each OPAMP will also contribute its own offset and noise errors into the pass band region. Selecting a smaller cut-off frequency is the adopted strategy for this project.

To determine the amount of attenuation necessary for a specific A/D converter, the following equation can be used [34] :

$$Att\ [dB] = 6n \tag{3.2.3}$$

where:

n = A/D resolution in bits

Equation (3.2.3) indicates the amount of attenuation needed to make the amplitude of any component above $f_s/2$ smaller than the A/D converter's LSB (Least Significant Bit). In practice, noise-signal amplitudes almost never equal the amplitudes of signal components of interest [34], so equation (3.2.3) represents worst case.

The analog low-pass anti-aliasing filters for this project are realized by making use of active, second-order Butterworth filters. Active filters make use of OPAMPs which provide excellent isolation between stages. The isolation is possible due to the OPAMP's high input impedance and low output impedance. The Butterworth filter provides the flattest passband response [36] and a low component count.

The Sallen-Key topology provides better passband unity gain than the multiple feedback topology [34] and is utilized in this project. Figure 3.3 illustrates an active second-order Sallen-Key Butterworth filter.

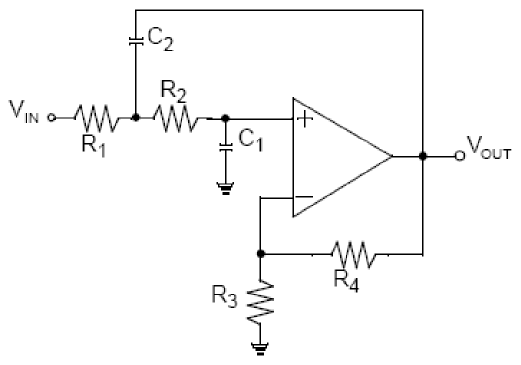


Figure 3.3: Active second-order Sallen-Key Butterworth filter [33]

$$f_{cut-off} = \frac{1}{2\pi\sqrt{R_1R_2C_1C_2}} \tag{3.2.4}$$

$$K = 1 + \frac{R_4}{R_3} \tag{3.2.5}$$

$$\frac{V_{out}}{V_{in}}(s) = \frac{K}{s^2C_1C_2R_1R_2 + s[C_1(R_1+R_2) + (1-K)C_2R_1] + 1} \tag{3.2.6}$$

The sensors are sampled at 1000 Hz [30], implying that $f_{cut-off}$ must be below 500 Hz. To make component selection easy and to achieve good attenuation at the Nyquist frequency, a value of $f_{cut-off} \approx 95$ Hz is chosen. R_3 is open and R_4 is shorted, resulting in unity gain, K. The component values are as follows:

$$\begin{aligned} R_1 &= 4.7\,\text{k}\Omega \\ R_2 &= 18\,\text{k}\Omega \\ C_1 &= 100\,\text{nF} \\ C_2 &= 330\,\text{nF} \end{aligned}$$

$$\begin{aligned} R_3 &= \infty\,\Omega \\ R_4 &= 0\,\Omega \end{aligned}$$

The *OPA4350* from *Texas Instruments* is a rail-to-rail CMOS (Complementary Metal Oxide Semiconductor) operational amplifier optimized for low voltage, single supply operation. Rail-to-rail input/output, low noise (5nV/$\sqrt{\text{Hz}}$), high speed operation (38 MHz, 22V/μs), low cost and availability make the *OPA4350* ideal for driving sampling A/D converters [37] and it is used in this project.

Analog low-pass filtering is only part of the data acquisition signal chain and is illustrated in figure 3.4.

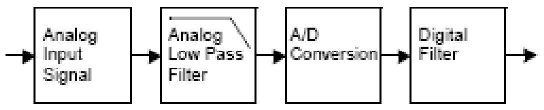


Figure 3.4: Data acquisition signal chain [33]

Noise can be injected during the A/D conversion process, making it necessary to design and implement digital filters. Digital filters are programmable and enable the user to program the cut-off frequency and output data rates. Section 3.3.1 discusses the digital filter design and illustrates the combined response of the analog and digital filters.

3.2.1.5 Power distribution

The *NodeSense* unit receives +12V, -12V and +5V from the separate power board (section 3.2.4). Analog components (eg. sensors) are powered by the +12V and -12V supplies, while the digital components (eg. CPU) are powered by the +5V supply. Decoupling capacitors provide adequate filtering for the digital power supply. However, analog electronics are very sensitive to unwanted high frequencies and power supply ripple, making it necessary to design and implement additional analog power supply filtering.

A LC filter design is selected for the analog power supply filtering, since it reduces output ripple and high frequency components [38]. The LC filter also provides a low component count, resulting in easy implementation. Figure 3.5 depicts a basic LC filter design.

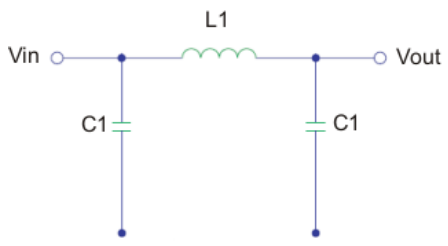


Figure 3.5: LC filter

The following equation is used to calculate the required values of the filter components in order to achieve the desired damping ratio [38] :

$$Min\ required\ damping\ of\ LC\ filter = \frac{V_{max\ ripple}}{V_{Oac}} = \frac{X_C}{X_C + X_L} \tag{3.2.7}$$

where:

$V_{max\ ripple}$ = maximum allowed output ripple of LC filter
V_{Oac} = listed output ripple of DC/DC converter (section 3.2.4)

$X_C = \frac{1}{2\pi f C}[\Omega]$
$X_L = 2\pi f L[\Omega]$
f = switching frequency of the DC/DC converter
C = capacitance
L = inductance

Using [39], the following values are known:

$$\begin{aligned} V_{max\,ripple} &= 10\,\mathrm{m}V \\ V_{Oac} &= 75\,\mathrm{m}V \\ f &= 300\,\mathrm{kHz} \end{aligned}$$

By using equation (3.2.7) the minimum required damping can be calculated as follows:

$$Min\,required\,damping\,of\,LC\,filter = \frac{10}{75} = 0.133 \tag{3.2.8}$$

A convenient capacitance and inductance of 1 μF and 1 mH respectively satisfies the minimum damping requirement.

Not all of the analog components of the *NodeSense* unit require a +12V or -12V supply. An analog +5V supply as well as a +5V reference voltage are required. The A/D converters require a very stable reference and make use of the +5V reference voltage.

The *REG104* from *Texas Instruments* is a low-noise, low-dropout linear regulator capable of providing a regulated +5V supply from a +12V input [40]. The *MAX6350* from *Maxim* is a low-noise, precision voltage reference with extremely low temperature coefficients and excellent accuracy, capable of providing a +5V reference voltage from a +12V input [41]. Both the *REG104* and the *MAX6350* are used in this project, due to their low cost, accuracy and availability.

A summary of the *NodeSense* unit can now be shown in the following functional block diagram:

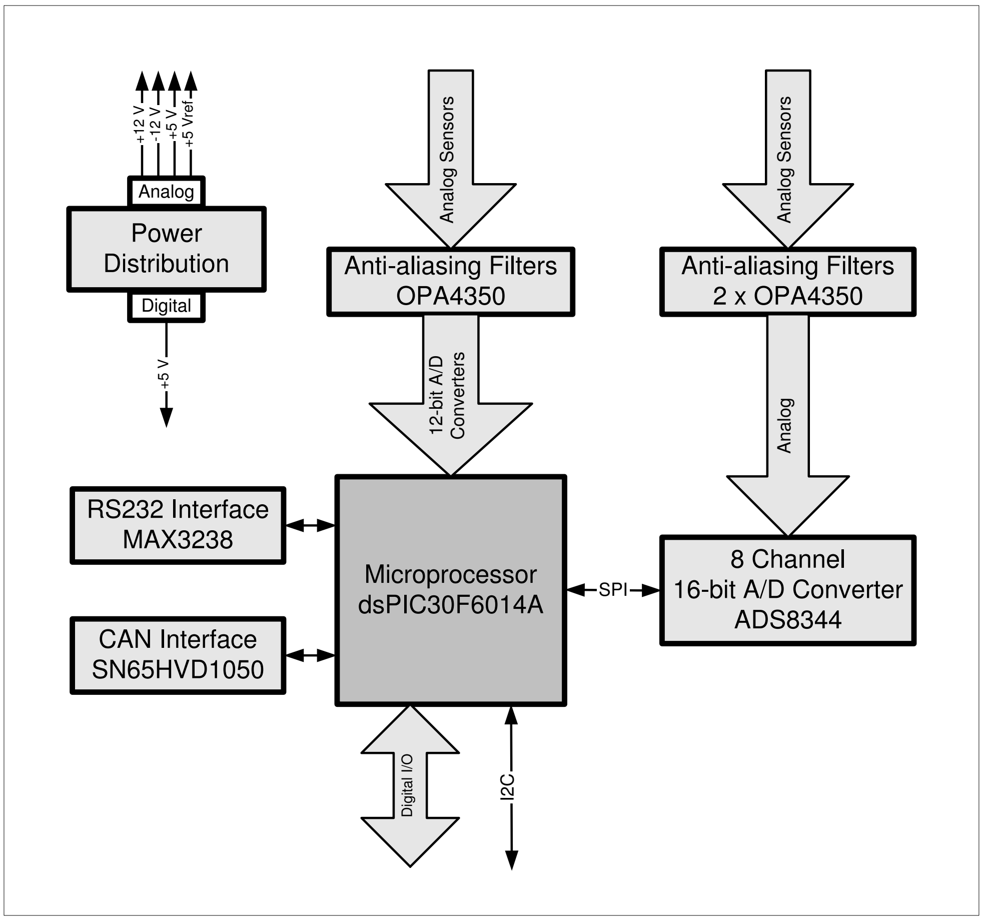

+12 V
-12 V
+5 V
+5 Vref
Analog
Power Distribution
Digital
+5 V
Analog Sensors
Anti-aliasing Filters OPA4350
12-bit A/D Converters
Analog Sensors
Anti-aliasing Filters 2 x OPA4350
Analog
RS232 Interface MAX3238
CAN Interface SN65HVD1050
Microprocessor dsPIC30F6014A
SPI
8 Channel 16-bit A/D Converter ADS8344
Digital I/O
I2C

Figure 3.6: *NodeSense* functional block diagram

3.2.2 IMU

The rate gyroscopes and accelerometers form part of the IMU. Rate gyroscopes provide angular rate data, while accelerometers provide static and dynamic acceleration data. A triad of rate gyroscopes and accelerometers is needed to create a 6-DOF IMU.

The *ADXRS401* MEMS rate gyroscope from *Analog Devices* has a dynamic range of $\pm 75^\circ/s$ and is chosen for this project, due to its low cost, small size and availability. It is also used by [5], [6] and [30] in the ESL. The single rate output of the *ADXRS401* is a voltage proportional to the angular rate about the axis normal to the top surface of the sensor package [42]. A precision reference and a temperature output are also provided for compensation techniques. *Analog*'s *ADXRS401* measures the angular rate in one axis only, making it necessary to purchase three units and mount them orthogonal with respect to each other.

The *LIS3L02AS4* from *STMicroelectronics* is a low-power three-axes MEMS linear accelerometer with a dynamic range of $\pm 2g/\pm 6g$ [43]. It is used in this project to test its capabilities as a low-cost three-axes accelerometer, since it has not been used in the ESL before. Each output of the *LIS3L02AS4* is a voltage proportional to the measured static and dynamic accelerations in the specific axis.

3.2.3 Magnetometer and pressure sensor

The *FG-D2* from the *HMO (Hermanus Magnetic Observatory)* is a three-axes magnetometer capable of measuring the earth's magnetic field vector. The magnetometer is given for use in this project. Each output of the *FG-D2* is a voltage proportional to the measured magnetic field in the specific axis. Unfortunately, no data sheet could be found for the *FG-D2*. Only basic output information is available.

The *MPX4115A* from *Freescale Semiconductor* is a monolithic, signal conditioned, silicon pressure sensor capable of measuring absolute pressures from 15 kPa to 115 kPa [44]. It is used in this project, due to its low cost, small size and availability. It is also utilized by [6] and [30] in the ESL. The output of the *MPX4115A* is a voltage proportional to the applied pressure.

3.2.4 Power board

The main power supply for an AUV is simulated in this project by using two +12V batteries connected in series, creating a +24V supply. The *NodeSense* unit requires +12V, -12V and +5V to operate correctly (see section 3.2.1.5), making it necessary to use DC-DC converters in order to convert the +24V to +12V, -12V and +5V. The *TEL5-2411* and *TEL-2422* from *Traco Power* are high power density DC-DC converters with high efficiency, regulated outputs and a wide input range of 18V - 36V [39]. Both the *TEL5-2411* and *TEL-2422* are used in this project, due to their excellent cost/performance ratio and availability. The *TEL5-2411* has an output voltage of +5V, while the *TEL5-2422* has an output voltage of ±12V.

The DC-DC converters are mounted on a separate PCB, due to their rather large packaging. A separate PCB architecture will also simplify the implementation of future power supply requirement changes.

A summary of the power supply PCB can now be shown in the following functional block diagram:

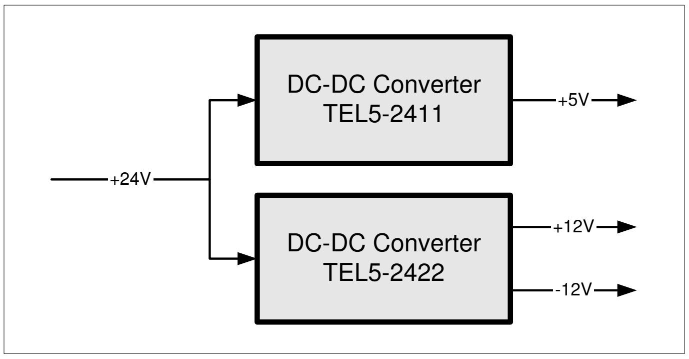


Figure 3.7: Main power supply functional block diagram

The complete *NodeSense* unit, inertial sensors and main power supply unit are illustrated as follows:

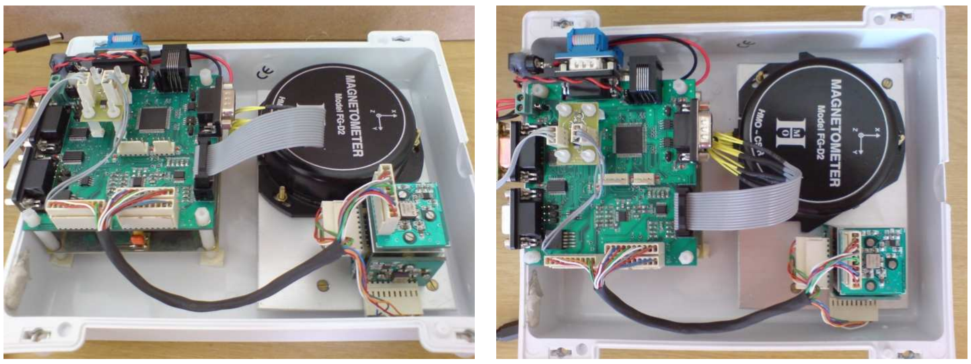
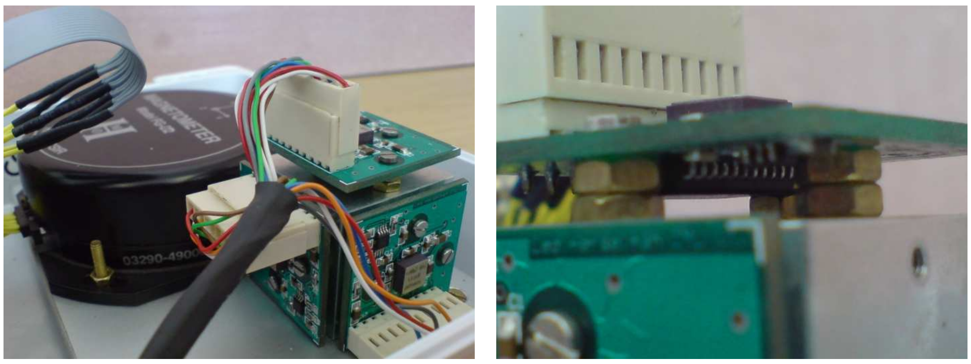
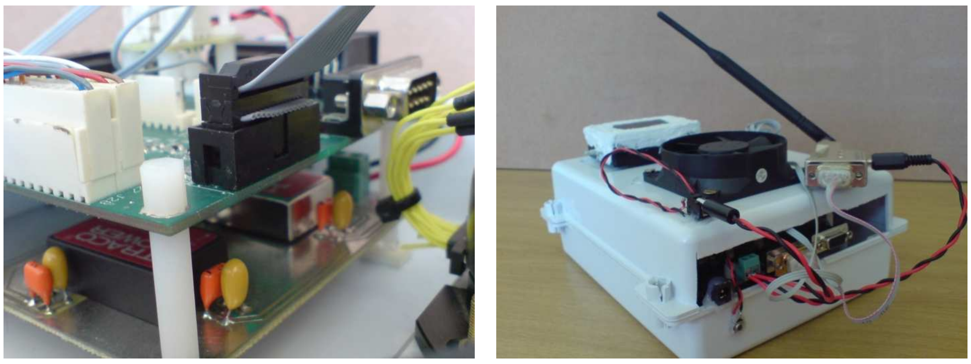


Figure 3.8: Top views of the hardware test bed



(a) (b)

Figure 3.9: (a) Black magnetometer and IMU cube, (b) Accelerometer mounted underneath rate gyroscope



(a) (b)

Figure 3.10: (a) Main power supply PCB underneath *NodeSense* PCB, (b) Enclosed hardware test bed with wireless RS232 transceiver

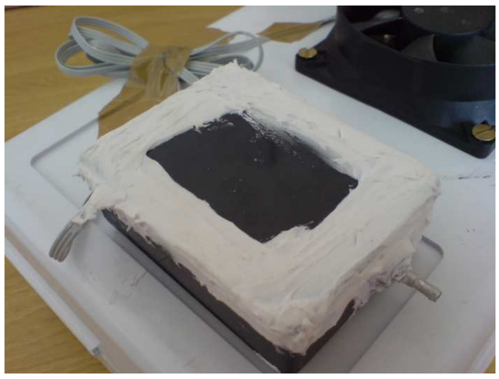


Figure 3.11: Enclosed pressure sensor

3.3 Embedded software

The embedded software for the *dsPIC30F6014A* is divided into three main parts:

- The digital filtering of the sampled sensor data (section 3.3.1).
- The implementation of the necessary communication protocols (section 3.3.2).
- The timing of all the different processes (section 3.3.3).

3.3.1 Digital filtering

Digital filters are used to remove noise injected during the A/D conversion process [33]. There are two types of digital filters:

- FIR (Finite Impulse Response) filters
- IIR (Infinite Impulse Response) filters

FIR filters utilize past inputs, while IIR filters utilize past inputs and outputs. The IIR filter is chosen for this project, since it is characterized by a significantly lower order than the corresponding FIR filter [45]. A lower order filter executes faster and also uses less processing resources. Another advantage of the IIR filter is the fact that an analog filter transfer function can be designed and converted to a transfer function in the digital domain.

An IIR second order Butterworth filter is chosen for this project, due to its maximally flat passband response. The Butterworth filter also has minimal phase shift over the passband region when compared to other conventional filters [45]. The Butterworth digital filter can be written in the following form [30]:

$$a_1 y(n) = b_1 x(n) + b_2 x(n-1) + b_3 x(n-2) - a_2 y(n-1) - a_3 y(n-2) \tag{3.3.1}$$

where:

$a_{1..3}, b_{1..3}$ = filter coefficients
x = A/D samples
y = filter output values

Equation (3.3.1) shows that the digital filter is the sum of product terms. The *dsPIC30F6014A* can multiply two 16-bit values and store the 32-bit result in one of its two 40-bit accumulators. Addition and subtraction operations, performed on the accumulators and registers, are also supported.

The *dsPIC30F6014A* microprocessor does not have any hardware to handle floating-point mathematics, therefore any such operations must be emulated with integer math. The sampled A/D values are 16-bit integers and can be represented as follows:

$$\begin{array}{rcl} \textit{integer bits} & . & \textit{decimal bits} \\ 16 & . & 0 \end{array}$$

The filter coefficients for this project are 16-bits wide and in the range of 0 to 2 (see table 3.3). Multiplication or division by 2 on a microprocessor is a fast operation, since the bits are simply shifted left or right. Thus, normalizing the coefficients to be in the range of 0 to 1 will allow the coefficients to be represented as follows:

$$\begin{array}{rcl} \textit{integer bits} & . & \textit{decimal bits} \\ 0 & . & 16 \end{array}$$

Now, the multiplication of an A/D sample and its coefficient can be written as follows:

$$\begin{array}{rc} A/D\ sample & 16.0 \\ \times Filter\ coefficient & 0.16 \\ & -\,-\,-\,-\,- \\ & 16.16 \\ & -\,-\,-\,-\,- \end{array} \tag{3.3.2}$$

The result of equation (3.3.2) is a 32-bit fixed-point value where the integer part is 16 bits wide and the decimal part is also 16 bits wide. Addition and subtraction of 32-bit values will also result in 32-bit values.

Now, the multiplication of a filter output value and its coefficient can be written as follows:

$$\begin{array}{rc}
Filter\,output & 16.16 \\
\times Filter\,coefficient & 0.16 \\
 & -\,-\,-\,-\,- \\
 & 16.32 \\
 & -\,-\,-\,-\,-
\end{array} \tag{3.3.3}$$

The result of equation (3.3.3) is a 48-bit fixed-point value. However, the 32-bit decimal value exceeds 16 bits, making it necessary to discard the lower 16 bits. Discarding the lower 16 bits creates a 32-bit fixed-point value without any significant loss in accuracy.

The digital filter reduces A/D noise. However, the noise can be reduced even more by making use of oversampling. Sampling and filtering the analog sensor data at a high rate, f_{high}, and taking the average value every N^{th} sample as the sensor module output, creates a $log_2(N)$ times reduction in the A/D noise [30]. The high rate, f_{high}, for this project is 1000 Hz (section 3.2.1.4) and the sensor module outputs data at a rate of 50 Hz. The selected output rates have also been used by [5], [6] and [30] in the ESL. A high sampling rate of 1000 Hz and an output rate of 50 Hz creates a ($log_2(20) = 4.32$) times noise reduction, more than adequate for most applications [30]. The 50 Hz output rate makes it necessary to design the digital filter to provide enough attenuation at 25 Hz (Nyquist frequency) in order to prevent secondary aliasing. A bandwidth of 8 Hz is selected to reduce the computational complexity of the digital filter, while still providing good attenuation at 25 Hz.

The digital filter coefficients are calculated by using the *Matlab*™ function, *butter()*. Since only 16 bits are used for the coefficient representation, there will be an error between a calculated filter coefficient and its 16-bit representation. The calculated digital filter coefficients, their 16-bit representation and the percentage quantization error are listed as follows (coefficient a_1, equation 3.3.1, is normalized to 1) :

	Calculated	**16-bit**	**Error [%]**
b_1	0.00060985471872	0.00061035156250	0.081469
b_2	0.00121970943743	0.00122070312500	0.081469
b_3	0.00060985471872	0.00061035156250	0.081469
a_2	-1.92894226325203	-1.92895507812500	0.000664
a_3	0.93138168212690	0.93139648437500	0.001589

Table 3.3: Filter coefficient quantization

The following figures illustrate the analog (section 3.2.1.4), digital and combined filter response as well as the effect of the quantization error:

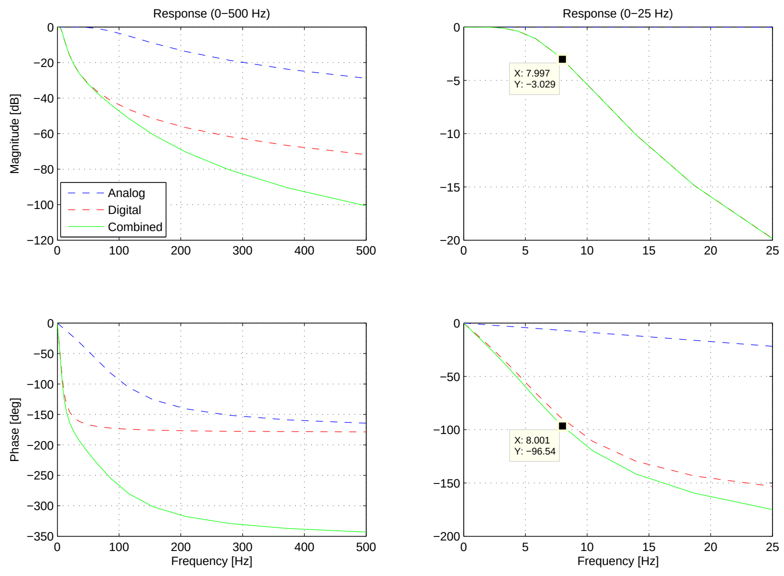


Figure 3.12: Magnitude and phase response of analog, digital and combined filter design

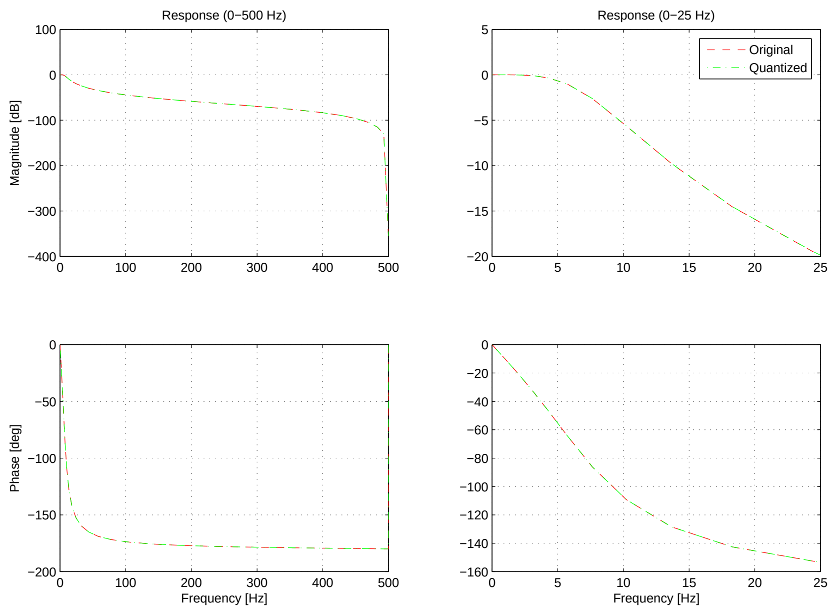


Figure 3.13: Magnitude and phase response of original and quantized digital filter

Figure 3.12 shows that the filter designs provide satisfactory attenuation. The phase of the analog anti-aliasing filter also has little effect on the output within the system bandwidth. Figure 3.13 shows that the effect of quantization error is negligible.

The digital filter algorithm implemented on the *dsPIC30F6014A* can be divided into three main parts:

1. For each coefficient-sample pair ($[b_1,\ x(n)]$, $[b_2,\ x(n-1)]$, $[b_3, x(n-2)]$) the following is done:
   - Load the normalized coefficient and sensor A/D sample into 16-bit registers.
   - Multiply the two registers and add the product to accumulator A (40-bit register).
2. For each coefficient-filter output pair ($[a_2,\ y(n-1)]$, $[a_3,\ y(n-2)]$) the following is done:

- Load the normalized coefficient and higher 16 bits of the 32-bit filter output into 16-bit registers.
- Multiply the two registers and add the product to accumulator A.
- Load the lower 16 bits of the filter output into a register.
- Multiply the lower 16 bits register with its appropriate coefficient and store the result in accumulator B (40-bit register).
- Shift accumulator B 16 bits to the right and add/subtract the result to/from accumulator A.

3. Since all the coefficients are normalized, correct the final result in accumulator A by multiplying it by two. Check for overflow errors and update the necessary states.

The filter implementation is shown in the following functional block diagram:

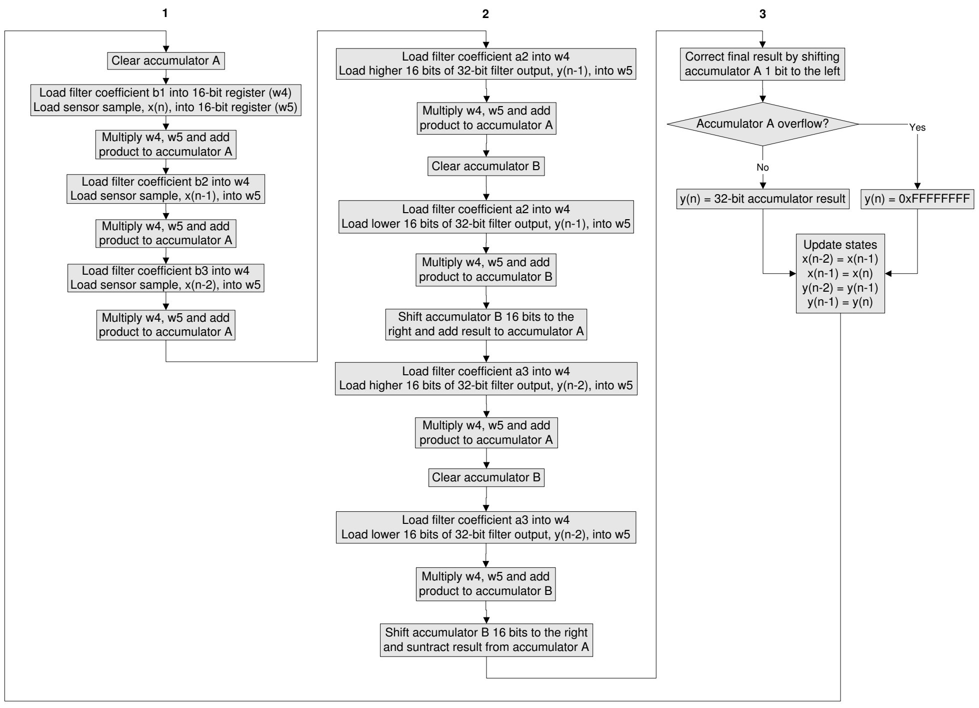

1
Clear accumulator A
Load filter coefficient b1 into 16-bit register (w4)
Load sensor sample, x(n), into 16-bit register (w5)
Multiply w4, w5 and add product to accumulator A
Load filter coefficient b2 into w4
Load sensor sample, x(n-1), into w5
Multiply w4, w5 and add product to accumulator A
Load filter coefficient b3 into w4
Load sensor sample, x(n-2), into w5
Multiply w4, w5 and add product to accumulator A
2
Load filter coefficient a2 into w4
Load higher 16 bits of 32-bit filter output, y(n-1), into w5
Multiply w4, w5 and add product to accumulator A
Clear accumulator B
Load filter coefficient a2 into w4
Load lower 16 bits of 32-bit filter output, y(n-1), into w5
Multiply w4, w5 and add product to accumulator B
Shift accumulator B 16 bits to the right and add result to accumulator A
Load filter coefficient a3 into w4
Load higher 16 bits of 32-bit filter output, y(n-2), into w5
Multiply w4, w5 and add product to accumulator A
Clear accumulator B
Load filter coefficient a3 into w4
Load lower 16 bits of 32-bit filter output, y(n-2), into w5
Multiply w4, w5 and add product to accumulator B
Shift accumulator B 16 bits to the right and suntract result from accumulator A
3
Correct final result by shifting accumulator A 1 bit to the left
Accumulator A overflow?
Yes
No
y(n) = 32-bit accumulator result
y(n) = 0xFFFFFFFF
Update states
x(n-2) = x(n-1)
x(n-1) = x(n)
y(n-2) = y(n-1)
y(n-1) = y(n)

Figure 3.14: Digital filter implementation on *dsPIC30F6014*

3.3.2 Communication protocols

As discussed in section 3.2.1.2, the physical layer of RS232, SPI, I^{2}C and CAN communication are supported by the sensor module. However, only the RS232 and SPI protocols are implemented on the *dsPIC30F6014*. The RS232 protocol is used to enable communication between the sensor module and a PC, while the SPI protocol is used to enable communication between the *dsPIC30F6014* and the A/D converter. Although the I^{2}C and CAN protocols are not implemented at this stage, it can easily be done when future implementations create the need for the sensor module to interface with I^{2}C or CAN sensors/devices. The RS232 and SPI protocol implementations are discussed in the following sections.

3.3.2.1 RS232

The RS232 implementation can be divided into three main parts:

1. Initialize the UART by loading the correct values into the required registers (see [27]).

2. To send a string of bytes the following is done:

   - Load the bytes that need to be transmitted into a user-defined transmit buffer.
   - Load the first byte from the transmit buffer into the UART's 8-bit transmit buffer.
   - The first byte is automatically sent and the transmit ISR (Interrupt Service Routine) is called.
   - Customize the ISR to clear the transmit flag and load the next byte into the UART's transmit buffer.
   - The ISR will now automatically send the rest of the bytes contained in the user-defined buffer.

3. When bytes are received, the following is done:

   - Each byte received is stored in the UART's 8-bit receive buffer.
   - Create an ISR to clear the receive flag and store the received byte in a user-defined buffer.

- The ISR will now automatically store all the received bytes in the user-defined buffer.

The RS232 implementation is shown in the following functional block diagram:

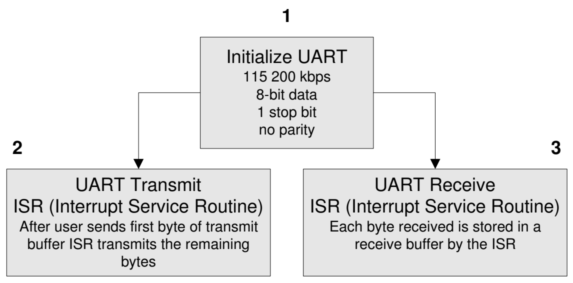

1
Initialize UART
115 200 kbps
8-bit data
1 stop bit
no parity
2
UART Transmit
ISR (Interrupt Service Routine)
After user sends first byte of transmit buffer ISR transmits the remaining bytes
3
UART Receive
ISR (Interrupt Service Routine)
Each byte received is stored in a receive buffer by the ISR

Figure 3.15: RS232 implementation

3.3.2.2 SPI

The SPI implementation can be divided into two main parts:

1. Initialize the SPI module by loading the correct values into the required registers (see [27]).
2. To send/store a string of bytes the following is done:
   - Load the bytes that need to be transmitted into a user-defined transmit buffer.
   - Load the first byte from the transmit buffer into the SPI module's transmit/receive buffer.
   - The byte is automatically transmitted.
   - The SPI module immediately executes a receive, stores the received byte in the transmit/receive buffer and calls the ISR.
   - Customize the ISR to clear the interrupt flag, store the received byte in a user-defined buffer and load the next byte to be transmitted into the SPI module's transmit/receive buffer.
   - The ISR will now automatically send the rest of the bytes that need to be transmitted and save the received byte after each send.

The SPI implementation is shown in the following functional block diagram:

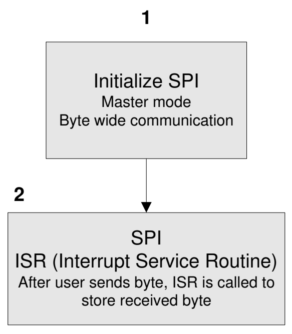

1
Initialize SPI
Master mode
Byte wide communication
2
SPI
ISR (Interrupt Service Routine)
After user sends byte, ISR is called to store received byte

Figure 3.16: SPI implementation

3.3.3 Timing

The main routine of the *dsPIC30F6014A*'s embedded software is illustrated in the following functional block diagram:

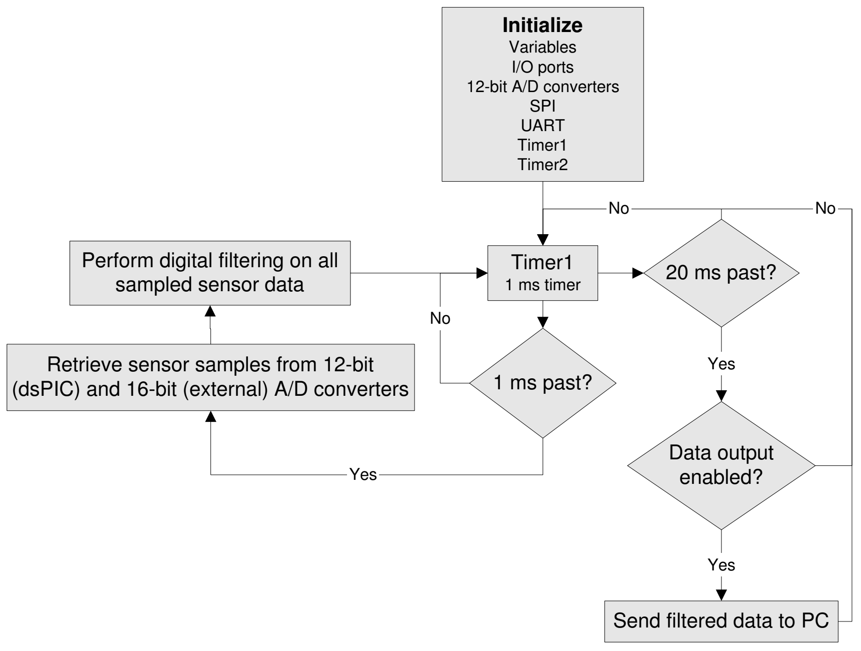


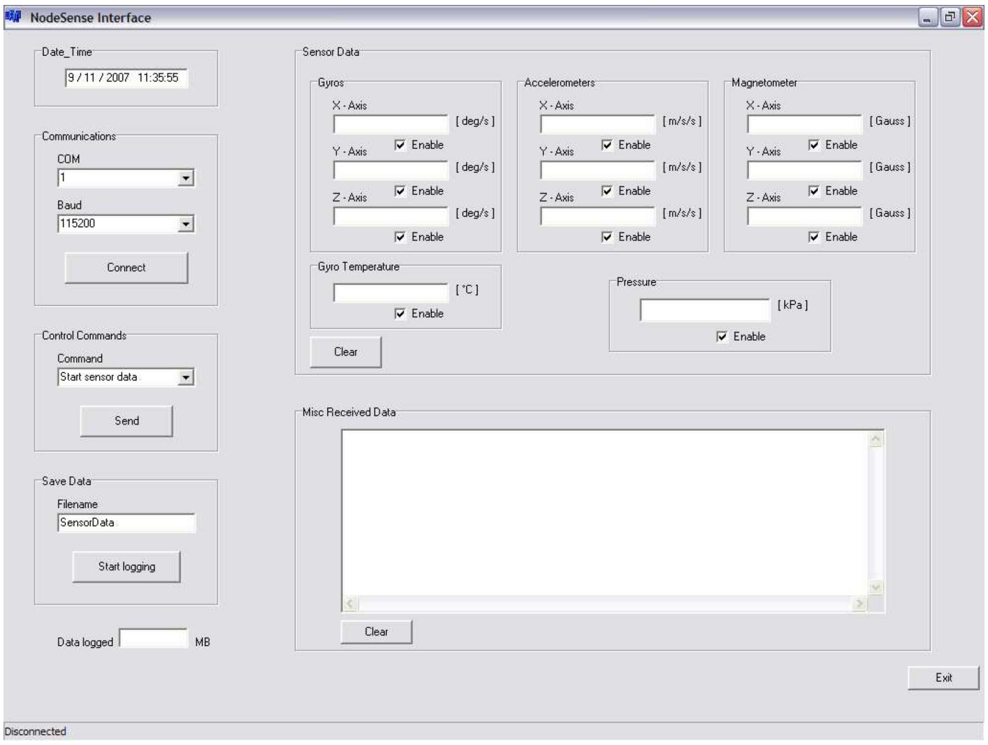
Figure 3.17: Main routine of embedded software

3.4 Interface GUI

The GUI developed for the sensor module executes on a PC and is capable of performing the following tasks:

- Establishment of communication with the sensor module (section 3.4.1).
- Enabling/disabling of sensor data flow from the sensor module to the PC (section 3.4.2).
- Logging of sensor data received from the sensor module (section 3.4.3).
- Displaying of current sensor data received from the sensor module (section 3.4.4).

The complete GUI is illustrated in figure 3.18.



NodeSense Interface

Date_Time: 9 / 11 / 2007 11:35:55

Communications: COM 1; Baud 115200

Control Commands: Command Start sensor data

Save Data: Filename SensorData

Disconnected

Figure 3.18: Sensor module GUI

The GUI tasks are discussed in the following four sections.

3.4.1 Communications

The communication setup allows the user to select the required RS232 COM port number and baud rate. After the selection is made, the user must click on the *Connect* button to initiate the RS232 communication with the sensor module. The *Connect* button will change to a *Disconnect* button if the communication setup is successful. Clicking on the *Disconnect* button will disable the RS232 connection. The communication setup is shown in figure 3.19.

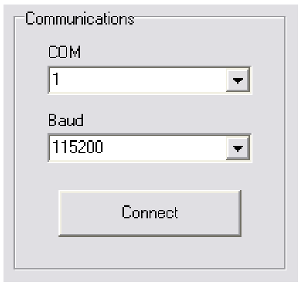


Figure 3.19: GUI communication setup

3.4.2 Control commands

The control interface allows the user to send a command to the sensor module. There are two commands to choose from: *Start sensor data* and *Stop sensor data*. If the sensor module receives the first command, it will start to output sensor data at 50 Hz. The second command tells the sensor module to halt all data sending. By selecting the required command and clicking on the *Send* button, the command is sent to the sensor module. The control interface is shown in figure 3.20.

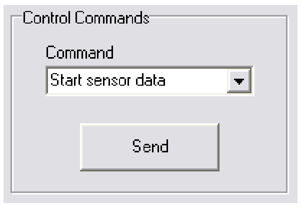


Figure 3.20: GUI control interface

3.4.3 Data logging

The logging interface allows the user to start and stop the logging of received sensor data. By clicking on the *Start logging* button all sensor data are stored in a text file. The title of the text file will contain the string in the *Filename* text box as well as the date and time. When data logging is initiated the *Start logging* button changes to a *Stop logging* button. Clicking on the *Stop logging* button will disable the data logging process. The size of the text file, containing the logged sensor data, is displayed in the *Data logged* text box. Figure 3.21 illustrates the logging interface.

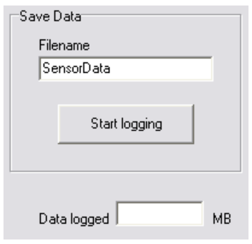


Figure 3.21: GUI logging interface

3.4.4 Sensor data

Figure 3.22 illustrates the display area where all received sensor data is shown.

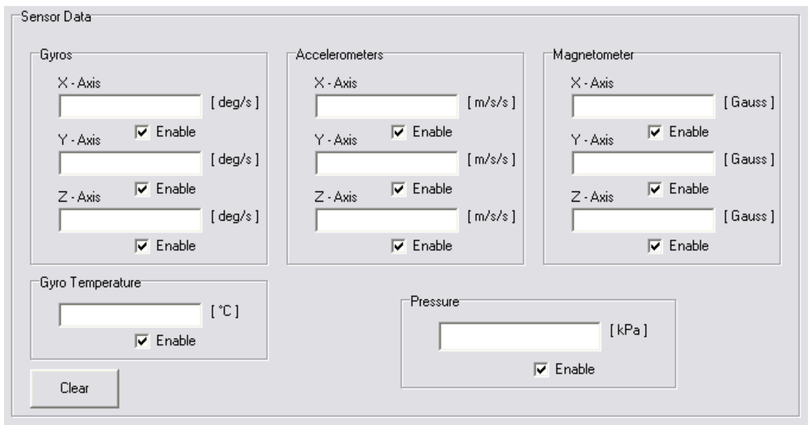


Figure 3.22: Sensor data display

3.5 Sensor calibration

Sensor calibration is necessary to compensate for nominal gain deviations, sensor misalignments, sensor bias values and temperature effects. The gain deviation and sensor misalignment effects are combined in a 3×3 matrix, also known as a cross coupling matrix. A separate vector is used for the sensor bias values. The cross coupling matrix is equal to a unity 3×3 matrix if the sensors are perfectly aligned with the body axial system and if the sensor gains match the theoretical (data sheet) gains. However, even with small misalignments and gain deviations the cross coupling matrix is still close to a 3×3 unity matrix. All calibrations are done with respect to available references. It is important to select an accurate reference, since the calibration is only accurate if the reference is accurate [5].

3.5.1 Cross coupling matrix and bias vector

The cross coupling matrix and bias vector are defined in the following equation:

$$\begin{bmatrix} X_{cal} \\ Y_{cal} \\ Z_{cal} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \begin{bmatrix} X_{A/D} \\ Y_{A/D} \\ Z_{A/D} \end{bmatrix} + \begin{bmatrix} O_x \\ O_y \\ O_z \end{bmatrix} \tag{3.5.1}$$

where:

$[\,X_{cal}\ Y_{cal}\ Z_{cal}\,]^T$ = sensor values after calibration

$\begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix}$ = cross coupling matrix

$[\,X_{A/D}\ Y_{A/D}\ Z_{A/D}\,]^T$ = sensor values before calibration

$[\,O_x\ O_y\ O_z\,]^T$ = vector containing bias values

3.5.1.1 Gyroscopes

The rate gyroscopes are calibrated by using one of the following two methods:

- Mount the sensor module on a rate table and spin each of the three axes (x, y, z) at known rotation rates. The rotation direction must also be varied to increase the calibration accuracy. Log the measured rotation rates and fit them with the known rotation rates.

- Rotate the sensor module through known angles. Log the measured rotation rates and integrate them to form the measured angles. Fit the measured angles with the known angles.

The first method is used, since a working rate table is available in the ESL. A least squares fit on the data provides accurate calibration results. If MEMS sensors are used, it is important to make sure that the sensors are exposed to constant static acceleration during the calibration process, since the bias is sensitive to accelerations.

3.5.1.2 Magnetometer

The magnetometer is calibrated by making use of the fact that the maximum magnetic field in Stellenbosch (test location) is 0.26 Tesla. The sensor module is rotated so that the three axes of the magnetometer cut the maximum magnetic field an equal number of times. It is important not to let any of the magnetometer axes cut the magnetic field notably more, since the least squares algorithm used will then weight some of the parameters more than it should. The magnetometer axes will cut the magnetic field an equal number of times by executing the following rotations:

- Point the nose (x-axis) of the sensor module in a northern direction. All movements are made through 360 degrees, first in one direction and then in the other.

- Pitch, and then roll the sensor module.

- Roll the sensor module through 90 degrees and rotate it through 360 degrees in the northern direction, again in both directions.

Note that the rotation movements do not have to be perfect in order for the algorithm to be effective. The rotation movements can be done very slowly, as long as they are done at the same pace throughout.

A least squares algorithm is now applied in Microsoft's Excel to optimize the cross coupling matrix and the bias vector (equation 3.5.1). The algorithm ensures that the minimum error between the maximum magnetic field and the calculated magnetic field is obtained. Thus, the error to be minimized can be defined as follows:

$$\sum(0.26^{-}r)^2 \tag{3.5.2}$$

where:

$$\begin{aligned} r^2 &= (C_{11}X_{A/D} + C_{12}Y_{A/D} + C_{13}Z_{A/D} + O_x)^2 \\ &\quad + (C_{21}X_{A/D} + C_{22}Y_{A/D} + C_{23}Z_{A/D} + O_y)^2 \\ &\quad + (C_{31}X_{A/D} + C_{32}Y_{A/D} + C_{33}Z_{A/D} + O_z)^2 \\ r &= \sqrt{r^2} \end{aligned}$$

Figure 3.23 illustrates the calibration procedure of the magnetometer used in this project. The top row shows the uncalibrated measurements and the bottom row depicts the measurements after calibration. Tell tale signs to know if the calibration is successful are that the rotations appear spherical and that they touch the four sides of the 0.26 Tesla box.

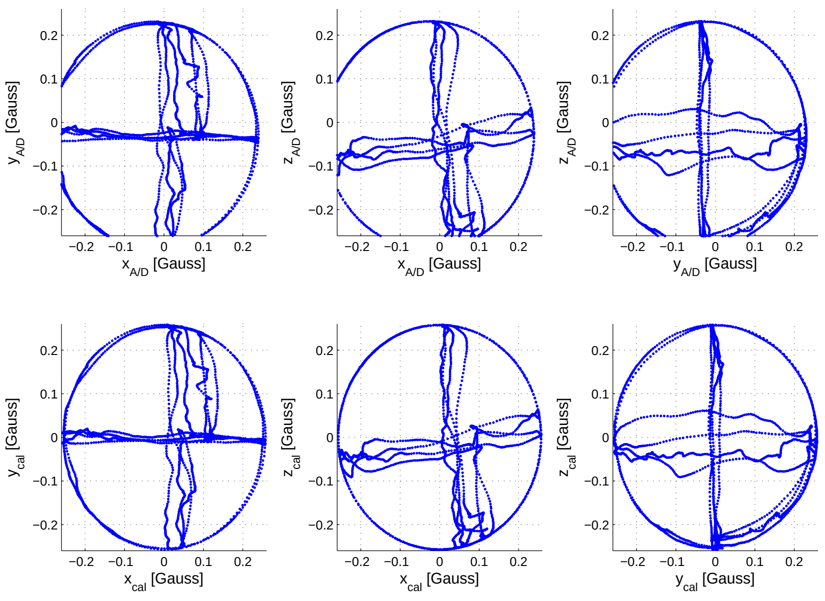


Figure 3.23: Magnetometer calibration

3.5.1.3 Accelerometers

The magnetometer calibration process is also used for the accelerometers, except that the maximum magnetic field (0.26 T) is replaced by the earth's gravity constant (assumed to be 9.81 m/s^2). Figure 3.24 illustrates the calibration results.

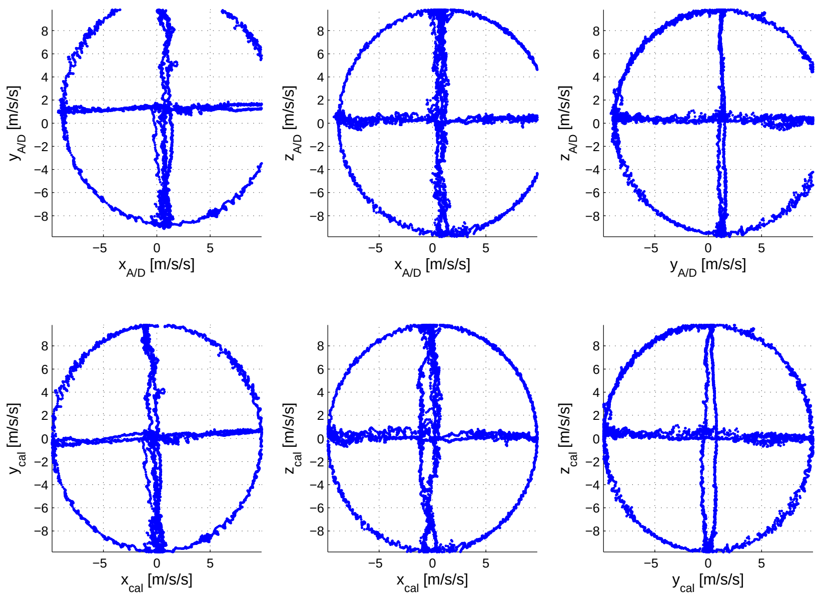


Figure 3.24: Accelerometer calibration

3.5.1.4 Pressure sensor

The pressure sensor is calibrated by applying known pressures, logging the measured pressures and minimizing the error between the known and measured pressures. However, the calibration process shows that the gain and offset values from the pressure sensor data sheet are accurate enough.

3.5.2 Temperature

The bias value of a MEMS sensor is temperature dependent [5], making it necessary to perform temperature calibration in order to compensate for any temperature variations. Only the rate gyroscopes and accelerometers show significant, linear bias drift with temperature variations. Figures 3.25 to 3.30 illustrate each sensor's bias value with respect to temperature. A linear fit is done on the sensors' data.

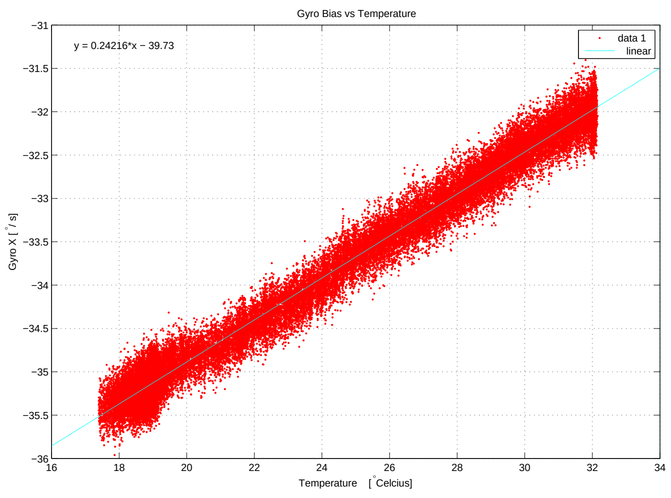


Figure 3.25: Gyroscope X temperature calibration

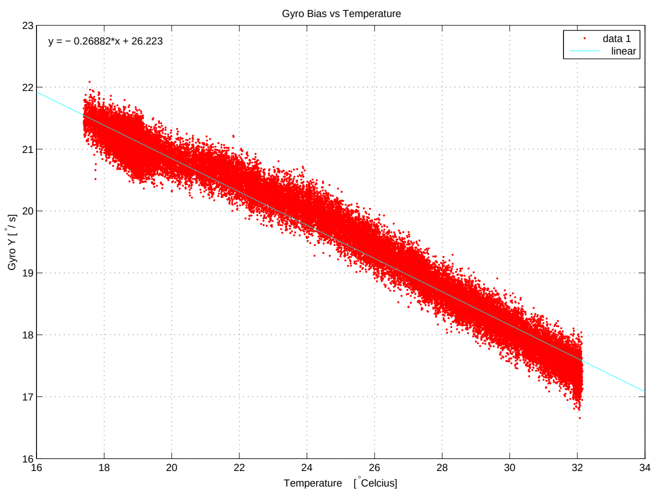


Figure 3.26: Gyroscope Y temperature calibration

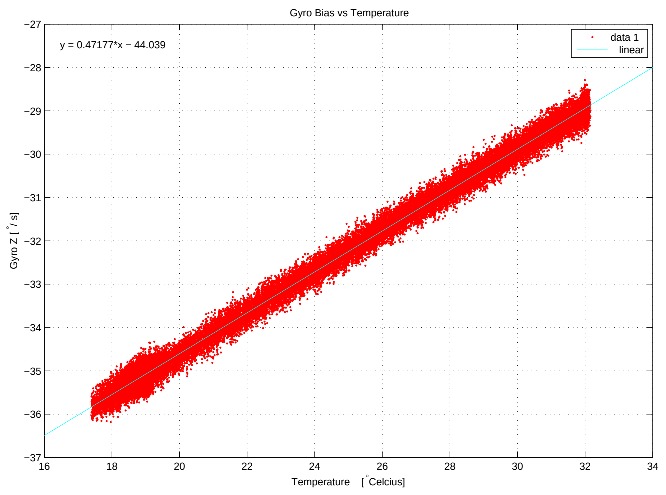


Figure 3.27: Gyroscope Z temperature calibration

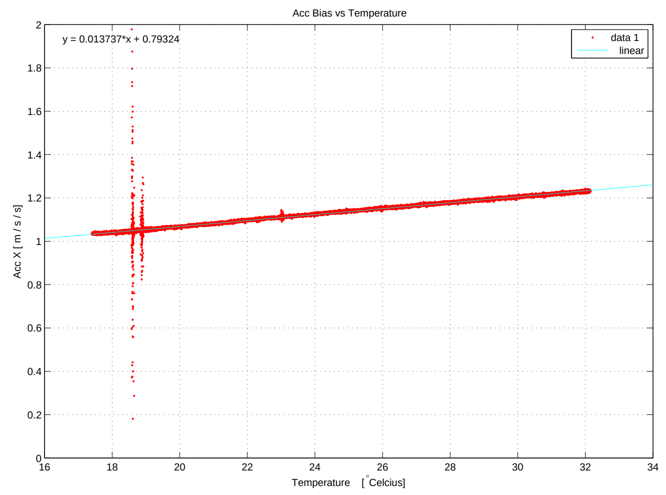


Figure 3.28: Accelerometer X temperature calibration

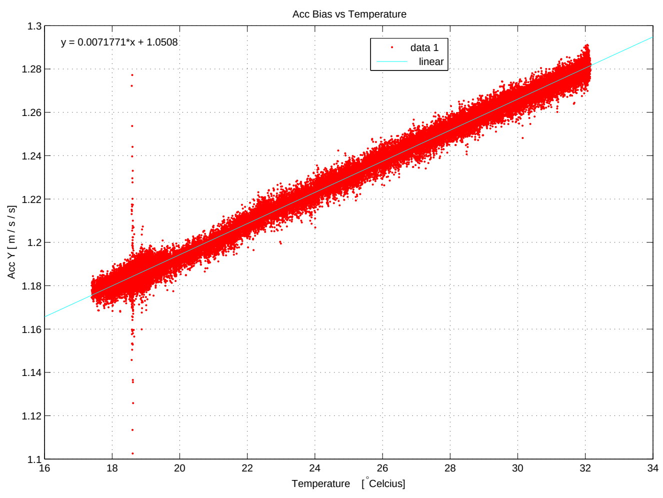


Figure 3.29: Accelerometer Y temperature calibration

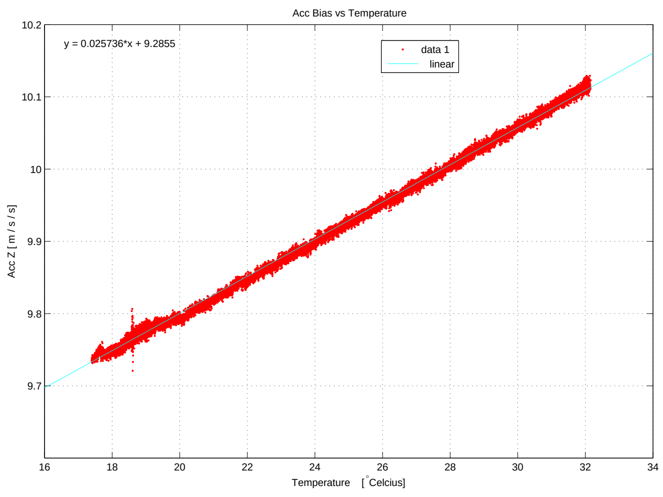


Figure 3.30: Accelerometer Z temperature calibration

3.6 Summary

Chapter 3 discusses the hardware and embedded software developed for this project. The hardware consists of four main parts: a capturing and processing unit, a 6-DOF IMU, other inertial sensors and a power supply unit. Section 3.2 focuses on the four main hardware parts. The embedded software mainly consists of digital filter and communication protocol implementations and these are discussed in section 3.3. Section 3.4 focuses on the GUI developed for the hardware and finally, section 3.5 discusses the sensor calibration methods used in this project.

The hardware and embedded software were tested and operate as specified by the requirements. The hardware schematics and embedded software are available on a disc included with this thesis.

Chapter 4

Navigation Simulation Software

4.1 Overview

Chapter 4 focuses on the development of a navigation algorithm simulation, capable of using simulated and real-time sensor data to simulate sensor combinations and therefore evaluate the best configuration for each mission. The modeling of sensors and the reproduction of sensor data are discussed in section 4.2, while section 4.3 focuses on the implementation of the navigation algorithms. Section 4.4 focuses on the development of a simulation GUI and finally, section 4.5 gives a brief summary of chapter 4.

4.2 Sensor data simulation

The simulation of sensor data can be divided into two main parts:

- Forces, moments and the equations of motion (section 2.3) are used to generate vital navigation data.
- The navigation data, from the first part, and sensor models are used to generate the required sensor data.

All simulations are done in Matlab™ and Simulink™.

4.2.1 Forces and moments

For simulation purposes the body forces and moments are usually generated by the vehicle model. However, an AUV model is not available for this project and the development of a model falls outside the scope of this project. The required forces and moments are generated by making use of two developed controllers: a speed-to-force and an angle-to-moment controller. The speed-to-force controller receives NED speed references and the vehicle mass as input, while the angle-to-moment controller receives attitude references as input. A standard six degrees of freedom block, implementing the equations of motion, receives the generated forces and moments as input and gives accelerations, velocities, angular accelerations, angular rates and angles (attitude) as output. The two controllers and the 6-DOF block are illustrated in figure 4.1.

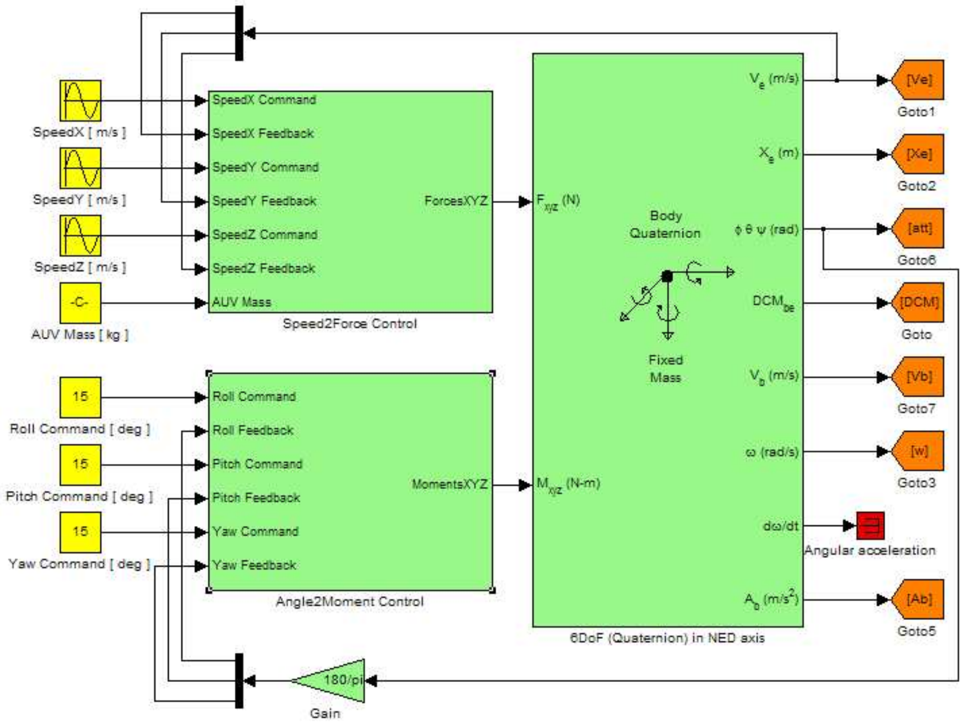
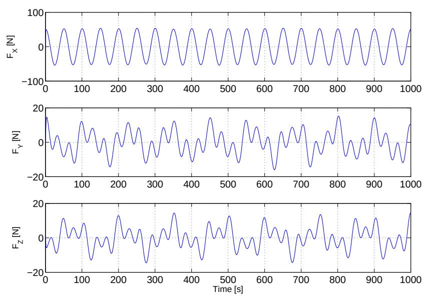
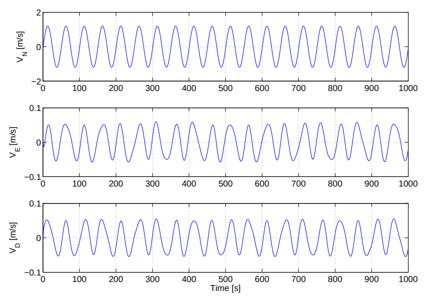


Figure 4.1: Simulink model of simulated AUV motion

Information received from IMT indicates that their developed AUV has a maximum speed of 1.2 m/s and a mass of 360 kg. Taking the received information into account, the forward speed reference is varied between -1.2 m/s and 1.2 m/s, while the other two speed references are kept as close to zero as possible. A mass reference of 360 kg and attitude references between -15° and 15° are also used. The generated forces, moments and AUV motion are shown in the following figures:

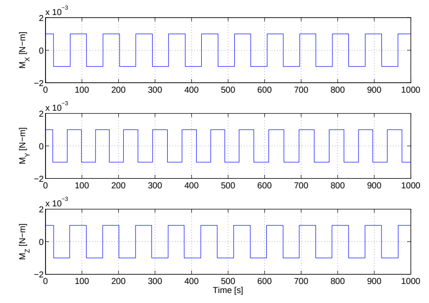


(a)

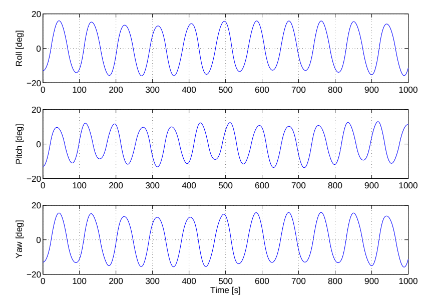


(b)

Figure 4.2: (a) Simulated body forces (b) and resulting NED velocities of AUV



(a)



(b)

Figure 4.3: (a) Simulated body moments (b) and resulting attitude of AUV

Figure 4.2(b) shows how V_N varies between -1.2 m/s and 1.2 m/s, while the variations of V_E and V_D are very small. Figure 4.3(b) shows how the roll, pitch and yaw vary between -15° and 15°. The velocities have a period of 40 - 50 seconds, while the attitude has a period of 80 - 90 seconds. The long periods represent the slow movement of an AUV. The simulated motion in figures 4.2(b) and 4.3(b) accurately represents the movements of an AUV and is used, together with the other 6-DOF block outputs, as inputs for the sensor models (section 4.2.2).

4.2.2 Sensor models

Each sensor simulated in this project has its own model. The model simulates real-life sensor data, making it very useful if certain sensors are unavailable or too expensive to purchase. A GPS, rate gyroscopes, accelerometers, a magnetometer, a sonar, tilt sensors, a pressure sensor and a water wheel are modeled in this project. Each model is discussed in the following sections and is based upon the research done by [5].

4.2.2.1 GPS

Although there is no GPS signal underwater, the sensor is modeled to illustrate the difference between navigation with and without a GPS. A GPS provides position and velocity measurements. The position is given in latitude, longitude and height (LLH), while the velocity is given in NED axes. Figure 4.4 illustrates the GPS model. A discussion of the model follows after the figure.

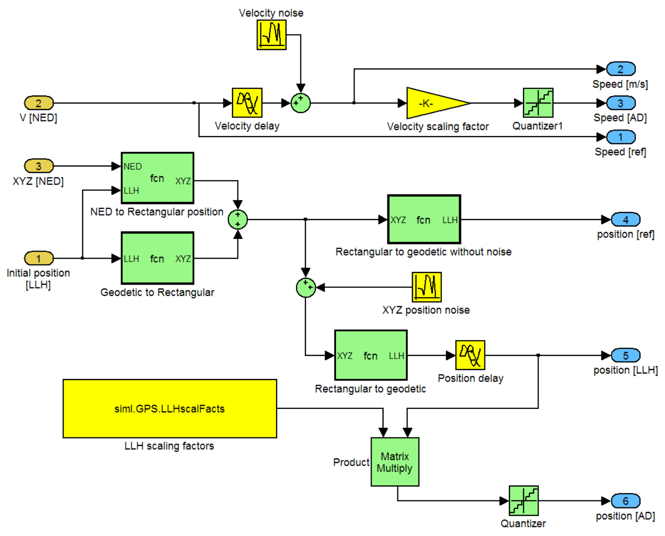


Figure 4.4: Simulink model of GPS

The GPS model receives velocity, displacement and initial position as input. The displacement and velocity are provided by the model in figure 4.1, while the initial position is provided by a user input. The displacement and velocity are both in NED axes, while the initial position is given in LLH. To add the displacement to the initial position, the initial position is converted to ECEF rectangular format by using equations (2.2.4) to (2.2.6). The displacement is converted to rectangular format by using the following equation [5] :

$$\begin{bmatrix} X_{rec} \\ Y_{rec} \\ Z_{rec} \end{bmatrix} = \begin{bmatrix} -\sin\lambda\cos\varphi & -\sin\varphi & -\cos\lambda\cos\varphi \\ -\sin\lambda\sin\varphi & \cos\varphi & -\cos\lambda\sin\varphi \\ \cos\lambda & 0 & -\sin\lambda \end{bmatrix} \begin{bmatrix} X_N \\ Y_E \\ Z_D \end{bmatrix} \tag{4.2.1}$$

The sum of the displacement and initial position in rectangular format is combined with GPS position noise and converted back to LLH by using equations (2.2.1) to (2.2.3). GPS signal delay is added to the converted position and results in a simulated GPS position output. GPS velocity noise and signal delay are added to the input velocity and result in a simulated GPS velocity output. The position and velocity output values are also scaled and converted to integer values to simulate a digital GPS position and velocity output.

4.2.2.2 Gyroscopes

Rate gyroscopes provide angular rate measurements. The angular rates, in rad/s, are given in body axes. Figure 4.5 illustrates the gyroscope model. A discussion of the model follows after the figure.

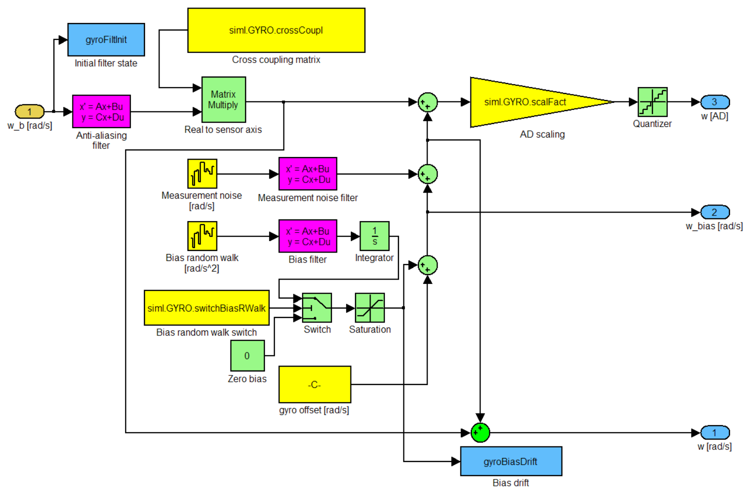


Figure 4.5: Simulink model of rate gyroscopes

The gyroscope model receives body angular rates as input. The angular rates are provided by the model in figure 4.1. A second-order anti-aliasing filter is applied to the input data to simulate the sensor filtering process. The filtered angular rates are then multiplied with a cross coupling matrix (section 3.5) to simulate gain deviation and sensor misalignment effects. Measurement noise (section 2.5.1), bias drift (section 2.5.2) and sensor offset are added to the angular rates and result in a simulated three-axes rate gyroscope output. The output of the noise sources is filtered through second-order Butterworth filters to set the bandwidth. The angular rate output values are also scaled and converted to integer values to simulate a digital three-axes rate gyroscope output.

4.2.2.3 Accelerometers

Accelerometers measure static and dynamic accelerations. The accelerations, in m/s^2, are given in body axes. Figure 4.6 illustrates the accelerometer model. A discussion of the model follows after the figure.

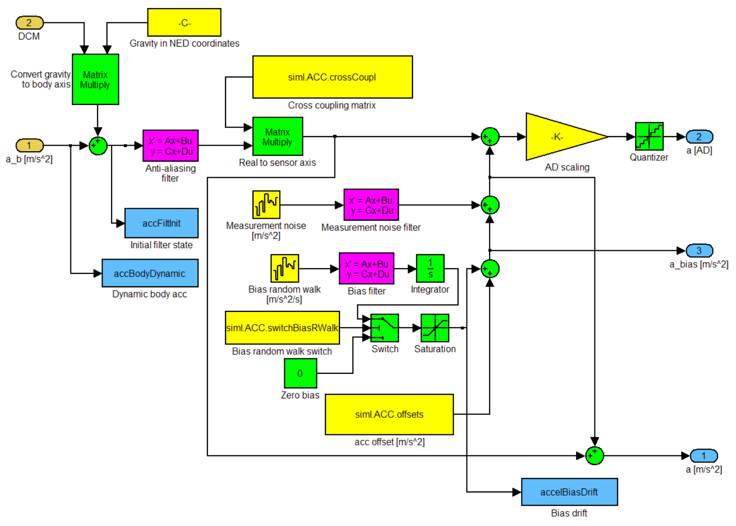


Figure 4.6: Simulink model of accelerometers

The accelerometer model receives dynamic accelerations and the DCM as input. The dynamic accelerations and the DCM are provided by the model in figure 4.1. Static accelerations are added to the dynamic accelerations by converting the earth's gravity vector in NED axes to a gravity vector in body axes. The DCM is used for the axes transformation. A second-order anti-aliasing filter is applied to the sum of accelerations to simulate the sensor filtering process. The filtered accelerations are then multiplied with a cross coupling matrix to simulate gain deviation and sensor misalignment effects. Measurement noise, bias drift and sensor offset are added to the accelerations and result in a simulated three-axes accelerometer output. The output of the noise sources is filtered through second-order Butterworth filters to set the bandwidth. The acceleration output values are also scaled and converted to integer values to simulate a digital three-axes accelerometer output.

4.2.2.4 Magnetometer

The magnetometer measures the magnitude of a magnetic field in three axes. The magnetometer measurements, in Gauss, are given in body axes. Figure 4.7 shows the magnetometer model. A discussion of the model follows after the figure.

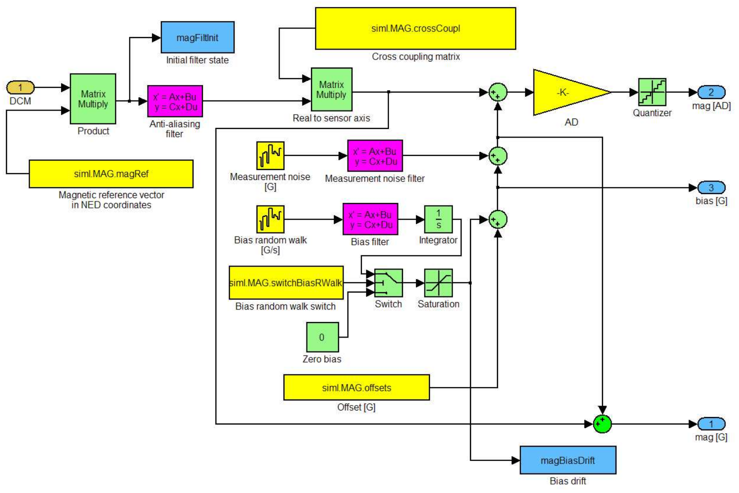


Figure 4.7: Simulink model of magnetometer

The magnetometer model receives the DCM as input. The DCM is provided by the model in figure 4.1. The magnetometer measurements are simulated by converting the earth's magnetic reference vector in NED axes to a vector in body axes. The DCM is used for the axes transformation. A second-order anti-aliasing filter is applied to the magnetometer measurements to simulate the sensor filtering process. The filtered magnetometer measurements are then multiplied with a cross coupling matrix to simulate gain deviation and sensor misalignment effects. Measurement noise, bias drift and sensor offset are added to the magnetometer measurements to simulate noise, hard iron and soft iron effects [5]. The magnetometer measurements with added noise result in a simulated three-axes magnetometer output. The output of the noise sources is filtered through second-order Butterworth filters to set the bandwidth. The magnetometer measurement values are also scaled and converted to integer values to simulate a digital three-axes magnetometer output.

4.2.2.5 Sonar

The sonar provides velocity measurements with respect to the ocean floor. The velocity measurements, in m/s, are given in body axes. Figure 4.8 shows the sonar model. A discussion of the model follows after the figure.

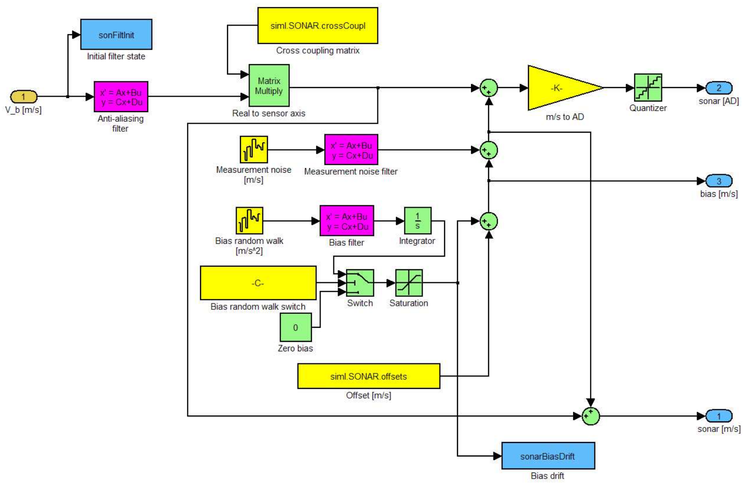


Figure 4.8: Simulink model of sonar

The sonar model receives body velocity measurements as input. The velocity measurements are provided by the model in figure 4.1. A second-order anti-aliasing filter is applied to the velocity measurements to simulate the sensor filtering process. The filtered velocity measurements are then multiplied with a cross coupling matrix to simulate gain deviation and sensor misalignment effects. Measurement noise, bias drift and sensor offset are added to the velocity measurements to simulate noise and the condition of the ocean floor (e.g. sandy or rocky). The velocity measurements with added noise result in a simulated three-axes sonar output. The output of the noise sources is filtered through second-order Butterworth filters to set the bandwidth. The velocity output values are also scaled and converted to integer values to simulate a digital three-axes sonar output.

4.2.2.6 Tilt sensors

The tilt sensors provide roll and pitch measurements in radians. Figure 4.9 shows the tilt sensor model. A discussion of the model follows after the figure.

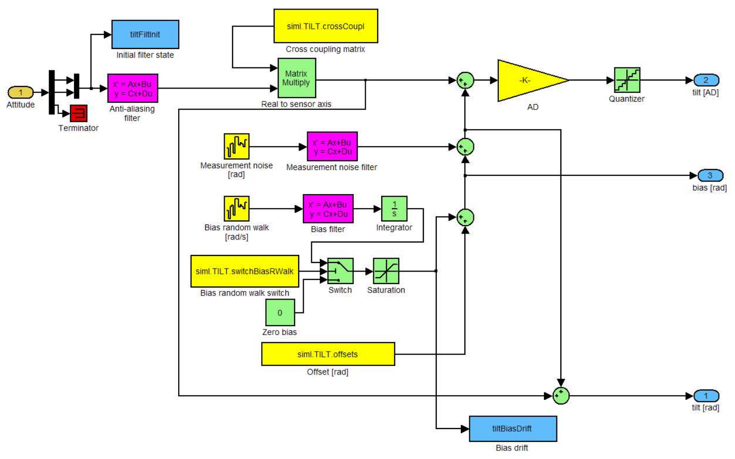


Figure 4.9: Simulink model of tilt sensors

The tilt sensor model receives roll and pitch measurements as input. The roll and pitch measurements are provided by the model in figure 4.1. A second-order anti-aliasing filter is applied to the input data to simulate the sensor filtering process. The filtered roll and pitch measurements are then multiplied with a cross coupling matrix to simulate gain deviation and sensor misalignment effects. Measurement noise, bias drift and sensor offset are added to the roll and pitch measurements and result in a simulated two-axes tilt sensor output. The output of the noise sources is filtered through second-order Butterworth filters to set the bandwidth. The roll and pitch output values are also scaled and converted to integer values to simulate a digital two-axes tilt sensor output.

4.2.2.7 Pressure sensor

The pressure sensor provides pressure measurements that can be converted to underwater depth measurements in meters. Figure 4.10 shows the pressure sensor model. A discussion of the model follows after the figure.

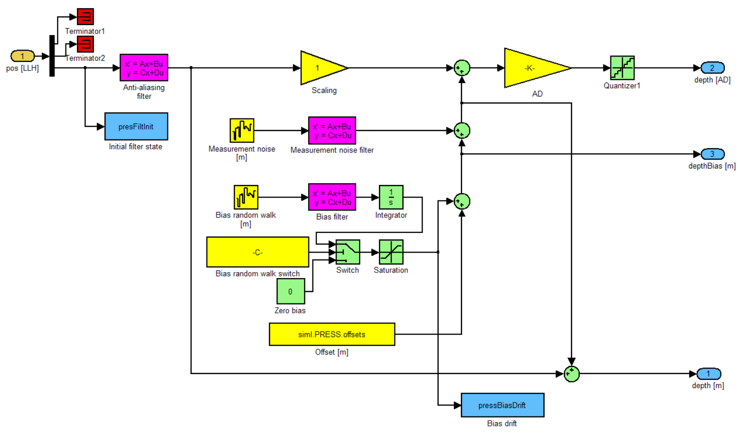


Figure 4.10: Simulink model of pressure sensor

The pressure sensor model receives height measurements as input. The height measurements are provided by the GPS model (figure 4.4). Negative height indicates the underwater depth of the vehicle. A second-order anti-aliasing filter is applied to the input data to simulate the sensor filtering process. The filtered height measurements are then multiplied with a scaling gain to simulate gain deviation effects. The pressure sensor is a single-axis sensor, thus no cross coupling matrix is required. Measurement noise, bias drift and sensor offset are added to the height measurements and result in a simulated depth output. The output of the noise sources is filtered through second-order Butterworth filters to set the bandwidth. The depth output values are also scaled and converted to integer values to simulate a digital depth output.

4.2.2.8 Water wheel

The water wheel provides forward body velocity (in m/s) with respect to the ocean current flow. Figure 4.11 shows the water wheel model. A discussion of the model follows after the figure.

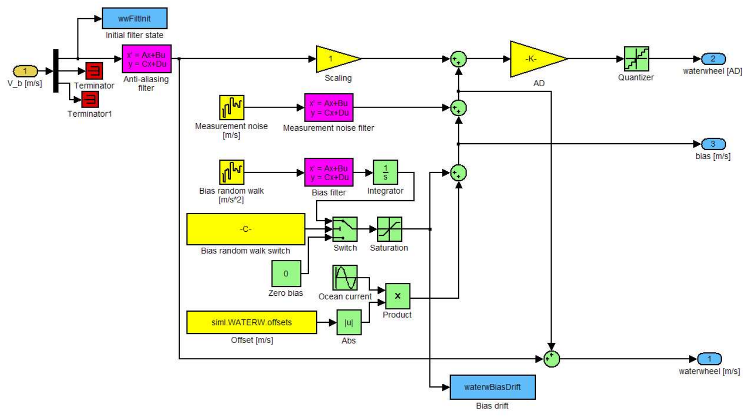


Figure 4.11: Simulink model of water wheel

The water wheel model receives forward body velocity measurements as input. The forward velocity measurements are provided by the model in figure 4.1. A second-order anti-aliasing filter is applied to the input data to simulate the sensor filtering process. The filtered forward velocity measurements are then multiplied with a scaling gain to simulate gain deviation effects. The water wheel is a single-axis sensor, thus no cross coupling matrix is required. Ocean currents are simulated by adding a sinusoidal offset to the forward velocity measurements. Measurement noise and bias drift are also added to the forward velocity measurements and result in a simulated water wheel output. The output of the noise sources is filtered through second-order Butterworth filters to set the bandwidth. The forward velocity output values are also scaled and converted to integer values to simulate a digital water wheel output.

4.3 Navigation algorithms

The two navigation algorithms implemented in this project are the MEKF (section 2.6.2) and the Position and Velocity EKF (section 2.6.3). Each algorithm implementation is divided into two parts: an initialization and update part. The initialization part initializes the algorithm, while the update part is executed at each time step for the duration of the simulation.

4.3.1 MEKF

The MEKF estimates the quaternion vector which is used to calculate the DCM (equation 2.2.17) and the attitude (equations 2.2.23 to 2.2.25) of a vehicle. The MEKF also estimates the three rate gyroscope biases. Simulated sensor data (section 4.2) and real-time sensor data can be used for the MEKF input.

4.3.1.1 Initialization

The initialization routine is illustrated in the following functional block diagram:

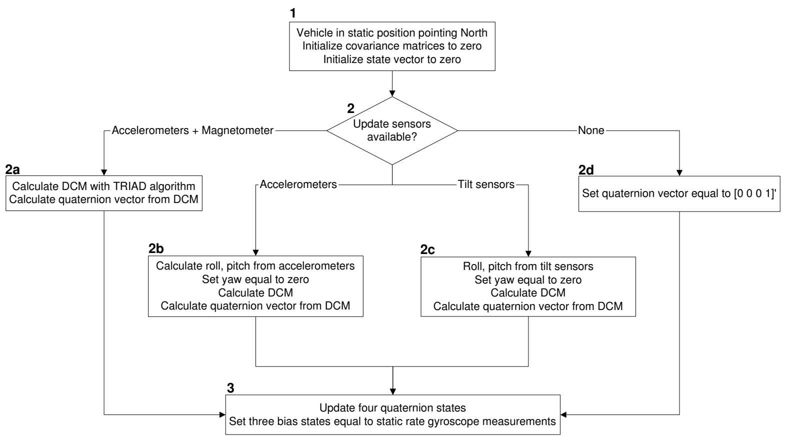

1
Vehicle in static position pointing North
Initialize covariance matrices to zero
Initialize state vector to zero
2
Update sensors available?
Accelerometers + Magnetometer
None
Accelerometers
Tilt sensors
2a
Calculate DCM with TRIAD algorithm
Calculate quaternion vector from DCM
2b
Calculate roll, pitch from accelerometers
Set yaw equal to zero
Calculate DCM
Calculate quaternion vector from DCM
2c
Roll, pitch from tilt sensors
Set yaw equal to zero
Calculate DCM
Calculate quaternion vector from DCM
2d
Set quaternion vector equal to [0 0 0 1]'
3
Update four quaternion states
Set three bias states equal to static rate gyroscope measurements

Figure 4.12: MEKF initialization routine

The functional block diagram in figure 4.12 is now discussed in more detail:

1. With the vehicle in a static position and pointing North, the covariance matrices and the state vector are initialized to zero.

2. The choice of update sensors determines the next step.

   a) The accelerometer measurements, gravity reference vector, magnetometer measurements and magnetic field reference vector are used with the TRIAD algorithm (section 2.4) to calculate the DCM. The quaternion vector is then calculated by using equations (2.2.26) to (2.2.30).

   b) The accelerometer measurements are used to calculate roll and pitch, while yaw is set to zero. Equation (2.2.10) is used to calculate the DCM. The quaternion vector is then calculated by using equations (2.2.26) to (2.2.30).

   c) The tilt sensors provide roll and pitch, while yaw is set to zero. Equation (2.2.10) is used to calculate the DCM. The quaternion vector is then calculated by using equations (2.2.26) to (2.2.30).

   d) The quaternion vector is initiated to $[\,0\,0\,0\,1\,]^T$.

3. The quaternion states are updated, while the three bias states are set equal to the static rate gyroscope measurements.

4.3.1.2 Update

The update routine is illustrated in the following functional block diagram:

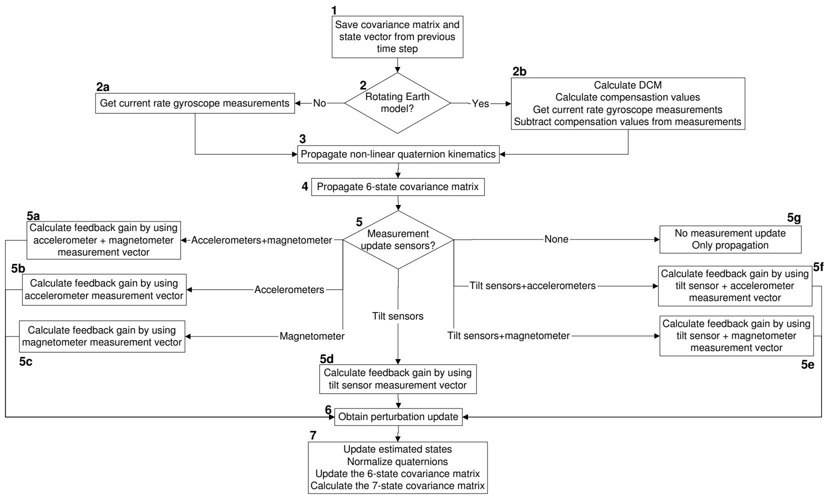

1
Save covariance matrix and state vector from previous time step
2
Rotating Earth model?
No
Yes
2a
Get current rate gyroscope measurements
2b
Calculate DCM
Calculate compensastion values
Get current rate gyroscope measurements
Subtract compensation values from measurements
3
Propagate non-linear quaternion kinematics
4
Propagate 6-state covariance matrix
5
Measurement update sensors?
5a
Calculate feedback gain by using accelerometer + magnetometer measurement vector
Accelerometers+magnetometer
5b
Calculate feedback gain by using accelerometer measurement vector
Accelerometers
5c
Calculate feedback gain by using magnetometer measurement vector
Magnetometer
Tilt sensors
5d
Calculate feedback gain by using tilt sensor measurement vector
None
5g
No measurement update
Only propagation
Tilt sensors+accelerometers
5f
Calculate feedback gain by using tilt sensor + accelerometer measurement vector
Tilt sensors+magnetometer
5e
Calculate feedback gain by using tilt sensor + magnetometer measurement vector
6
Obtain perturbation update
7
Update estimated states
Normalize quaternions
Update the 6-state covariance matrix
Calculate the 7-state covariance matrix

Figure 4.13: MEKF update routine

The functional block diagram in figure 4.13 is now discussed in more detail:

1. The covariance matrix and state vector from the previous time step are saved.

2. The choice of an Earth rotation model determines the next step.

   a) A non-rotating model is adopted and the estimated gyroscope biases, from the previous time step, are subtracted from the current rate gyroscope measurements.

   b) A rotating model is adopted and compensation must be done (section 2.6.2.4). The DCM is calculated by using the quaternion vector from the previous time step. Position and velocity values, estimated by the Position and Velocity EKF, together with the DCM and the earth's rotation rate are substituted into equation (2.6.42) and the adjusted rate gyroscope measurements are calculated. The estimated gyroscope biases, from the previous time step, are also subtracted from the adjusted rate gyroscope measurements.

3. The non-linear quaternion kinematics are propagated by integrating equation (2.6.20) and adding the result to the estimated quaternions from the previous time step. The gyroscope biases are propagated by using the estimated bias states from the previous time step.

4. The perturbation covariance matrix is propagated as follows:

   - The discrete matrix, $\boldsymbol{\Phi}_k$, is calculated by using equation (2.6.10) and the continuous matrix, $\boldsymbol{F}$, from equation (2.6.39).
   - The discrete process-noise matrix, $\boldsymbol{Q}_k$, is calculated by using equation (2.6.11).
   - Equation (2.6.12) is then used for the covariance propagation.

5. The choice of update sensors determines how update measurements are integrated into the system:

   a) The gravity and magnetic field reference vector in NED coordinates are transformed to vectors in body axes by making use of the calculated DCM. Since two vector updates are available, two of the matrices in equation (2.6.48) are combined to form a $6{\times}6$

measurement matrix. The vectors in body axes are normalized and substituted into the combined measurement matrix. Equation (2.6.13) is used to calculate the optimal feedback gain matrix. The innovation vector is obtained by subtracting the transformed reference vectors from the sensor measurement vectors.

b) The gravity reference vector in NED coordinates is transformed to a vector in body axes by making use of the calculated DCM. Since only one vector update is available, the matrix in equation (2.6.48) is used for the measurement matrix. The vector in body axes is normalized and substituted into the measurement matrix. Equation (2.6.13) is used to calculate the optimal feedback gain matrix. The innovation vector is obtained by subtracting the transformed gravity reference vector from the accelerometer measurement vector.

c) The magnetic field reference vector in NED coordinates is transformed to a vector in body axes by making use of the calculated DCM. Since only one vector update is available, the matrix in equation (2.6.48) is used for the measurement matrix. The vector in body axes is normalized and substituted into the measurement matrix. Equation (2.6.13) is used to calculate the optimal feedback gain matrix. The innovation vector is obtained by subtracting the transformed magnetic field reference vector from the magnetometer measurement vector.

d) The tilt sensor measurements are integrated into the system by using equation (2.2.23), equation (2.2.24) and the following measurement matrix:

$$\boldsymbol{H} = \begin{bmatrix} \frac{\partial\phi}{\partial q_1} & \frac{\partial\phi}{\partial q_2} & \frac{\partial\phi}{\partial q_3} & 0 & 0 & 0 \\ \frac{\partial\theta}{\partial q_1} & \frac{\partial\theta}{\partial q_2} & \frac{\partial\theta}{\partial q_3} & 0 & 0 & 0 \end{bmatrix} \tag{4.3.1}$$

Equation (2.6.13) is used to calculate the optimal feedback gain matrix. The innovation vector is obtained by subtracting the estimated roll and pitch (equations 2.2.23 and 2.2.24) from the tilt sensor measurement vector.

e) The measurement matrix from 5(c) and 5(d) are combined to form a 6×6 measurement matrix. Equation (2.6.13) is used to calculate the optimal feedback gain matrix. The innovation vector is obtained as discussed in 5(c) and 5(d).

f) The measurement matrix from 5(b) and 5(d) are combined to form a 6×6 measurement matrix. Equation (2.6.13) is used to calculate the optimal feedback gain matrix. The innovation vector is obtained as discussed in 5(b) and 5(d).

g) No sensor update measurements are available. Only the propagation, described in 3 and 4, takes place. The full seven-state (four quaternions and three gyroscope biases) covariance matrix is calculated by using an adapted version of equation (2.6.35) and is written as follows:

$$\boldsymbol{P}_{7\times 7} = \boldsymbol{S}\,\boldsymbol{P}_{6\times 6}\,\boldsymbol{S}' \tag{4.3.2}$$

where:

$$\boldsymbol{S} = \begin{bmatrix} \hat{q}_4 & -\hat{q}_3 & \hat{q}_2 & 0 & 0 & 0 \\ \hat{q}_3 & \hat{q}_4 & -\hat{q}_1 & 0 & 0 & 0 \\ -\hat{q}_2 & \hat{q}_1 & \hat{q}_4 & 0 & 0 & 0 \\ -\hat{q}_1 & -\hat{q}_2 & -\hat{q}_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

6. The perturbation update vector is obtained by multiplying the optimal feedback gain matrix with the innovation vector.

7. The four propagated quaternion states are updated by using equation (2.6.34), while the three gyroscope bias states are updated by adding the bias perturbation vector to the propagated bias states. The quaternions are normalized and the six-state perturbation covariance matrix is updated by using equation (2.6.14). Equation (4.3.2) is used to calculate the full seven-state covariance matrix.

4.3.2 Position and Velocity EKF

The PVEKF estimates the position and velocity of a vehicle. The position is given in LLH, while the velocity is given in NED axes. Simulated sensor data (section 4.2) and real-time sensor data can be used for the PVEKF input.

4.3.2.1 Initialization

The initialization routine is illustrated in the following functional block diagram:

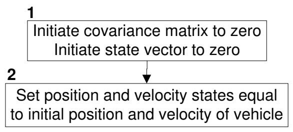

1
Initiate covariance matrix to zero
Initiate state vector to zero
2
Set position and velocity states equal to initial position and velocity of vehicle

Figure 4.14: PVEKF initialization routine

The functional block diagram in figure 4.14 is now discussed in more detail:

1. The covariance matrix and the state vector are initialized to zero.
2. The position and velocity states are set equal to the initial position and velocity of the vehicle. The initial position and velocity can be entered manually or it can be retrieved from onboard sensors (e.g. GPS).

4.3.2.2 Update

The update routine is illustrated in the following functional block diagram:

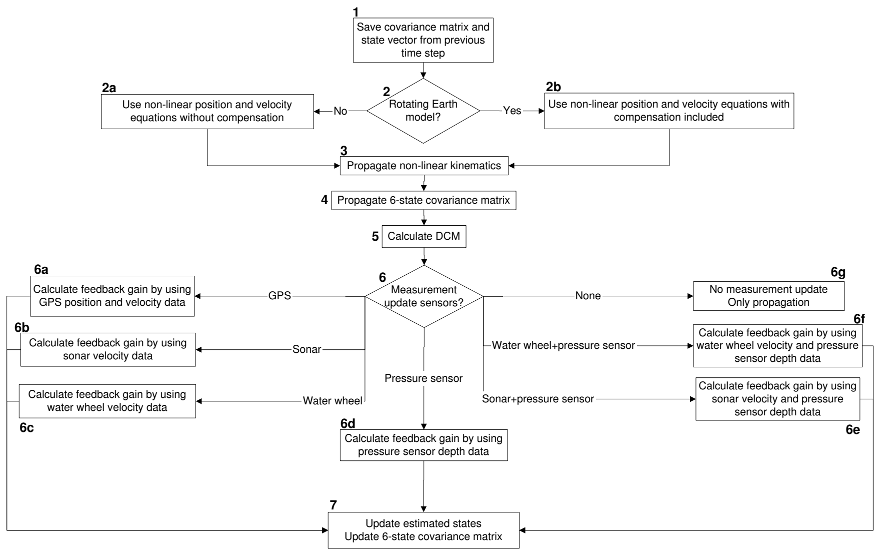

1
Save covariance matrix and state vector from previous time step
2
Rotating Earth model?
No
Yes
2a
Use non-linear position and velocity equations without compensation
2b
Use non-linear position and velocity equations with compensation included
3
Propagate non-linear kinematics
4
Propagate 6-state covariance matrix
5
Calculate DCM
6
Measurement update sensors?
GPS
Sonar
Water wheel
Pressure sensor
None
Water wheel+pressure sensor
Sonar+pressure sensor
6a
Calculate feedback gain by using GPS position and velocity data
6b
Calculate feedback gain by using sonar velocity data
6c
Calculate feedback gain by using water wheel velocity data
6d
Calculate feedback gain by using pressure sensor depth data
6e
Calculate feedback gain by using sonar velocity and pressure sensor depth data
6f
Calculate feedback gain by using water wheel velocity and pressure sensor depth data
6g
No measurement update Only propagation
7
Update estimated states Update 6-state covariance matrix

Figure 4.15: PVEKF update routine

The functional block diagram in figure 4.15 is now discussed in more detail:

1. The covariance matrix and state vector from the previous time step are saved.
2. The choice of an Earth rotation model determines the next step.
   a) Equations (2.6.49) to (2.6.51) are used for the non-linear position kinematics. Equation (2.6.55), without the compensation terms, is used for the non-linear velocity kinematics.
   b) Equations (2.6.49) to (2.6.51) are used for the non-linear position kinematics. Equation (2.6.55), with the compensation terms included, is used for the non-linear velocity kinematics.
3. The non-linear position and velocity kinematics are propagated by integrating the non-linear equations and adding the result to the estimated position and velocity states from the previous time step.
4. The six-state covariance matrix is propagated as follows:
   - The discrete matrix, $\boldsymbol{\Phi}_k$, is calculated by using equation (2.6.10) and the continuous matrix, $\boldsymbol{F}$, from equation (2.6.57).
   - The discrete process-noise matrix, $\boldsymbol{Q}_k$, is calculated by using equation (2.6.11).
   - Equation (2.6.12) is then used for the covariance propagation.
5. The DCM is calculated by using equation (2.2.17). The quaternions are estimated by the MEKF.
6. The choice of update sensors determines how update measurements are integrated into the system:
   a) All six states are updated by the GPS measurements. The following measurement matrix is used:

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3.3}$$

Equation (2.6.13) is used to calculate the optimal feedback gain matrix. The innovation vector is obtained by subtracting the propagated position and velocity states from the GPS measurement vector.

b) The sonar measurement vector in body axes is transformed to a vector in NED axes by using the inverse DCM. All three velocity states are updated by the NED measurement vector. The following measurement matrix is used:

$$\boldsymbol{H} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3.4}$$

Equation (2.6.13) is used to calculate the optimal feedback gain matrix. The innovation vector is obtained by subtracting the propagated velocity states from the NED measurement vector.

c) The water wheel measurement in body axes is transformed to a measurement in NED axes by using the inverse DCM. Only one velocity state, V_N, is updated by the NED measurement. The following measurement matrix is used:

$$\boldsymbol{H} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.3.5}$$

Equation (2.6.13) is used to calculate the optimal feedback gain matrix. The innovation value is obtained by subtracting the propagated velocity state, V_N, from the NED measurement.

d) The pressure sensor measurement is converted to a depth measurement and incorporated into the system by using the following measurement matrix:

$$\boldsymbol{H} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{4.3.6}$$

Equation (2.6.13) is used to calculate the optimal feedback gain matrix. The innovation value is obtained by subtracting the propagated height state from the depth measurement.

e) The measurement matrix from 6(b) and 6(d) are combined to form a 4×6 measurement matrix. Equation (2.6.13) is used to calculate the optimal feedback gain matrix. The innovation vector is obtained as discussed in 6(b) and 6(d).

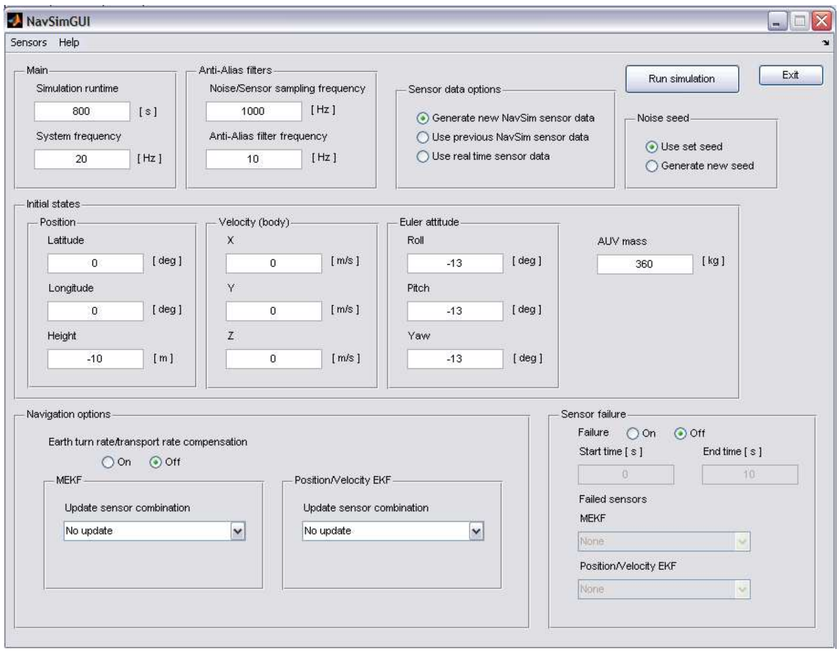
f) The measurement matrix from 6(c) and 6(d) are combined to form a 2×6 measurement matrix. Equation (2.6.13) is used to calculate the optimal feedback gain matrix. The innovation vector is obtained as discussed in 6(c) and 6(d).

g) No sensor update measurements are available. Only the propagation, described in 3 and 4, takes place.

7. The optimal feedback gain matrix is multiplied with the innovation vector and the result is added to the six propagated states, resulting in updated states. Equation (2.6.14) is used to update the covariance matrix.

4.4 User interface

The GUI developed for the simulation software executes in the Matlab™ environment and is illustrated in figure 4.16.



Main
Simulation runtime 800 [s]
System frequency 20 [Hz]

Anti-Alias filters
Noise/Sensor sampling frequency 1000 [Hz]
Anti-Alias filter frequency 10 [Hz]

Sensor data options
[✓ Generate new NavSim sensor data]
Use previous NavSim sensor data
Use real time sensor data

Noise seed
[✓ Use set seed]
Generate new seed

Initial states
Position
Latitude 0 [deg]
Longitude 0 [deg]
Height -10 [m]

Velocity (body)
X 0 [m/s]
Y 0 [m/s]
Z 0 [m/s]

Euler attitude
Roll -13 [deg]
Pitch -13 [deg]
Yaw -13 [deg]

AUV mass 360 [kg]

Navigation options
Earth turn rate/transport rate compensation
On [✓ Off]

MEKF
Update sensor combination No update

Position/Velocity EKF
Update sensor combination No update

Sensor failure
Failure On [✓ Off]
Start time [s] 0
End time [s] 10
Failed sensors
MEKF None
Position/Velocity EKF None

Figure 4.16: Simulation software GUI

The GUI in figure 4.16 helps the user to run a navigation simulation and displays the necessary simulation results. Each part of the GUI is discussed in the following sections.

4.4.1 Sensors and Help menu

By clicking on the *Sensors* drop-down menu, illustrated in figure 4.17, the user is able to select any of the modeled sensors (section 4.2.2) and enter the sensor's characteristics. The characteristics include measurement noise, bias drift, offset, cross coupling and a few sensor-specific aspects. By clicking on the *Help* drop-down menu, the user is provided with information that explains the use of each GUI function.

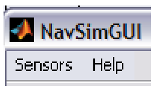
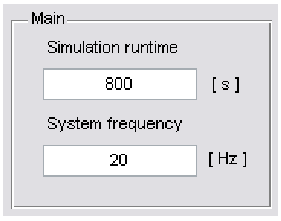


Figure 4.17: Sensors and Help menu

4.4.2 Main

The *Main* section enables the user to enter the simulation runtime and the system update frequency. Figure 4.18 shows the *Main* section.



Main

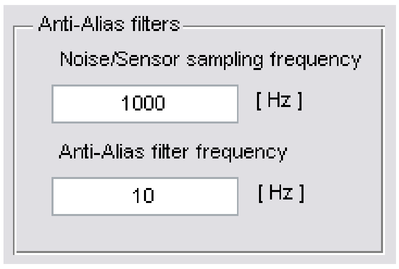
Simulation runtime: 800 [s]

System frequency: 20 [Hz]

Figure 4.18: Main

4.4.3 Anti-alias filters

The *Anti-Alias filters* section, illustrated in figure 4.19, enables the user to set the sensors' sampling frequency and the anti-aliasing filters' cut-off frequency.


Anti-Alias filters

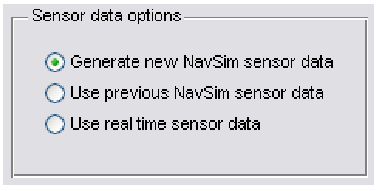
Noise/Sensor sampling frequency: 1000 [Hz]

Anti-Alias filter frequency: 10 [Hz]

Figure 4.19: Anti-alias filters

4.4.4 Sensor data options

The *Sensor data options* section enables the user to choose between the use of newly generated sensor data, previously saved sensor data or saved real-time sensor data. New sensor data are generated by the sensor simulation software (section 4.2), while real-time sensor data are generated by a vehicle's sensors and saved in a text file. Figure 4.20 illustrates the *Sensor data options* section.

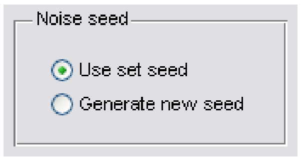

Sensor data options

● Generate new NavSim sensor data
○ Use previous NavSim sensor data
○ Use real time sensor data

Figure 4.20: Sensor data options

4.4.5 Noise seed

The *Noise seed* section enables the user to choose between a constant or newly generated seed value. A constant seed value will cause the random number generators, in the simulation software, to produce the same sequence of values with each simulation. A new seed value will cause the random number generators to produce a different sequence of values. The noise seed option is useful when different sensor combinations are compared and the noise sequence needs to stay the same with each simulation. Figure 4.21 shows the *Noise seed* section.



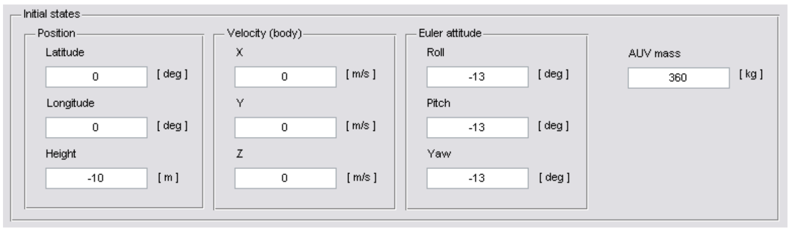
Noise seed

[✓ Use set seed] Generate new seed

Figure 4.21: Noise seed

4.4.6 Initial states

The *Initial states* section enables the user to enter the initial position, velocity and attitude of an AUV or any other simulated vehicle. The mass of the vehicle can also be entered. Figure 4.22 illustrates the *Initial states* section.



Initial states

Position
Latitude: 0 [deg]
Longitude: 0 [deg]
Height: -10 [m]

Velocity (body)
X: 0 [m/s]
Y: 0 [m/s]
Z: 0 [m/s]

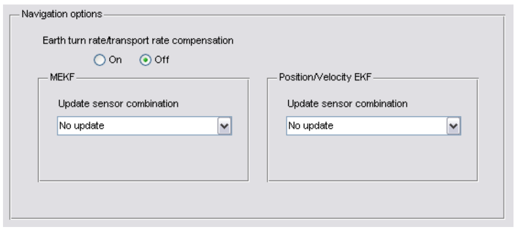
Euler attitude
Roll: -13 [deg]
Pitch: -13 [deg]
Yaw: -13 [deg]

AUV mass: 360 [kg]

Figure 4.22: Initial states

4.4.7 Navigation options

The *Navigation options* section enables the user to select the sensor combinations that need to be simulated. Sensor combinations can be selected for the MEKF and the PVEKF. The user also has a choice between a rotating and a non-rotating Earth model. Figure 4.23 illustrates the *Navigation options* section.



Navigation options

Earth turn rate/transport rate compensation
On [✓ Off]

MEKF
Update sensor combination: No update

Position/Velocity EKF
Update sensor combination: No update

Figure 4.23: Navigation options

4.4.8 Sensor failure

The *Sensor failure* section enables the user to simulate sensor failures for certain periods of time. Additional information about a sensor combination can be obtained by analyzing the navigation results during sensor failure. Figure 4.24 shows the *Sensor failure* section.

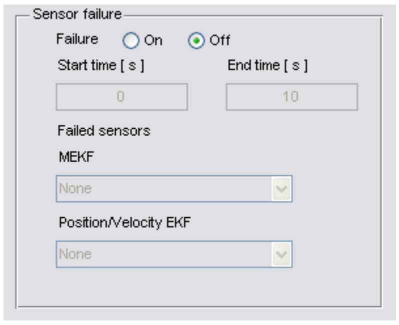

Sensor failure
Failure On [✓ Off]
Start time [s] 0
End time [s] 10
Failed sensors
MEKF: None
Position/Velocity EKF: None

Figure 4.24: Sensor failure

4.4.9 Simulation execution

By clicking on the *Run simulation* button, the GUI collects all the entered information and calls the necessary procedures. After a successful simulation all the necessary outputs are generated and displayed in a user-friendly format. The simulation output shows the position, velocity and attitude of the vehicle during the simulation. Statistical error information is also part of the simulation output and is used to compare sensor combinations with each other. The simulation software discussed in this chapter is summarized in the following functional block diagram:

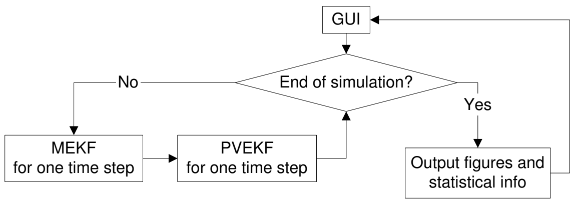


Figure 4.25: Summary of simulation software

4.5 Summary

Chapter 4 discusses the navigation simulation software developed for this project. Section 4.2 focuses on AUV movement simulation and sensor modeling. The implementation of the MEKF and PVEKF algorithms is discussed in section 4.3, while the simulation GUI is discussed in section 4.4.

The simulation software was tested and operates as specified by the requirements. The software is available on a disc included with this thesis.

Chapter 5

Results

5.1 Overview

Chapter 5 illustrates and discusses the navigation simulation results of this project. Section 5.2 focuses on simulations where real-time sensor data are used as input, while section 5.3 focuses on simulations where simulated sensor data are used as input. Finally, section 5.4 gives a brief summary of Chapter 5.

5.2 Real-time data

Real-time sensor data are used to test the simulation software discussed in Chapter 4. Real-time data are also used to make sure that the simulations are realistic enough. The sensor module discussed in Chapter 3 is used to collect real-time sensor data. Three maneuvers are simulated: a straight line, wave and turn maneuver. With each maneuver the sensor module is manually moved by a human to simulate the specific movements. The real-time sensor data are logged and used as input for the simulation software.

Before a simulation with real-time data is executed, the quality of the sensor module's sensors must be determined by calculating their noise characteristics. The Allan variance method (section 2.5) is used to calculate the noise characteristics and these are listed in tables 5.1 to 5.4. The noise characteristics are entered into the simulation by using the simulation GUI (section 4.4).

Rate gyroscope axis	**X**	**Y**	**Z**	**Avg**
Angle random walk [$1\times10^{-4}\ rad/s/\sqrt{Hz}$]	6.84	7.27	7.45	7.19
Rate random walk [$1\times10^{-5}\ rad/s/s/\sqrt{Hz}$]	2.45	3.09	4.22	3.26

Table 5.1: Noise characteristics of rate gyroscopes

Accelerometer axis	**X**	**Y**	**Z**	**Avg**
Velocity random walk [$1\times10^{-4}\ m/s^2/\sqrt{Hz}$]	6.88	6.68	11.3	8.28
Rate random walk [$1\times10^{-5}\ m/s^2/s/\sqrt{Hz}$]	4.62	3.08	12.2	6.62

Table 5.2: Noise characteristics of accelerometers

Magnetometer axis	**X**	**Y**	**Z**	**Avg**
Magnetic random noise [$1\times10^{-5}\ Gauss/\sqrt{Hz}$]	3.92	5.20	3.69	4.27
Rate random walk [$1\times10^{-5}\ Gauss/s/\sqrt{Hz}$]	2.45	7.00	1.32	3.59

Table 5.3: Noise characteristics of magnetometer

Pressure sensor axis	**Z**
Pressure random noise [$1\times10^{-3}\ kPa/\sqrt{Hz}$]	1.65
Rate random walk [$1\times10^{-4}\ kPa/s/\sqrt{Hz}$]	2.99

Table 5.4: Noise characteristics of pressure sensor

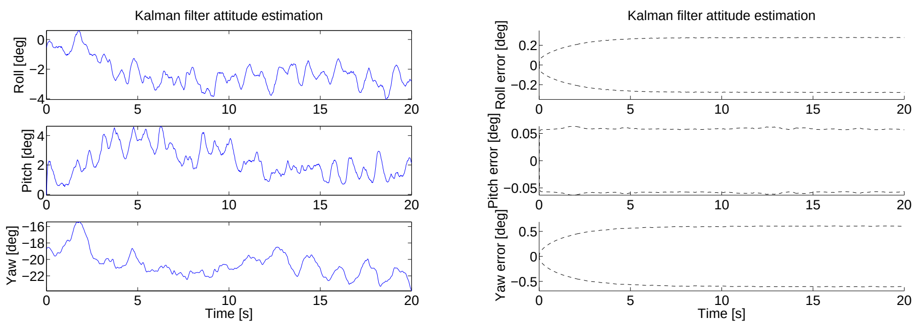
5.2.1 Straight line maneuver

While keeping a constant height and attitude, the sensor module is moved forward in a straight line. The logged data are used as input for the simulation software. The sensor module provides rate gyroscope, accelerometer, magnetometer and pressure sensor measurements (sections 3.2.2 and 3.2.3). By taking the given sensors into consideration and using the information from section 4.3, the following states are expected to be updated:

Attitude	Roll √	Pitch √	Yaw √
Position	North X	East X	Down √
Velocity	North X	East X	Down √

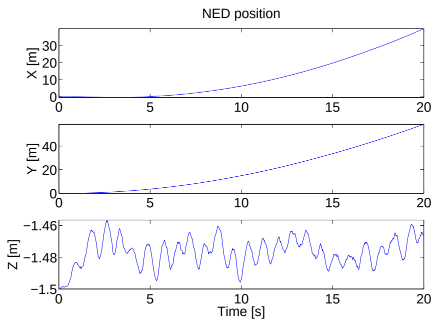
Table 5.5: Expected updated states

The pressure sensor will update the down position state. However, the down velocity state will also be updated, since velocity is equal to differentiated position. To make the simulation results more intuitive, the position and velocity states are displayed in NED axes. The results of the straight line maneuver simulation are illustrated in figures 5.1 to 5.3.

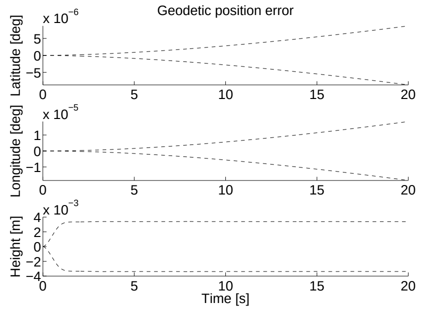


(a) (b)

Figure 5.1: Straight line maneuver simulation with (a) estimated attitude and (b) 2σ error bounds

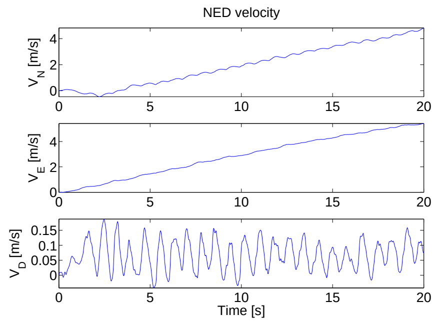


(a)

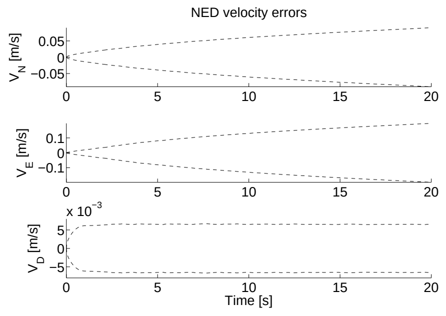


(b)

Figure 5.2: Straight line maneuver simulation with (a) estimated position and (b) 2σ error bounds



(a)



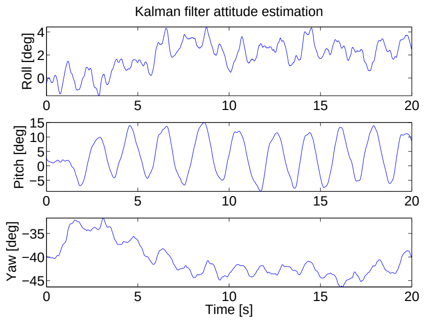
(b)

Figure 5.3: Straight line maneuver simulation with (a) estimated velocity and (b) 2σ error bounds

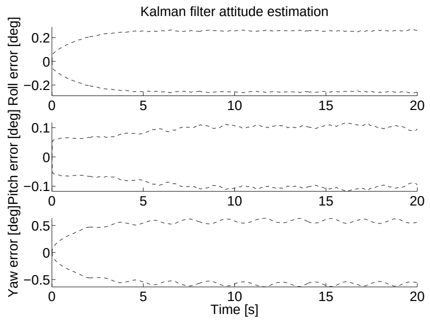
Figures 5.1 to 5.3 confirm the expected results listed in table 5.5. The estimated attitude in figure 5.1(a) changes by small amounts and confirms a constant attitude. Figure 5.1(b) shows that the 2σ (95%) error bounds converge to constant values. The converged error bounds indicate that the estimation errors are kept within bounds by update sensor measurements. Figure 5.2(a) illustrates the updated down position state with its coverged error bounds in figure 5.2(b). The other two position states in figure 5.2(a) show diverging behavior. The diverging behavior is explained by the diverging error bounds in figure 5.2(b), indicating that the estimation errors are not kept within bounds by update sensor measurements. Figure 5.3(a) shows the estimated velocity states. Since velocity is equal to differentiated position, the down velocity state is also updated. However, the other two velocity states diverge.

5.2.2 Wave maneuver

The wave maneuver is executed by performing a straight line maneuver, while inducing positive and negative pitch to create a wave movement. The results listed in table 5.5 are also expected for the wave maneuver, since the same sensor module is used. Figures 5.4 to 5.6 illustrate the wave maneuver simulation results.

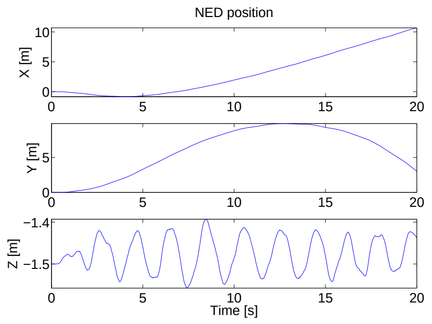


(a)

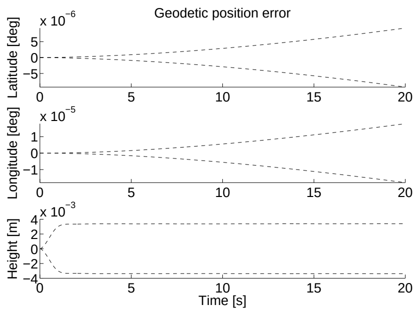


(b)

Figure 5.4: Wave maneuver simulation with (a) estimated attitude and (b) 2σ error bounds

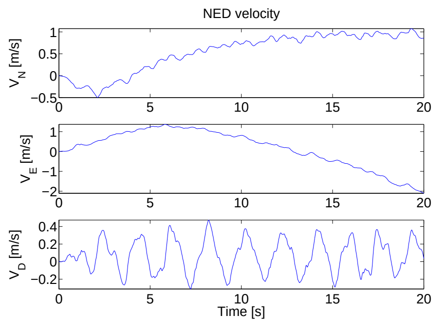


(a)

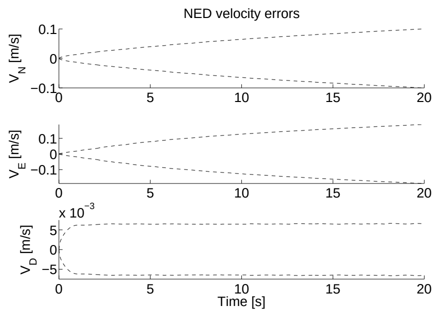


(b)

Figure 5.5: Wave maneuver simulation with (a) estimated position and (b) 2σ error bounds



(a)



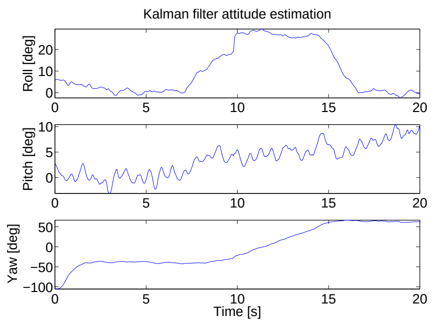
(b)

Figure 5.6: Wave maneuver simulation with (a) estimated velocity and (b) 2σ error bounds

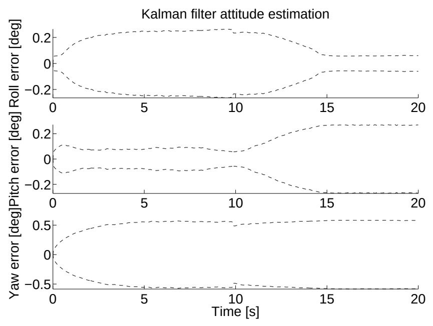
Figures 5.4 to 5.6 confirm the expected results listed in table 5.5. The pitch state in figure 5.4(a) illustrates the wave movement. The converged error bounds in figure 5.4(b) show that all the attitude states are updated by sensor measurements. The wave maneuver also causes the sensor module to experience increases and decreases in height. The variations in height are illustrated by the down position state in figure 5.5(a). Figure 5.5(b) shows that the down position state is updated by sensor measurements. The other two position states in figure 5.5(a) show diverging behavior. The diverging behavior is explained by the diverging error bounds in figure 5.5(b). Figure 5.6(a) shows the estimated velocity states. Since velocity is equal to differentiated position, the down velocity state is also updated. However, the other two velocity states diverge.

5.2.3 Turn maneuver

The turn maneuver is executed by performing a positive roll, positive pitch, positive yaw and negative roll sequence. If the sensor module faces North, then it will face East after the maneuver is executed. The results listed in table 5.5 are also expected for the turn maneuver, since the same sensor module is used. Figures 5.7 to 5.9 illustrate the turn maneuver simulation results.

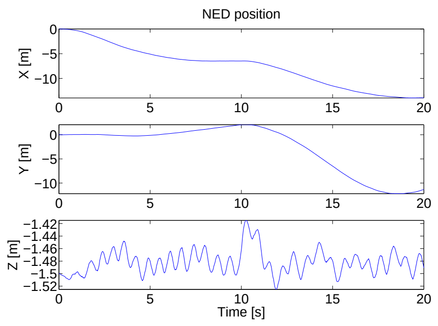


(a)

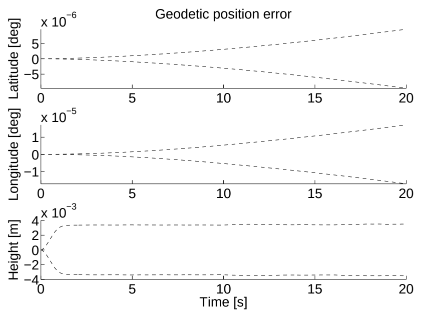


(b)

Figure 5.7: Turn maneuver simulation with (a) estimated attitude and (b) 2σ error bounds

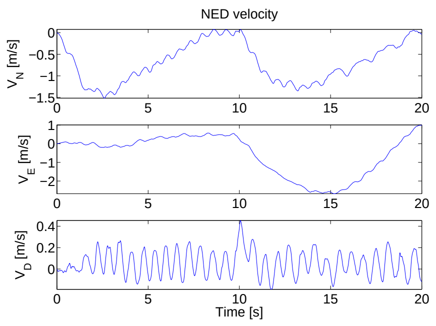


(a)

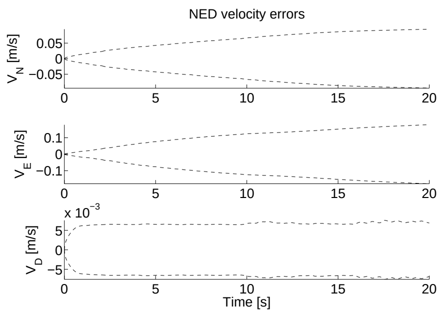


(b)

Figure 5.8: Turn maneuver simulation with (a) estimated position and (b) 2σ error bounds



(a)



(b)

Figure 5.9: Turn maneuver simulation with (a) estimated velocity and (b) 2σ error bounds

Figures 5.7 to 5.9 confirm the expected results listed in table 5.5. The roll, pitch and yaw states in figure 5.7(a) show how the sequence of movements is executed. The converged error bounds in figure 5.7(b) show that all the attitude states are updated by sensor measurements. Figure 5.7(b) also shows that there are variations in the convergence values of the roll and pitch states' error bounds. The variations are caused by dynamic accelerations. The accelerometers help to update the roll and pitch states by providing the MEKF with a gravity (static acceleration) measurement vector (section 4.3.1). However, dynamic accelerations, caused by vehicle movement, can make the measured gravity vector more "noisy" since the accelerometers measure static and dynamic accelerations. A decrease in the quality of the gravity measurement vector will cause the error uncertainty (bound) to converge at a higher value. However, an AUV or any vehicle with slow dynamics will only generate small dynamic accelerations and will have little effect on the gravity measurement vector.

Figure 5.8(b) shows that the down position state is also updated by sensor measurements. The other two position states in figure 5.8(a) show diverging behavior. The diverging behavior is explained by the diverging error bounds in figure 5.8(b). Figure 5.9(a) shows the estimated velocity states. Since velocity is equal to differentiated position, the down velocity state is also updated. However, the other two velocity states diverge.

5.3 Simulations

Section 5.2 shows that the navigation simulation software functions correctly when real-time sensor data are used as input. In section 5.3 simulated sensor data (section 4.2) are used as input for the simulation software. Different sensor combinations are simulated and the RMS (Root Mean Square) estimation errors are compared to one another. The quality of each sensor is also varied by changing its noise characteristics (section 2.5). Each sensor is simulated with high noise and low noise values to simulate a low and high quality sensor respectively. The Allan variance method (section 2.5) and sensor data sheets are used to form realistic sensor noise values. In cases where the Allan variance method and data sheets are not used, hypothetical values are used. The sensors used in the simulations, together with their noise characteristics, are listed in table 5.6. If a sensor's noise characteristics are gathered from a data sheet, then its model number is also listed in table 5.6. Each sensor combination is discussed in a separate case study.

Sensor	**Measurement noise**	**Bias drift**
Rate gyroscope A *Analog Devices ADXRS401 MEMS*	$7.19\times10^{-4}\ rad/s/\sqrt{Hz}$	$3.26\times10^{-5}\ rad/s/s/\sqrt{Hz}$
Rate gyroscope B *Honeywell GG1320AN Laser*	$1.02\times10^{-6}\ rad/s/\sqrt{Hz}$	$1.02\times10^{-7}\ rad/s/s/\sqrt{Hz}$
Accelerometer A *STMicroelectronics LIS3L02AS4 MEMS*	$8.28\times10^{-4}\ m/s^2/\sqrt{Hz}$	$6.62\times10^{-5}\ m/s^2/s/\sqrt{Hz}$
Accelerometer B *Dytran 7500A1 Variable Capacitance*	$7.85\times10^{-5}\ m/s^2/\sqrt{Hz}$	$7.85\times10^{-6}\ m/s^2/s/\sqrt{Hz}$
Magnetometer A *HMO FG-D2 Fluxgate*	$4.27\times10^{-5}\ Gauss/\sqrt{Hz}$	$3.59\times10^{-5}\ Gauss/s/\sqrt{Hz}$
Magnetometer B *MEMSense MAG3 AMR*	$6.80\times10^{-8}\ Gauss/\sqrt{Hz}$	$6.80\times10^{-9}\ Gauss/s/\sqrt{Hz}$
Sonar A	$1.00\times10^{-3}\ m/s/\sqrt{Hz}$	$1.00\times10^{-4}\ m/s/s/\sqrt{Hz}$
Sonar B	$1.00\times10^{-6}\ m/s/\sqrt{Hz}$	$1.00\times10^{-7}\ m/s/s/\sqrt{Hz}$
Tilt sensor A *Crossbow CXTD02 MEMS*	$2.34\times10^{-3}\ rad/\sqrt{Hz}$	$2.34\times10^{-4}\ rad/s/\sqrt{Hz}$
Tilt sensor B *Whirlybird WTILT-02D MEMS*	$1.66\times10^{-4}\ rad/\sqrt{Hz}$	$1.66\times10^{-5}\ rad/s/\sqrt{Hz}$
Water wheel A	$1.00\times10^{-2}\ m/s/\sqrt{Hz}$	$1.00\times10^{-3}\ m/s/s/\sqrt{Hz}$
Water wheel B	$1.00\times10^{-6}\ m/s/\sqrt{Hz}$	$1.00\times10^{-7}\ m/s/s/\sqrt{Hz}$
Pressure sensor A *Freescale MPX4115A*	$1.65\times10^{-3}\ kPa/\sqrt{Hz}$	$2.99\times10^{-4}\ kPa/s/\sqrt{Hz}$
Pressure sensor B *Paroscientific Series 8000*	$5.5\times10^{-5}\ kPa/\sqrt{Hz}$	$5.5\times10^{-6}\ kPa/s/\sqrt{Hz}$
	Position deviation	**Velocity deviation**
GPS A *Ublox 4Hz update*	$4\ m$	$0.1\ m/s$
GPS B *Novatel 20Hz update*	$0.45\ m$	$0.03\ m/s$

Table 5.6: Simulated sensors and their different noise characteristics

5.3.1 Case study 1

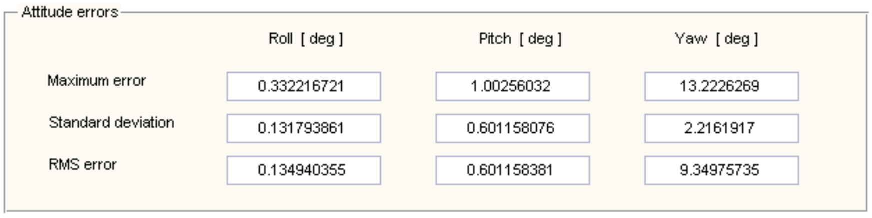
The first four case studies focus on sensor combinations for the MEKF (section 4.3.1.2). The MEKF estimates attitude and thus, only attitude estimation errors are compared. The first case study focuses on a rate gyroscope and accelerometer combination. The rate gyroscopes are the core sensors of the

MEKF, while the accelerometers provide the MEKF with gravity vector update measurements. The expected updated states are listed in table 5.7.

Attitude	Roll $\surd$	Pitch $\surd$	Yaw X
Position	North X	East X	Down X
Velocity	North X	East X	Down X

Table 5.7: Case study 1: Expected updated states

The noise characteristics of rate gyroscope A and accelerometer A (table 5.6) are used and the simulation results are illustrated in figure 5.10.


Attitude errors

	Roll [deg]	Pitch [deg]	Yaw [deg]
Maximum error	0.332216721	1.00256032	13.2226269
Standard deviation	0.131793861	0.601158076	2.2161917
RMS error	0.134940355	0.601158381	9.34975735

Figure 5.10: Simulation results when rate gyroscope A and accelerometer A noise characteristics are used

The quality of the rate gyroscopes and accelerometers are varied and the simulation results are listed in table 5.8. The simulations are executed with and without bias drift to show the effect of sensor bias drift. The results in table 5.8 are with respect to the RMS estimation errors in figure 5.10. The down arrow (↓) means that the RMS error is "x" times smaller than the reference RMS error, while the up arrow (↑) means that the RMS error is "x" times larger than the reference RMS error. The values listed in table 5.8 represent the "x" factor. NSC is the abbreviation for "No Significant Change".

Noise characteristics	**RMS error [Drift on]**	**RMS error [Drift off]**
Rate gyroscope B Accelerometer A	Roll 5.88 ↓ Pitch 25.00 ↓	Roll 25.00 ↓ Pitch 25.00 ↓
Rate gyroscope A Accelerometer B	Roll NSC Pitch NSC	Roll NSC Pitch NSC
Rate gyroscope B Accelerometer B	Roll 7.14 ↓ Pitch 25.00 ↓	Roll 25.00 ↓ Pitch 25.00 ↓

Table 5.8: Case study 1 additional results

Figure 5.10 shows that only the roll and pitch states are accurately updated. The large yaw error can be explained by the fact that the gravity vector has a big downward component. When the AUV is yawing around the downward axis, the measured vector will not change as much as when the AUV is rolling or pitching, resulting in bigger estimation errors in the yaw angle.

Using the results listed in table 5.8, the factor by which the sensors are improved is compared with the average RMS error change. The comparison results are listed in table 5.9. The sensor improvement factor is calculated by using the measurement noise values listed in table 5.6. The RMS error change is also with respect to the RMS values in figure 5.10. NC is the abbreviation for "No Change".

Sensor improvement factor	**Avg RMS error [Drift on]**	**Avg RMS error [Drift off]**
Rate gyroscopes 704.9 Accelerometers NC	15.40 ↓	25.00 ↓
Rate gyroscopes NC Accelerometers 10.5	NSC	NSC
Rate gyroscopes 704.9 Accelerometers 10.5	16.10 ↓	25.00 ↓

Table 5.9: Case study 1 comparison results

The results listed in table 5.9 show that a large rate gyroscope improvement factor of 704.9 is necessary to decrease the average RMS error by a factor of 15.4. By calibrating the sensors and thereby reducing the bias drift, the RMS errors are decreased even more. The small accelerometer improvement factor results in no significant RMS error changes.

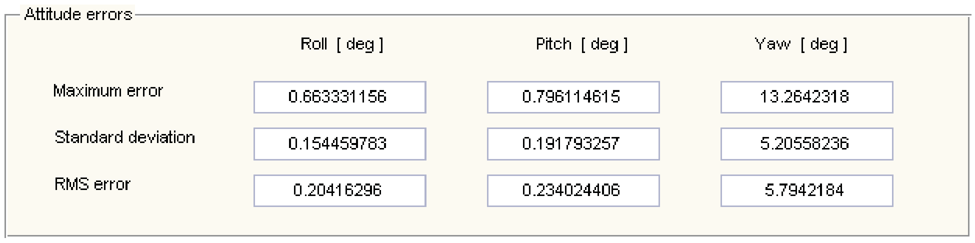
5.3.2 Case study 2

The second case study focuses on a rate gyroscope and tilt sensor combination. The rate gyroscopes are the core sensors of the MEKF, while the tilt sensors provide the MEKF with roll and pitch update measurements. The expected updated states are listed in table 5.10.

Attitude	Roll $\surd$	Pitch $\surd$	Yaw X
Position	North X	East X	Down X
Velocity	North X	East X	Down X

Table 5.10: Case study 2: Expected updated states

The noise characteristics of rate gyroscope A and tilt sensor A (table 5.6) are used and the simulation results are illustrated in figure 5.11.


Attitude errors

	Roll [deg]	Pitch [deg]	Yaw [deg]
Maximum error	0.663331156	0.796114615	13.2642318
Standard deviation	0.154459783	0.191793257	5.20558236
RMS error	0.20416296	0.234024406	5.7942184

Figure 5.11: Simulation results when rate gyroscope A and tilt sensor A noise characteristics are used

The quality of the rate gyroscopes and tilt sensors are varied and the simulation results are listed in table 5.11. The simulations are executed with and without bias drift to show the effect of sensor bias drift. The results in table 5.11 are with respect to the RMS estimation errors in figure 5.11. The down arrow (↓) means that the RMS error is "x" times smaller than the reference RMS error, while the up arrow (↑) means that the RMS error is "x" times larger than the reference RMS error. The values listed in table 5.11 represent the "x" factor. NSC is the abbreviation for "No Significant Change".

Noise characteristics	**RMS error [Drift on]**	**RMS error [Drift off]**
Rate gyroscope B Tilt sensor A	Roll 1.25 ↓ Pitch 1.20 ↓	Roll 25.00 ↓ Pitch 16.67 ↓
Rate gyroscope A Tilt sensor B	Roll 5.26 ↓ Pitch 5.88 ↓	Roll 5.56 ↓ Pitch 6.67 ↓
Rate gyroscope B Tilt sensor B	Roll 11.10 ↓ Pitch 12.50 ↓	Roll 14.28 ↓ Pitch 25.00 ↓

Table 5.11: Case study 2 additional results

Figure 5.11 shows that only the roll and pitch states are accurately updated. Using the results listed in table 5.11, the factor by which the sensors are improved is compared with the average RMS error change. The comparison results are listed in table 5.12. The sensor improvement factor is calculated by using the measurement noise values listed in table 5.6. The RMS error change is also with respect to the RMS values in figure 5.11.

Sensor improvement factor	**Avg RMS error [Drift on]**	**Avg RMS error [Drift off]**
Rate gyroscopes 704.9 Tilt sensors NC	1.23 ↓	20.83 ↓
Rate gyroscopes NC Tilt sensors 14.1	5.57 ↓	6.12 ↓
Rate gyroscopes 704.9 Tilt sensors 14.1	11.80 ↓	19.64 ↓

Table 5.12: Case study 2 comparison results

The results listed in table 5.12 show that by increasing the rate gyroscope quality by a factor of 704.9 and keeping the quality of the tilt sensors the same, the average RMS error is decreased by a factor of only 1.23. The previous result shows that it is important to keep the variation in quality between different navigation sensors small, especially if no drift calibration is done. The small variation is necessary, otherwise the low-quality sensors (tilt sensors) may influence the effect of the high-quality sensors (rate gyroscopes) when they are combined in a navigation algorithm. By calibrating the sensors and thereby reducing the bias drift, the RMS errors are decreased even more.

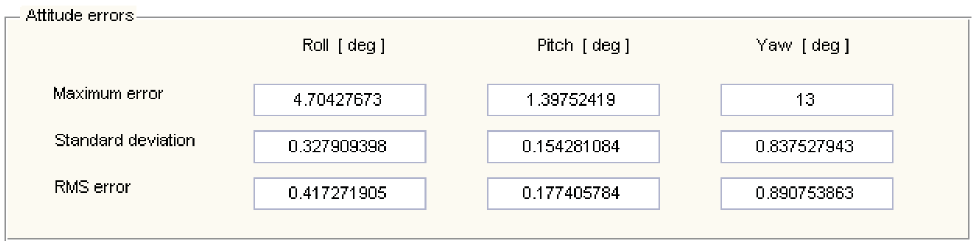
5.3.3 Case study 3

The third case study focuses on a rate gyroscope, tilt sensor and magnetometer combination. The rate gyroscopes are the core sensors of the MEKF. The tilt sensors provide the MEKF with roll and pitch update measurements, while the magnetometer provides the MEKF with magnetic field vector update measurements. The expected updated states are listed in table 5.13.

Attitude	Roll $\surd$	Pitch $\surd$	Yaw $\surd$
Position	North X	East X	Down X
Velocity	North X	East X	Down X

Table 5.13: Case study 3: Expected updated states

The noise characteristics of rate gyroscope A, tilt sensor A and magnetometer A (table 5.6) are used and the simulation results are illustrated in figure 5.12. The quality of the rate gyroscopes, tilt sensors and magnetometer are varied and the simulation results are listed in table 5.14. The simulations are executed with and without bias drift to show the effect of sensor bias drift. The


Attitude errors

	Roll [deg]	Pitch [deg]	Yaw [deg]
Maximum error	4.70427673	1.39752419	13
Standard deviation	0.327909398	0.154281084	0.837527943
RMS error	0.417271905	0.177405784	0.890753863

Figure 5.12: Simulation results when rate gyroscope A, tilt sensor A and magnetometer A noise characteristics are used

results in table 5.14 are with respect to the RMS estimation errors in figure 5.12. The down arrow (↓) means that the RMS error is "x" times smaller than the reference RMS error, while the up arrow (↑) means that the RMS error is "x" times larger than the reference RMS error. The values listed in table 5.14 represent the "x" factor.

Noise characteristics	**RMS error [Drift on]**	**RMS error [Drift off]**
Rate gyroscope B Tilt sensor A Magnetometer A	Roll 5.90 ↑ Pitch 7.47 ↑ Yaw 7.20 ↑	Roll 5.60 ↑ Pitch 7.44 ↑ Yaw 7.28 ↑
Rate gyroscope A Tilt sensor B Magnetometer A	Roll 4.00 ↓ Pitch 1.75 ↓ Yaw NSC	Roll 8.33 ↓ Pitch 5.88 ↓ Yaw 4.00 ↓
Rate gyroscope A Tilt sensor A Magnetometer B	Roll 1.28 ↓ Pitch 1.33 ↓ Yaw NSC	Roll 1.56 ↓ Pitch 2.22 ↓ Yaw 1.33 ↓
Rate gyroscope B Tilt sensor B Magnetometer B	Roll 2.78 ↑ Pitch 3.76 ↑ Yaw 3.35 ↑	Roll 2.78 ↑ Pitch 3.76 ↑ Yaw 3.35 ↑

Table 5.14: Case study 3 additional results

Figure 5.12 shows that the roll, pitch and yaw states are accurately updated. The larger yaw error can be explained by the fact that the magnetic field vector has a big downward component. When the AUV is yawing around the downward axis, the measured vector will not change as much as when the AUV is rolling or pitching, resulting in bigger estimation errors on the yaw angle.

Using the results listed in table 5.14, the factor by which the sensors are improved is compared with the average RMS error change. The comparison results are listed in table 5.15. The sensor improvement factor is calculated by using the measurement noise values listed in table 5.6. The RMS error change is also with respect to the RMS values in figure 5.12.

Sensor improvement factor	**Avg RMS error [Drift on]**	**Avg RMS error [Drift off]**
Rate gyroscopes 704.9 Tilt sensors NC Magnetometer NC	6.86 ↑	6.77 ↑
Rate gyroscopes NC Tilt sensors 14.1 Magnetometer NC	2.25 ↓	6.07 ↓
Rate gyroscopes NC Tilt sensors NC Magnetometer 627.9	1.20 ↓	1.70 ↓
Rate gyroscopes 704.9 Tilt sensors 14.1 Magnetometer 627.9	3.29 ↑	3.29 ↑

Table 5.15: Case study 3 comparison results

The results listed in table 5.15 show that by increasing the rate gyroscope quality by a factor of 704.9 and keeping the quality of the tilt sensors and magnetometer the same, the average RMS error is increased by a factor of 6.86. The average RMS error is also increased when the quality factor of all three sensors are increased. The previous two result show that it is important to keep the variation in quality between different navigation sensors small. The small variation is necessary, otherwise the sensor measurements may influence the MEKF negatively and cause the RMS error to increase instead of decrease. By calibrating the sensors and thereby reducing the bias drift, the RMS errors are decreased even more. The results of this case study also show that the magnetometer helps to update the yaw state.

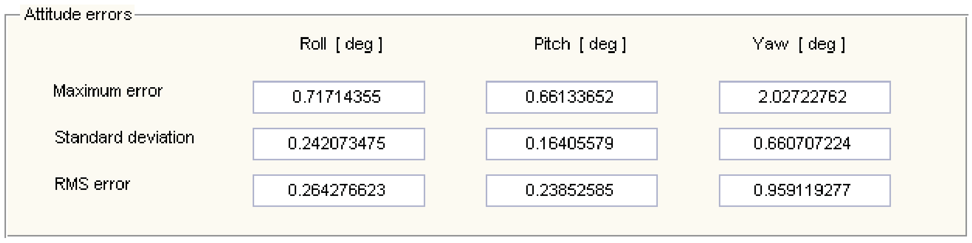
5.3.4 Case study 4

The fourth case study focuses on a rate gyroscope, accelerometer and magnetometer combination. The rate gyroscopes are the core sensors of the MEKF. The accelerometers provide the MEKF with gravity vector update measurements, while the magnetometer provides the MEKF with magnetic field vector update measurements. The expected updated states are listed in table 5.16.

Attitude	Roll √	Pitch √	Yaw √
Position	North X	East X	Down X
Velocity	North X	East X	Down X

Table 5.16: Case study 4: Expected updated states

The noise characteristics of rate gyroscope A, accelerometer A and magnetometer A (table 5.6) are used and the simulation results are illustrated in figure 5.13.



Attitude errors

	Roll [deg]	Pitch [deg]	Yaw [deg]
Maximum error	0.71714355	0.66133652	2.02722762
Standard deviation	0.242073475	0.16405579	0.660707224
RMS error	0.264276623	0.23852585	0.959119277

Figure 5.13: Simulation results when rate gyroscope A, accelerometer A and magnetometer A noise characteristics are used

The quality of the rate gyroscopes, accelerometers and magnetometer are varied and the simulation results are listed in table 5.17. The simulations are executed with and without bias drift to show the effect of sensor bias drift. The results in table 5.17 are with respect to the RMS estimation errors in figure 5.13. The down arrow (↓) means that the RMS error is "x" times smaller than the reference RMS error, while the up arrow (↑) means that the RMS error is "x" times larger than the reference RMS error. The values listed in table 5.17 represent the "x" factor.

Figure 5.13 shows that the roll, pitch and yaw states are accurately updated. The larger yaw error can be explained by the fact that the magnetic field and gravity vectors have a big downward component. When the AUV is yawing around the downward axis, the measured vectors will not change as much as when the AUV is rolling or pitching, resulting in bigger estimation errors in the yaw angle.

Using the results listed in table 5.17, the factor by which the sensors are improved is compared with the average RMS error change. The comparison results are listed in table 5.18. The sensor improvement factor is calculated

Noise characteristics	RMS error [Drift on]	RMS error [Drift off]
Rate gyroscope B Accelerometer A Magnetometer A	Roll 1.27 ↓ Pitch 1.10 ↓ Yaw 1.39 ↓	Roll 14.29 ↓ Pitch 20.00 ↓ Yaw 33.33 ↓
Rate gyroscope A Accelerometer B Magnetometer A	Roll NSC Pitch NSC Yaw NSC	Roll 1.15 ↓ Pitch 2.08 ↓ Yaw 1.69 ↓
Rate gyroscope A Accelerometer A Magnetometer B	Roll 1.16 ↓ Pitch 2.17 ↓ Yaw 1.69 ↓	Roll 1.16 ↓ Pitch 2.17 ↓ Yaw 1.72 ↓
Rate gyroscope B Accelerometer B Magnetometer B	Roll 7.69 ↓ Pitch 12.50 ↓ Yaw 14.29 ↓	Roll 7.69 ↓ Pitch 12.50 ↓ Yaw 14.29 ↓

Table 5.17: Case study 4 additional results

by using the measurement noise values listed in table 5.6. The RMS error change is also with respect to the RMS values in figure 5.13.

Sensor improvement factor	Avg RMS error [Drift on]	Avg RMS error [Drift off]
Rate gyroscopes 704.9 Accelerometers NC Magnetometer NC	1.25 ↓	22.54 ↓
Rate gyroscopes NC Accelerometers 10.5 Magnetometer NC	NSC	1.64 ↓
Rate gyroscopes NC Accelerometers NC Magnetometer 627.9	1.67 ↓	1.68 ↓
Rate gyroscopes 704.9 Accelerometers 10.5 Magnetometer 627.9	11.49 ↓	11.49 ↓

Table 5.18: Case study 4 comparison results

The results of this case study once again show the importance of sensor calibration. By calibrating the sensors and thereby reducing the bias drift, the RMS errors in some cases are significantly decreased . The case study results also show that the magnetometer helps to update the yaw state. The sensor combination used in this case study shows the best overall performance when compared to the results of case studies one to three. Case study three also shows good results. However, accelerometer sensor data are useful for the MEKF and PVEKF, while tilt sensor data are only useful for the MEKF. The rate gyroscope, accelerometer and magnetometer sensor combination is recommended for the attitude estimation of an AUV or vehicle with slow dynamics. By performing sensor calibration, the attitude estimation errors will decrease even more.

The next four case studies focus on sensor combinations for the PVEKF (section 4.3.2.2). Thus, only position and velocity estimation errors are compared. However, the PVEKF requires the estimated MEKF quaternion states. The rate gyroscope, accelerometer and magnetometer sensor combination is used with the MEKF to estimate the required quaternion states.

5.3.5 Case study 5

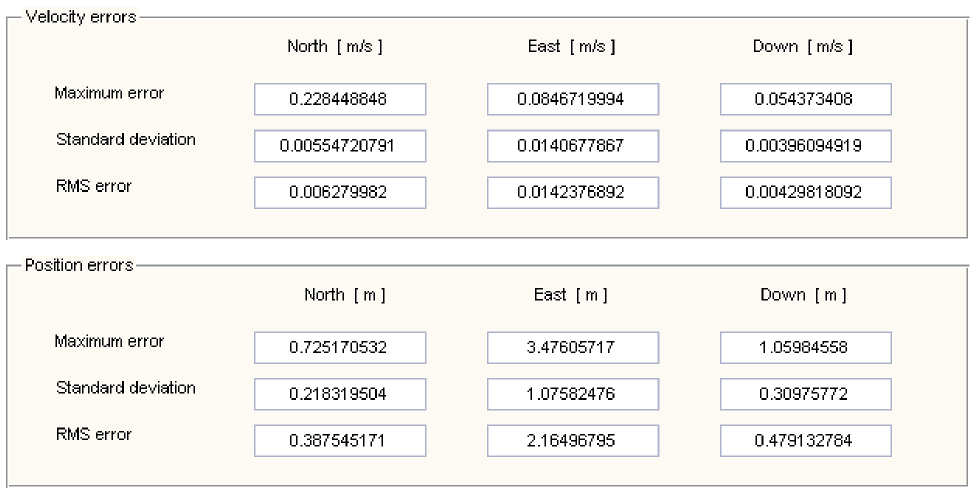
The fifth case study focuses on an attitude sensor and sonar sensor combination. The attitude sensors (rate gyroscopes, accelerometers and magnetometer) are used with the MEKF to estimate the required quaternion states. The sonar provides the PVEKF with body velocity update measurements. The expected updated states are listed in table 5.19. The attitude sensors are discussed in study cases one to four, thus only PVEKF estimation errors are compared in this case study.

Attitude	Roll √	Pitch √	Yaw √
Position	North X	East X	Down X
Velocity	North √	East √	Down √

Table 5.19: Case study 5: Expected updated states

The noise characteristics of rate attitude sensors A (rate gyroscope A, accelerometer A, magnetometer A) and sonar A (table 5.6) are used and the simulation results are illustrated in figure 5.14.

The quality of the attitude sensors and sonar are varied and the simulation results are listed in table 5.20. The simulations are executed with and without bias drift to show the effect of sensor bias drift. The results in table 5.20 are with respect to the RMS estimation errors in figure 5.14. The down arrow ($\downarrow$) means that the RMS error is "x" times smaller than the reference RMS error, while the up arrow ($\uparrow$) means that the RMS error is "x" times larger than the reference RMS error. The values listed in table 5.20 represent the "x" factor.



Velocity errors

	North [m/s]	East [m/s]	Down [m/s]
Maximum error	0.228448848	0.0846719994	0.054373408
Standard deviation	0.00554720791	0.0140677867	0.00396094919
RMS error	0.006279982	0.0142376892	0.00429818092

Position errors

	North [m]	East [m]	Down [m]
Maximum error	0.725170532	3.47605717	1.05984558
Standard deviation	0.218319504	1.07582476	0.30975772
RMS error	0.387545171	2.16496795	0.479132784

Figure 5.14: Simulation results when attitude sensors A and sonar A noise characteristics are used

Noise characteristics	RMS error [Drift on]	RMS error [Drift off]
Attitude sensors B Sonar A	V_N 2.23 ↑ V_E 2.27 ↓ V_D 1.18 ↓	V_N 2.20 ↑ V_E 2.27 ↓ V_D 1.30 ↓
Attitude sensors A Sonar B	V_N 2.04 ↓ V_E 1.04 ↓ V_D 1.22 ↓	V_N 2.04 ↓ V_E 1.92 ↓ V_D 2.70 ↓
Attitude sensors B Sonar B	V_N 2.08 ↓ V_E 11.11 ↓ V_D 8.33 ↓	V_N 2.08 ↓ V_E 11.11 ↓ V_D 8.33 ↓

Table 5.20: Case study 5 additional results

Figure 5.14 shows that the north, east and down velocity states are accurately updated. Using the results listed in table 5.20, the factor by which the sensors are improved is compared with the average RMS error change. The comparison results are listed in table 5.21. The sensor improvement factor is calculated by using the measurement noise values listed in table 5.6. The RMS error change is also with respect to the RMS values in figure 5.14.

The results listed in table 5.21 show that the velocity RMS errors are decreased the most when the quality of the attitude sensors and sonar is increased. However, by increasing the sonar quality by a factor of 1000 and keeping the quality of the attitude sensors the same, the average RMS error is only decreased by a factor of 1.43. By calibrating the sensors and thereby reducing the bias drift, the RMS errors are decreased even more. Accurate

Sensor improvement factor	Avg RMS error [Drift on]	Avg RMS error [Drift off]
Attitude sensors B Sonar NC	0.407 ↓	0.457 ↓
Attitude sensors NC Sonar 1000	1.43 ↓	2.22 ↓
Attitude sensors B Sonar 1000	7.17 ↓	7.17 ↓

Table 5.21: Case study 5 comparison results

sonar velocity updates also improve the estimation of the position states, since position is equal to integrated velocity. However, any velocity noise will also be integrated and will result in position random walk. The sonar provides velocity measurements with respect to the ocean floor, thus the accuracy of the sonar largely depends on the terrain where it is used.

5.3.6 Case study 6

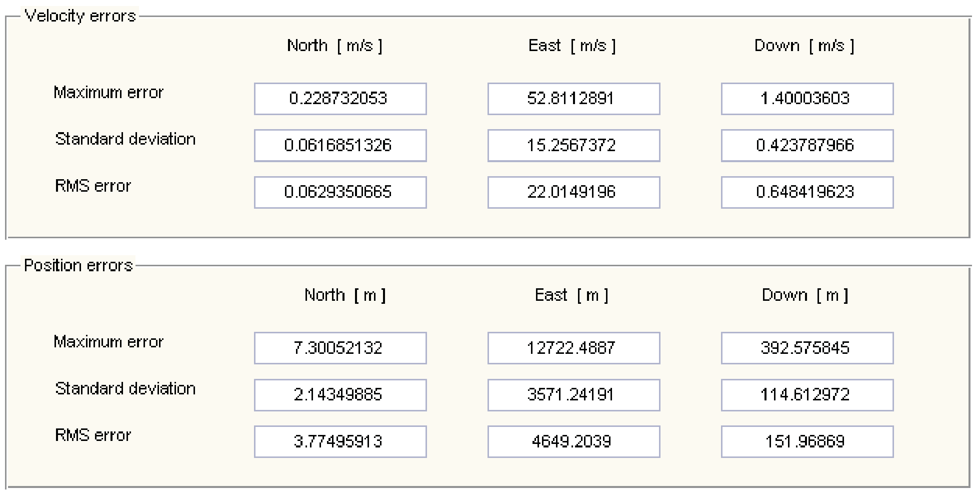
The sixth case study focuses on an attitude sensor and water wheel sensor combination. The attitude sensors (rate gyroscopes, accelerometers and magnetometer) are used with the MEKF to estimate the required quaternion states. The water wheel provides the PVEKF with forward body velocity update measurements. The expected updated states are listed in table 5.22. The attitude sensors are discussed in study cases one to four, thus only PVEKF estimation errors are compared in this case study.

Attitude	Roll $\surd$	Pitch $\surd$	Yaw $\surd$
Position	North X	East X	Down X
Velocity	North $\surd$	East X	Down X

Table 5.22: Case study 6: Expected updated states

The noise characteristics of attitude sensors A (rate gyroscope A, accelerometer A, magnetometer A) and water wheel A (table 5.6) are used and the simulation results are illustrated in figures 5.15 and 5.16. A very weak ocean current is modeled for the water wheel simulations.

The quality of the attitude sensors and water wheel are varied and the simulation results are listed in table 5.23. The simulations are executed with and without bias drift to show the effect of sensor bias drift. The results in table



Velocity errors

	North [m/s]	East [m/s]	Down [m/s]
Maximum error	0.228732053	52.8112891	1.40003603
Standard deviation	0.0616851326	15.2567372	0.423787966
RMS error	0.0629350665	22.0149196	0.648419623

Position errors

	North [m]	East [m]	Down [m]
Maximum error	7.30052132	12722.4887	392.575845
Standard deviation	2.14349885	3571.24191	114.612972
RMS error	3.77495913	4649.2039	151.96869

Figure 5.15: Simulation results when attitude sensors A and water wheel A noise characteristics are used

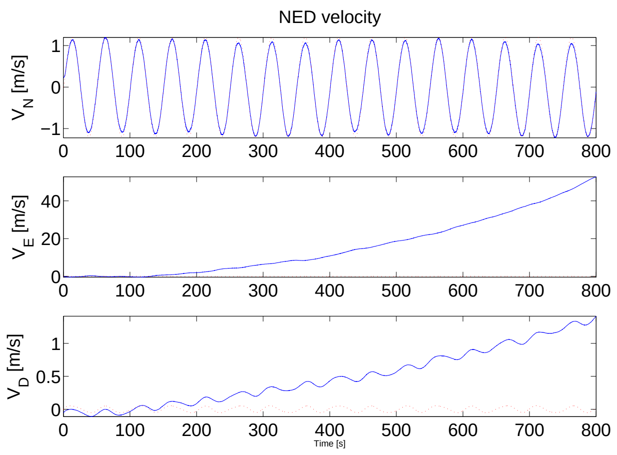


Figure 5.16: Estimated NED velocity

5.23 are with respect to the RMS estimation errors in figure 5.15. The down arrow (↓) means that the RMS error is "x" times smaller than the reference RMS error, while the up arrow (↑) means that the RMS error is "x" times larger than the reference RMS error. The values listed in table 5.23 represent the "x" factor.

Noise characteristics	**RMS error [Drift on]**	**RMS error [Drift off]**
Attitude sensors B Water wheel A	V_N NSC	V_N 1.02 ↓
Attitude sensors A Water wheel B	V_N 1.20 ↓	V_N 1.22 ↓
Attitude sensors B Water wheel B	V_N 1.22 ↓	V_N 1.22 ↓

Table 5.23: Case study 6 additional results

Figure 5.16 shows that the north velocity state is accurately updated. The east and down velocity states are not updated and diverge over time. Using the results listed in table 5.23, the factor by which the sensors are improved is compared with the average RMS error change. The comparison results are listed in table 5.24. The sensor improvement factor is calculated by using the measurement noise values listed in table 5.6. The RMS error change is also with respect to the RMS values in figure 5.15.

Sensor improvement factor	**Avg RMS error [Drift on]**	**Avg RMS error [Drift off]**
Attitude sensors B Water wheel NC	NSC	1.02 ↓
Attitude sensors NC Water wheel 10000	1.20 ↓	1.22 ↓
Attitude sensors B Water wheel 10000	1.22 ↓	1.22 ↓

Table 5.24: Case study 6 comparison results

The results listed in table 5.24 show that by increasing the water wheel quality by a factor of 10000 and keeping the quality of the attitude sensors the same, the average RMS error is only decreased by a factor of 1.20. By calibrating the sensors and thereby reducing the bias drift, the RMS errors are decreased even more. Accurate water wheel velocity updates also improve the estimation of the north position state, since position is equal to integrated velocity. However, any velocity noise will also be integrated and will result in position random walk. The water wheel provides velocity measurements with respect to the water current flow, thus the accuracy of the water wheel largely depends on the strength and direction of the current.

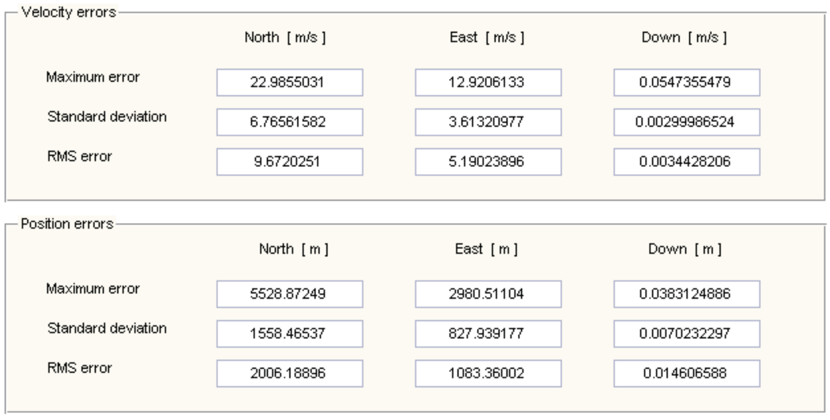
5.3.7 Case study 7

The sixth case study focuses on an attitude sensor and pressure sensor combination. The attitude sensors (rate gyroscopes, accelerometers and magnetometer) are used with the MEKF to estimate the required quaternion states. The pressure sensor provides the PVEKF with depth update measurements. The expected updated states are listed in table 5.25. The attitude sensors are discussed in study cases one to four, thus only PVEKF estimation errors are compared in this case study.

Attitude	Roll $\surd$	Pitch $\surd$	Yaw $\surd$
Position	North X	East X	Down $\surd$
Velocity	North X	East X	Down X

Table 5.25: Case study 7: Expected updated states

The noise characteristics of rate attitude sensors A (rate gyroscope A, accelerometer A, magnetometer A) and pressure sensor A (table 5.6) are used and the simulation results are illustrated in figures 5.17.



Velocity errors

	North [m/s]	East [m/s]	Down [m/s]
Maximum error	22.9855031	12.9206133	0.0547355479
Standard deviation	6.76561582	3.61320977	0.00299986524
RMS error	9.6720251	5.19023896	0.0034428206

Position errors

	North [m]	East [m]	Down [m]
Maximum error	5528.87249	2980.51104	0.0383124886
Standard deviation	1558.46537	827.939177	0.0070232297
RMS error	2006.18896	1083.36002	0.014606588

Figure 5.17: Simulation results when attitude sensors A and pressure sensor A noise characteristics are used

The quality of the attitude sensors and pressure sensor are varied and the simulation results are listed in table 5.26. The simulations are executed with and without bias drift to show the effect of sensor bias drift. The results in table 5.26 are with respect to the RMS estimation errors in figure 5.15. The down arrow (↓) means that the RMS error is "x" times smaller than the reference RMS error, while the up arrow (↑) means that the RMS error is "x" times larger than the reference RMS error. The values listed in table 5.26 represent the "x" factor.

Noise characteristics	**RMS error [Drift on]**	**RMS error [Drift off]**
Attitude sensors B Pressure sensor A	Z_D NSC	Z_D 1.92 ↓
Attitude sensors A Pressure sensor B	Z_D 14.29 ↓	Z_D 16.67 ↓
Attitude sensors B Pressure sensor B	Z_D 16.67 ↓	Z_D 16.67 ↓

Table 5.26: Case study 7 additional results

Figure 5.17 shows that the down position state is accurately updated. The north and east position states are not updated and diverge over time. Using the results listed in table 5.26, the factor by which the sensors are improved is compared with the average RMS error change. The comparison results are listed in table 5.27. The sensor improvement factor is calculated by using the measurement noise values listed in table 5.6. The RMS error change is also with respect to the RMS values in figure 5.17.

Sensor improvement factor	**Avg RMS error [Drift on]**	**Avg RMS error [Drift off]**
Attitude sensors B Pressure sensor NC	NSC	1.92 ↓
Attitude sensors NC Pressure sensor 30	14.29 ↓	16.67 ↓
Attitude sensors B Pressure sensor 30	16.67 ↓	16.67 ↓

Table 5.27: Case study 7 comparison results

The results listed in table 5.27 show that by increasing the pressure sensor quality by a factor of 30 and keeping the quality of the attitude sensors the same, the average RMS error is decreased by a factor of 14.29. By calibrating the sensors and thereby reducing the bias drift, the RMS errors are decreased even more. Accurate pressure sensor depth updates also improve the estimation of the down velocity state, since velocity is equal to differentiated position.

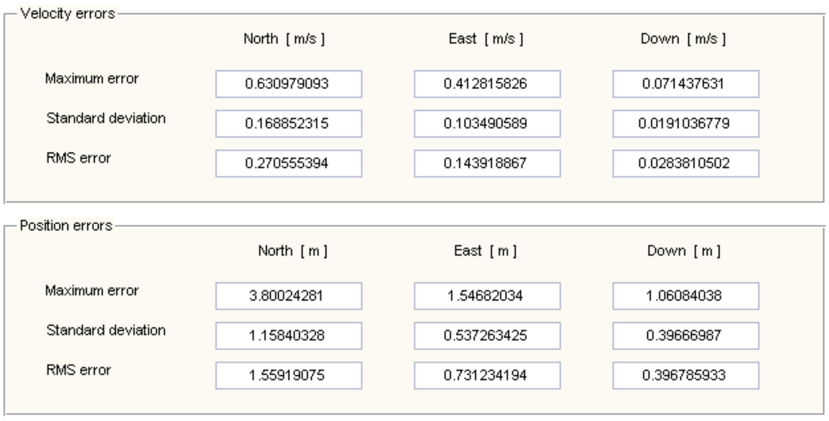
5.3.8 Case study 8

The eighth case study focuses on an attitude sensor and GPS combination. The attitude sensors (rate gyroscopes, accelerometers and magnetometer) are used with the MEKF to estimate the required quaternion states. The GPS provides the PVEKF with position and velocity update measurements. The expected updated states are listed in table 5.28. The attitude sensors are discussed in study cases one to four, thus only PVEKF estimation errors are compared in this case study.

Attitude	Roll $\surd$	Pitch $\surd$	Yaw $\surd$
Position	North $\surd$	East $\surd$	Down $\surd$
Velocity	North $\surd$	East $\surd$	Down $\surd$

Table 5.28: Case study 8: Expected updated states

The noise characteristics of rate attitude sensors A (rate gyroscope A, accelerometer A, magnetometer A) and GPS A (table 5.6) are used and the simulation results are illustrated in figures 5.18.



Velocity errors

	North [m/s]	East [m/s]	Down [m/s]
Maximum error	0.630979093	0.412815826	0.071437631
Standard deviation	0.168852315	0.103490589	0.0191036779
RMS error	0.270555394	0.143918867	0.0283810502

Position errors

	North [m]	East [m]	Down [m]
Maximum error	3.80024281	1.54682034	1.06084038
Standard deviation	1.15840328	0.537263425	0.39666987
RMS error	1.55919075	0.731234194	0.396785933

Figure 5.18: Simulation results when attitude sensors A and GPS A noise characteristics are used

The quality of the attitude sensors and GPS are varied and the simulation results are listed in table 5.29. The simulations are executed with and without bias drift to show the effect of sensor bias drift. The results in table 5.29 are with respect to the RMS estimation errors in figure 5.18. The down arrow (⇂) means that the RMS error is "x" times smaller than the reference RMS error, while the up arrow (↾) means that the RMS error is "x" times larger than the reference RMS error. The values listed in table 5.29 represent the "x" factor.

Noise characteristics	**RMS error [Drift on]**	**RMS error [Drift off]**
Attitude sensors B GPS A	V_N 2.44 ⇂ V_E NSC V_D 1.02 ⇂ X_N 3.88 ↾ Y_E 5.15 ↾ Z_D 4.10 ↾	V_N 2.70 ⇂ V_E NSC V_D 1.52 ⇂ X_N 3.74 ↾ Y_E 5.5 ↾ Z_D 3.70 ↾
Attitude sensors A GPS B	V_N 5.26 ⇂ V_E 6.67 ⇂ V_D 5.26 ⇂ X_N 5.56 ⇂ Y_E 20 ⇂ Z_D 12.5 ⇂	V_N 7.69 ⇂ V_E 11.11 ⇂ V_D 7.14 ⇂ X_N 5.56 ⇂ Y_E 25 ⇂ Z_D 12.5 ⇂
Attitude sensors B GPS B	V_N 12.5 ⇂ V_E 6.25 ⇂ V_D 6.25 ⇂ X_N 5.26 ⇂ Y_E 6.67 ⇂ Z_D 7.69 ⇂	V_N 12.5 ⇂ V_E 5.88 ⇂ V_D 11.11 ⇂ X_N 5.26 ⇂ Y_E 6.25 ⇂ Z_D 8.33 ⇂

Table 5.29: Case study 8 additional results

Figure 5.18 shows that the north, east and down position and velocity states are accurately updated. Using the results listed in table 5.29, the factor by which the sensors are improved is compared with the average RMS error change. The comparison results are listed in table 5.30. The sensor improvement factor is calculated by using the GPS position and velocity noise deviation values listed in table 5.6. The RMS error change is also with respect to the RMS values in figure 5.18.

Sensor improvement factor	**Avg RMS error [Drift on]**	**Avg RMS error [Drift off]**
Attitude sensors B GPS NC	Vel 1.49 ⇂ Pos 4.38 ↾	Vel 1.74 ⇂ Pos 4.31 ↾
Attitude sensors NC GPS pos 8.9 vel 3.3	Vel 5.73 ⇂ Pos 12.69 ⇂	Vel 8.65 ⇂ Pos 14.35 ⇂
Attitude sensors B GPS pos 8.9 vel 3.3	Vel 8.33 ⇂ Pos 6.54 ⇂	Vel 9.83 ⇂ Pos 6.61 ⇂

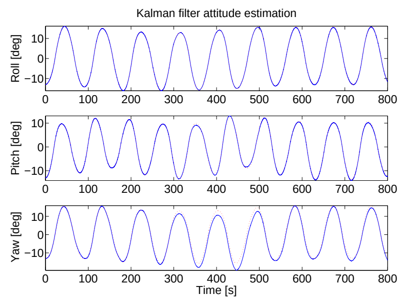
Table 5.30: Case study 8 comparison results

The results listed in table 5.30 show that the combination of high-quality attitude sensors and a low-quality GPS causes an increase in the position RMS error. However, by using a higher-quality GPS the position and velocity RMS errors are decreased. By calibrating the sensors and thereby reducing the bias drift, the RMS errors are decreased even more.

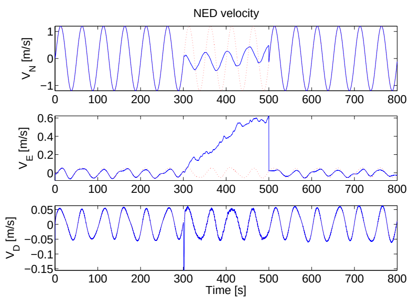
The sensor combination used in this case study shows the best overall performance when compared to the results of case studies five to seven. However, GPS signals are not available under water. With no GPS signals, a sonar and pressure sensor combination is recommended to provide the necessary PVEKF sensor update measurements. However, the sonar and pressure sensor combination will only update the north, east and down velocity states and the down position state. The accuracy of the sonar also largely depends on the terrain where it is used. Thus, it is recommended that the AUV surface at fixed time intervals to receive GPS updates. If surfacing is not possible, then a buoy system is recommended. Buoys in a harbor environment can be designed to transmit position data under water and provide an AUV with the necessary updates.

5.3.9 Case study 9

This case study shows an example of simulated sensor failure. A rate gyroscope, accelerometer and magnetometer combination is used for the MEKF, while a sonar and pressure sensor combination is used for the PVEKF. The



(a)



(b)

Figure 5.19: Sensor failure simulation with (a) estimated attitude and (b) NED velocity

magnetometer and sonar are simulated to fail between 300 and 500 seconds and the results are illustrated in figure 5.19. The yaw state is slightly affected by the failure of the magnetometer, while the north and east velocity states are significantly affected by the failure of the sonar. The pressure sensor updates the down position state and that is why the down velocity state is still updated after the sonar failure.

5.4 Summary

Chapter 5 focuses on the results of navigation simulations where different sensor combinations are used. The quality of each sensor is also varied. A rate gyroscope, accelerometer, magnetometer, pressure sensor and sonar sensor combination with periodic position updates via GPS or a buoy positioning system is recommended for accurate AUV navigation. The navigation performance sensitivity with respect to the quality variation of each recommended sensor is listed in tables 5.18, 5.21, 5.27 and 5.30. Sensor bias drift calibration also helps to improve navigation accuracy.

Chapter 6

Project Summary and Recommendations

6.1 Summary

Navigation simulation software capable of using real-time and simulated sensor data is developed in this project. The simulation software is used to simulate different sensor combinations and therefore evaluate the best configuration for each AUV mission. A sensor module is also developed to capture real-time sensor data. The sensor module includes a low-cost 6-degree-of-freedom inertial measurement module (rate gyroscopes and accelerometers), a three-axes magnetometer and other sensor interfaces. The real-time sensor data are used to test and calibrate the navigation simulation software.

Different sensor combinations are evaluated by using the navigation simulation software with simulated sensor data as input. The quality of each sensor is varied by changing its noise characteristics. A rate gyroscope, accelerometer, magnetometer, pressure sensor and sonar sensor combination with periodic position updates via GPS or a buoy positioning system is recommended for accurate AUV navigation. A navigation performance sensitivity study, with respect to the quality variation of the sensors and sensor combinations used in this project, is also done and the results are listed in Chapter 5.

All the requirements specified in Chapter 1 have been fulfilled by the work done in this project. The largest contribution of this project is a navigation performance sensitivity study with respect to the quality variation of sensors and sensor combinations. The performance study, together with the developed simulation tools, will simplify the process of selecting a sensor combination to fulfill a specific navigation accuracy requirement.

6.2 Recommendations

For future work on AUV navigation simulation, more accurate sensor models can be used. Only measurement noise and bias drift are modeled in this project. More noise characteristics (see [14]) can be added to the current sensor models. A sonar provides measurements with respect to the ocean floor, while a water wheel provides measurements with respect to the water current. Thus, accurate environmental modeling is also recommended for future work. By increasing the accuracy of the sensor and environmental models, more accurate navigation simulations are made possible.

For simulation purposes the body forces and moments are usually generated by the vehicle model. However, an AUV model is not available for this project. An AUV model is recommended for future work. The model must receive navigation commands as input and generate the necessary forces and moments as output. An AUV model will make it possible to simulate a wide variety of AUV maneuvers by simply providing the model with the necessary navigation commands.

An AUV must be used for future real-time data capturing. Data from an AUV in its operating environment can provide more accurate real-time sensor data. The sensor data can also help to model environmental aspects more accurately.

Bibliography

[1] Lok, J., Mine-countermeasures forces emerge from splendid isolation, Jane's International Defence Review, 2006.

[2] Britting, K.R., Inertial Navigation Systems Analysis, 1971.

[3] Biezad, D.J., Integrated Navigation and Guidance Systems, 1999.

[4] Titterton, D.H., Weston, J.L., Strapdown Inertial Navigation Technology, 1997.

[5] Bijker, J., Development of an Attitude and Heading Reference System for an Airship, 2006.

[6] Peddle, I.K., Autonomous Flight of a Model Aircraft, 2005.

[7] Hough, W.J., Autonomous Aerobatic Flight of a Fixed Wing Unmanned Aerial Vehicle, 2007.

[8] [Online]. Available: http://www.istockphoto.com /file_closeup/how/style_and_design/illustrations/ 96063_vector_grid.php?id=96063.

[9] Wilson, J., Design of an Estimator for a Vertical Take-off and Landing Vehicle, 2006.

[10] Milne, G.W., Simplifying Vector Rotations and Coordination, AIAA Atmospheric Flight Mechanics Conference and Exhibit, 2001.

[11] Cook, M.V., Flight Dynamic Principles, 1997.

[12] Etkin, B., Reid, L.D., Dynamics of Flight - Stability and Control, 1996.

[13] Shuster, M.D., Oh, S.D., Three-Axis Attitude Determination from Vector Observations, Journal of Guidance and Control, Vol. 4, Nr 1, 1981.

[14] Hou, H., Modeling Inertial Sensors Errors Using Allan Variance, 2004.

[15] Cemenska, J., Sensor Modeling and Kalman Filter Applied to Satellite Attitude Determination, 2003.

[16] Simon, D., Optimal State Estimation, 2006.

[17] Gelb, A., Applied Optimal Estimation, 1974.

[18] Bar-Shalom, Y., Rong Li, X., Kirubarajan, T., Estimation with Applications to Tracking and Navigation, 2001.

[19] Zarchan, P., Musoff, H., Fundamentals of Kalman Filtering, 2000.

[20] Lefferts, E.J., Markley, F.L., Shuster, M.D., Kalman Filtering for Spacecraft Attitude Estimation, Journal of guidance, control and dynamics, Vol. 5, Nr 5, 1982.

[21] Treurnicht, J., Notes on Attitude Filtering, 2004.

[22] Markley, F.L., Attitude Error Representation for Kalman Filtering, Journal of guidance, control and dynamics, Vol. 26, Nr 2, 2003.

[23] LaViola, J.J., A Comparison of Unscented and Extended Kalman Filtering for Estimating Quaternion Motion, Proceedings of the 2003 American control conference, 2003.

[24] Markley, F.L., Multiplicative versus Additive Filtering for Spacecraft Attitude Determination, Dynamics and control systems and structures in space (DCSSS), 6th conference, 2004.

[25] Lawrence, A., Modern Inertial Technology: Navigation, Guidance and Control, 1998.

[26] ChatField, A.B., Fundamentals of High Accuracy Inertial Navigation, 1997.

[27] Microchip Technology Inc., dsPIC30F6014A Data Sheet, 2005.

[28] Texas Instruments Inc., MAX3238 Data Sheet, 2004.

[29] Texas Instruments Inc., SN65HVD1050 Data Sheet, 2005.

[30] Groenewald, S., Development of a Rotary-Wing Test Bed for Autonomous Flight, 2005.

[31] Treurnicht, J., Sun sensor development notes, 2004.

[32] Texas Instruments Inc., ADS8344 Data Sheet, 2003.

[33] Baker, B.C., Anti-Aliasing, Analog Filters for Data Acquisition Systems, Microchip Technology Inc., 1999.

[34] Frequency Devices Inc., Analog Electronic Filter Design Guide, 2003. [Online]. Available: http://www.freqdev.com/guide/fullguide.html

[35] Franklin, G.F., Powell, J.D., Workman, M., Digital Control of Dynamic Systems, 1998.

[36] Horowitz, P., Winfield, H., The Art of Electronics, 1989.

[37] Texas Instruments Inc., OPA4350 Data Sheet, 2005.

[38] Ericsson, Output Filter Design, Design Note 011, 2005.

[39] Traco Power, DC/DC Converters Tel 5 Series 5/6 Watt Data Sheet, 2003.

[40] Texas Instruments Inc., REG104 Data Sheet, 2005.

[41] Maxim IC, MAX6350 Data Sheet, 2001.

[42] Analog Devices Inc, ADXRS401 Data Sheet, 2004.

[43] STMicroelectronics, LIS3L02AS4 Data Sheet, 2005.

[44] Freescale Semiconductor Inc., MPX4115A Data Sheet, 2005.

[45] Ginde, S.V., Noronha, J.A.N., Design of IIR Filters, 2001.