Olav Austrheim Stenså

Choice and design of 6 degree-of-freedom observer for ROVs operating in fish farms in the presence of time-varying environmental disturbances

Master's thesis in Cybernetics and Robotics
Supervisor: Torleiv Håland Bryne
Co-supervisor: Herman Biørn Amundsen
June 2023

Olav Austrheim Stenså

Choice and design of 6 degree-of-freedom observer for ROVs operating in fish farms in the presence of time-varying environmental disturbances

Master’s thesis in Cybernetics and Robotics
Supervisor: Torleiv Håland Bryne
Co-supervisor: Herman Biørn Amundsen
June 2023

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

NTNU **Norwegian University of** **Science and Technology**	**Faculty of Information Technology** **and Electrical Engineering** **Department of Engineering Cybernetics**

MASTER THESIS DESCRIPTION SHEET

Name:	Olav Austrheim Stenså
Department:	Engineering Cybernetics
Thesis title (Norwegian):	Valg og design av seks frihetsgraders observer for ROV til havbruk påvirket av tidsvarierende miljøforstyrrelser
Thesis title (English):	Choice and design of 6 degree-of-freedom observer for ROVs operating in fish farms in the presence of time-varying environmental disturbances

Thesis Description:
In the CHANGE project, SINTEF Ocean is targeting research to address challenges in using UUVs in dynamically changing environments such as fish farms. For underwater vehicles operating autonomously in an aquaculture context, challenges include operating in the presence of harsh and time-varying sealoads from waves and currents, and in unstructured surroundings consisting of flexible structures and large biomass quantities. An essential task of a control system for autonomous UUV operations in aquaculture is therefore to have an accurate and robust observer that precisely estimates the vehicle states, also in the case of when sensor readings are unreliable or subject to noise.

The main goal of the project is to design an observer that can estimate the 6 degree-of-freedom (DOF) states of an ROV. The ROV is a BlueROV2 which is actuated in all 6 DOFs and has a sensor payload that includes

- an inertial measurement unit (IMU),
- a pressure sensor for measuring depth,
- a Doppler velocity log (DVL) for measurement of linear velocity, and
- an ultrashort baseline (USBL) system for acoustic-based measurements of position.

As underwater signal transmission is susceptible to problems such as signal attenuation, scattering, and multipath propagation, which are even more frequent inside fish cages stocked with dense biomass, it is critical that an observer can use dead-reckoning to compensate for the loss of DVL and USBL measurements. The observer should be implemented in FhSim (a simulation framework for C++) and validated via sea trials.

The project can be continued as a master project, and the student will be given the opportunity to participate in a field trip to the SINTEF ACE fish farm infrastructure for field testing of the implemented navigation system.

The following tasks should be considered included those carried out during the specialization project:

1. Literature review in relation to:
   a. Kinematic and dynamic modelling of ROVs
   b. Underwater navigation systems and state estimators, such as the Extended Kalman Filter (EKF), the Multiplicative EKF, complementary filters, and passive observers.
   c. UUV operations at fish farms.
2. Identify and conclude on the most suited observer design(s) for ROVs operating in fish farms in the presence of environmental disturbances based on your review.
3. Adapt the chosen observer to the specific vehicle and sensor suite.
4. Develop a suitable methodology to handle DVL outliers.
5. Develop a suitable methodology to handle ROV operations in the wave zone preventing wave induced ROV motion from be considered by the real-time ROV control systems operating inside a fish farm.
6. Implement the chosen observer design in FhSim.
7. Conduct full scale experiments in a fish farm (such as SINTEF ACE fish farm).
8. Present and discuss your results.
9. Conclude on your thesis.

Start date:	2023-01-09
Due date:	2023-06-19
Thesis performed at:	Department of Engineering Cybernetics, NTNU
Supervisor:	Associate professor Torleiv H. Bryne, Dept. of Eng. Cybernetics, NTNU
Co-Supervisor):	Herman Biørn Amundsen Dept. of Eng. Cybernetics, NTNU SINTEF Ocean AS

Abstract

Remotely Operated Vehicles (ROVs) have become useful tools for inspection and maintenance in Aquaculture. This thesis documents the choice, design, implementation, and full-scale testing of an observer for an ROV operating in fish farms surrounded by fish, and subject to waves and currents. The ROV is equipped with an Inertial Measurement Unit (IMU), a Doppler Velocity Log (DVL), and an Ultra-Short Baseline (USBL) positioning system.

Currently, ROVs at fish farms are remotely controlled by operators on the surface. A robust and precise navigation system is required to automate the tasks that today require manual labor. The goal is to increase efficiency and reduce the cost of the inspection and maintenance needed to prevent fish from escaping the fish farms.

The thesis presents the information about the vehicle, sensors, and environment relevant for choosing an observer, as well as observer candidates that can solve the problem. Inertial navigation aided by DVL and USBL is chosen as the observer and designed to fit the vehicle and sensors. Extra functionality to help with operation in waves is added. The system is implemented in object-oriented C++ and tested at full scale in a fish farm. Results are presented and discussed before the choice, design, and implementation are evaluated and changes are proposed.

A Multiplicative Extended Kalman Filter (MEKF) is developed and adapted for the specific vehicle and sensor suite. A method for modeling biases in the pressure sensor measurement is proposed and estimation of these biases is integrated into the MEKF.

A fish farm is a challenging environment to navigate; fish, nets, and moorings are complicating factors. It becomes clear that USBL and DVL measurements in such an environment are of poor quality and not always available at all. The chosen observer works, but the quality of the position estimates in particular is insufficient for the purpose. An IMU of better quality is necessary to increase performance to a level sufficient for ROV automation.

Although the results of the work in this thesis do not quite reach the objective, will this thesis contribute with an overview, details, considerations, and experiences that are important for continued efforts to solve the problem.

Sammendrag

Fjernstyrte undervannsfarkoster (ROVer) har blitt et nyttig verktøy for inspeksjon og vedlikehold i oppdrettsnæringen. Denne oppgaven dokumenterer valg, utvikling, implementering og fullskala testing av en observer til en ROV som skal operere i oppdrettsmerder i et miljø med fisk, strømning og bølger. ROVen er utstyrt med treghetssensorer (IMU), en Doppler Velocity Log (DVL) og et Ultra-Short Baseline (USBL) posisjoneringssystem.

På nåværende tidspunkt styres ROVer manuelt ved hjelp av fjernstyring. Et robust og presist navigasjonssystem vil legge grunnlaget for automatisering av oppgaver som i dag krever manuelt arbeid. Formålet med dette er å effektivisere og redusere kostnader med inspeksjon og vedlikehold av oppdrettsmerder, spesielt med tanke på å oppdage og utbedre hull i merdene og dermed unngå rømming av fisk.

Oppgaven presenterer informasjon om fartøy, sensorer og miljø som er relevant med tanke på valg av observer, samt en rekke observere som kan løse oppgaven. Treghetsnavigasjon hjulpet av DVL og USBL velges som observer og tilpasses til ROVen og de tilgjengelige sensorene. Ekstra funksjonalitet legges til for at ROVen lettere skal kunne operere i bølger. Systemet implementeres i objektorientert C++ og testes fullskala i en oppdrettsmerd. Resultater presenteres og diskuteres, før valg, design og implementasjon evalueres og endringer foreslås.

Et Multiplikativt Utvidet Kalman Filter (MEKF) utvikles og tilpasses for den spesifikke farkosten og de tilgjengelige sensorene. En metode for å modellere biaser i trykksensormålinger foreslås og estimering av biasene integreres i observeren.

En oppdrettsmerd er et utfordrende sted å navigere i; fisk, nett og fortøyninger kompliserer operasjoner. Det er tydelig at USBL og DVL målinger i et slikt miljø blir av dårlig kvalitet og ikke alltid tilgjengelig. Den valgte observerløsningen fungerer, men kvaliteten på spesielt posisjonsestimatene blir for dårlige til formålet. Treghetssensorer av høyere kvalitet er nødvendig for å forbedre ytelsen til ett nivå som er tilstrekkelig for automatisering av fartøyet.

Selv om resultatet av arbeidet i oppgaven ikke når opp til målsetningen, bidrar oppgaven med oversikt, detaljer, betraktninger og erfaringer som er viktige for videre forsøk på å løse problemstillingen.

Acknowledgements

This thesis is written in the spring of 2023 as the final assignment in a 5-year MSc program in Cybernetics at the Norwegian University of Science and Technology (Norges Teknisk-Naturvitenskapelige Universitet, NTNU). This thesis is a result of collaboration with SINTEF Ocean and is a part of SINTEF Ocean's CHANGE project.

First of all, I want to thank my supervisors Thorleiv Haaland Bryne and Herman Biørn Amundsen; Thorleiv, for the interesting and in-depth discussions on the subject of navigation and for your immense attention to detail. Herman, for organizing the practical side of the project, for a great field trip, and for sharing valuable hands-on experience with ROVs and aquaculture.

Thanks to SINTEF Ocean for the opportunity, for making a full-scale fish farm available for testing, and for access to the ROV, sensors, and equipment necessary to complete the project. This project was implemented in SINTEF Ocean's FhSim, which is a simulation framework. This greatly simplified the process of integrating my code with existing control systems and sensor interfaces.

I also want to mention a couple of NTNU courses that were of great help in completing this project; "Guidance, Navigation and Control of Vehicles", which I consider a great general overview of the subject of marine craft navigation and control by Thor I. Fossen, and "Sensor Fusion", a course on Kalman filtering by Edmund Brekke.

Trondheim, June 2023

Olav Austrheim Stenså

Preface

The topic of this thesis is a hybrid between aquaculture robotics and more classical navigation challenges. This project was chosen because of my interest in marine craft navigation and control and a desire to learn more about, in particular, attitude kinematics and representations, Kalman filtering, and operations in waves. The thesis is therefore more directed toward solving a navigation system engineering challenge within aquaculture, than working in aquaculture robotics itself. The author is in general very happy with the learning outcomes from the project.

The thesis is a continuation of a specialization project which was handed in by the author in the fall of 2022 with the title "Choice and design of a 6 degree-of-freedom observer for an UUV operating in fish farms in the presence of time-varying environmental disturbances"(Stenså, 2022). The introductory work (background, literature reviews, etc.) is reused in this thesis and because of this, certain chapters, paragraphs, sentences, and figures are similar or identical. However, the end product; design, implementation, and results are considered to be completely independent.

Contents

List of Figures

List of Tables

List of Abbreviations

AHRS	Attitude Heading Reference System
CO	Coordinate Origin
DOA	Direction Of Arrival
DOF	Degree-Of-Freedom
DVL	Doppler Velocity Log
EKF	Extended Kalman Filter
ESKF	Error State Kalman Filter
FFT	Fast Fourier Transform
GNC	Guidance, Navigation and Control
GNSS	Global Navigation Satellite System
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
LBL	Long BaseLine
LOS	Line-of-Sight
MEKF	Multiplicative Extended Kalman Filter
NED	North-East-Down
NIS	Normalized Innovation Squared
PID	Proportional-Integral-Derivative
ROV	Remotely Operated underwater Vehicle
SBL	Short BaseLine
UML	Unified Modeling Language
USBL	Ultra-Short BaseLine

Chapter 1

Introduction

1.1 Background

Aquaculture is one of the fastest growing sectors in the global food industry, being the fastest growing sector in 2018 (FAO, 2018). Aquaculture plays an important role in meeting the world's ever-growing need for protein. Rapid growth comes with numerous challenges, such as environmental concerns, disease management, and operational efficiency. To overcome these challenges and ensure the long-term viability of aquaculture, technological advancements are essential.

1.2 Motivation

One of the challenges in aquaculture is fish escaping the fish farms (Thorvaldsen et al., 2015). Escaped fish may carry and spread diseases (another major challenge) to the local wildlife. To prevent escapes, regular inspection and maintenance of the net pens are necessary. Traditionally, this was a job for human divers, but Remotely Operated underwater Vehicle (ROV)s have become common for these tasks, reducing the risk of injuries (Føre et al., 2018). The safety of operators is also a concern as fish farms are moved further offshore where the weather is harsh (Bjelland et al., 2015). This is a key motivation for SINTEF Ocean's CHANGE project[1] which this thesis is a part of. ROVs are as the name suggests remotely operated, meaning they require human operators in what is a time-consuming, and depending on the weather, a difficult task. This is a motivation for moving to more sophisticated methods of controlling the ROVs and achieving some degree of automation and even autonomy. To achieve this it is absolutely vital for the vehicle (and the operators) to know where the vehicle is positioned. Previous research has shown that navigation inside net pens using Doppler Velocity Log (DVL) and Ultra-Short BaseLine (USBL) is possible (Rundtop and Frank, 2016). Since the measurements available underwater are not the most reliable, it is important for the vehicle to have a robust observer which is able to estimate its position. The development, implementation, and evaluation of such an observer is the main goal of this thesis.

[1]CHANGE project website: `https://www.sintef.no/en/projects/2021/change-an-underwater-robotics-concept-for-dynamically-changing-environments/`

1.3 Outline

The thesis is structured in chapters, each covering an important topic, or important stage in the process. The first chapter, Chapter 2 presents the ROV and its sensors. This chapter is placed early to give an idea of what we have to work with, which is important for choices and considerations in many of the later chapters. This is followed by Chapter 3, which introduces the basic notation used in the thesis. Chapter 4 defines the reference frames used in this thesis, as well as the notation for vectors in these frames. Related to this, is Chapter 5, a separate chapter on attitude representations. A literature review on topics relevant to the choice of observer is presented in Chapter 6, before the alternatives are discussed and an observer is chosen in Chapter 7.

The remaining chapters follow a more classical report structure, starting with Chapter 8 presenting the necessary theory for the chosen observer. Chapter 9 covers the modeling stage. Chapter 10 covers tailoring of the observer to suit the ROV, sensor suite, and the environment. This is simply referred to as observer design. Chapter 11 documents the code implementation stage, a large part of this project. Experimental setup and tuning are covered by Chapter 12, named Method. The results are presented and discussed in Chapter 13. Finally, conclusions and further work are presented in Chapter 14.

Chapter 2

The ROV

The goal of this thesis is to choose and design an observer for a ROV. To do this, we need to know about the ROV in question. This chapter presents what we have to work with in terms of the vehicle and sensors and is taken from Stenså (2022). Section 2.1 presents the BlueROV2. Section 2.2 presents the available sensors.

2.1 BlueROV2

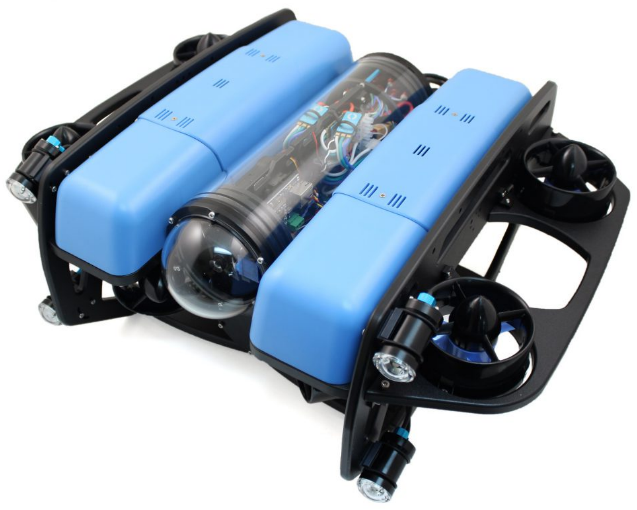

Figure 2.1: The BlueROV2 with heavy configuration kit (picture from the BlueROV2 website[2])

The observer will be specifically (but not exclusively) designed for the BlueROV2, shown in Figure 2.1. The BlueROV2 is a small and affordable consumer-grade UUV suitable for inspection and underwater exploration. This particular BlueROV2 is a “Heavy configuration” and its basic dimensions are shown in Figure 2.2.

[2]BlueROV2 website: `https://bluerobotics.com/store/rov/bluerov2-upgrade-kits/brov2-heavy-retrofit/`

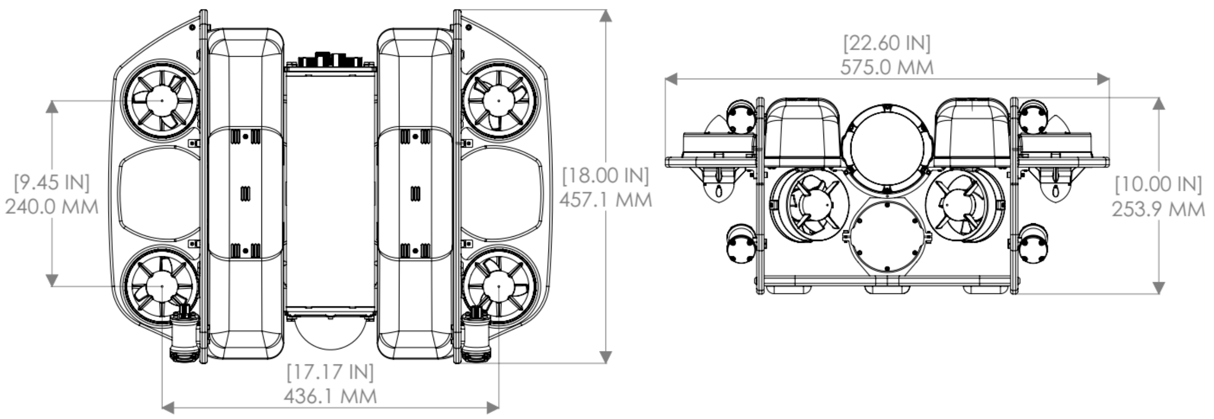

Figure 2.2: Dimensions of the BlueROV 2 heavy configuration (from the BlueROV2 website)

The ROV has a tether for communication with operators and equipment at the surface. It weighs 11.5 kg and is designed to be slightly positively buoyant to ensure that the ROV returns to the surface in case of propulsion or power system failure. The ROV design is based on a frame containing a watertight enclosure with electronics. The vehicle is actuated in all 6 degrees of freedom through 8 thrusters inside the frame. The maximum rated depth is 100m depth and it has a maximum forward speed of 1.5 m/s. Lights and a tilting camera are fitted to allow vision for human operators on the surface. Rechargeable batteries on the vehicle allow for about 2 hours of usage time under "normal use" (Blue Robotics, 2022).

2.2 Sensors

The BlueROV2 is configured with the following sensors from factory (Blue Robotics, 2022):

- 3-DOF Gyro
- 3-DOF Accelerometer
- 3-DOF Magnetometer
- Pressure sensor
- Temperature sensor
- Internal barometer
- Current and voltage sensors
- Leak detection

In addition, a Doppler Velocity Log (DVL) and an Ultra-Short BaseLine (USBL) transponder are fitted for navigation purposes. Some of the factory sensors are for safety system purposes only. The rest of this chapter is dedicated to a more in-depth presentation of sensors relevant to the later chapters.

2.2.1 Accelerometer

The accelerometer on BlueROV2 is an LSM303D (ST Microelectronics, 2013b). The accelerometer measures specific force along three axes in a frame fixed to the accelerometer. When the local gravity is known, the specific force measurement can be used to:

1. Estimate the acceleration of the craft
2. Estimate the roll and pitch of the craft

The accelerometer is a high-rate sensor. This is especially useful with small, quick, and maneuverable vehicles. The accelerometer output is subject to significant measurement noise. Being a consumer-grade sensor, the accelerometer is prone to a significant measurement bias. This bias can vary with time.

2.2.2 Gyro

The gyro on BlueROV2 is an L3GD20H from (ST Microelectronics, 2013a). The gyro measures angular velocity about three axes in a frame fixed to the gyro. The gyro can be used to estimate the angular velocity of the craft. The measurement is subject to measurement noise. The gyro measurements are subject to a significant measurement bias. The bias can vary with time.

2.2.3 Magnetometers

The magnetometer on BlueROV2 is an LSM303D (ST Microelectronics, 2013b), this is the same chip as the accelerometer. These sensors measure the local magnetic field along three axes fixed to the magnetometer. This includes the Earth's magnetic field as well as disturbances from nearby ferromagnetic items or electronics. The measurements can be used to determine the vehicle's orientation, specifically the heading. The magnetometer measurements are subject to some noise, but it is the aforementioned magnetic disturbances that are the main sources of error.

2.2.4 Pressure sensor

The pressure sensor on the BlueROV2 is an MS5837-30BA (TE Connectivity, 2019). The pressure sensor measures the pressure of the water surrounding the craft. The measurement can be used to calculate depth below the surface. The pressure sensor measures absolute pressure, meaning that it does not only measure the water pressure due to gravity but also the pressure of the air on the surface, appearing as a time-varying bias. The pressure sensor is subject to noise.

2.2.5 Doppler Velocity Log

The Doppler Velocity Log fitted to the BlueROV2 is an A50 (Waterlinked, 2018). The DVL measures the Doppler shift of four hydroacoustic beams reflecting back from a surface. The relative velocity and distance along the beams are then calculated. To use the net pen walls as a reflecting surface, the DVL is mounted facing forward on the BlueROV2. This is not very common since the most consistent reflecting surface usually is the seabed and therefore DVLs are traditionally mounted facing down. The DVL returns velocity measurements along three axes fixed to the DVL itself and an estimated 3×3 covariance matrix that represents the uncertainty of the velocity measurements. The A50 was at the time of writing the world's smallest commercially available DVL - by far, according to their website (Waterlinked, 2018).

2.2.6 Ultra-Short BaseLine

The USBL system fitted to the BlueROV2 is a Sonardyne Micro-Ranger. This is a very small and light system with basic performance. Quoting their own website[3] for the Micro-Ranger 2: "For when good is good enough". USBL is an underwater acoustic positioning system consisting of a transceiver and a transponder. The transceiver is typically mounted to a mothership, while the transponder is mounted to the UUV. Since the system consists of only two parts, the attitude of the transceiver becomes very important. The transceiver is therefore fitted with internal attitude sensors for roll and pitch, and a magnetic compass used to determine yaw. GNSS positioning is also required to determine the transceiver position. The USBL system outputs the position measurement in a three-dimensional vector. Additionally, it outputs an estimate of the standard deviation of the horizontal part of the measurement.

2.2.7 Attitude and Heading Reference System

The BlueROV2 also has a built-in observer for attitude, which is referred to as Attitude Heading Reference System (AHRS). The observer estimates roll, pitch, and yaw using the gyro, accelerometer, and magnetometer.

[3]Micro-Ranger 2 website: `https://www.sonardyne.com/products/micro-ranger-2-shallow-water-usbl-system/`

Chapter 3

Notation

This thesis contains a lot of mathematical notation. When possible, parameters are gathered in vectors and matrices to simplify notation and keep the number of variables under control. It is worth noting that this chapter only introduces the notation, and it is further developed through the coming chapters. Section 3.1 describes how scalars, vectors, and matrices are distinguished. Section 3.2 explains how functions are denoted. Lastly, Section 3.3 introduces a special operator that will be used to simplify notation later in the thesis.

3.1 Scalars, vectors, and matrices

In order to distinguish scalars, vectors, and matrices, they have different notations. Scalars are denoted with lowercase letters, vectors with lowercase bold letters, and matrices with uppercase bold letters. Examples are shown Table 3.1.

Table 3.1: Notation for scalars, vectors, and matrices

Type	**Case**	**Bold**	**Example**
Scalar	lower	No	a
Vector	lower	**Yes**	$\boldsymbol{q}$
Matrix	UPPER	**Yes**	$\boldsymbol{R}$

In order to distinguish covariance matrices from other matrices, they are written in calligraphic font. An example of a covariance matrix is $\boldsymbol{\mathcal{R}}$. Parameter errors are denoted as a δ (greek letter delta) in front of the parameter. Examples are: δa (scalar), $\boldsymbol{\delta a}$ (vector), and $\boldsymbol{\delta A}$ (matrix). These errors are to be thought of as close to zero, not infinitesimal, but in a range where small angle approximations like $\sin x \approx x$ are valid. Estimates of true parameters are denoted with a hat. Example: $\hat{\boldsymbol{x}}$ is the estimate of the vector x.

3.2 Functions

Functions also have different notations based on what they return. This notation is the same as for parameters except that the symbols are upright instead of in italics, see Table 3.2

Table 3.2: Notation for functions

Return type	**Case**	**Bold**	**Example**
Scalar	lower	No	$\sin(\cdot)$
Vector	lower	**Yes**	$\mathbf{q}(\cdot)$
Matrix	UPPER	**Yes**	$\mathbf{R}(\cdot)$

3.3 Discrete variable update operator

In a discrete system, it is common to update a variable based on the previous value of the same variable. An example where x is the variable and k is the time step:

$$x[k+1] = x[k] + 1 \tag{3.1}$$

To avoid the cluttering and often unnecessary square brackets notation, we will use an arrow instead of the equal sign. The same example becomes:

$$x \leftarrow x + 1 \tag{3.2}$$

, which can be read as; “update x using the existing version of x”. This notation is inspired by Solà (2017).

Chapter 4

Reference Frames and Vectors

The reference frames used in this thesis are defined in Section 4.1. The notation used to denote vectors within these frames is found in Section 4.2, and operators used on vectors in the frames in Section 4.3.

4.1 Reference frames

4.1.1 North East Down frame

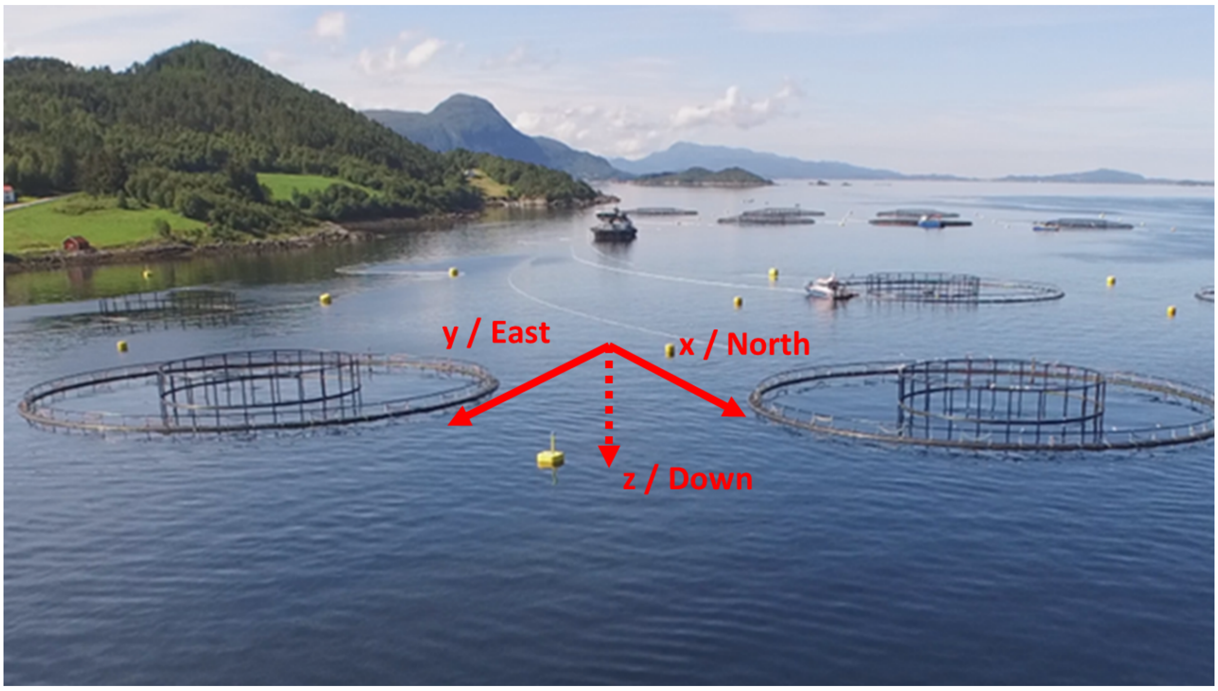

Figure 4.1: An example of a NED frame at the SINTEF ACE fish farm at Korsneset (Picture from SINTEF[4], modified with axes).

[4]Picture from: `https://www.sintef.no/en/all-laboratories/ace/`

A North-East-Down (NED)-frame will be used as a reference to describe the position of the ROV. A NED frame has its x-axis pointing north, y-axis to the east, and z-axis down. See an example of such a frame in Figure 4.1. The USBL positioning system returns measurements in a NED frame, and in order to keep things simple we choose to just inherit that frame. This means that the coordinate origin is earth fixed (not moving with the vehicle) and that the exact location of the origin is chosen in the USBL system. Since the surface of the earth is curved, it is important that the coordinate origin is chosen reasonably close to where the ROV will operate.

4.1.2 The body frame

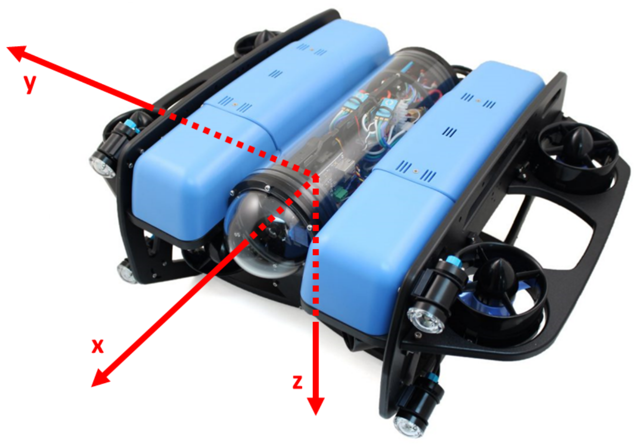

Figure 4.2: The body frame (picture from the BlueROV2 website, modified with axes)

The UUVs body frame is defined according to the convention in Fossen (2021). The x-axis points forwards, the y-axis starboard, and the z-axis down as can be seen in Figure 4.2. The coordinate origin is set to the center of the IMU. This is recommended in Fossen (2021) to avoid numerical derivation of measurements when the accelerometer measurements are used in an observer. "b" is used as a superscript to denote vectors in the body frame.

4.1.3 DVL frame

The DVL is mounted to the ROV in an orientation where its axes do not align with the body frame. Because of this, it is necessary to define a frame for the DVL. The axes are defined in the DVLs online documentation (Waterlinked, 2018) and are drawn in 4.3.

- X axis is pointing forward (LED is forward, cable backward)
- Y axis is pointing right
- Z axis is pointing down (mounting holes are up, transducers are down)

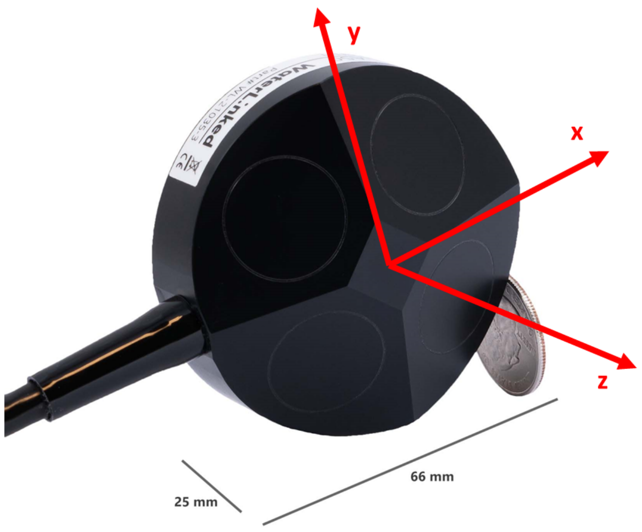

Figure 4.3: The DVL frame (picture from Waterlinked (2018), modified with axes)

Superscript "d" is used to denote vectors in the DVL frame.

4.2 Vector notation

A lot of three-dimensional vectors will be used in this thesis. They will generally have sub- and superscripts to denote which reference frames they belong to and what they describe. The superscripts denote the reference frame in which the vector is represented. Subscripts are used to denote what the vector describes. We will explain this through an example; the vector $\boldsymbol{p}^c_{a/b}$ is the position of the coordinate origin of frame a relative to the coordinate origin of frame b given in frame c. Since we very often end up with b/n in the subscript, the subscripts are not used in those cases, meaning that $\boldsymbol{p}^c$ is the position of the coordinate origin of the body frame relative to the coordinate origin of the NED given in frame c. The notation of vectors used in this thesis is summarized in Table 4.1.

Table 4.1: List of vectors

Parameter	**Symbol**	**Description**
Position	$\boldsymbol{p}$	Distance along x,y,z axes
Velocity	$\boldsymbol{v}$	Velocity along x,y,z axes
Acceleration	$\boldsymbol{a}$	Acceleration along x,y,z axes
Rotation vector	$\boldsymbol{\alpha}$	Angle about direction in space
Angular velocity	$\boldsymbol{\omega}$	Angular velocity about x,y,z axes
Bias	$\boldsymbol{b}$	Bias along/about x,y,z axes
Process noise	$\boldsymbol{w}$	Noise along/about x,y,z axes
Measurement noise	$\boldsymbol{\eta}$	Noise along/about x,y,z axes
Lever arm	$\boldsymbol{r}$	Distance along x,y,z axes

4.3 Vector Operators

4.3.1 Rotations

Rotation matrices are used to transform vectors between coordinate frames. This thesis follows the passive body-to-ned convention for rotation matrices, meaning:

$$\boldsymbol{p}_{b/n}^{n} = \boldsymbol{R}\boldsymbol{p}_{b/n}^{b} \tag{4.1}$$

The inverse operation is the transpose of the rotation matrix:

$$\boldsymbol{p}_{b/n}^{b} = \boldsymbol{R}^{\top}\boldsymbol{p}_{b/n}^{n} \tag{4.2}$$

The symbol $\boldsymbol{R}$ is used exclusively for rotations between the body frame and NED. We also need to define a matrix to do the constant rotation between the DVL frame and body. Let $\boldsymbol{D}$ be defined such that:

$$\boldsymbol{v}_{d/n}^{b} = \boldsymbol{D}\boldsymbol{v}_{d/n}^{d} \tag{4.3}$$

4.3.2 Vector cross products and skew-symmetric matrices

Let $\boldsymbol{a}^c = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^\top$ be a vector in frame c. The skew symmetric matrix operator $\mathbf{S}$ on a is defined as:

$$\mathbf{S}(\boldsymbol{a}^c) = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{4.4}$$

The matrix resulting from the operator $\mathbf{S}$ has the property $\mathbf{S}(\boldsymbol{a}^c)^\top = -\mathbf{S}(\boldsymbol{a}^c)$, which is the definition of a skew-symmetric matrix. The inverse operation from matrix to vector is here referred to as the vex-operator and is defined such that:

$$\boldsymbol{a}^c = \operatorname{vex}\left(\mathbf{S}(\boldsymbol{a}^c)\right) \tag{4.5}$$

The vector cross-product is often useful for analysis in 3-dimensional reference frames. The skew-symmetric operator is related to the cross-product in the following way:

$$\boldsymbol{a}^c \times \boldsymbol{b}^c = \mathbf{S}(\boldsymbol{a}^c)\boldsymbol{b}^c \tag{4.6}$$

In this thesis, the cross-product is replaced by the skew-symmetric matrix operator for consistency. Also, note that the operator follows the same anticommutative property as the cross product:

$$\begin{gathered} \boldsymbol{a}^c \times \boldsymbol{b}^c = -(\boldsymbol{b}^c \times \boldsymbol{a}^c) \\ \Updownarrow \\ \mathbf{S}(\boldsymbol{a}^c)\boldsymbol{b}^c = -\mathbf{S}(\boldsymbol{b}^c)\boldsymbol{a}^c \end{gathered} \tag{4.7}$$

Chapter 5

Attitude representations

The orientation of the body frame in relation to NED will in this thesis be referred to as attitude. Attitude can be represented in a number of ways. This chapter presents the different attitude representations used in the later chapters and is taken from Stenså (2022). Section 5.1 presents the attitude representations, and section 5.2 lists the relevant conversion between them.

5.1 Attitude representations

There are multiple ways to represent attitude. The four that are most relevant for this thesis are (Groves, 2008):

- Euler angles
- Rotation matrix
- Quaternions
- Rotation vector

5.1.1 Euler angles

The Euler angles describe three subsequent rotations about three coordinate axes. These rotations are called roll, pitch, and yaw. This makes it perhaps the most intuitive representation of attitude. Euler angles are the standard parametrization for attitude in guidance, navigation, and control systems for surface vessels in Fossen (2021). The symbols used for Euler angles in this project are ϕ, θ, and ψ describing roll, pitch, and yaw respectively. The Euler angle parametrization suffers from singularities. No matter how you define the rotations you always end up with at least two. For the standard Euler angles, which are used here, the singularities occur at $\theta = \pm 90°$. This is usually not a problem for surface craft (if the bow is pointing straight down there is little use for navigation), but it is likely that a ROV to end up in such orientation.

5.1.2 Rotation matrix

Rotation matrices describe the rotation of a vector from one frame to another. They are mathematically very intuitive as the conversion is just a matrix multiplication and the reverse operation involves multiplication with the transpose of the matrix. As for notation, $\boldsymbol{R}$ is used to denote a rotation from the body frame to the NED frame. This follows the passive body-to-ned convention used in Solà (2017). The opposite rotation is achieved through the transpose; $\boldsymbol{R}^\top$. This eliminates the need for cluttering sub- and super-scripts.

5.1.3 Unit quaternions

Unit quaternions is a four-parameter, singularity-free attitude representation (Solà, 2017; Fossen, 2021). A quaternion can be viewed as a complex number with 1 real and 3 complex parts. The intuition behind how quaternions work is therefore related to how traditional complex numbers work. As for notation, the letter $\boldsymbol{q}$ is used for the quaternion. The quaternion convention used here is the Hamiltonian convention used in Solà (2017) and is represented in vector form. This follows the passive body-to-world convention, similar to what is used for the rotation matrix in 5.1.2. $\mathbf{q}(\cdot)$ is used to denote the quaternion as a function of something else. For some operations it is necessary to split the quaternion into its real (η) and complex (ϵ) parts:

$$\boldsymbol{q} = \begin{bmatrix} \eta \\ \boldsymbol{\epsilon} \end{bmatrix}, \quad \eta \in \mathbb{R}, \quad \boldsymbol{\epsilon} \in \mathbb{R}^3 \tag{5.1}$$

Quaternions are considered to be computationally light weigh and have easy-to-define operations like the quaternion product (Solà, 2017):

$$\boldsymbol{q}_a \otimes \boldsymbol{q}_b = \begin{bmatrix} \eta_a \eta_b - \boldsymbol{\epsilon}_a^\top \boldsymbol{\epsilon}_b \\ \eta_a \boldsymbol{\epsilon}_b + \eta_b \boldsymbol{\epsilon}_a + \boldsymbol{\epsilon}_a \times \boldsymbol{\epsilon}_b \end{bmatrix} \tag{5.2}$$

The quaternion can be used to rotate vectors directly without converting to a rotation matrix first. The quaternion rotation is defined such that a vector $\boldsymbol{z}$ can be rotated from body to NED in the following way (Solà, 2017):

$$\begin{bmatrix} 0 \\ \boldsymbol{z}^n \end{bmatrix} = \boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{z}^b \end{bmatrix} \otimes \boldsymbol{q}^* \tag{5.3}$$

, where $\boldsymbol{q}^*$ is the quaternion inverse, defined as:

$$\boldsymbol{q}^* = \begin{bmatrix} \eta \\ -\boldsymbol{\epsilon} \end{bmatrix} \tag{5.4}$$

5.1.4 Rotation vector

The rotation vector, often also referred to as an axis-angle parametrization is perhaps the most basic way to represent attitude. The vector is 3-dimensional and describes a direction in space. The length of the vector describes a rotation around that direction in space. The symbol $\boldsymbol{\alpha}$ (as in $\boldsymbol{\alpha}$ttitude or $\boldsymbol{\alpha}$xis-$\boldsymbol{\alpha}$ngle) is used to denote rotation vectors in this project. The angle of rotation is just given as $\|\boldsymbol{\alpha}\|$ to keep the notation simple. As for all 3-parameter representations, it suffers from singularities. A really unfortunate one where $\|\boldsymbol{\alpha}\| = 0$ (a vector with no length has no direction), and the representation is non-unique for $\|\boldsymbol{\alpha}\| = \pi$ (the negation will represent the same rotation).

5.2 Conversion

When using multiple attitude parametrizations in the same system, it is necessary to be able to convert between them. The relevant formulas are presented here.

5.2.1 Euler angles to rotation matrix

The rotation matrix can be constructed as subsequent rotations about three axes (Fossen, 2021):

$$\begin{aligned}\mathbf{R}(\phi,\theta,\psi) &= \mathbf{R}(\psi)\,\mathbf{R}(\theta)\,\mathbf{R}(\phi) \\ &= \begin{bmatrix}\cos\psi & -\sin\psi & 0\\ \sin\psi & \cos\psi & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\cos\theta & 0 & \sin\theta\\ 0 & 1 & 0\\ -\sin\theta & 0 & \cos\theta\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\phi & -\sin\phi\\ 0 & \sin\phi & \cos\phi\end{bmatrix}\end{aligned} \tag{5.5}$$

5.2.2 Euler angles to quaternion

A quaternion can be constructed from the Euler angles as follows (Fossen, 2021):

$$\boldsymbol{q}(\phi,\theta,\psi) = \begin{bmatrix}\cos\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2}\\ \sin\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} - \cos\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2}\\ \cos\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2}\\ \cos\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} - \sin\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2}\end{bmatrix} \tag{5.6}$$

5.2.3 Rotation matrix to Euler angles

Let:

$$\boldsymbol{R} = \begin{bmatrix}R_{11} & R_{12} & R_{13}\\ R_{21} & R_{22} & R_{23}\\ R_{31} & R_{32} & R_{33}\end{bmatrix} \tag{5.7}$$

Then, according to Fossen (2021), the euler angles can be found as:

$$\phi(\boldsymbol{R}) = \operatorname{atan2}(R_{32}, R_{33}) \tag{5.8a}$$
$$\theta(\boldsymbol{R}) = -\operatorname{asin}(R_{31}), \quad \theta \neq \pm 90° \tag{5.8b}$$
$$\psi(\boldsymbol{R}) = \operatorname{atan2}(R_{21}, R_{11}) \tag{5.8c}$$

5.2.4 Quaternion to rotation matrix

A simple formula is found in Fossen (2021):

$$\mathbf{R}(\boldsymbol{q}) = \boldsymbol{I}_3 + 2\eta\,\mathbf{S}(\boldsymbol{\epsilon}) + 2\,\mathbf{S}^2(\boldsymbol{\epsilon}) \tag{5.9}$$

5.2.5 Rotation vector to quaternion

From Solà (2017). Note that curly brackets $\{\cdot\}$ denote “as a function of a rotation vector”.

$$\mathbf{q}\{\boldsymbol{\alpha}\} = \begin{bmatrix}\cos\frac{\|\boldsymbol{\alpha}\|}{2}\\ \frac{\boldsymbol{\alpha}}{\|\boldsymbol{\alpha}\|}\sin\frac{\|\boldsymbol{\alpha}\|}{2}\end{bmatrix} \tag{5.10}$$

5.2.6 Rotation vector to rotation matrix

The mathematical way to convert rotation vectors into rotation matrices is through the Rodrigues rotation formula (Solà, 2017):

$$\mathbf{R}\{\boldsymbol{\alpha}\} = \boldsymbol{I}_3 + \frac{\sin\|\boldsymbol{\alpha}\|}{\|\boldsymbol{\alpha}\|}\mathbf{S}(\boldsymbol{\alpha}) + \frac{1-\cos\|\boldsymbol{\alpha}\|}{\|\boldsymbol{\alpha}\|^2}\mathbf{S}^2(\boldsymbol{\alpha}) \tag{5.11}$$

5.2.7 Attitude jacobian

It is handy to also have a way of converting between Euler angles and rotation vectors using a Jacobian. This is only intended for small angles $\delta\phi$, $\delta\theta$, $\delta\psi$ about ϕ, θ, ψ respectively. The jacobian is derived in Appendix A.3. The jacobian $\boldsymbol{T}_{att}$ is defined such that:

$$\boldsymbol{\delta\alpha}^n = \boldsymbol{T}_{att}\begin{bmatrix}\delta\phi\\ \delta\theta\\ \delta\psi\end{bmatrix} \quad \text{and} \quad \begin{bmatrix}\delta\phi\\ \delta\theta\\ \delta\psi\end{bmatrix} = \boldsymbol{T}_{att}^{-1}\boldsymbol{\delta\alpha}^n \tag{5.12}$$

, where $\boldsymbol{\delta\alpha}^n$ is a small rotation vector in NED and $\delta\phi$, $\delta\theta$, and $\delta\psi$ are small errors angles.

$$\boldsymbol{T}_{att} = \begin{bmatrix}\cos\hat{\psi}\cos\hat{\theta} & -\sin\hat{\psi} & 0\\ \sin\hat{\psi}\cos\hat{\theta} & \cos\hat{\psi} & 0\\ -\sin\hat{\theta} & 0 & 1\end{bmatrix} \tag{5.13}$$

$$\boldsymbol{T}_{att}^{-1} = \begin{bmatrix}\frac{\cos\hat{\psi}}{\cos\hat{\theta}} & \frac{\sin\hat{\psi}}{\cos\hat{\theta}} & 0\\ -\sin\hat{\psi} & \cos\hat{\psi} & 0\\ \frac{\cos\hat{\psi}\sin\hat{\theta}}{\cos\hat{\theta}} & \frac{\sin\hat{\psi}\sin\hat{\theta}}{\cos\hat{\theta}} & 1\end{bmatrix} \tag{5.14}$$

Chapter 6

Literature Review

This chapter contains a preliminary literature review providing background for choosing an observer for the BlueROV2 and is largely taken from Stenså (2022). Section 6.1 is concerned with modeling of ROVs. Potential observers are presented in Section 6.2. Finally, Section 6.3 concerns ROV operations at fish farms.

6.1 Modeling of ROVs

6.1.1 Hydrodynamic modeling

When designing Guidance, Navigation and Control (GNC) systems for marine craft, the first step is often to find a model for the craft. Having a model of a marine craft is useful for many reasons, the most important for this scenario are:

- Simulation
- Model-based observer design and tuning
- Model-based controller design and tuning

Modeling of marine craft is thoroughly described in Fossen (2021). In GNC applications, the resulting model is often on the form:

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{6.1a}$$

$$\boldsymbol{M}_{RB}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\boldsymbol{\eta}) = -\boldsymbol{M}_A\dot{\boldsymbol{\nu}}_r - \boldsymbol{C}_A(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r - \boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{\tau} + \boldsymbol{\tau}_{wave} \tag{6.1b}$$

where $\boldsymbol{\eta}$ is a state vector containing position and attitude in NED, $\boldsymbol{\nu}$ is a state vector containing linear and angular velocity in the body frame, and $\boldsymbol{\nu}_r$ is a state vector containing linear and angular velocity relative to the surrounding water in the body frame. The matrix $\boldsymbol{J}$ is a mapping between the body and the world frame. $\boldsymbol{M}_{RB}$, $\boldsymbol{C}_{RB}$ are the rigid body inertia and coriolis matrices. $\boldsymbol{g}$ is a vector describing gravity and buoyancy. $\boldsymbol{M}_A$, $\boldsymbol{C}_A$ and $\boldsymbol{D}$ are hydrodynamical terms describing added mass, added Coriolis, and damping. $\boldsymbol{\tau}$ are forces that the vehicle's actuators (rudders, propellers, thrusters, etc.) apply. $\boldsymbol{\tau}_{wave}$ wave forces. Forces due to currents are captured in hydrodynamical terms since relative velocities are used.

6.1.2 Kinematic model

It is also possible to avoid hydrodynamical modeling and instead rely on measurements of acceleration. The relation between the states can be described through differential equations. Acceleration, velocity, and position are related, and so are the angular velocity and attitude. For the translational model, there are two practical options to choose from, depending on which frame you want to express velocity in. The model with velocity in NED is:

$$\dot{\boldsymbol{p}}^n = \boldsymbol{v}^n \tag{6.2a}$$

$$\dot{\boldsymbol{v}}^n = \boldsymbol{a}^n \tag{6.2b}$$

, where $\boldsymbol{p}$, $\boldsymbol{v}$, and $\boldsymbol{a}$ are three-dimensional vectors describing position, velocity, and acceleration respectively. This is the model used in Solà (2017). Choosing to model the velocity in the body frame results in a slightly more complicated model:

$$\dot{\boldsymbol{p}}^n = \mathbf{R}(\boldsymbol{q})\boldsymbol{v}^b \tag{6.3a}$$

$$\dot{\boldsymbol{v}}^b = \boldsymbol{a}^b - \mathbf{S}(\boldsymbol{\omega}^b)\boldsymbol{v}^b \tag{6.3b}$$

, where $\boldsymbol{\omega}^b$ is the angular velocity. This model is used in Farrell (2008) and is also the basis for the hydrodynamical model in Section 6.1.1. When using quaternions as attitude representation, the attitude model becomes the quaternion differential equation:

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega}^b \end{bmatrix} \triangleq \frac{1}{2}\boldsymbol{q} \otimes \boldsymbol{\omega}^b \tag{6.4}$$

, where $\boldsymbol{q}$ is the quaternion. This is the most common, as the unit quaternion is a minimal singularity-free representation of attitude. Farrell (2008), Solà (2017) and Fossen (2021) all recommend this approach. Alternatively, it is possible to use the rotation matrix differential equation:

$$\dot{\boldsymbol{R}} = \boldsymbol{R}\mathbf{S}(\boldsymbol{\omega}^b) \tag{6.5}$$

where $\boldsymbol{R}$ is the rotation matrix. Gade (1997) uses this representation of attitude.

6.2 Observers

This section is a summary of the different observers that were considered for the ROV in this thesis. The main requirement is that they must be able to work in 6-Degree-Of-Freedom (DOF).

6.2.1 Complementary filter

Mahony et al. (2008) uses a classical control theory approach with feedback loops and integral bias estimation to aid the gyro measurements. The aiding is based on reference vectors. One for the direction of gravity measured through the accelerometer and one linearly independent of gravity measured through the magnetometer. The reference vectors are normalized to avoid any errors due to modeling or acceleration other than gravity or magnetic field other than that of the Earth. Orthogonalization of the reference vectors may also be used in order to improve performance. The version used here is from Grip et al. (2017) with slightly modified notation.

$$\boldsymbol{r}_m^b = \frac{\boldsymbol{y}_{mag}^b}{\left\|\boldsymbol{y}_{mag}^b\right\|} \tag{6.6a}$$

$$\boldsymbol{r}_f^b = \frac{\boldsymbol{y}_{acc}^b}{\|\boldsymbol{y}_{acc}^b\|} \tag{6.6b}$$

$$\boldsymbol{\sigma} = k_m(\boldsymbol{r}_m^b \times \mathbf{R}^\top(\hat{\boldsymbol{q}})\bar{\boldsymbol{r}}_m^n) + k_f(\boldsymbol{r}_f^b \times \mathbf{R}^\top(\hat{\boldsymbol{q}})\bar{\boldsymbol{r}}_f^n) \tag{6.6c}$$

$$\hat{\boldsymbol{\omega}}^b = \boldsymbol{y}_{gyro}^b - \hat{\boldsymbol{b}}_{gyro}^b \tag{6.6d}$$

$$\dot{\hat{\boldsymbol{q}}} = \frac{1}{2}\hat{\boldsymbol{q}} \otimes (\hat{\boldsymbol{\omega}} + \boldsymbol{\sigma}) \tag{6.6e}$$

$$\dot{\hat{\boldsymbol{b}}}_{gyro}^b = -\boldsymbol{K}_i\boldsymbol{\sigma} \tag{6.6f}$$

Where $\bar{\boldsymbol{r}}_m^n$ and $\bar{\boldsymbol{r}}_f^n$ are the reference vectors for the direction of the magnetic field and gravity in NED respectively. $\boldsymbol{\sigma}$ is the attitude error. $\boldsymbol{K}_i$, k_m and k_f are tuning constants.

There are also other options for complimentary filters, for instance, the gradient descent observer by Madgwick et al. (2011). Instead of the classical feedback loop, a gradient descent algorithm is used to fuse the aiding measurements into the attitude estimate. The observer is based on quaternions.

6.2.2 Integration filter

Dukan (2014) proposes a sensor-based nonlinear observer consisting of; the accelerometer measurement equation, translational kinematics, and a bias model. Injection terms are added in classical nonlinear observer fashion.

$$\hat{\boldsymbol{a}}^b = \boldsymbol{y}_{acc}^b - \hat{\boldsymbol{b}}_{acc}^b + \mathbf{R}^\top(\hat{\boldsymbol{q}})\boldsymbol{g}^n \tag{6.7a}$$

$$\dot{\hat{\boldsymbol{p}}}^n = \mathbf{R}(\hat{\boldsymbol{q}})\hat{\boldsymbol{v}}^b + K_{11}\boldsymbol{\delta y}_{pos}^n + K_{21}\,\mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{\delta y}_{vel}^b \tag{6.7b}$$

$$\dot{\hat{\boldsymbol{v}}}^b = \hat{\boldsymbol{a}}^b - \mathbf{S}(\hat{\boldsymbol{\omega}}^b)\hat{\boldsymbol{v}}^b + K_{12}\,\mathbf{R}^\top(\hat{\boldsymbol{q}})\boldsymbol{\delta y}_{pos}^n + K_{22}\boldsymbol{\delta y}_{vel}^b \tag{6.7c}$$

$$\dot{\hat{\boldsymbol{b}}}_{acc}^b = -K_{13}\,\mathbf{R}^\top(\hat{\boldsymbol{q}})\boldsymbol{\delta y}_{pos}^n - K_{23}\boldsymbol{\delta y}_{vel}^b \tag{6.7d}$$

$$\boldsymbol{\delta y}_{pos}^n = \boldsymbol{y}_{pos}^n - \hat{\boldsymbol{p}}^n \tag{6.7e}$$

$$\boldsymbol{\delta y}_{vel}^b = \boldsymbol{y}_{vel}^b - \hat{\boldsymbol{v}}^b \tag{6.7f}$$

Where $\boldsymbol{y}_{pos}^n$ is the available position measurement, in this case, a combination of USBL and pressure sensor measurements and $\boldsymbol{y}_{vel}^b$ would be the DVL measurement in this case. K_{ij} are tuning parameters.

6.2.3 Aided INS

Aided Inertial Navigation System (INS) is an observer based on an INS. The INS is a type of navigation system where the actual movement of the vessel is measured in terms of acceleration and angular velocity, and integrated to obtain position, velocity, and attitude. The INS needs to be ‘aided” using external measurements in order to counteract the drift caused by this (otherwise) open loop integration. In this case, we refer to aided INS as an INS aided using a Kalman filter. This results in an observer which allows for more advanced modeling of noise and otherwise inherit the strengths of the Kalman filter. Typically, the unit quaternion is used as attitude representation and then the observer is often called Multiplicative Extended Kalman Filter (MEKF) (Fossen, 2021). Aided INS can be used to estimate all 6-DOF through a neat trick where the attitude error is represented using 3 parameters.

6.2.4 Model-based Nonlinear passive obverver

The nonlinear passive observer is a model-based observer designed specifically for marine craft. The observer design is based on a model in the form of (6.1) and passivity arguments from nonlinear theory to simplify tuning and guarantee stability. The original observer was proposed for 3-DOF surface vessels (Fossen and Strand, 1999), but it can easily be extended to 6-DOF which is done in for instance Hval (2012). The passive observer requires at least an external measurement of position and a measurement/estimate of attitude in order to work. It is also possible to utilize a velocity measurement.

6.2.5 Model-based Extended Kalman filter

The Extended Kalman filter (EKF) is a Bayesian observer based on the original Kalman filter (Kalman, 1960). The EKF overcomes the limitation of the original Kalman Filter, which assumes linearity in system dynamics and measurements. The EKF is designed to estimate the system states by combining measurements from sensors (like USBL and DVL) with a mathematical model of the system, like the hydrodynamical model (6.1). The EKF approximates the system's nonlinear behavior by linearizing the model around the current estimated state. This linearization process involves calculating the Jacobian matrix, which represents the partial derivatives of the model equations with respect to the state variables. The EKF operates in two steps: the prediction step, where the system state is estimated based on the previous state and the system model, and the update step, where the predicted state is corrected using measurement data. The prediction step is run iteratively, with the update step following whenever measurements are available. The downsides of the EKF are primarily related to the linearization around current state estimates which leads to a filter where stability cannot be proven theoretically.

6.3 ROV operations at fish-farms

6.3.1 The environment in a fish farm

The inside of a net cage is an underwater, Global Navigation Satellite System (GNSS)-denied location where the ROV may not be possible to locate visually from the surface. The water is saline and gravity may be different from location to location but can be considered constant inside the limited area of the fish farm. Since the fish farms may be placed further from the coast than traditionally, the environmental conditions can be fairly rough. This includes winds while operating at the surface, but more importantly, ocean currents and dynamically changing wave loads since at least parts of the net pens are located in the wave zone. The net pens are filled with a large amount of fish. These may get in the way of sensors and cameras and may complicate operations. Lastly, the net pens themselves are constantly moving with currents and waves and the location and shape of the net are therefore difficult to model/predict.

6.3.2 Autonomous inspection through net following

Fish cages are fairly large structures, and manually controlling an ROV to inspect the entirety of the cages net can be a challenging and time-consuming/labor-intensive job. This is extra challenging in rough weather conditions when waves and currents apply forces which is difficult for a human to predict and counteract manually. This is motivation to design more advanced control systems for the ROV. Ideally, the ROV should be able to complete the task of traversing the net pen walls by itself. This is the objective of the master thesis of Amundsen (2020). The basic idea is to navigate using a DVL mounted to the front of the ROV. The DVL returns the distance to the net pen wall along 4 non-parallel beams. The attitude and distance relative to the wall become observable through these 4 measurements and along with a sufficient velocity estimate, this is sufficient to provide feedback for aLine-of-Sight (LOS) path following controller (Fossen, 2021) which integral effect can take care of the environmental forces acting on the vehicle. This means that the job of the ROV operator is reduced to setting a desired speed, distance to the net wall, and direction so that the ROV will traverse the net pen and deliver more or less smooth video for inspection of the net.

6.4 Sensors for underwater vehicles

6.4.1 Ultra Short Baseline

USBL is a type of acoustic positioning system used underwater. A USBL system consists of two parts; a transceiver, which is wired and returns the measurements to the surface, and a transponder which is wireless and only used as a beacon to locate for instance a ROV. The system works in the following way (Jaffre et al., 2005):

1. The transceiver sends an ”interrogation signal” (asking the transponder to respond).
2. The transponder receives the signal and waits a fixed amount of time before it answers with a signal.
3. The signal is received by the transceiver. The direction in which the signal is received (Direction Of Arrival (DOA)) is measured and combined with the receiver's attitude sensors to calculate the azimuth and elevation of the transponder relative to the transceiver.
4. The transceiver uses the time elapsed between sending and receiving to calculate the range to the transponder.
5. Cartesian coordinates are calculated from the spherical (range, azimuth, and elevation).

In general, USBL systems are the most basic type of underwater positioning since it only requires two parts. More advanced systems are characterized by the need for multiple transmitters in order to triangulate position. Examples are Short BaseLine (SBL) and

Long BaseLine (LBL), where the ”length of the baseline” refers to the distance between transmitters, and more distance means better triangulation.

6.4.2 Doppler Velocity log

Doppler Velocity Logs (DVL) are acoustic sensors that can be fitted to moving crafts in order to measure velocity (and potentially also distance). The DVLs working principle is based on the Doppler effect. Four non-parallel transmitters send a signal toward a reflecting surface and await the echo to return. When the echo is received, the frequency is measured and compared to the original signal. The change of frequency determines the speed along the transmitter direction. Since the four transmitters are non-parallel, velocity can be calculated in three dimensions. The time between sending and receiving can be used to calculate the distance to the reflecting surface (Farrell, 2008).

Chapter 7

Choise of Observer

This chapter covers the considerations made in the process of choosing the best observer for this specific vehicle, operating in the specific environment using the specific sensor suite. Section 7.1 states the performance requirements for the observer. Section 7.2 discusses the choice between a model-based and sensor-based observer design. Section 7.3 is a brief discussion on the choice between a Kalman filter and a nonlinear observer. Section 7.4 concludes on which type of design is chosen.

7.1 Performance requirements

The vehicle should be able to navigate using feedback from the observer. When used for inspection and maintenance it will be inside the net pens and fairly close to the net. This means that the position (and velocity) estimates must be fairly accurate to avoid the UUV getting tangled in the net, or worse; creating holes in the net.

Since fish, nets, and moorings are present, the hydroacoustic sensors (USBL and DVL) can be blocked. In these cases, the sensors will not return correct measurements or not return measurements at all (signal loss). This can happen every now and then, but also over longer time periods (multiple seconds). The observer must be robust to give sufficiently good estimates in these cases. This is often referred to as dead-reckoning capability.

The vehicle is light, small, fast, and agile. Acceleration, velocity, and angular velocity can change significantly in a very short amount of time. Since the observer output will be used as feedback to a control system, it is important that it is able to follow these changes to prevent unnecessary phase lag in the feedback loop. A high observer update rate is required to achieve this.

The vehicle is actuated in 6-DOF and can be controlled to any attitude. Whether the use case requires this to be possible is a different discussion, but for the robustness of the observer and to not impose limitations on the design of the control systems, the observer should be able to estimate all attitudes singularity free.

Since the vehicle will work in the wave zone and in locations where the current can be strong and dynamically changing, the observer must be able to function even though the external forces on the vehicle are non-constant.

7.2 Model-based vs sensor-based

There are two main categories of observers to choose from; model-based and sensor-based observers. This section discusses the pros and cons of the two.

7.2.1 Model-based observers and hydrodynamical modeling

Model-based observers are the go-to solution for larger ships and other crafts that have a hull that is easy to model using hydrodynamical models (Section 6.1.1). In the case of the BlueROV2, the shape of the hull is much more difficult to model. The BlueROV2 is built like a frame with thrusters inside. Finding coefficients for the drag and added mass is nontrivial. It should also be noted that the BlueROV2 (heavy) has a massive amount of thrust available compared to its size making it very agile and quick. The exact effect of this is thought difficult to model. The thrusters are also located and oriented in such a manner that their wake may affect each other as well as the hull/frame.

Nextly, the effect of the dynamically changing environment is difficult to take into consideration in a model-based observer. External forces are usually modeled as a residual force (a bias) with the assumption that it is constant or slowly varying. This assumption may not be reasonable.

A model-based observer would have to be designed as a hydrodynamical model with the vehicle's thrust as the input and USBL, DVL, and the pressure sensor as aiding measurements for position and velocity as well as the AHRS for attitude aiding. This utilizes the most important sensors in our sensor suite except the accelerometer.

7.2.2 Sensor-based observers and sensor quality

Sensor-based observers are based on measuring the actual movement of the vehicle. For comparison are model-based observers based on calculating what the movement should be. Measuring the movement renders hydrodynamical modeling and estimation of external forces unnecessary.

The quality of the sensors is vital to the performance of the observers. Whether the consumer-grade inertial sensors on the BlueROV2 are sufficient is uncertain. Inertial sensors are prone to biases causing drift in the observer estimates and the severity of these effects are dependent on the quality (and price tag) of the sensors themselves. The required performance for this application is that the inertial sensors must be able to maintain a sufficiently precise measurement during time periods when USBL measurements are not available. This is assumed to be periods of up to 5 seconds.

A sensor-based observer would require the accelerometer and gyro to be used to measure the movement of the vehicle, and USBL, DVL, and the pressure sensor for position and velocity aiding. Additionally, the AHRS can be used for aiding the attitude estimates.

7.3 Kalman filter vs nonlinear observer

It is also necessary to choose between Kalman filter observers and nonlinear observers. Nonlinear observers have the strength that they can be proven stable, which nonlinear extensions of the Kalman filter cannot. Nonlinear observers are also in general less demanding on computational power. The main strength of the Kalman filter is that along with estimating the system states, it also estimates the covariance of the state estimate. This is helpful to ensure fast convergence after the loss of signal from acoustic sensors. A Kalman filter will also utilize the covariance information reported by the USBL and DVL. This could help the observer to avoid putting too much emphasis on bad measurements.

7.4 Conclusion

In order to avoid hydrodynamical modeling and to ensure that the observer is able to capture the impact of the dynamically changing environment, we will go with a sensor-based approach. We will use a Kalman filter to incorporate the covariance information returned from the DVL and USBL. The combination is referred to as "Aided INS". There are still some decisions to be made. First and foremost, the attitude parametrization must be chosen. The quaternion will be used over the rotation matrix for its computational efficiency and overall simplicity. This allows for a singularity-free representation of attitude. Secondly, we must choose a kinematic model for the translational motion. The question is whether velocity should be represented in the body or NED-frame. In this case, the velocity sensor (the DVL) returns measurements in a frame that is rotating with the body frame. This means that if we choose to represent velocity in the body frame, the DVL measurements can enter the observer in a linear fashion (does not need to be rotated). For this reason, the velocity will be represented in the body frame.

Chapter 8

Background Information and Theory

This chapter provides the theory specific to the modeling and design phase of the observer. Sections 8.1 and 8.2 concern modeling of waves and water pressure due to waves. Section 8.3 contains theory specific to the aided INS observer design. Section 8.4 is a section on the discretization of stochastic state space systems, which is important in Kalman filtering. The concept of Normalized Innovation Squared (NIS) is introduced in Section 8.5, followed by a note on outlier rejection in Section 8.6. Dynamic positioning and wave filtering are presented in Section 8.7. Lastly, frequency estimation is discussed in Section 8.8.

8.1 Wave modeling

Ocean waves are often modeled using a wave spectrum describing which wave frequencies are present in the sea. Multiple such spectra have been introduced over the years, for instance, the JONSWAP spectrum and the Modified Pierson-Moskowitz spectrums (Fossen, 2021). Since the modeling of waves is incredibly complex, they all have in common that they are to some degree based on experimental data and are therefore approximations. In practice, wave spectrums have a peak frequency (the frequency with the largest magnitude), and have a bell shape around this frequency. A simplified model to generate such a spectrum is a second-order linear system driven by white noise on the form (Fossen, 2021):

$$\xi(s) = \frac{2\lambda\omega_0 s}{s^2 + 2\lambda\omega_0 s + \omega_0^2}\varepsilon(s) \tag{8.1}$$

, where ξ is the generated wave spectrum, ω_0 is the peak frequency, λ is a damping term that determines the shape of the wave spectrum, and ε is the driving white noise term, which variance governs the overall strength of the waves. The model (8.1) can be used to generate waves in for instance a simulator or to model waves in a Kalman filter or other observers, for instance, the passive observer in Fossen and Strand (1999).

8.2 Pressure due to waves

On the surface, waves appear to be vertical movements of water. This is also the case under the surface where the water molecules move vertically. Sub-surface waves can be considered as fluctuations in pressure at a specific point. The pressure can be modeled as a function of the waves at the surface. A 2D model is presented in Willumsen et al. (2007) which will be used as the primary source on this specific subject. The model is based on linear wave theory which is a topic we will not dive deeper into than absolutely necessary.

Let x be the horizontal and z the vertical position (below average surface level) of the point we want to examine. The depth from the surface to the sea bed is h. ρ and g are the water density and local gravity. The amplitude of the surface wave at time t and horizontal position x is $\eta(t,x)$, while k is the wave number. The component of the pressure that is due to waves becomes $p_w(x,z,t)$:

$$p_w(x,z,t) = -\rho g \frac{\cosh(k(h-z))}{\cosh(kh)} \eta(x,t) \tag{8.2}$$

From (8.2) we can see that the fraction reaches its minima at the bottom ($z = h$) and that it is equal to 1 at the surface ($z = 0$). In general, the effect of surface waves on pressure attenuates with depth. The model (8.2) can also be converted to the frequency domain (Willumsen et al., 2007) in order to observe how depth below the surface affects the wave spectrum. In addition to the general attenuation of amplitude, the higher frequencies tend to attenuate faster, leading to a peak frequency that gets lower with depth (and always lower than that on the surface).

8.3 Aided INS

8.3.1 Inertial navigation system

An INS is a 6-DOF dead-reckoning navigation system. The INS is based on integrating inertial measurements of angular velocity and specific force (acceleration including the acceleration due to gravity) into estimates of attitude, velocity, and position (Groves, 2008). The estimates in the INS will drift and become inaccurate over a relatively short time. Therefore, the INS needs so-called “aiding”, meaning external measurements are used to correct the INS-estimates. In aided INS, this is typically done using an Error State Kalman Filter (ESKF)

8.3.2 Error State Kalman Filter

The ESKF is a Bayesian estimation algorithm that can be used to estimate the error in a state estimate. The ESKF algorithm is practically the same as the Extended Kalman Filter (EKF). The ESKF algorithm can be divided into two parts, one is the prediction step:

$$\hat{\boldsymbol{\mathcal{P}}} \leftarrow \boldsymbol{F}\hat{\boldsymbol{\mathcal{P}}}\boldsymbol{F}^\top + \boldsymbol{\mathcal{Q}} \tag{8.3}$$

, where $\hat{\boldsymbol{\mathcal{P}}}$ is the state error covariance matrix, $\boldsymbol{F}$ is the discrete time error state transition matrix, and $\boldsymbol{\mathcal{Q}}$ is the discrete time process noise covariance matrix. The prediction step is run whenever the state estimate is updated through the process model. The other step is known as the update step:

$$\boldsymbol{\mathcal{S}} = \boldsymbol{H}\hat{\boldsymbol{\mathcal{P}}}\boldsymbol{H}^\top + \boldsymbol{\mathcal{R}} \tag{8.4a}$$
$$\boldsymbol{K} = \hat{\boldsymbol{\mathcal{P}}}\boldsymbol{H}^\top\boldsymbol{\mathcal{S}}^{-1} \tag{8.4b}$$
$$\boldsymbol{\delta x} = \boldsymbol{K}\boldsymbol{\delta y} \tag{8.4c}$$
$$\hat{\boldsymbol{\mathcal{P}}} \leftarrow (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\hat{\boldsymbol{\mathcal{P}}}(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})^\top + \boldsymbol{K}\boldsymbol{\mathcal{R}}\boldsymbol{K}^\top \tag{8.4d}$$

, where $\boldsymbol{\mathcal{S}}$ is the innovation covariance matrix, $\boldsymbol{H}$ is the measurement jacobian, $\boldsymbol{\mathcal{R}}$ is the measurement covariance matrix, $\boldsymbol{K}$ is the Kalman gain, $\boldsymbol{\delta x}$ is the estimated state error, and $\boldsymbol{\delta y}$ is the innovation. The update step is run whenever one or more measurements are available (Solà, 2017).

8.3.3 Aiding measurements

Aiding measurements are measurements used to correct the INS state estimates through the ESKF update step. The aiding measurements are (usually) modeled on the form:

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{\eta} \tag{8.5}$$

, where $\boldsymbol{h}(\boldsymbol{x})$ is a function of the true state $\boldsymbol{x}$, and $\boldsymbol{\eta}$ is a measurement noise vector. A predicted measurement $\hat{\boldsymbol{y}}$ can be constructed from the INS state estimate $\hat{\boldsymbol{x}}$ such that:

$$\hat{\boldsymbol{y}} = \boldsymbol{h}(\hat{\boldsymbol{x}}) \tag{8.6}$$

In order to be usable in the ESKF, the innovation $\boldsymbol{\delta y} = \boldsymbol{y} - \hat{\boldsymbol{y}}$ is calculated. From the innovation, it is possible to find a measurement Jacobian $\boldsymbol{H}$ with respect to the error state $\boldsymbol{\delta x}$ such that (Solà, 2017):

$$\boldsymbol{\delta y} = \boldsymbol{y} - \hat{\boldsymbol{y}} \approx \boldsymbol{H}\boldsymbol{\delta x} + \boldsymbol{\eta} \tag{8.7}$$

Where $\boldsymbol{\eta}$ is the measurement noise arising from the aiding measurements measurement model (8.5).

8.4 Discretization of stochastic state space systems

A system on the form:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{G}\boldsymbol{w} \tag{8.8}$$

, where $\boldsymbol{x}$ is the state vector, $\boldsymbol{A}$ is the system matrix, $\boldsymbol{G}$ is the noise matrix and the noise vector $\boldsymbol{w}$ is distributed as:

$$\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\mathcal{W}}) \tag{8.9}$$

, can be discretized over a time step Δt, to the form:

$$\boldsymbol{x} \leftarrow \boldsymbol{F}\boldsymbol{x} + \boldsymbol{n} \tag{8.10}$$

, where the transition matrix $\boldsymbol{F}$ can be found as:

$$\boldsymbol{F} = \exp\left(\boldsymbol{A}\Delta t\right) \tag{8.11}$$

, and the noise term $\boldsymbol{n}$ is distributed as:

$$\boldsymbol{n} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\mathcal{Q}}) \tag{8.12}$$

$\boldsymbol{\mathcal{Q}}$ can be found as (Bar-Shalom et al., 2001):

$$\boldsymbol{\mathcal{Q}} = \int_0^{\Delta t} \exp\left(\boldsymbol{A}(\Delta t - \tau)\right)\boldsymbol{G}\boldsymbol{\mathcal{W}}\boldsymbol{G}^\top \exp\left(\boldsymbol{A}^\top(\Delta t - \tau)\right)d\tau \tag{8.13}$$

The integral in (8.13) can be calculated efficiently through a power series expansion using the method of Van Loan (1978):

$$\exp\left(\begin{bmatrix} -\boldsymbol{A} & \boldsymbol{G}\boldsymbol{\mathcal{W}}\boldsymbol{G}^\top \\ \mathbf{0} & \boldsymbol{A}^\top \end{bmatrix} \Delta t\right) = \begin{bmatrix} \times & \boldsymbol{V}_2 \\ \mathbf{0} & \boldsymbol{V}_1 \end{bmatrix} \tag{8.14}$$

, where $\boldsymbol{\mathcal{Q}}$ is found as:

$$\boldsymbol{\mathcal{Q}} = \boldsymbol{V}_1^\top \boldsymbol{V}_2 \tag{8.15}$$

8.5 Normalized innovation squared

Normalized innovation squared (NIS) is a consistency metric commonly used in Kalman filtering. The NIS, ϵ is calculated like:

$$\epsilon = \boldsymbol{\delta y}^\top \boldsymbol{\mathcal{S}}^{-1} \boldsymbol{\delta y} \tag{8.16}$$

, where $\boldsymbol{\delta y}$ is the innovation vector and $\boldsymbol{\mathcal{S}}$ is the innovation covariance (Bar-Shalom et al., 2001). The general idea is to check whether the innovation covariance is consistent with the innovations observed. The NIS metric will in general follow a χ^2 distribution. To check for consistency, it is common to use a χ^2 confidence interval with lower and upper bounds:

$$\epsilon_{lower} = \text{chi2inv}(\frac{\alpha}{2}, n_y) \tag{8.17a}$$

$$\epsilon_{upper} = \text{chi2inv}(1 - \frac{\alpha}{2}, n_y) \tag{8.17b}$$

, where n_y is the number of elements in the innovation $\boldsymbol{\delta y}$, chi2inv is the inverse cumulative distribution function of the χ^2-distribution, and the parameter $\alpha \in (0, 1)$ determines the size of the confidence interval. The consistency check can be interpreted as: "We can be $100(1-\alpha)\%$ certain on consistency when $\epsilon \in [\epsilon_{lower}, \epsilon_{upper}]$". The NIS will naturally vary a lot over time and it can therefore be better to look at the average of the NIS over a number of samples N. This leads to the Average Normalized Innovation Squared (ANIS) metric:

$$\bar{\epsilon} = \frac{1}{N}\sum_{k=1}^{N} \epsilon_k \tag{8.18}$$

The ANIS confidence intervals become:

$$\bar{\epsilon}_{lower} = \text{chi2inv}(\frac{\alpha}{2}, N n_y)/N \tag{8.19a}$$

$$\bar{\epsilon}_{upper} = \text{chi2inv}(1 - \frac{\alpha}{2}, N n_y)/N \tag{8.19b}$$

8.6 Outlier rejection

Some sensors have issues with measurements that are outright wrong, meaning that they do not reflect reality at all and are unpredictable such that they cannot be modeled. In these cases, it may be necessary to make sure that measurements that are incompatible with the sensor model are classified as what we will refer to as outliers. The idea of outlier rejection is to avoid outliers from entering the observer and impacting the estimates.

8.7 Dynamic Positioning

For fully actuated vessels and vehicles it is possible to use feedback control systems to control position and attitude. This is more commonly known as Dynamic Positioning (DP). For surface vessels, GNSS and gyrocompasses are combined in an observer for position and heading feedback. Thrusters, main propellers, and rudders are actuators. Dynamic positioning is also possible for sub-surface vehicles but with different sensors. Dynamic positioning gives the operator the ability to keep the vehicle more or less still and maintain position/attitude in potentially demanding environments where manual control would be challenging.

8.7.1 Wave filtering

Wave filtering is an essential part of any dynamic positioning (DP) system for ships (Fossen, 2021). Wave filtering is used to prevent wave motion from propagating through the control system and applying oscillations to the actuators. This avoids wearing out the actuators and saves a considerable amount of energy in the process. For model-based marine craft observers, wave filtering has become a natural part of the navigation problem and is often, and with great benefits, included in the observer itself. For example, the passive observer by Fossen and Strand (1999).

For large ships, it is possible to get away with lowpass filtering measurements or estimates to remove the wave frequencies, as the closed loop bandwidth of such ships is lower than the wave frequencies occurring regularly at sea (Fossen, 2021). This means that all frequencies that the ships are able to create using their own motion will slip through the low pass filter. For smaller ships that are able to maneuver quicker, it is often necessary for wave filtering to have bandstop characteristics since they make take frequencies inside or even higher than the wave spectrum.

8.8 Peak frequency estimation

To avoid the addition of unnecessary phase lag and limit filtering out movement that is not due to wave motion, the wave filters should be tuned to only filter out the wave frequencies that are actually present. These frequencies can be different from time to time and even change during long missions. When using wave filtering with notch filter characteristics, it is common to set the notch frequency to the most dominant frequency among the wave frequencies that are present, this will from now on be referred to as the peak frequency. The peak frequency is not constant and needs to be estimated

8.8.1 Sinusoidal signal frequency estimator

Given a sinusoidal signal with amplitude a and phase ϕ on the form:

$$u(t) = a \sin(\omega t + \phi) \tag{8.20}$$

, it is possible to estimate the frequency of the sinusoidal signal, ω from u using an estimator (Aranovskiy et al., 2007; Belleter et al., 2015) on the form:

$$\dot{\xi}_1 = \xi_2 \tag{8.21a}$$

$$\dot{\xi}_2 = -\omega_0^2 \xi_1 - 2\omega_0 \xi_2 + \omega_0^2 u \tag{8.21b}$$

$$\dot{\hat{\Omega}} = k\xi_1(\dot{\xi}_2 - \hat{\Omega}\xi_1) \tag{8.21c}$$

$$\hat{\omega} = \sqrt{-\hat{\Omega}} \tag{8.21d}$$

, where the adaption gain k is a tuning parameter for the convergence rate and the natural frequency ω_0 should be chosen such that $\omega < \omega_0$ (Belleter et al., 2015).

Chapter 9

Modeling

Modeling is important in any observer design. For the observer chosen in Chapter 7, the modeling of sensors is central. Section Section 9.1 covers the modeling of the different sensors. The kinematics of the vehicle itself is modeled in Section 9.2. Lastly, pressure sensor biases are modeled in Section 9.3.

9.1 Sensors

This section focuses on developing measurement models for the sensors that will be a part of the design. These are, in the order they are presented; the accelerometer, gyro, DVL, USBL, and pressure sensor.

9.1.1 Acceletometer

The acceleration measures specific force in the body frame. Measuring specific force means that it includes the acceleration due to gravity ($\boldsymbol{g}^n$) in addition to the acceleration ($\boldsymbol{a}^b$) of the sensor itself. The accelerometer is subject to a slowly varying bias ($\boldsymbol{b}^b_{acc}$), as well as significant measurement noise ($\boldsymbol{w}^b_{acc}$). The measurement equation thus becomes:

$$\boldsymbol{y}^b_{acc} = \boldsymbol{a}^b - \mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{g}^n + \boldsymbol{b}^b_{acc} + \boldsymbol{w}^b_{acc} \tag{9.1}$$

9.1.2 Gyro

The gyro measures angular velocity ($\boldsymbol{\omega}^b_{gyro}$) in the body frame. This measurement is subject to a slowly varying bias ($\boldsymbol{b}^b_{gyro}$), and measurement noise ($\boldsymbol{w}^b_{acc}$). The resulting measurement equation is:

$$\boldsymbol{y}^b_{gyro} = \boldsymbol{\omega}^b_{gyro} + \boldsymbol{b}^b_{gyro} + \boldsymbol{w}^b_{acc} \tag{9.2}$$

9.1.3 Attitude and Heading Reference System

As mentioned in Chapter 2, the BlueROV2 has an internal AHRS based on its gyro, accelerometer, and magnetometer. This is to be used instead of the magnetometer. The main motivation for this approach is to be able to utilize the calibration software of the BlueROV that takes care of magnetic offsets and biases. The AHRS returns its attitude estimates in Euler angles. Due to the discontinuous nature of the Euler angles, we will not attempt to use them directly but instead convert them to a rotation matrix and use that in the model. We will however model the noise terms in roll, pitch, yaw, and linearized using a Jacobian. Using the AHRS estimates for roll, pitch and yaw; y_ϕ, y_θ, and y_ψ, let the AHRS measurement model be:

$$\begin{aligned}\boldsymbol{Y}_{ahrs} &= \mathbf{R}(y_\phi + \eta_\phi, y_\theta + \eta_\theta, y_\psi + \eta_\psi) \\ &\approx (\boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{\eta}^n_{ahrs}))\,\mathbf{R}(\boldsymbol{q})\end{aligned} \tag{9.3}$$

, where $\boldsymbol{\eta}_{ahrs}$ are the individual noise terms η_ϕ, η_θ, η_ψ transformed into a rotation vector in NED. Let the Euler angle noise terms be distributed as:

$$\underbrace{\begin{bmatrix}\eta_\phi \\ \eta_\theta \\ \eta_\psi\end{bmatrix}}_{\boldsymbol{\eta}_{\phi\theta\psi}} \sim \mathcal{N}\left(\begin{bmatrix}0\\0\\0\end{bmatrix}, \underbrace{\begin{bmatrix}\sigma_\phi^2 & 0 & 0 \\ 0 & \sigma_\theta^2 & 0 \\ 0 & 0 & \sigma_\psi^2\end{bmatrix}}_{\boldsymbol{\mathcal{R}}_{\phi\theta\psi}}\right) \tag{9.4}$$

, where $\boldsymbol{\mathcal{R}}_{\phi\theta\psi}$ is the covariance matrix. The transformation jacobian (see Section 5.2.7) is defined such that:

$$\boldsymbol{\eta}^n_{ahrs} = \boldsymbol{T}(y_\theta, y_\psi)\boldsymbol{\eta}_{\phi\theta\psi} \tag{9.5}$$

This results $\boldsymbol{\eta}^n_{ahrs}$ being distributed as:

$$\boldsymbol{\eta}^n_{ahrs} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{T}(y_\theta, y_\psi)\boldsymbol{\mathcal{R}}_{\phi\theta\psi}\boldsymbol{T}(y_\theta, y_\psi)^\top) \tag{9.6}$$

9.1.4 Doppler Velocity Log

The Doppler Velocity Log measures velocity along three axes in a frame fixed to the DVL. The DVL has additive measurement noise in the same frame as the measurement. This results in the following measurement equation:

$$\boldsymbol{y}^d_{dvl} = \boldsymbol{v}^d_{d/n} + \boldsymbol{\eta}^d_{dvl} \tag{9.7}$$

The DVL is mounted to the vehicle's body frame at an orientation described by a rotation matrix named $\boldsymbol{D}$ (see Section 4.3.1). We use this matrix to rotate the measurement into the body frame, resulting in:

$$\boldsymbol{D}\boldsymbol{y}^d_{dvl} = \boldsymbol{v}^b_{d/n} + \boldsymbol{D}\boldsymbol{\eta}^d_{dvl} \tag{9.8}$$

The DVL is mounted with a lever arm $\boldsymbol{r}^b_{d/b}$ from the body CO such that:

$$\boldsymbol{D}\boldsymbol{y}^d_{dvl} = \boldsymbol{v}^b_{b/n} - \mathbf{S}(\boldsymbol{r}^b_{d/b})\boldsymbol{\omega}^b + \boldsymbol{D}\boldsymbol{\eta}^d_{dvl} \tag{9.9}$$

9.1.5 Ultra-Short BaseLine

The USBL measures the position of the transponder $p^n_{tp/n}$ in NED. The transponder is offset from the body Coordinate Origin (CO) by a lever arm $r^b_{tp/b}$. The measurement noise is modeled as additive with a white noise term $\boldsymbol{\eta}^n_{usbl}$. The measurement equation thus becomes:

$$\boldsymbol{y}^n_{usbl} = \boldsymbol{p}^n + \mathbf{R}(\boldsymbol{q})\boldsymbol{r}^b_{tp} + \boldsymbol{\eta}^n_{usbl} \tag{9.10}$$

9.1.6 Pressure Sensor

The pressure sensor on the ROV will be used to determine depth. For many applications, the conversion between pressure and depth are straight forward. However, since the ROV will operate in the sea where tides are present and in environments with significant waves, the picture is more complicated. This subsection covers modeling of the components that contribute to the pressure sensor measurement. Figure 9.1 is meant to give an overview of the four main contributors to the pressure measurement and these will be explained in more detail in the following paragraphs.

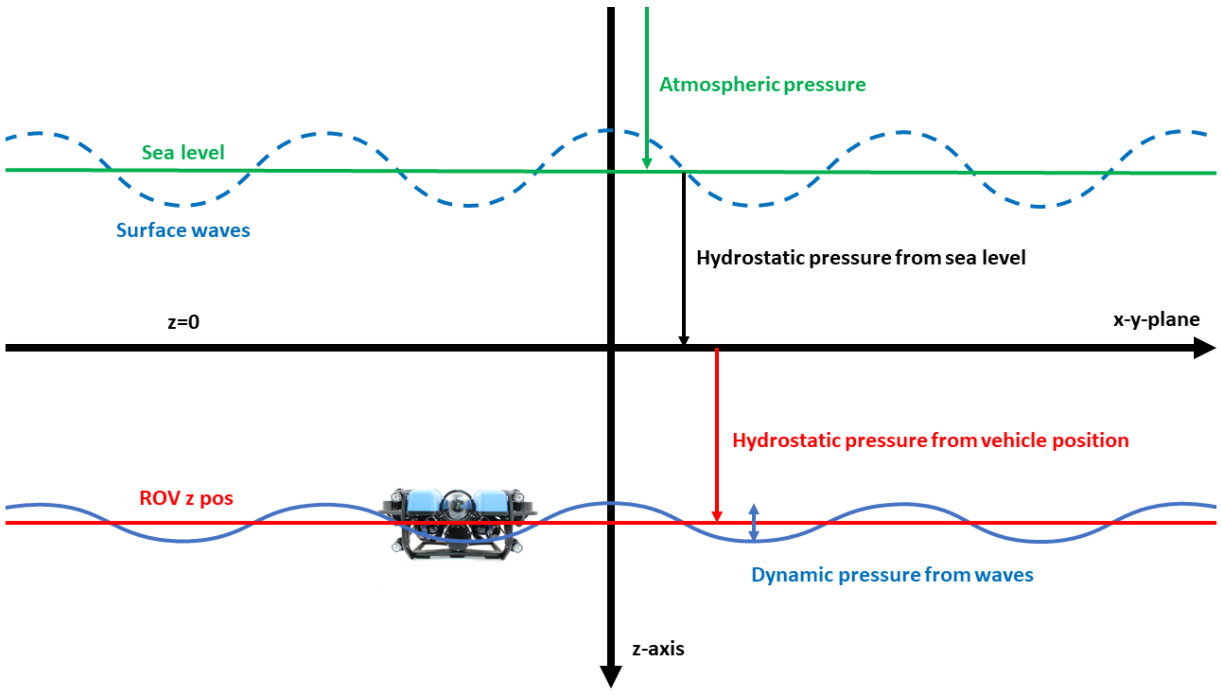
Atmospheric pressure
Sea level
Surface waves
Hydrostatic pressure from sea level
z=0
x-y-plane
Hydrostatic pressure from vehicle position
ROV z pos
Dynamic pressure from waves
z-axis

Figure 9.1: Pressure contributions as seen from the pressure sensor on the ROV.

Atmospheric pressure

The pressure sensor measures absolute pressure meaning that the atmospheric pressure will be a significant contributor to the total measurement. The atmospheric pressure at sea level is in general nonstatic and changes slowly over time.

Hydrostatic pressure from sea level

Another major contributor is the hydrostatic pressure resulting from the sea level being different than NED-zero ($z = 0$). This offset can change over time due to tides. It should be noted that when defined as in Figure 9.1, this pressure component can become negative. However, the sum of all contributors will always be positive and therefore physical. The rate of change of this component is in general very slow.

Hydrostatic pressure from vehicle position

The pressure component resulting from the ROVs vertical offset from the coordinate origin is the component we are interested in using. This can be used to compute the vertical offset. There are multiple formulas to do so, some take into account factors like; water temperature, salinity, density, and compressibility. It is also not uncommon for seawater to settle into layers of different densities. Since the ROV will not operate very deep and we want to keep things simple, we will assume that the pressure is proportional to the position through the density of the water and the local gravity:

$$pressure = \rho(\boldsymbol{g}^n)^\top \boldsymbol{p}^n_{ps/n} \tag{9.11}$$

Note that the formula also allows the ROV to operate above $z = 0$ which is important since the actual sea level can be above this plane as seen in figure 9.1.

Dynamic pressure from waves

As discussed in Section 8.2, the wave motion also induces wave frequencies in the pressure measurements. In rough conditions, these fluctuations are considered to be so significant that they will be inconsistent with the movement measured by the inertial sensors (IMU). For this reason, we want to model them as a part of the pressure measurement in order to filter them out. The frequencies present take the shape of a wave spectrum, and we expect the pressure measurement to be able to change with frequencies both higher and lower than this spectrum, put another way; the frequencies of the pressure fluctuations lie inside the closed-loop bandwidth of the ROV. This is important to consider when choosing a filtering technique for the pressure measurement.

Model

Finally, the sensor suffers from a certain amount of measurement noise. This can be added to the pressure contributions from the previous subsections to get a model:

$$\begin{aligned} y_{ps} &= b_{atm} + b_{lf} + b_{wf} + \rho(\boldsymbol{g}^n)^\top \boldsymbol{p}^n_{ps/n} + \eta_{ps} \\ &= b_{atm} + b_{lf} + b_{wf} + \rho(\boldsymbol{g}^n)^\top (\boldsymbol{p}^n_{b/n} + \mathbf{R}(\boldsymbol{q})\boldsymbol{r}^b_{ps/b}) + \eta_{ps} \end{aligned} \tag{9.12}$$

Here y_{ps} is the pressure sensor measurement, b_{atm} is a constant bias set to be the standard atmosphere, b_{lf} is a slowly varying bias consisting of changes in atmospheric pressure and sea level. b_{wf} is a bias resulting from waves and is therefore assumed to be oscillating at the wave frequency.

9.2 Kinematics model

To estimate the motion of the ROV, we need a model of the ROVs kinematics. Since this is a sensor-based observer, the model will be based on the measurement equations for the accelerometer (9.1) and the gyro (9.2). These can be rearranged to describe acceleration and angular velocity:

$$\boldsymbol{a}^b = \boldsymbol{y}^b_{acc} - \boldsymbol{b}^b_{acc} + \mathbf{R}^\top(\boldsymbol{q})\boldsymbol{g}^n - \boldsymbol{w}^b_{acc} \tag{9.13a}$$
$$\boldsymbol{\omega}^b = \boldsymbol{y}^b_{gyro} - \boldsymbol{b}^b_{gyro} - \boldsymbol{w}^b_{gyro} \tag{9.13b}$$

The acceleration and angular velocity can then be integrated through a kinematics model:

$$\dot{\boldsymbol{p}}^n = \mathbf{R}(\boldsymbol{q})\boldsymbol{v}^b \tag{9.13c}$$
$$\dot{\boldsymbol{v}}^b = \boldsymbol{a}^b - \mathbf{S}(\boldsymbol{\omega}^b)\boldsymbol{v}^b \tag{9.13d}$$
$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes \boldsymbol{\omega}^b \tag{9.13e}$$

Lastly, the biases for the accelerometer and gyro are modeled as slowly varying through Gauss-Markov processes. Their dynamics take the form:

$$\dot{\boldsymbol{b}}^b_{acc} = -\frac{1}{T_{acc}}\boldsymbol{b}^b_{acc} + \boldsymbol{w}^b_{b,acc} \tag{9.13f}$$
$$\dot{\boldsymbol{b}}^b_{gyro} = -\frac{1}{T_{gyro}}\boldsymbol{b}^b_{gyro} + \boldsymbol{w}^b_{b,gyro} \tag{9.13g}$$

Together, the equations (9.13) constitute a complete model of the dynamic behavior of the vehicle including sensor biases and noise.

9.3 Pressure biases

As discussed in Section 9.1.6, the pressure sensor will measure a slowly varying bias due to variations in sea level and atmospheric pressure. Additionally, there will be a component that is oscillating due to pressure from waves. We want a stochastic model for both to allow for use in a Kalman filter. The slow varying component due to sea level and atmospheric pressure is modeled as a gauss-markov process:

$$b_{lf}(s) = \frac{1}{s + \frac{1}{T_{lf}}} w_{lf}(s) \tag{9.14}$$

, where T_{lf} is a (large) time constant and w_{lf} is a unit white noise term.

The wave frequency bias is less straightforward, Willumsen et al. (2007) models the wave motion as a Gauss-Markov process. This is reasonable when the closed-loop bandwidth is lower than the wave frequencies since it gives the filter low-pass properties. In this case, the bandwidth overlaps with the wave frequencies and may even be higher. To model the bias we will use a second-order wave spectrum approximation driven by white noise (see Section 8.1):

$$b_{wf}(s) = \frac{2\lambda\omega_w s}{s^2 + 2\lambda\omega_w s + \omega_w^2} w_{wf}(s) \tag{9.15}$$

This approach is the same as is used to achieve wave filtering in the passive observer in Fossen (2021). A realization of (9.15) is:

$$\dot{b}_{wf} = -2\lambda\omega_w(t)b_{wf} - \omega_w^2(t)b_{int} + 2\lambda\omega_w(t)w_{wf} \tag{9.16a}$$
$$\dot{b}_{int} = b_{wf} \tag{9.16b}$$

, we couple this with the realization of (9.14):

$$\dot{b}_{lf} = -\frac{1}{T_{lf}}b_{lf} + w_{lf} \tag{9.17}$$

, to obtain a linear time-variant state space system:

$$\underbrace{\begin{bmatrix} \dot{b}_{lf} \\ \dot{b}_{wf} \\ \dot{b}_{int} \end{bmatrix}}_{\dot{\boldsymbol{b}}_{pbe}} = \underbrace{\begin{bmatrix} -\frac{1}{T_{lf}} & 0 & 0 \\ 0 & -2\lambda\omega_w(t) & -\omega_w^2(t) \\ 0 & 1 & 0 \end{bmatrix}}_{\boldsymbol{A}_{pbe}(t)} \underbrace{\begin{bmatrix} b_{lf} \\ b_{wf} \\ b_{int} \end{bmatrix}}_{\boldsymbol{b}_{pbe}} + \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 2\lambda\omega_w(t) \\ 0 & 0 \end{bmatrix}}_{\boldsymbol{G}_{pbe}(t)} \underbrace{\begin{bmatrix} w_{lf} \\ w_{wf} \end{bmatrix}}_{\boldsymbol{w}_{pbe}} \tag{9.18}$$

, which is stable for positive T_{lf}, λ and ω_e. The noise vector $\boldsymbol{w}_{pbe}$ is distributed as:

$$\boldsymbol{w}_{pbe} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\mathcal{W}}_{pbe}) \tag{9.19}$$

, where:

$$\boldsymbol{\Sigma}_{pbe} = \begin{bmatrix} \sigma_{lf}^2 & 0 \\ 0 & \sigma_{wf}^2 \end{bmatrix} \tag{9.20}$$

Chapter 10

Observer Design

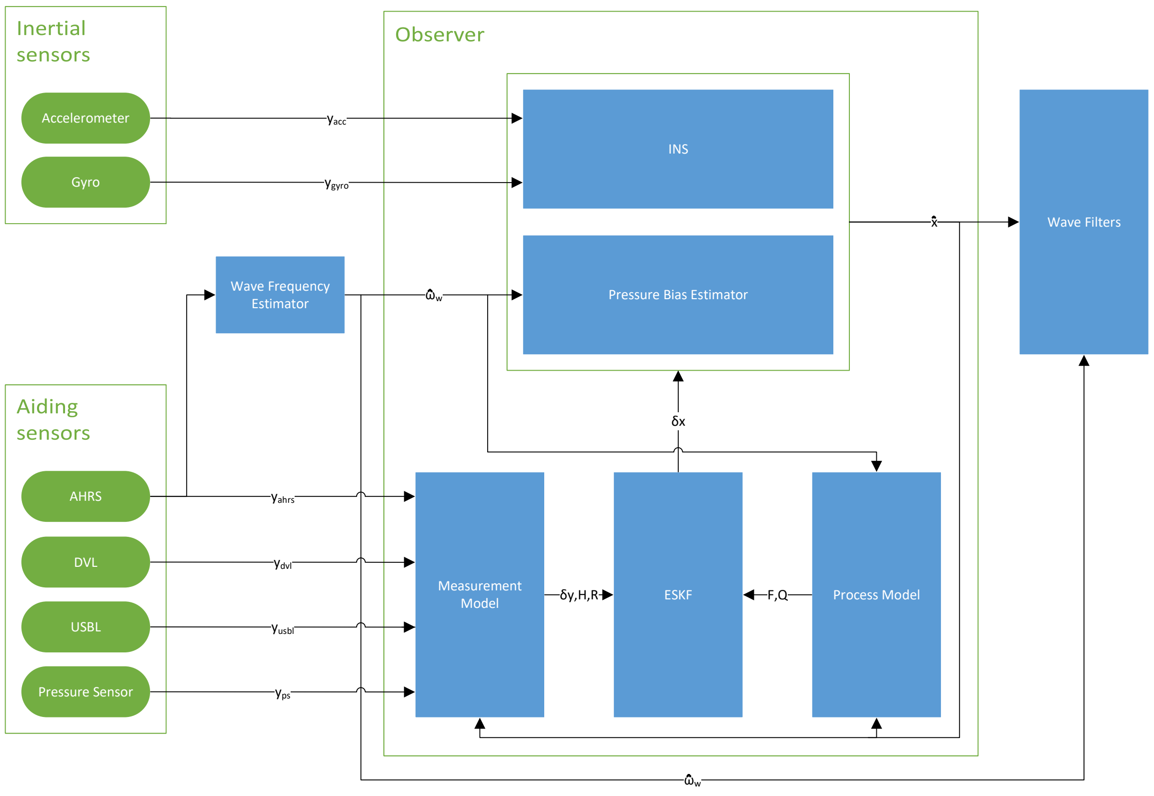

Figure 10.1: A block diagram of the observer design

This chapter covers the mathematical design of the observer. The chapter is structured module-wise. The modules and the signals between them are depicted in Figure 10.1. The core of the observer is the INS which is derived in Section 10.1. Parallel to this, the pressure bias estimator will be used to compensate for biases in the pressure measurement. Section 10.2 covers the design of this module. The INS and the pressure bias estimator needs to be corrected to avoid drift. This is done via an ESKF which is presented in Section 10.3. The ESKF requires a process model of the INS and the pressure bias estimator to be able to work. This is presented in Section 10.4 and the mathematical derivations are in Appendix A.1. To be able to utilize external measurements, the ESKF requires measurement models of the individual sensors. These are presented in Section 10.5 and derived in Appendix A.2. The wave filter is presented in Section 10.6. Finally, the wave frequency estimator is put together in Section 10.7.

10.1 Inertial navigation system

The idea of the inertial navigation system is to integrate accelerometer and gyro measurements to obtain position, velocity, and attitude. The equations for the inertial navigation system are found as the deterministic part of the kinematics model (9.13). The resulting continuous time inertial navigation system equations become:

$$\hat{\boldsymbol{a}}^b = \boldsymbol{y}^b_{acc} - \hat{\boldsymbol{b}}^b_{acc} + \mathbf{R}^\top(\hat{\boldsymbol{q}})\bar{\boldsymbol{g}}^n \tag{10.1a}$$
$$\hat{\boldsymbol{\omega}}^b = \boldsymbol{y}^b_{gyro} - \hat{\boldsymbol{b}}^b_{gyro} \tag{10.1b}$$
$$\dot{\hat{\boldsymbol{p}}}^n = \mathbf{R}(\hat{\boldsymbol{q}})\hat{\boldsymbol{v}}^b \tag{10.1c}$$
$$\dot{\hat{\boldsymbol{v}}}^b = \hat{\boldsymbol{a}}^b - \mathbf{S}(\hat{\boldsymbol{\omega}}^b)\hat{\boldsymbol{v}}^b \tag{10.1d}$$
$$\dot{\hat{\boldsymbol{q}}} = \frac{1}{2}\hat{\boldsymbol{q}} \otimes \hat{\boldsymbol{\omega}}^b \tag{10.1e}$$
$$\dot{\hat{\boldsymbol{b}}}^b_{acc} = -\frac{1}{T_{acc}}\hat{\boldsymbol{b}}^b_{acc} \tag{10.1f}$$
$$\dot{\hat{\boldsymbol{b}}}^b_{gyro} = -\frac{1}{T_{gyro}}\hat{\boldsymbol{b}}^b_{gyro} \tag{10.1g}$$

10.1.1 Discretization

The INS equations must be discretized before implementation.

$$\hat{\boldsymbol{a}}^b = \boldsymbol{y}^b_{acc} - \hat{\boldsymbol{b}}^b_{acc} + \mathbf{R}^\top(\hat{\boldsymbol{q}})\bar{\boldsymbol{g}}^n \tag{10.2a}$$
$$\hat{\boldsymbol{\omega}}^b = \boldsymbol{y}^b_{gyro} - \hat{\boldsymbol{b}}^b_{gyro} \tag{10.2b}$$
$$\hat{\boldsymbol{p}}^n \leftarrow \hat{\boldsymbol{p}}^n + \mathbf{R}(\hat{\boldsymbol{q}})\hat{\boldsymbol{v}}^b\Delta t + \frac{1}{2}\mathbf{R}(\hat{\boldsymbol{q}})\hat{\boldsymbol{a}}^b\Delta t^2 \tag{10.2c}$$
$$\hat{\boldsymbol{v}}^b \leftarrow \hat{\boldsymbol{v}}^b + (\hat{\boldsymbol{a}}^b - \mathbf{S}(\hat{\boldsymbol{\omega}}^b)\hat{\boldsymbol{v}}^b)\Delta t \tag{10.2d}$$
$$\hat{\boldsymbol{q}} \leftarrow \hat{\boldsymbol{q}} \otimes \mathbf{q}\{\hat{\boldsymbol{\omega}}^b\Delta t\} \tag{10.2e}$$
$$\hat{\boldsymbol{b}}^b_{acc} \leftarrow (1 - \frac{\Delta t}{T_{acc}})\hat{\boldsymbol{b}}^b_{acc} \tag{10.2f}$$
$$\hat{\boldsymbol{b}}^b_{gyro} \leftarrow (1 - \frac{\Delta t}{T_{gyro}})\hat{\boldsymbol{b}}^b_{gyro} \tag{10.2g}$$

The integration technique used here is from Solà (2017). This is mainly Euler discretization, however, in the quaternion differential equation, the angular velocity estimate is integrated into a rotation vector ($\hat{\boldsymbol{\omega}}^b\Delta t$) before being used to update the quaternion estimate. This avoids the bad practice of Euler integration on quaternions which results in a quaternion of non-unit length.

10.1.2 State errors and INS injection

If the INS is just propagated using the equations found in the section above, its estimates would quickly drift away from the true states. To compensate, we will use an ESKF to calculate the error in the states. We define the state errors as follows:

$$\boldsymbol{p}^n = \hat{\boldsymbol{p}}^n + \boldsymbol{\delta p}^n \tag{10.3a}$$
$$\boldsymbol{v}^b = \hat{\boldsymbol{v}}^b + \boldsymbol{\delta v}^b \tag{10.3b}$$
$$\boldsymbol{q} = \boldsymbol{\delta q} \otimes \hat{\boldsymbol{q}} \tag{10.3c}$$
$$\boldsymbol{b}^b_{acc} = \hat{\boldsymbol{b}}^b_{acc} + \boldsymbol{\delta b}^b_{acc} \tag{10.3d}$$
$$\boldsymbol{b}^b_{gyro} = \hat{\boldsymbol{b}}^b_{gyro} + \boldsymbol{\delta b}^b_{gyro} \tag{10.3e}$$

, where the error state ($\boldsymbol{\delta}$-state) is the error between the true state and the INS estimates ($\hat{}$-state). Note that the quaternion error is premultiplied with the quaternion estimate. This leads to a rotation vector defined in NED when the quaternion error is swapped with a rotation vector like:

$$\boldsymbol{\delta q} \triangleq \mathrm{q}\{\boldsymbol{\delta \alpha}^n\} \tag{10.4}$$

This has a significant impact on the dynamics of this error state and is recommended by Li and Mourikis (2012). According to (10.3), we have to alter the INS state estimates by the error state in order to obtain the true state. This gives birth to the error state injection procedure:

$$\hat{\boldsymbol{p}^n} \leftarrow \hat{\boldsymbol{p}}^n + \boldsymbol{\delta p}^n \tag{10.5a}$$
$$\hat{\boldsymbol{v}}^b \leftarrow \hat{\boldsymbol{v}}^b + \boldsymbol{\delta v}^b \tag{10.5b}$$
$$\hat{\boldsymbol{q}} \leftarrow \mathrm{q}\{\boldsymbol{\delta \alpha}^n\} \otimes \hat{\boldsymbol{q}} \tag{10.5c}$$
$$\hat{\boldsymbol{b}}^b_{acc} \leftarrow \hat{\boldsymbol{b}}^b_{acc} + \boldsymbol{\delta b}^b_{acc} \tag{10.5d}$$
$$\boldsymbol{b}^b_{gyro} \leftarrow \hat{\boldsymbol{b}}^b_{gyro} + \boldsymbol{\delta b}^b_{gyro} \tag{10.5e}$$

The injection is to be run whenever an error state estimate is available from the ESKF.

10.2 Pressure bias estimator

Similar to the inertial navigation system, the pressure bias estimator is the deterministic part of the pressure bias model (9.18):

$$\underbrace{\begin{bmatrix} \dot{b}_{lf} \\ \dot{b}_{wf} \\ \dot{b}_{int} \end{bmatrix}}_{\dot{\hat{\boldsymbol{b}}}_{pbe}} = \underbrace{\begin{bmatrix} -\frac{1}{T_{lf}} & 0 & 0 \\ 0 & -2\lambda\hat{\omega}_w(t) & -\hat{\omega}^2_w(t) \\ 0 & 1 & 0 \end{bmatrix}}_{\hat{\boldsymbol{A}}_{pbe}(t)} \underbrace{\begin{bmatrix} b_{lf} \\ b_{wf} \\ b_{int} \end{bmatrix}}_{\hat{\boldsymbol{b}}_{pbe}} \tag{10.6}$$

, where the peak frequency has been replaced by an estimate of the peak frequency to make the bias estimator adaptive.

10.2.1 Discretization

Assuming that the encounter frequency is constant (in practice: slowly varying) over the time step Δt, the system (10.6) can be discretized using first-order Euler discretization.

$$\boldsymbol{F}_{pbe}(t) = \begin{bmatrix} 1 - \frac{\Delta t}{T_{lf}} & 0 & 0 \\ 0 & 1 - 2\lambda\hat{\omega}_w(t)\Delta t & -\hat{\omega}_w^2(t)\Delta t \\ 0 & \Delta t & 1 \end{bmatrix} \tag{10.7}$$

The bias estimator can be updated the following way:

$$\hat{\boldsymbol{b}}_{pbe} \leftarrow \boldsymbol{F}_{pbe}(t)\hat{\boldsymbol{b}}_{pbe} \tag{10.8}$$

10.2.2 Pressure bias error and injection

Similar to the INS, the error state is defined as:

$$\boldsymbol{b}_{pbe} = \hat{\boldsymbol{b}}_{pbe} + \boldsymbol{\delta b}_{pbe}, \tag{10.9}$$

which lead to the injection step:

$$\hat{\boldsymbol{b}}_{pbe} \leftarrow \hat{\boldsymbol{b}}_{pbe} + \boldsymbol{\delta b}_{pbe} \tag{10.10}$$

10.3 Error state Kalman filter

An ESKF will be used to calculate the state error $\boldsymbol{\delta x}$ which we define as:

$$\boldsymbol{\delta x} = \begin{bmatrix} \boldsymbol{\delta p}^n \\ \boldsymbol{\delta v}^b \\ \boldsymbol{\delta \alpha}^n \\ \boldsymbol{\delta b}^b_{acc} \\ \boldsymbol{\delta b}^b_{gyro} \\ \boldsymbol{\delta b}_{pbe} \end{bmatrix} \tag{10.11}$$

This is achieved through an estimate of the covariance of $\boldsymbol{\delta x}$ called $\hat{\boldsymbol{\mathcal{P}}}$.

10.3.1 Prediction Step

The covariance is updated every time the INS and pressure bias estimator are updated using what is known as the prediction step defined as:

$$\hat{\boldsymbol{\mathcal{P}}} \leftarrow \boldsymbol{F}\hat{\boldsymbol{\mathcal{P}}}\boldsymbol{F}^\top + \boldsymbol{\mathcal{Q}} \tag{10.12}$$

, where $\boldsymbol{F}$ is a the discrete time state transition matrix and $\boldsymbol{\mathcal{Q}}$ is the discretized process noise covariance matrix. Expressions for these matrices will be derived later.

10.3.2 Update step

The ESKF relies on measurements to calculate the state error $\boldsymbol{\delta x}$ through what is known as the update step:

$$\boldsymbol{\mathcal{S}} = \boldsymbol{H}\hat{\boldsymbol{\mathcal{P}}}\boldsymbol{H}^\top + \boldsymbol{\mathcal{R}} \tag{10.13a}$$

$$\boldsymbol{K} = \hat{\boldsymbol{\mathcal{P}}}\boldsymbol{H}^\top\boldsymbol{\mathcal{S}}^{-1} \tag{10.13b}$$

$$\boldsymbol{\delta x} = \boldsymbol{K}\boldsymbol{\delta y} \tag{10.13c}$$

$$\hat{\boldsymbol{\mathcal{P}}} \leftarrow (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})\hat{\boldsymbol{\mathcal{P}}}(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})^\top + \boldsymbol{K}\boldsymbol{\mathcal{R}}\boldsymbol{K}^\top \tag{10.13d}$$

, where $\boldsymbol{H}$ is the measurement jacobian, $\boldsymbol{\mathcal{R}}$ is the measurement covariance, and $\boldsymbol{\delta y}$ is the measurement innovation (difference between the actual measurement and what the measurement is expected to be according to observer state). $\boldsymbol{\mathcal{S}}$ is the covariance of the innovation $\boldsymbol{\delta y}$. $\boldsymbol{K}$ is an intermediary matrix commonly known as the Kalman gain. $\boldsymbol{H}$ and $\boldsymbol{\mathcal{R}}$ can be concatenated from the measurements available at each time step. We will derive expressions for $\boldsymbol{\delta y}$, $\boldsymbol{H}$, and $\boldsymbol{\mathcal{R}}$ for each sensor later.

10.3.3 Reset

When the ESKF is used with attitude error as a state, Solà (2017) recommends a third step in the ESKF algorithm called the reset step. This is to be run along with the INS injection Section 10.1.2 to ensure that the attitude error covariance is correct <u>after</u> injection. With our choice of states, the reset operation is a jacobian on the form:

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \boldsymbol{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{I}_3 + \frac{1}{2}\mathrm{S}(\boldsymbol{\delta\alpha}^n) & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{I}_3 \end{bmatrix} \tag{10.14}$$

, which is used to reset the covariance as follows:

$$\hat{\boldsymbol{P}} \leftarrow \boldsymbol{J}\hat{\boldsymbol{P}}\boldsymbol{J}^\top \tag{10.15}$$

10.4 Process model

The goal of this section is to arrive at a process model for use in the ESKF prediction step, i.e. find expressions for the $\boldsymbol{F}$ and $\boldsymbol{\mathcal{Q}}$ matrices. Section 10.4.1 present the linearized error state dynamical system which is, in turn, discretized in Section 10.4.2.

10.4.1 Error state dynamics

. We want to obtain differential equations for the error state. Time differentiation of the error state definition in (10.3) and (10.9) yields:

$$\dot{\boldsymbol{\delta p}}^n = \dot{\boldsymbol{p}}^n - \dot{\hat{\boldsymbol{p}}}^n \tag{10.16a}$$

$$\dot{\boldsymbol{\delta v}}^b = \dot{\boldsymbol{v}}^b - \dot{\hat{\boldsymbol{v}}}^b \tag{10.16b}$$

$$\dot{\boldsymbol{\delta q}} = \dot{\boldsymbol{q}} \otimes \hat{\boldsymbol{q}}^* + \boldsymbol{q} \otimes \dot{\hat{\boldsymbol{q}}}^* \tag{10.16c}$$

$$\dot{\boldsymbol{\delta b}}^b_{acc} = \dot{\boldsymbol{b}}^b_{acc} - \dot{\hat{\boldsymbol{b}}}^b_{acc} \tag{10.16d}$$

$$\dot{\boldsymbol{\delta b}}^b_{gyro} = \dot{\boldsymbol{b}}^b_{gyro} - \dot{\hat{\boldsymbol{b}}}^b_{gyro} \tag{10.16e}$$

$$\dot{\boldsymbol{\delta b}}_{pbe} = \dot{\boldsymbol{b}}_{pbe} - \dot{\hat{\boldsymbol{b}}}_{pbe} \tag{10.16f}$$

, notice that the quaternion error requires the product rule when differentiated due to its multiplicative nature. The error state dynamics are derived from inserting the true state model (9.13) and the INS differential equations (10.1) into (10.16). The derivation is carried out in Appendix A.1. The result is a set of differential equations that can be linearized around INS state estimates and put on matrix form:

$$\underbrace{\begin{bmatrix} \dot{\boldsymbol{\delta p}}^n \\ \dot{\boldsymbol{\delta v}}^b \\ \dot{\boldsymbol{\delta \alpha}}^n \\ \dot{\boldsymbol{\delta b}}^b_{acc} \\ \dot{\boldsymbol{\delta b}}^b_{gyro} \\ \dot{\boldsymbol{\delta b}}_{pbe} \end{bmatrix}}_{\dot{\boldsymbol{\delta x}}} = \underbrace{\begin{bmatrix} \mathbf{0}_3 & \mathbf{R}(\hat{\boldsymbol{q}}) & -\mathbf{S}(\mathbf{R}(\hat{\boldsymbol{q}})\hat{\boldsymbol{v}}^b) & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & -\mathbf{S}(\hat{\boldsymbol{\omega}}^b) & \mathbf{R}^\top(\hat{\boldsymbol{q}})\,\mathbf{S}(\boldsymbol{g}^n) & -\boldsymbol{I}_3 & -\mathbf{S}(\hat{\boldsymbol{v}}^b) & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & -\mathbf{R}(\hat{\boldsymbol{q}}) & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \frac{-1}{T_{acc}}\boldsymbol{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \frac{-1}{T_{gyro}}\boldsymbol{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \hat{\boldsymbol{A}}_{pbe} \end{bmatrix}}_{\boldsymbol{A}(\hat{\boldsymbol{x}})} \underbrace{\begin{bmatrix} \boldsymbol{\delta p}^n \\ \boldsymbol{\delta v}^b \\ \boldsymbol{\delta \alpha}^n \\ \boldsymbol{\delta b}^b_{acc} \\ \boldsymbol{\delta b}^b_{gyro} \\ \boldsymbol{\delta b}_{pbe} \end{bmatrix}}_{\boldsymbol{\delta x}}$$

$$+ \underbrace{\begin{bmatrix} \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_{3\times2} \\ -\boldsymbol{I}_3 & -\mathbf{S}(\hat{\boldsymbol{v}}^b) & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_{3\times2} \\ \mathbf{0}_3 & -\mathbf{R}(\hat{\boldsymbol{q}}) & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_{3\times2} \\ \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{I}_3 & \mathbf{0}_3 & \mathbf{0}_{3\times2} \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{I}_3 & \mathbf{0}_{3\times2} \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \hat{\boldsymbol{G}}_{pbe} \end{bmatrix}}_{\boldsymbol{G}(\hat{\boldsymbol{x}})} \underbrace{\begin{bmatrix} \boldsymbol{w}^b_{acc} \\ \boldsymbol{w}^b_{gyro} \\ \boldsymbol{w}^b_{b,acc} \\ \boldsymbol{w}^b_{b,gyro} \\ \boldsymbol{w}_{pbe} \end{bmatrix}}_{\boldsymbol{w}} \tag{10.17}$$

Where the rotation vector $\boldsymbol{\delta\alpha}^n$ is used to represent the attitude error and the noise vector $\boldsymbol{w}$ is distributed as:

$$\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\mathcal{W}}) \tag{10.18}$$

And the diagonal noise covariance matrix $\boldsymbol{\Sigma}$ is defined as:

$$\boldsymbol{\mathcal{W}} = \begin{bmatrix} \boldsymbol{I}_3\sigma^2_{acc} & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_{3\times2} \\ \mathbf{0}_3 & \boldsymbol{I}_3\sigma^2_{gyro} & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_{3\times2} \\ \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{I}_3\sigma^2_{b,acc} & \mathbf{0}_3 & \mathbf{0}_{3\times2} \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{I}_3\sigma^2_{b,gyro} & \mathbf{0}_{3\times2} \\ \mathbf{0}_{2\times3} & \mathbf{0}_{2\times3} & \mathbf{0}_{2\times3} & \mathbf{0}_{2\times3} & \boldsymbol{\Sigma}_{pb} \end{bmatrix} \tag{10.19}$$

10.4.2 Discretization

The deterministic part of the error state dynamics must be discretized in order to be useful in the discrete Kalman filter.

$$\boldsymbol{F} = \exp\left(\boldsymbol{A}(\hat{\boldsymbol{x}})\Delta t\right) \tag{10.20}$$

$$\approx \begin{bmatrix} \boldsymbol{I}_3 & \mathbf{R}(\hat{\boldsymbol{q}})\Delta t & -\mathbf{S}(\mathbf{R}(\hat{\boldsymbol{q}})\hat{\boldsymbol{v}}^b)\Delta t & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{R}^\top\{\hat{\boldsymbol{\omega}}^b\Delta t\} & \mathbf{R}^\top(\hat{\boldsymbol{q}})\,\mathbf{S}(\boldsymbol{g}^n)\Delta t & -\boldsymbol{I}_3\Delta t & -\mathbf{S}(\hat{\boldsymbol{v}}^b)\Delta t & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{I}_3 & \mathbf{0}_3 & -\mathbf{R}(\hat{\boldsymbol{q}})\Delta t & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{I}_3(1-\frac{\Delta t}{T_{acc}}) & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{I}_3(1-\frac{\Delta t}{T_{gyro}}) & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{F}_{pbe} \end{bmatrix} \tag{10.21}$$

The stochastic part must also be discretized in order to obtain the discrete noise matrix $\boldsymbol{\mathcal{Q}}$. Since the Eigen matrix library (Guennebaud, Jacob et al., 2010) in C++has a fast series expansion evaluation of the matrix exponential, Van Loans method (see Section 8.4) is the most precise way to achieve discretization:

$$\exp\left(\begin{bmatrix} -\boldsymbol{A} & \boldsymbol{G}\boldsymbol{\mathcal{W}}\boldsymbol{G}^\top \\ \mathbf{0} & \boldsymbol{A}^\top \end{bmatrix}\Delta t\right) = \begin{bmatrix} \times & \boldsymbol{V}_2 \\ \mathbf{0} & \boldsymbol{V}_1 \end{bmatrix} \tag{10.22}$$

, where $\boldsymbol{\mathcal{Q}}$ is found as:

$$\boldsymbol{\mathcal{Q}} = \boldsymbol{V}_1^\top \boldsymbol{V}_2 \tag{10.23}$$

10.5 Measurement model

To be able to utilize the sensors in the ESKF, they need to be modeled in error-variable form. The goal of this section is to arrive at measurement models for each of the sensors, that is expressions for the innovations $\boldsymbol{\delta y}$, the measurement matrices $\boldsymbol{H}$, and the measurement covariance matrices $\boldsymbol{\mathcal{R}}$. The USBL is modeled in Section 10.5.1 followed by the pressure sensor (10.5.2), DVL (10.5.3), and at last the AHRS in Section 10.5.4.

10.5.1 USBL

The USBL measurement is predicted to be:

$$\hat{\boldsymbol{y}}^n_{usbl} = \hat{\boldsymbol{p}}^n_{b/n} + \mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{r}^b_{tp/b} \tag{10.24}$$

The USBL measurement innovation is the difference between the measurement and the predicted measurement. The measurement jacobian is a result of the difference between the measurement model and the predicted measurement:

$$\delta\boldsymbol{y}^n_{usbl} = \boldsymbol{y}^n_{usbl} - \hat{\boldsymbol{y}}^n_{usbl} \tag{10.25}$$

(Derivation in Appendix A.2.1)

$$\approx \underbrace{\begin{bmatrix} \boldsymbol{I}_3 & \mathbf{0}_3 & -\mathbf{S}(\mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{r}^b_{tp/b}) & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix}}_{\boldsymbol{H}_{usbl}(\hat{\boldsymbol{x}})} \boldsymbol{\delta x} + \boldsymbol{\eta}^n_{usbl} \tag{10.26}$$

, where the measurement noise $\boldsymbol{\eta}^n_{usbl}$ is distributed as:

$$\boldsymbol{\eta}^n_{usbl} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\mathcal{R}}_{usbl}) \tag{10.27}$$

10.5.2 Pressure sensor

The pressure measurement is predicted to be:

$$y_{ps} = b_{atm} + \hat{b}_{lf} + \hat{b}_{wf} + \rho \boldsymbol{g}^{n\top}(\hat{\boldsymbol{p}}^n_{b/n} + \mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{r}^b_{ps/b}) \tag{10.28}$$

$$\delta y_{ps} = y_{ps} - \hat{y}_{ps} \tag{10.29}$$

(Derivation in Appendix A.2.2)

$$\approx \underbrace{\begin{bmatrix}\rho \boldsymbol{g}^{n\top} & \mathbf{0_{1x3}} & -\rho \boldsymbol{g}^{n\top}\mathrm{S}(\mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{r}^b_{ps}) & \mathbf{0_{1x3}} & \mathbf{0_{1x3}} & \begin{bmatrix}1 & 1 & 0\end{bmatrix}\end{bmatrix}}_{\boldsymbol{H}_{ps}(\hat{\boldsymbol{x}})} \boldsymbol{\delta x} + \eta^n_{ps} \tag{10.30}$$

, where the measurement noise η^n_{ps} is distributed as:

$$\eta^n_{ps} \sim \mathcal{N}(0, \mathcal{R}_{ps}) \tag{10.31}$$

10.5.3 DVL

The DVL measurement is predicted to be:

$$\hat{\boldsymbol{y}}^b_{dvl} = \hat{\boldsymbol{v}}^b_{b/n} - \mathrm{S}(\boldsymbol{r}^b_{d/b})\hat{\boldsymbol{\omega}}^b_{b/n} \tag{10.32}$$

$$\boldsymbol{\delta y}^b_{dvl} = \boldsymbol{D}\boldsymbol{y}^d_{dvl} - \hat{\boldsymbol{y}}^b_{dvl} \tag{10.33}$$

(Derivation in Appendix A.2.3)

$$\approx \underbrace{\begin{bmatrix}\mathbf{0}_3 & \boldsymbol{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathrm{S}(\boldsymbol{r}^b_{dvl}) & \mathbf{0}_3\end{bmatrix}}_{\boldsymbol{H}_{dvl}} \boldsymbol{\delta x} + \boldsymbol{D}\boldsymbol{\eta}^d_{dvl} + \mathrm{S}(\boldsymbol{r}^b_{dvl})\boldsymbol{\eta}^b_{gyro} \tag{10.34}$$

, where the noise is distributed as:

$$\boldsymbol{D}\boldsymbol{\eta}^d_{dvl} + \mathrm{S}(\boldsymbol{r}^b_{dvl})\boldsymbol{\eta}^b_{gyro} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\mathcal{R}}_{dvl}) \tag{10.35}$$

, and:

$$\boldsymbol{\mathcal{R}}_{dvl} = \boldsymbol{D}\tilde{\boldsymbol{\mathcal{R}}}_{dvl}\boldsymbol{D}^\top + \mathrm{S}(\boldsymbol{r}^b_{dvl})\boldsymbol{\mathcal{R}}_{gyro}\,\mathrm{S}(\boldsymbol{r}^b_{dvl})^\top \tag{10.36}$$

, where $\tilde{\boldsymbol{\mathcal{R}}}_{dvl}$ is the covariance matrix in the DVL frame and $\boldsymbol{\mathcal{R}}_{gyro}$ is the gyro noise covariance matrix.

10.5.4 AHRS

The AHRS requires a slightly different approach. Rotation matrices are used in order to avoid issues with Euler angle discontinuities. Rotation matrices are multiplicative by nature as opposed to additive as is the case for the other measurements. The AHRS measurement is predicted to be:

$$\hat{\boldsymbol{Y}}_{ahrs} = \mathbf{R}(\hat{\boldsymbol{q}}) \tag{10.37}$$

$$\begin{aligned}\boldsymbol{\delta y}^n_{ahrs} &= \text{vex}(\boldsymbol{Y}_{ahrs}\hat{\boldsymbol{Y}}^\top_{ahrs}) && (10.38)\\ &\quad\text{(Derivation in Appendix A.2.4)}\\ &\approx \underbrace{\begin{bmatrix}\mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3\end{bmatrix}}_{\boldsymbol{H}_{ahrs}(\hat{\boldsymbol{x}})}\boldsymbol{\delta x} + \boldsymbol{\eta}^n_{ahrs} && (10.39)\end{aligned}$$

$$\boldsymbol{\eta}^n_{ahrs} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{T}(y_\theta, y_\psi)\boldsymbol{\mathcal{R}}_{\phi\theta\psi}\boldsymbol{T}(y_\theta, y_\psi)^\top) \tag{10.40}$$

, where $\boldsymbol{\mathcal{R}}_{\phi\theta\psi}$ is a tuning matrix containing Euler angle variances on the diagonal. $\boldsymbol{T}$ is the Euler angle to rotation vector jacobian (see Section 5.2.7).

10.5.5 Outlier rejection

Some measurements, from now referred to as outliers, are not consistent with the DVL measurement model in Section 10.5.3. This can for instance be due to fish swimming in front of the DVL resulting in a Doppler frequency shift that includes the fish's velocity. Nevertheless, we want to avoid measurements that are outright wrong from entering the observer. An outlier rejection method is needed for this. Since the observer is Kalman filter based, it is natural to base the outlier rejection on some form of consistency check, the simplest being the Normalized Innovation Squared (NIS). If the NIS is too high, it means that the innovation is too large compared to the covariance in the Kalman filter and this suggests that the measurement used to calculate the innovation is an outlier. When NIS is used in Kalman filter tuning, it is also common to consider a lower bound (see Section 8.5). This is not applicable to outlier rejection since discarding a measurement for being "too good" is absurd. The NIS is calculated as:

$$\epsilon_{dvl} = \boldsymbol{\delta y}^\top_{dvl}\boldsymbol{\mathcal{S}}^{-1}_{dvl}\boldsymbol{\delta y}_{dvl} \tag{10.41}$$

The upper bound is:

$$\epsilon_{upper} = \text{chi2inv}(1 - \frac{\alpha}{2}, 3) \tag{10.42}$$

This is implemented as: "Reject $\boldsymbol{\delta y}_{dvl}$ if $\epsilon_{dvl} > \epsilon_{upper}$"

10.6 Wave filtering

In general, we want to implement wave filtering on the state estimates that are to be used as feedback for control systems. Most important are the position and attitude estimates. To allow for PD control we also want to filter velocity and angular velocity. Position, velocity, and angular velocity are orthogonal vectors and can be filtered along three axes. Attitude is more difficult. Ideally, the attitude should be filtered in such a way that singularities and discontinuities will not be a problem. However, there is no trivial way to do notch filtering on attitude. It is possible to realize some kind of quaternion low-pass filter (MathWorks, 2023) based on quaternion SLERP (Solà, 2017). Higher-order filter realizations are non-trivial. To keep things simple we will transform the attitude into Euler angles, apply notch filtering directly to each component and keep in mind that unwanted effects can arise close to the Euler angle singular points.

A notch filter on the form

$$h_n(s) = \frac{s^2 + 2\zeta\omega_n + \omega_n^2}{(s+\omega_n)^2} \tag{10.43}$$

, will be used to remove wave frequency components Fossen (2021). ω_n is the notch frequency, while ζ determines the notch band-width. In order to realize (10.43) as strictly proper, it can be cascaded with a first-order low-pass filter

$$h_{lp}(s) = \frac{\omega_{lp}}{s+\omega_{lp}} \tag{10.44}$$

, where the cut-off frequency ω_{lp} is set high and way outside the bandwidth of the vehicle to prevent unnecessary phase lag. We want to make the wave filter adaptive so that the notch frequency (ω_n) corresponds with the peak frequency of the wave spectrum (ω_w). We will use an estimate of the peak frequency and set $\omega_n = \hat{\omega}_w$ The cascade can then be realized as a third-order SISO state-space system on observable canonical form with matrices:

$$\boldsymbol{A}_{wf} = \begin{bmatrix} 0 & 0 & -\hat{\omega}_w^2\omega_{lp} \\ 1 & 0 & -(2\hat{\omega}_w\omega_{lp} + \hat{\omega}_w^2) \\ 0 & 1 & -(2\hat{\omega}_w + \omega_{lp}) \end{bmatrix} \tag{10.45a}$$

$$\boldsymbol{B}_{wf} = \begin{bmatrix} \hat{\omega}_w^2\omega_{lp} \\ 2\zeta\hat{\omega}_w\omega_{lp} \\ \omega_{lp} \end{bmatrix} \tag{10.45b}$$

$$\boldsymbol{C}_{wf} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tag{10.45c}$$

A total of 12 of these filters are implemented to filter position (3 states), velocity(3), Euler angles (3), and angular velocity (3).

10.7 Peak frequency estimator

The frequency estimator for sinusoidal signals by Aranovskiy et al. (2007) is to be used for estimating the peak frequency. The exact design is dependent on which signal is used as the input. The signal must be one that is excited by waves and therefore contains the sinusoidal component we want to estimate the frequency of. For 3-DOF vessels, it is common to choose roll and/or pitch which will oscillate around zero. Movement in heave is also possible as it is often near-aligned with NED-z. For 6 DOF-vessels none of this is true; roll and pitch are not necessarily close to zero and heave can therefore correspond to any direction in NED.

It is tempting to use state estimates as input to the frequency estimator as they are precise, low noise, and ideally unbiased. In the case of this system doing so will introduce a loop between the frequency estimator and the pressure bias estimator since the bias estimator impacts the state estimates (see Figure 10.1). Whether this would work or not will not be examined here. In general, introducing these kinds of feedback loops is bad practice as modularity in design and code tends to weaken. For this reason, we are left with using raw measurements as input. USBL and DVL are low rate, low precision, and unreliable sensors and are out of the picture. This leaves four candidates, all of which have high rates and are reliable:

- Accelerometer
- Gyro
- Pressure sensor
- AHRS

The accelerometer and gyro turned out to have too low a signal-to-noise ratio in test runs and are ruled out. The pressure measurement has an excellent signal-to-noise ratio, but the vehicle is expected to move a lot in the vertical direction which would disturb the estimation. Therefore, it was decided to use the AHRS pitch estimate. The pitch is expected to stay constant (excluding wave motion) over time. Since the pitch does not necessarily oscillate around zero, it is necessary to do modifications to the basic estimator design.

In Belleter et al. (2015), the Aranovskiy et al. (2007) estimator is modified with an adjustable natural frequency. We will use this as a starting point for our observer design. The observer dynamics are:

$$\dot{\xi}_1 = \xi_2 \tag{10.46a}$$

$$\dot{\xi}_2 = -\omega_f^2\xi_1 - 2\omega_f\xi_2 + \omega_f^2 u' \tag{10.46b}$$

$$\dot{\hat{\Omega}} = k\xi_1(\dot{\xi}_2 - \hat{\Omega}\xi_1) \tag{10.46c}$$

, where u' is the input. For the purpose of modifying, let the output $y = \xi_1$ such that the adaption law becomes:

$$\dot{\hat{\Omega}} = ky(\ddot{y} - \hat{\Omega}y) \tag{10.47}$$

, the transfer function from u' to y is

$$\frac{y}{u'}(s) = \frac{\omega_f^2}{(s+\omega_f)^2} \tag{10.48}$$

Now let the driving signal u' be the output of a highpass filter with input u such that

$$\frac{u'}{u}(s) = \frac{s}{s + \omega_{hp}} \tag{10.49}$$

, where ω_{hp} is the high pass cut-off-frequency. When cascaded with (10.48), the resulting transfer function becomes:

$$\frac{y}{u}(s) = \frac{\omega_f^2 s}{(s + \omega_f)^2 (s + \omega_{hp})} \tag{10.50}$$

, which is stable for ω_f, $\omega_{hp} > 0$. The transfer function (10.50) is strictly proper and can be realized as:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\omega_f^2 \omega_{hp} & -2\omega_f \omega_{hp} - \omega_f^2 & -\omega_{hp} - 2\omega_f \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega_f^2 \end{bmatrix} u \tag{10.51a}$$

$$\dot{\hat{\Omega}} = k_f x_2 (\dot{x}_3 - \hat{\Omega} x_2) \tag{10.51b}$$

$$\hat{\omega}_w = \sqrt{-\hat{\Omega}} \tag{10.51c}$$

This means that the adaption law is kept the same as the original.

Chapter 11

Implementation

This chapter is intended to provide insight into the process of implementing the observer design in object-oriented C++code. Due to the nature of programming, this section is a lot less formal than the rest of the thesis and is intended to be so. Section 11.1 is a summary of details in the implementation and engineering choices made to reduce computational burden and improve real-time performance. Section 11.2 is an overview of how the system was split into modules for implementation.

11.1 Implementation details

11.1.1 Kalman filter update step

The Kalman update step calculates the state error estimate $\boldsymbol{\delta x}$ and updates the state error covariance $\boldsymbol{\mathcal{P}}$. As a part of that, it is necessary to calculate the Kalman gain $\boldsymbol{K}$. This involves a matrix inversion and is on the form:

$$\boldsymbol{K} = \hat{\boldsymbol{\mathcal{P}}}\boldsymbol{H}^\top\boldsymbol{\mathcal{S}}^{-1} \tag{11.1}$$

The innovation covariance is quadratic with the dimension of the innovation $\boldsymbol{\delta y}$, and this dimension can in this case be up to 10 if all measurements arrive at the same time step. Inverting a dense 10×10 matrix is a significant task in a real-time system and it is in general bad practice to do pure matrix inversions if not strictly necessary. It is therefore common to do the following trick:

Let:

$$\boldsymbol{Z}^\top = \boldsymbol{H}^\top\boldsymbol{\mathcal{S}}^{-1}$$

Note that $\boldsymbol{\mathcal{S}}$ is a covariance matrix and thus symmetric such that:

$$\boldsymbol{Z} = \boldsymbol{\mathcal{S}}^{-1}\boldsymbol{H}$$

It is now clear that $\boldsymbol{Z}$ is the solution to the linear system:

$$\boldsymbol{\mathcal{S}}\boldsymbol{Z} = \boldsymbol{H}$$

This system can be solved through a solver algorithm for linear systems so that the Kalman gain is calculated as:

$$\boldsymbol{Z} = \text{linsolve}(\boldsymbol{\mathcal{S}}, \boldsymbol{H}) \tag{11.2a}$$
$$\boldsymbol{K} = \hat{\boldsymbol{\mathcal{P}}}\boldsymbol{Z}^\top \tag{11.2b}$$

Linsolve algorithms are often based on some kind of decomposition of the matrix $\boldsymbol{\mathcal{S}}$. Since covariance matrices are positive definite, it is possible to use Cholesky decomposition which is very fast. The eigen library has two variants of the Cholesky decomposition which are both excellent choices. It was decided to use “Robust Cholesky decomposition with pivoting” (Guennebaud, Jacob et al., 2010, function name: LDLT) for some extra accuracy as well as robustness with respect to the properties of the matrix used as input.

11.1.2 Calculating the Normalized Innovation Squared

The NIS is another example where inverting of covariance matrices is something we want to avoid, especially for outlier rejection which happens in real-time. The NIS is calculated as:

$$\epsilon = \boldsymbol{\delta y}^\top \boldsymbol{\mathcal{S}}^{-1} \boldsymbol{\delta y}$$

This could simply be calculated as:

$$\begin{aligned} \boldsymbol{z} &= \text{linsolve}(\boldsymbol{\mathcal{S}}, \boldsymbol{\delta y}) \\ \epsilon &= \boldsymbol{\delta y}^\top \boldsymbol{z} \end{aligned}$$

, using Cholesky decomposition on $\mathbf{S}$. It is however possible to exploit the properties of the Cholesky decomposition (Guennebaud, Jacob et al., 2010, function name: LLT) even more. Cholesky decomposition factors $\boldsymbol{\mathcal{S}}$ into a lower triangular matrix $\boldsymbol{L}$, such that:

$$\boldsymbol{\mathcal{S}} = \boldsymbol{L}\boldsymbol{L}^\top$$

, leading to:

$$\begin{aligned} \epsilon &= \boldsymbol{\delta y}^\top (\boldsymbol{L}\boldsymbol{L}^\top)^{-1} \boldsymbol{\delta y} \\ &= \boldsymbol{\delta y}^\top \boldsymbol{L}^{-1} (\boldsymbol{L}^\top)^{-1} \boldsymbol{\delta y} \\ &= ((\boldsymbol{L}^\top)^{-1} \boldsymbol{\delta y})^\top ((\boldsymbol{L}^\top)^{-1} \boldsymbol{\delta y}) \\ &= ||(\boldsymbol{L}^\top)^{-1} \boldsymbol{\delta y}||^2 \end{aligned}$$

Since $\boldsymbol{L}$ is lower triangular, it is trivial to solve, and we find the final pseudocode:

$$\boldsymbol{L} = \text{Cholesky}(\boldsymbol{\mathcal{S}}) \tag{11.3a}$$
$$\boldsymbol{z} = \text{solve}(\boldsymbol{L}^\top, \boldsymbol{\delta y}) \tag{11.3b}$$
$$\epsilon = ||\boldsymbol{z}||^2 \tag{11.3c}$$

11.1.3 Converting rotation vector to quaternion

Rotation vectors are converted to quaternions at two stages in the observer algorithm; first, when integrating the angular velocity estimate into the quaternion estimate in the INS:

$$\hat{\boldsymbol{q}} \leftarrow \hat{\boldsymbol{q}} \otimes \mathbf{q}\{\hat{\boldsymbol{\omega}}^b \Delta t\}$$

, and second, when injecting the attitude error into the quaternion estimate:

$$\hat{\boldsymbol{q}} \leftarrow \mathbf{q}\{\boldsymbol{\delta\alpha}^n\} \otimes \hat{\boldsymbol{q}}$$

When using a reasonably small time step, the rotation vector argument ($\boldsymbol{\alpha}$) is expected to be close to zero in both cases.

Looking at the conversion formula for the rotation vector it is clear that there is a risk for division by zero:

$$\mathbf{q}\{\boldsymbol{\alpha}\} = \begin{bmatrix} \cos\frac{\|\boldsymbol{\alpha}\|}{2} \\ \frac{\boldsymbol{\alpha}}{\|\boldsymbol{\alpha}\|}\sin\frac{\|\boldsymbol{\alpha}\|}{2} \end{bmatrix}$$

Common programming practice suggests:

$$\mathbf{q}\{\boldsymbol{\alpha}\} = \begin{bmatrix} \cos\frac{\|\boldsymbol{\alpha}\|}{2} \\ \frac{\boldsymbol{\alpha}}{\|\boldsymbol{\alpha}\|}\sin\frac{\|\boldsymbol{\alpha}\|}{2} \end{bmatrix} \quad \text{for } \|\alpha\| > \delta$$
$$\mathbf{q}\{\boldsymbol{\alpha}\} = \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} \quad \text{else}$$

, where the parameter δ ensures that we do not get too close to a division by zero. However, this would lead to a loss of accuracy in an interval around zero and/or division by a very small number (bad practice). This is an interval where we expect most of the arguments to be. For this reason, we will instead look at an option where the loss of accuracy occurs far from zero. The vector part of the quaternion can be rewritten as:

$$\boldsymbol{\epsilon}\{\boldsymbol{\alpha}\} = \frac{1}{2}\boldsymbol{\alpha}\frac{\sin\frac{\|\boldsymbol{\alpha}\|}{2}}{\frac{\|\boldsymbol{\alpha}\|}{2}}$$

Let $z = \frac{\|\boldsymbol{\alpha}\|}{2}$, since arguments are expected to be around zero, a Taylor series expansion around zero is a reasonable approximation:

$$\cos z = 1 - \frac{z^2}{2!} + \frac{z^4}{4!} - \frac{z^6}{6!} + ...$$
$$\frac{\sin z}{z} = 1 - \frac{z^2}{3!} + \frac{z^4}{5!} - \frac{z^6}{7!} + ...$$

(Gradshteĭn et al., 2007), this leads to a conversion formula on the form:

$$\mathbf{q}\{\boldsymbol{\alpha}\} = \begin{bmatrix} 1 - \frac{\|\boldsymbol{\alpha}\|^2}{2!2^2} + \frac{\|\boldsymbol{\alpha}\|^4}{4!2^4} - \frac{\|\boldsymbol{\alpha}\|^6}{6!2^6} + ... \\ \frac{1}{2}\boldsymbol{\alpha}(1 - \frac{\|\boldsymbol{\alpha}\|^2}{3!2^2} + \frac{\|\boldsymbol{\alpha}\|^4}{5!2^4} - \frac{\|\boldsymbol{\alpha}\|^6}{7!2^6} + ...) \end{bmatrix} \tag{11.4}$$

In practical implementation, the Taylor series must be truncated. Four terms are considered sufficient for this application. Also, note that the two Taylor series contain a lot of common factors. This can be exploited to reduce the number of operations needed.

11.1.4 Rotation vector to rotation matrix

The rotation vector to rotation matrix operator can also be redefined to avoid division by zero. Using the method in Section 11.1.3 and converting via the quaternion, we redefine:

$$\mathbf{R}\{\boldsymbol{\alpha}\} \triangleq \mathbf{R}(\mathbf{q}\{\boldsymbol{\alpha}\}) \tag{11.5}$$

11.1.5 Quaternion normalization

The quaternion is only a valid representation of the vehicle attitude when it is at unit length. This constraint is the price we pay for using four parameters to describe 3-DOF. It is therefore essential that the quaternion estimate is kept at unit length throughout the real-time implementation of the system. Quaternion products suffer from numerical drift, and the implementation of the rotation vector-to-quaternion operator (see the paragraph above) results in a quaternion that is not perfectly unit length. Thankfully, quaternion normalization is an incredibly computationally cheap operation:

$$\boldsymbol{q} \leftarrow \frac{\boldsymbol{q}}{\|\boldsymbol{q}\|} \tag{11.6}$$

For robustness, the quaternion estimate will be normalized every time it is altered.

11.2 Classes

This section gives a brief insight into how the system is modularized for implementation. An overview of the modules and how they interact can be seen in the block diagram in Figure 10.1. To avoid a second copy of this block diagram, the modules are here represented as individual tables and not in a full Unified Modeling Language (UML) class diagram. References to chapters are also added such that the tables serve as pretty good documentation along with the block diagram. Each table contains:

1. Internal state
2. The module interface (public functions)
3. Important helper functions that are specific to this thesis (private functions).

Internal state and the interface are what the author considers to be vital to have planned out before starting programming. In addition, some helper functions have been added to indicate where important features of the design are implemented.

11.2.1 Inertial navigation system

The inertial navigation system is the core of the observer and the most obvious module from which to make a class. Note that the acceleration and angular velocity estimates are not internal states since they can be calculated directly from measurements and biases. See Table 11.1.

Table 11.1: Module: Inertial navigation system

Module Name	InertialNavigationSystem	10.1
Internal State	$\hat{p}$	
	$\hat{v}$	
	$\hat{q}$	
	$\hat{b}_{acc}$	
	$\hat{b}_{gyro}$	
Public Functions	Integrate(y_{acc}, y_{gyro})	10.1.1
	Inject(δx)	10.1.2
Private functions	$q =$ RotationVectorToQuaternion(α)	11.1.3

11.2.2 Pressure bias estimator

The pressure bias estimator is tempting to implement into the INS, but since the two modules do not require access to each other's internal state, good programming practice suggests that they should be separate modules. This allows the INS to be run independently in case the pressure bias estimator turns out to be unnecessary. See Table 11.2.

Table 11.2: Module: Pressure bias estimator

Module Name	PressureBiasEstimator	10.2
Internal State	$\hat{b}_{lf}$	
	$\hat{b}_{wf}$	
	$\hat{b}_{int}$	
Public Functions	Step($\hat{\omega}_w$)	10.2.1
	Inject(δx)	10.2.2

11.2.3 Error state Kalman filter

The ESKF is kept general in implementation, meaning we do not want to incorporate model-specific functionality. The exception from this rule is the reset function which requires knowledge about the system states (which ones correspond to attitude). The idea is to keep an already large module as simple as possible and to allow for the reuse of a rather useful and versatile module. The model-specific functionality is put into modules that feed the ESKF. See Table 11.3.

Table 11.3: Module: Error state Kalman filter

Module Name	ErrorStateKalmanFilter	10.3
Internal State	$\hat{P}$	
Public Functions	Predict(F, Q)	10.3.1
	$\delta x =$ Update$(\delta y, H, R)$	10.3.2
	Reset(δx)	10.3.3
Private Functions	$K =$ KalmanGain(H, S)	11.1.1
	$R =$ RotationVectorToRotationMatrix(α)	11.1.4

11.2.4 Process model

This is a module that feeds the ESKF with a discrete model of the INS and the pressure bias estimator. The module has no state but is easily defined nevertheless. See Table 11.4.

Table 11.4: Module: Process model

Module Name	ProcessModel	10.4
Internal State	none	
Public Functions	$F =$ TransitionMatrix$(\hat{x})$	10.4.2
	$Q =$ ProcessCovariance(A, G, Σ)	
Private Functions	$A =$ SystemMatrix$(\hat{x})$	10.4.1
	$G =$ NoiseMatrix$(\hat{x})$	
	$Q =$ VanLoan(A, G, Σ)	10.4.2

11.2.5 Measurement model

This is a module that feeds the ESKF with a model of the aiding measurements and the innovation. The module is also responsible for rejecting outliers (meaning it only feeds the ESKF with "good" measurements). The module has no state. See Table 11.5.

Table 11.5: Module: Measurement model

Module Name	MeasurementModel	10.5
Internal State	none	
Public Functions	$\delta y =$ Innovation$(y, \hat{x})$	10.5.1 to 10.5.4
	$H =$ MeasurementMatrix$(\hat{x}, \hat{b})$	
	$R =$ MeasurementCovariance()	
Private Functions	$\epsilon =$ CalculateNis$(\delta y, S)$	11.1.2
	OutlierRejection(ϵ)	10.5.5

11.2.6 Wave filter

The wave filter is a straightforward implementation of the design in Section 10.6.

11.2.7 Wave frequency estimator

The wave frequency estimator is a straightforward implementation of the design in Section 10.7.

Chapter 12

Data collection, Configuration , and Calibration

This chapter describes how data was collected for experimental evaluation of the observer design and implementation, how the observer was tuned on that data, and how the Kalman filter is initialized. The experimental setup is presented in Section 12.1. The tuning used to achieve the later results is presented in Section 12.2. The final section, 12.3, concerns Kalman filter initialization.

12.1 Experimental setup and data gathering

The observer design in Chapter 10 is to be tested experimentally at a full-scale fish farm. Since the time available at the fish farm is limited, the decision was not to attempt tuning and debugging at the fish farm and instead focus on recording data that could later be used to test the design. This should have no impact on the performance of the observer itself as the data can be used as input to the observer exactly how it is recorded.

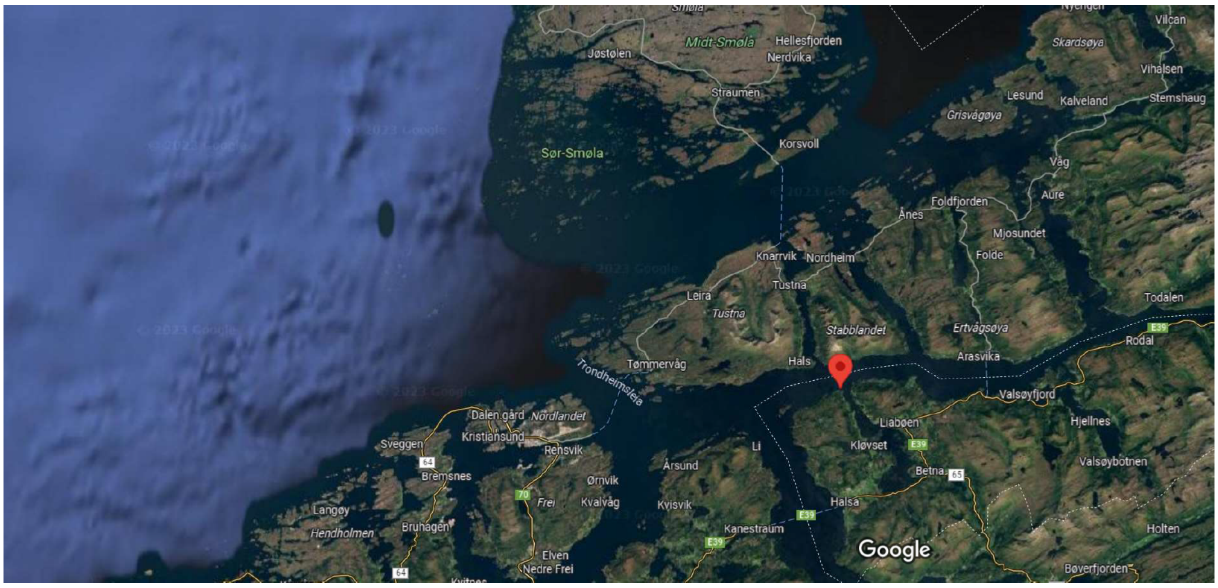

Figure 12.1: The location of the fish farm at Korsneset (from Google Maps[5]).

The fish farm is located at Korsneset (Figure 12.1), east of Kristiansund in Norway, and is available through SINTEF. It is located inside a fjord and is a shielded location where the seas are usually quite calm.

The trials were conducted on 13.04.2023. The weather was windy (quite strong continuous wind) and because of that, there were significant waves. The fish inside the cage were in the process of being fed or had just been fed, meaning they were quite active (swimming, leaping). In general, the conditions were well-suited to generate challenging input for the observer and really put it to the test.

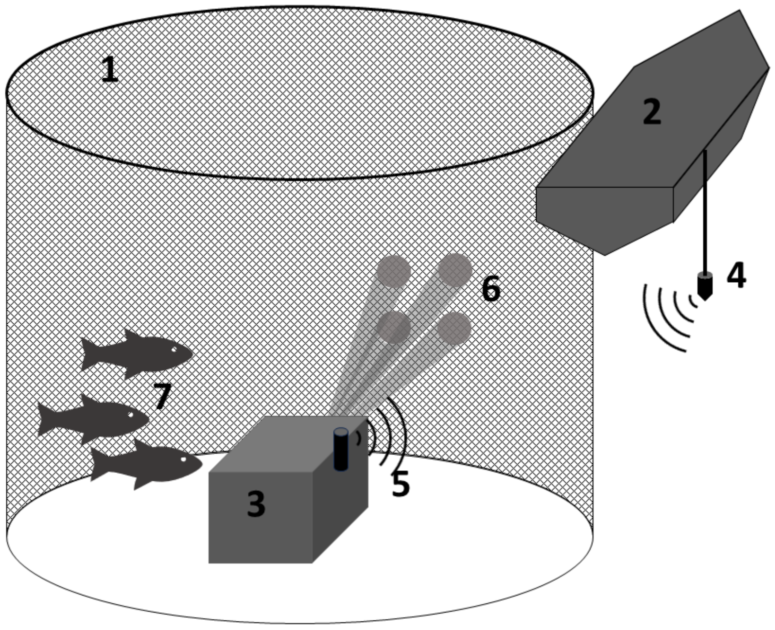

Figure 12.2: Sketch of the experimental setup. 1: The net pen. 2: The mothership, moored to the pen. 3: The BlueROV2. 4: USBL transceiver. 5: USBL transponder. 6: DVL beams reflecting on the net pen walls. 7: Fish inside the net pen. (Not to scale)

Figure 12.2 is a sketch of the setup for gathering data. The cylinder (1) is the net cage from the waterline and down. The front half is drawn as see-through, only the back wall is visible. The shape of the bottom is not relevant and not shown here. The mothership (2) is moored to the outside of the cage and the USBL transceiver (4) is more or less rigidly mounted to the mothership using a steel tube. The BlueROV2 (3) has the USBL transponder (5) mounted to the starboard side. This means that the USBL system communicates through the net pen wall. The DVL is mounted to the front of the BlueROV2 such that its beams are reflected off the net pen walls (6). Moving fish (7) are present in the cage and may block the USBL, the DVL, or even physically collide with the BlueROV2.

[5]Google Maps website: `https://www.google.com/maps`

12.2 Parameters and tuning

12.2.1 Gravity

The center of the fish farm at Korsneset is located at: 63°08'34.1" N 8°13'32.8" E. The altitude is assumed to be 0 (mean sea level). This is used as input to the gravity calculator (The International Gravimetric Bureau, n.d.) and the result is a gravity of 9.821786 m/s^2. Assuming that this aligns with the z-axis of the NED frame, we get the following vector:

$$\boldsymbol{g}^n = \begin{bmatrix} 0 & 0 & 9.821786 \end{bmatrix}^\top \text{ m/s}^2 \tag{12.1}$$

12.2.2 Athmospheric pressure

The average atmospheric pressure is assumed to be the standard atmosphere:

$$b_{atm} = 1\,\text{atm} = 101325\,\text{Pa} \tag{12.2}$$

12.2.3 Water density

The density of the seawater inside the fish farm is assumed to be:

$$\rho = 1025\,\text{kg/m}^3 \tag{12.3}$$

12.2.4 Mounting

The sensors on the BlueROV2 are mounted in various locations around the vehicle. These are measured in the body frame and become:

$$\boldsymbol{r}^b_{tp} = \begin{bmatrix} -0.09 & 0.22 & -0.11 \end{bmatrix}^\top \text{ m} \tag{12.4a}$$
$$\boldsymbol{r}^b_{ps} = \begin{bmatrix} -0.26 & 0 & 0 \end{bmatrix}^\top \text{ m} \tag{12.4b}$$
$$\boldsymbol{r}^b_{dvl} = \begin{bmatrix} 0.07 & 0 & 0.13 \end{bmatrix}^\top \text{ m} \tag{12.4c}$$

The matrix that represents the DVLs orientation in the body frame is:

$$\boldsymbol{D} = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \tag{12.5}$$

12.2.5 Noise

Tuning is a difficult task and it is often challenging to know where to start, for this reason, it is useful to have a rough idea of the order of magnitude of the parameters we want to choose. In Kalman filter applications, the tuning parameters represent the standard deviation (or variance if squared) of noise from sensors. For the accelerometer, gyro, and pressure sensor, the datasheet noise characteristics are used as a reference and multiplied with a scaling factor when tuning. The noise densities for those sensors are:

Table 12.1: Sensor parameters

Sensor	**Model**	**Data sheet parameter**	**Symbol**	**Value**	**Unit**
Accelerometer	ST Microelectronics LSM303D	Linear acceleration noise density	σ^*_{acc}	150	$\mathrm{mg}/\sqrt{\mathrm{Hz}}$
Gyro	ST Microelectronics L3GD20H	Rate noise density	σ^*_{gyro}	0.011	$^\circ/\mathrm{s}/\sqrt{\mathrm{Hz}}$
Pressure Sensor	TE Connectivity MS5837-30BA	Resolution RMS	σ^*_{ps}	0.20	mbar

The parameters in Table 12.1 are compensated for the update rate ($\sqrt{\mathrm{Hz}}$) and converted to SI units before use. The USBL returns a one-dimensional standard deviation ρ^*_{usbl} which will be scaled and used in the USBL measurement covariance. Similarly, the DVL returns a full covariance matrix $\boldsymbol{\mathcal{R}}^*_{dvl}$. This will be scaled and used in the DVL measurement covariance.

12.2.6 Tuning

The observer was tuned on experimental data. A lot of tuning parameters were tried. One good set of tuning parameters that made the Kalman filter somewhat consistent and led to a good performance is given in this section and is the one that gave the results in the next chapter.

INS

$$\sigma_{acc} = 50\sigma^*_{acc} \tag{12.6a}$$
$$\sigma_{gyro} = 50\sigma^*_{gyro} \tag{12.6b}$$
$$T_{acc} = 100000\,\mathrm{s} \tag{12.6c}$$
$$T_{gyro} = 100000\,\mathrm{s} \tag{12.6d}$$
$$\sigma_{b,acc} = 0.0001\,\mathrm{m/s^2} \tag{12.6e}$$
$$\sigma_{b,gyro} = 0.00001\,\mathrm{rad/s} \tag{12.6f}$$

Pressure bias estimator

$$\sigma_{lf} = 0.1\,\mathrm{Pa} \tag{12.7a}$$
$$\sigma_{wf} = 10\,\mathrm{Pa} \tag{12.7b}$$
$$T_{lf} = 10000\,\mathrm{s} \tag{12.7c}$$
$$\lambda = 0.1 \tag{12.7d}$$

Aiding measurements

$$\sigma_\phi = 0.001\,\text{rad} \tag{12.8a}$$
$$\sigma_\theta = 0.001\,\text{rad} \tag{12.8b}$$
$$\sigma_\psi = 0.001\,\text{rad} \tag{12.8c}$$
$$\tilde{\boldsymbol{\mathcal{R}}}_{dvl} = 300^2\boldsymbol{\mathcal{R}}^*_{dvl} \tag{12.8d}$$
$$\sigma_{ps} = 5\sigma^*_{ps} \tag{12.8e}$$
$$\sigma_{usbl} = 3\sigma^*_{usbl} \tag{12.8f}$$

Wave filter

$$\omega_{lp} = 10\,\text{rad/s} \tag{12.9a}$$
$$\zeta = 0.01\,\text{rad/s} \tag{12.9b}$$

Frequency estimator

$$\omega_f = 4\,\text{rad/s} \tag{12.10a}$$
$$\omega_{hp} = 0.5\,\text{rad/s} \tag{12.10b}$$
$$k = 20 \tag{12.10c}$$

Outlier rejection

The outlier rejection upper bound for DVL measurements is set to $\epsilon_{upper} = \text{chi2inv}(0.975, 3)$.

12.3 Initialization

Initialization is of great importance in Kalman filtering. A poorly initialized Kalman filter cannot be expected to perform well for a significant period of time and since the Kalman filter in this thesis is a nonlinear extension, it may also diverge completely. To avoid annoying and unnecessary manual labor, the INS and ESKF will be initialized using measurements.

12.3.1 Position initialization

The USBL is used to initialize the position such that:

$$\hat{\boldsymbol{p}}^n \leftarrow \boldsymbol{y}^n_{usbl} \tag{12.11}$$

The most reasonable initial covariance becomes:

$$\hat{\boldsymbol{\mathcal{P}}}_{pos} \leftarrow \boldsymbol{\mathcal{R}}_{usbl} \tag{12.12}$$

, where $\hat{\boldsymbol{\mathcal{P}}}_{pos}$ is the 3×3 submatrix of the state covariance corresponding to position.

12.3.2 Velocity initialization

The DVL can be used to initialize velocity. Assuming that the angular velocity is negligible (we do not want to initialize using the biased gyro), we get:

$$\hat{\boldsymbol{v}}^b \leftarrow \boldsymbol{D}\boldsymbol{y}^d_{dvl} \tag{12.13}$$

, with the covariance initialized to:

$$\hat{\boldsymbol{\mathcal{P}}}_{vel} \leftarrow \boldsymbol{D}\tilde{\boldsymbol{\mathcal{R}}}_{dvl}\boldsymbol{D}^\top \tag{12.14}$$

12.3.3 Attitude initialization

The attitude is initialized through the AHRS:

$$\hat{\boldsymbol{q}} \leftarrow \mathbf{q}(y_\phi, y_\theta, y_\psi) \tag{12.15}$$

The covariance is initialized using the attitude Jacobian:

$$\hat{\boldsymbol{\mathcal{P}}}_{att} \leftarrow \boldsymbol{T}_{att}(y_\theta, y_\psi)\boldsymbol{\mathcal{R}}_{\phi\theta\psi}\boldsymbol{T}^\top_{att}(y_\theta, y_\psi) \tag{12.16}$$

12.3.4 Pressure bias

The wave frequency bias is initialized to 0 but with a large covariance. The low-frequency bias is potentially large and needs proper initialization. The USBL and the pressure sensor will be used to do so.

$$\hat{b}_{lf} \leftarrow y_{ps} - \rho(\boldsymbol{g}^n)^\top\boldsymbol{y}^n_{usbl} \tag{12.17}$$

The covariance becomes a combination of USBL and pressure sensor noise:

$$\hat{\mathcal{P}}_{wf} \leftarrow \mathcal{R}_{ps} + \rho^2(\boldsymbol{g}^n)^\top\boldsymbol{\mathcal{R}}_{usbl}\boldsymbol{g}^n \tag{12.18}$$

Chapter 13

Results and Discussion

The results will be evaluated module-wise, meaning we will try to isolate the different components of the observer in order to determine the performance of the component and not just the entire system. This is a decision made because parts of the system perform less than ideal, while other parts perform as intended. It is therefore considered more constructive to separate them and see which modules should be kept and which need improvement/redesign. Section 13.1 presents the USBL and DVL measurements that the observer is tested with. Getting an idea of the quality and consistency of these measurements is important to the later results and discussion. Section 13.2 presents and discusses the performance of the observer core. This is followed by Section 13.3 which concerns the performance of the outlier rejection. The performance of the wave frequency estimator is discussed in Section 13.4, before the pressure bias estimator is evaluated in Section 13.5. Finally, the resulting estimates after wave filtering are presented and discussed in Section 13.6.

13.1 Datasets

This section discusses the data that the results in this chapter are based on. All the data was gathered in a full-scale real-world test at a fish farm (Section 12.1). The three datasets are picked over data from other test sets because they contain effects that we want to test the observer designs robustness against. The datasets are named numerically 1-3. We will focus on the hydroacoustic measurements because they are the most susceptible to unwanted effects like outliers, loss of signal, and overall poor measurement quality.

13.1.1 Dataset 1

This dataset is picked to test the outlier rejection on the DVL. The USBL measurements (Figure 13.2) are as good as they get; the reported measurement standard deviation is more or less constant, there are no signal losses, and no outliers except one at about 120 seconds. The Dvl measurement (Figure 13.1) is consistent and of high quality until the DVL is turned away from the net at about 70 seconds. At this point, some of the measurements are very uncertain (as indicated by the standard deviation in Figure 13.1), and the DVL seems to pick up something that is inconsistent (possibly swimming fish), but is reported as certain by the DVL through the standard deviation.

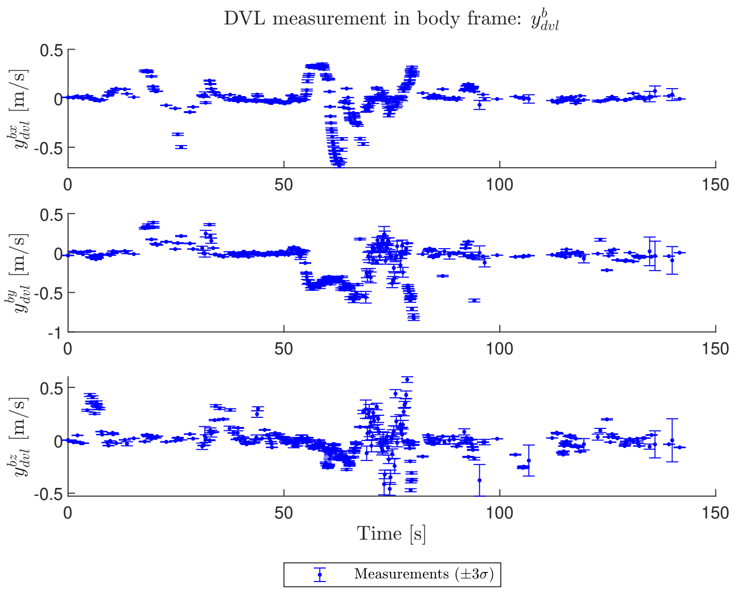

Figure 13.1: Dataset 1: Dvl measurements

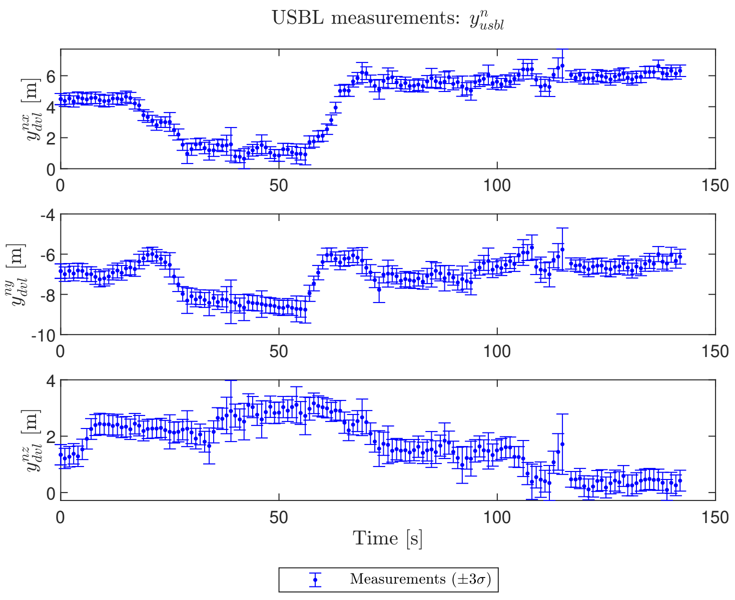

Figure 13.2: Dataset 1: USBL measurements

13.1.2 Dataset 2

Dataset 2 is chosen solely to test the observer robustness to USBL signal loss and poor USBL performance. At 60 seconds into the test run (Figure 13.4), the signal is lost for about 5 seconds. Why this is is uncertain, but it is unfortunately not uncommon in the complex environment of a fish farm (and it is therefore necessary that the observer is robust in this sense). The DVL performance (Figure 13.3) is typical with measurement losses every now and then as fish swim by or the vehicle attitude suggests that measurements are unlikely.

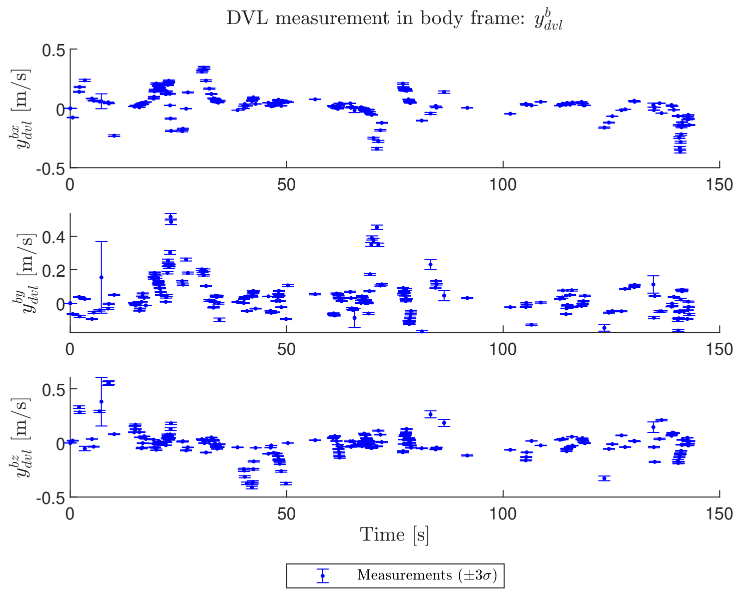

Figure 13.3: Dataset 2: Dvl measurements

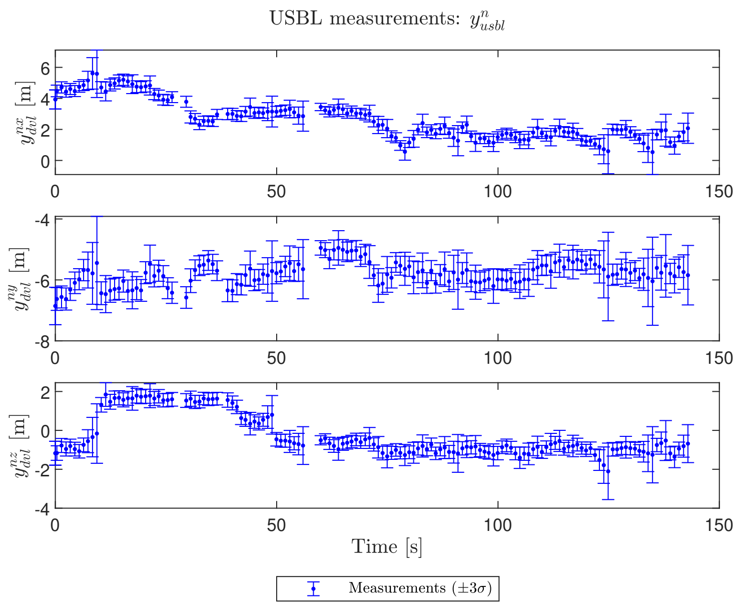

Figure 13.4: Dataset 2: USBL measurements

13.1.3 Dataset 3

Dataset 3 is a worst-case test set. The USBL data in Figure 13.6 is the worst achieved in all test runs. The signal loss, in the beginning, lasts about 20 seconds. For the first 200 seconds, the USBL reports uncertainties to a scale comparable to the size of the net pen that the vehicle operates in. To make matters worse, the DVL measurements (Figure 13.5) are lost before 120 seconds. The USBL performs well from 200 seconds and on. The intention of including this data set is to:

1. Test if the observer diverges with worst-case measurements.
2. To show how bad measurements can get in this environment

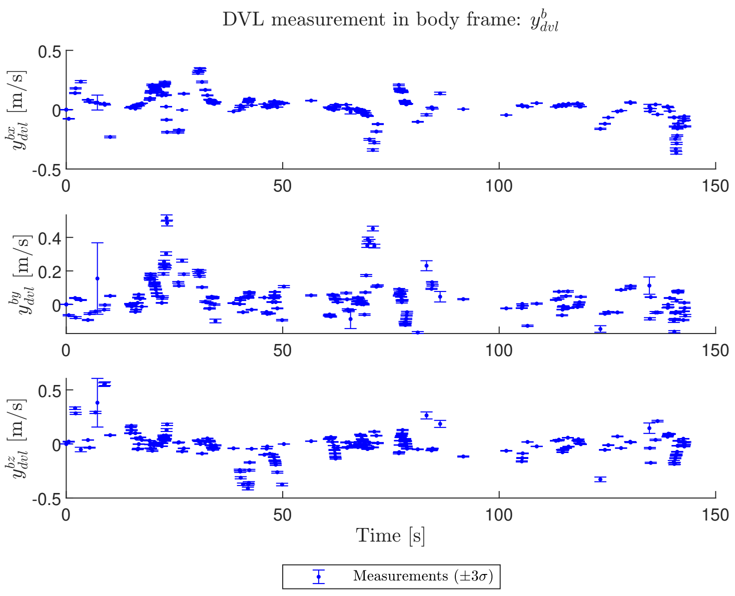

Figure 13.5: Dataset 3: DVL measurements

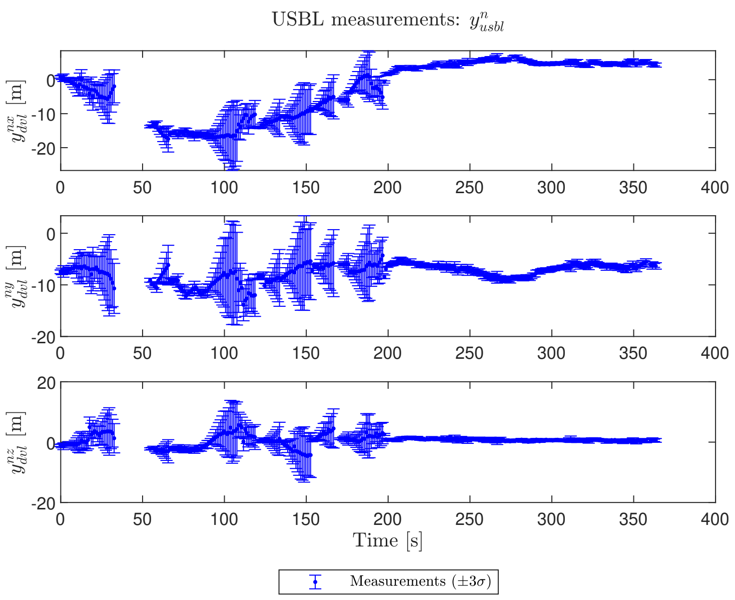

Figure 13.6: Dataset 3: USBL measurements

13.2 Observer performance

Firstly, and most importantly, we will evaluate the observer's performance. As we have no way of knowing the true state, we are left with evaluating the quality of the estimates. We will start with the data in Dataset 1 in order to make it easy for the observer.

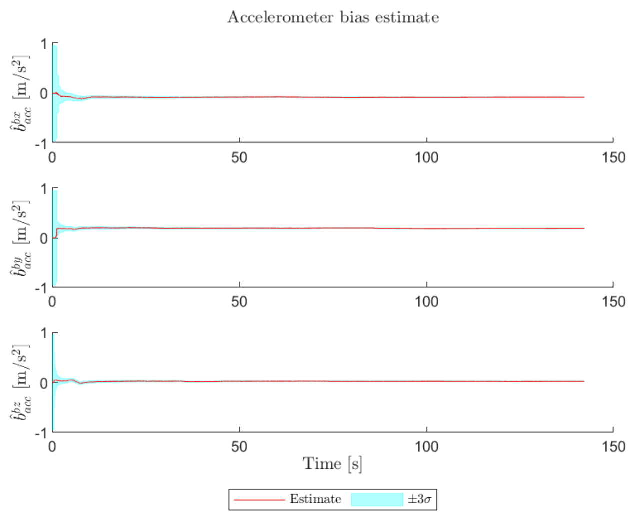

Figure 13.7: Dataset 1: Accelerometer bias estimate with standard deviation.

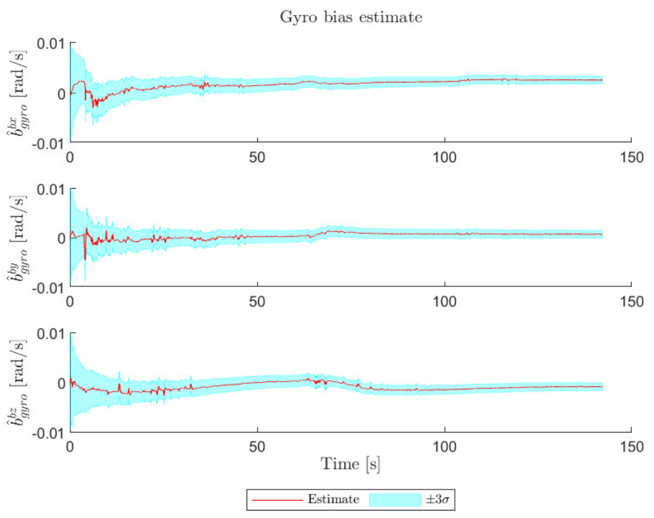

Figure 13.8: Dataset 1: Gyro bias estimate with standard deviation.

The ability to estimate the IMU biases is vital in inertial navigation as it is necessary to compensate for biases to prevent the estimates from drifting. Figures 13.7 and 13.8 show the convergence. The observer manages this fairly nicely; the accelerometer bias estimate is excellent, while the gyro bias estimate needs some time to converge.

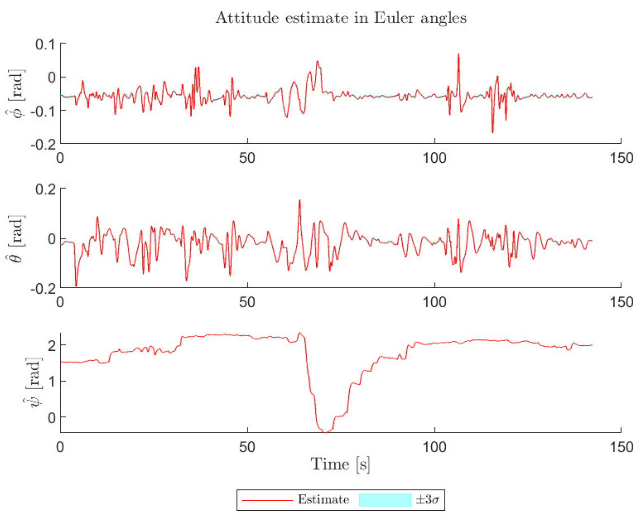

Figure 13.9: Dataset 1: Attitude estimate in Euler angles with standard deviation.

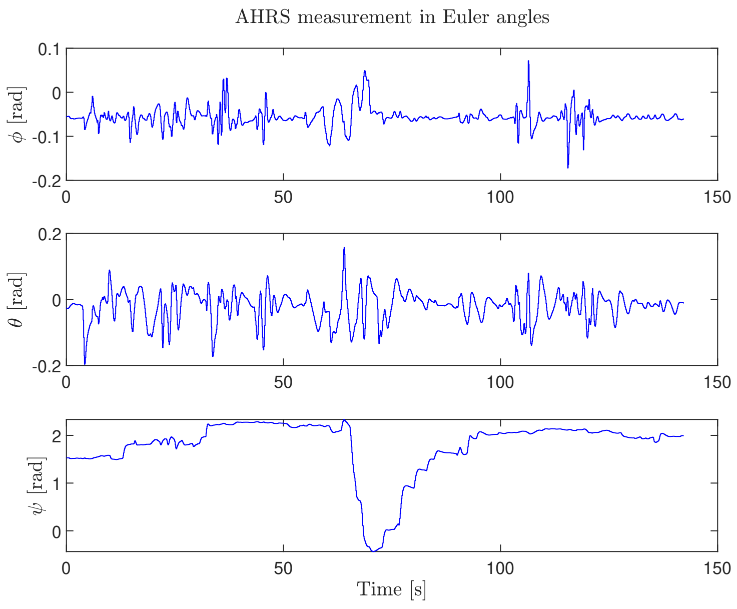

Figure 13.10: Dataset 1: AHRS attitude from the vehicle.

The attitude estimate (Figure 13.9) follows the AHRS estimates (Figure 13.10) reported from the vehicle. The accuracy of the estimate is difficult to judge, but it looks usable as feedback for a control system.

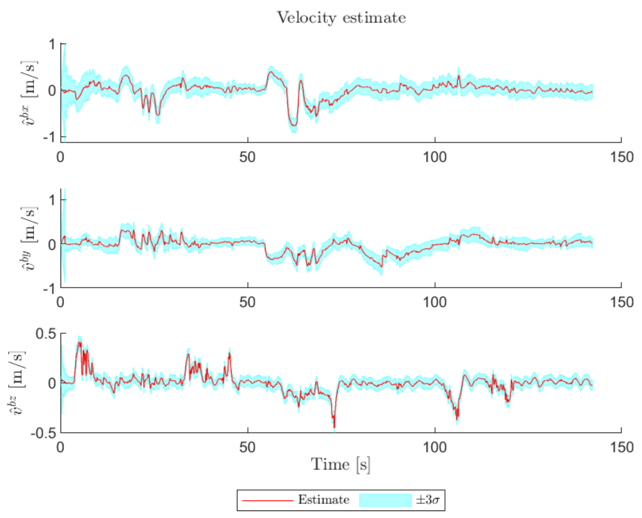

Figure 13.11: Dataset 1: Velocity estimate with standard deviation.

The velocity estimate (Figure 13.11) looks mostly satisfactory. The signal is smooth with oscillations from wave motion (which we expect to be present here). There is, however, a little more high-frequency noise than we would like present in the signal.

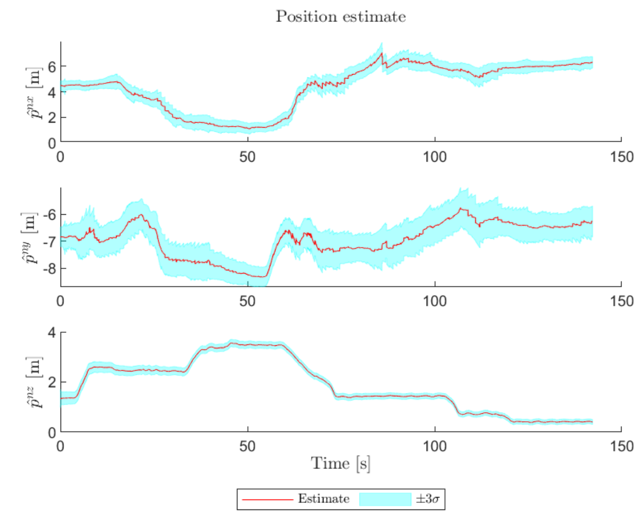

Figure 13.12: Dataset 1: Position estimate with standard deviation.

Figure 13.12 shows the position estimate along with three times the standard deviation estimated by the Kalman filter. The z-component (bottom plot) is estimated excellently, which is expected since both the USBL and the pressure sensor are available to estimate this and the pressure sensor is a high rate and high precision sensor. The x and y estimates are however far from ideal. The estimates are not smooth and closer to a sawtooth shape. This is because the estimate needs to be corrected for to much every time the USBL gives a measurement. Also note that it is impossible to distinguish wave motion from noise in the x-y estimates although waves are clearly there in the x-y velocity estimates in Figure 13.11. All of this indicates that the inertial navigation system is not performing well as it needs to be corrected too much. The primary error source here is probably the quality of the Inertial Measurement Unit (IMU). This is a consumer-grade IMU with a lot of noise. Additionally, the rate of the IMU measurements from the vehicle software was much lower in practice than expected getting as low as 10 Hz at times as opposed to the 50Hz that we want. We should also note that the use of a magnetometer-based attitude-aiding device (AHRS) is probably not making matters better, as a wrong yaw estimate is also expected to have a similar effect on the north and east estimates.

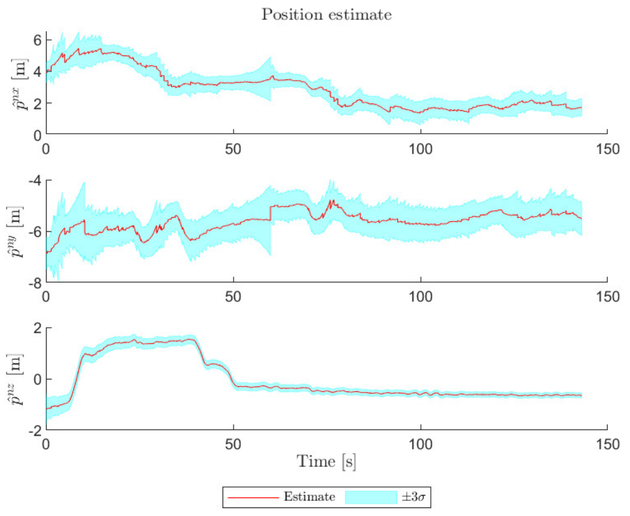

Figure 13.13: Dataset 2: Position estimate with standard deviation.

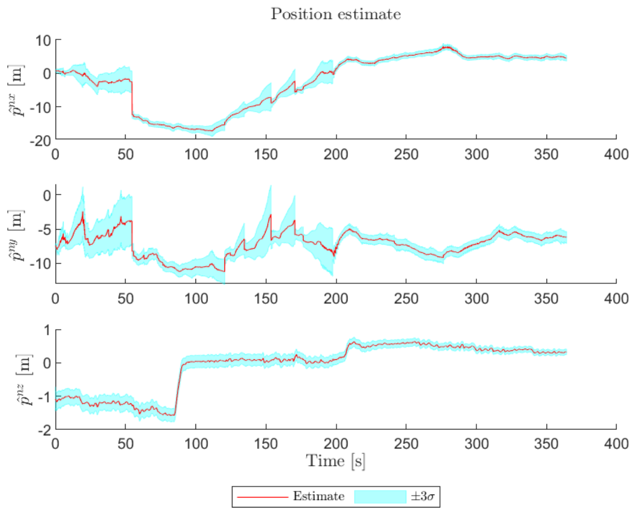

Figure 13.14: Dataset 3: Position estimate with standard deviation.

Having established that the observer works, although suboptimally, we will move on with the results from the two other datasets. Figure 13.13 shows the position estimate on Dataset 2 (the one with loss of USBL signal). The general performance is the same sawtooth signal as we saw earlier. More interesting is the performance when the USBL signal is lost at around 60 seconds. The inertial navigation system drifts as the signal is lost and a large jump (about 0.5 meters) in estimated position is required after a 5-second loss of signal. This is unacceptable. Figure 13.14 shows the performance on Dataset 3. Running any sort of feedback control systems on these estimates would surely not end well. Jumps of 10 meters; indicating errors in the same range would result in a collision with the net which is unacceptable.

13.3 Outlier rejection

The outlier rejection must be tested on a dataset where the observer is somewhat consistent, meaning we will use Dataset 1. This is also a dataset where the outlier rejection actually encounters measurements that need to be removed.

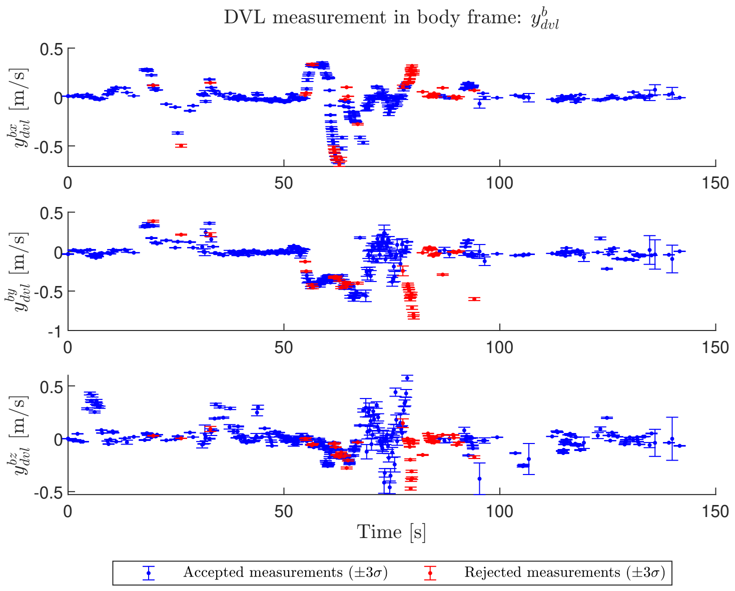

Figure 13.15: Dataset 1: Dvl measurement with rejected outliers.

Figure 13.15 show the DVL measurements. After roughly 70 seconds, the vehicle turns away from the net, meaning that the DVL is no longer directed normally towards the net pen wall, but instead on an angle. This also results in longer travel distance for the DVL beams (along the net) and therefore a higher risk of being reflected by fish. Exactly what causes the degradation in measurement quality is unknown. In the period 70-80 seconds, the DVL itself reports poor signal quality through a large standard deviation (indicated using error bars in the plots). Following that, the outlier rejection rejects a series of measurements that the DVL reports to have a low standard deviation. These measurements were not consistent with the INS leading to large innovations and NIS (Figure 13.16) and are therefore rejected. Note that the entire period of 70-90 seconds has questionable measurements, but that in the period from 70-80 seconds, few measurements are rejected because the DVL reports a high standard deviation resulting in a low NIS. Once the bad measurements are deemed certain by the DVL, they are rejected. This leads to a more robust outlier rejection as measurements that are uncertain, but not completely wrong are accepted, while the measurements that are outright wrong are rejected. We should also note that the outlier rejection also has some “random victims” along the way.

These measurements have NIS' that just about crosses the limit and are therefore rejected. As can be seen in Figure 13.16, the NIS of these measurements are significantly less than those belonging to the outliers.

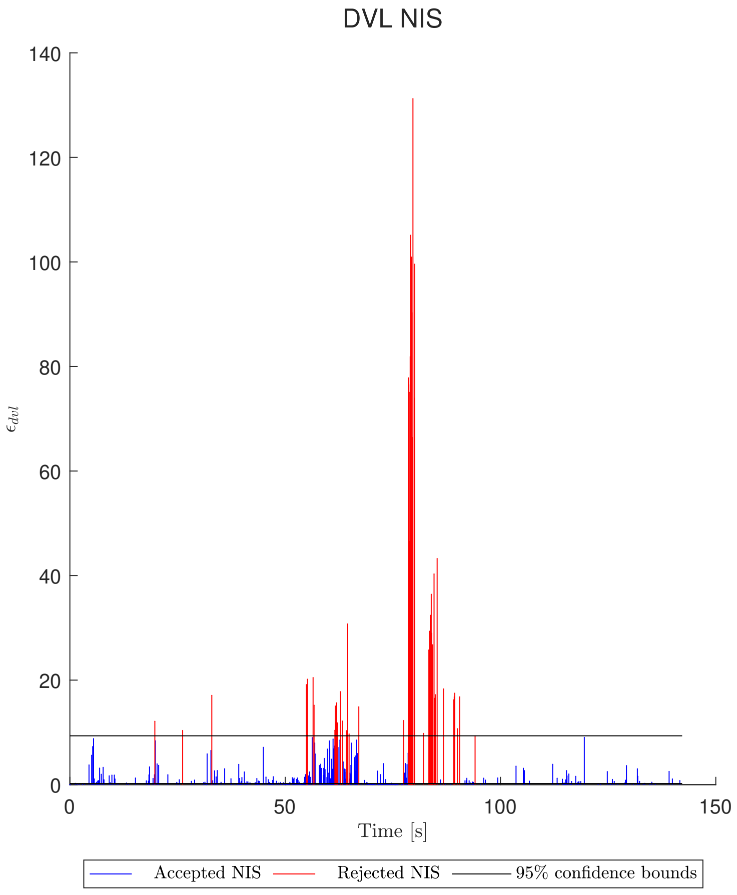

Figure 13.16: Dataset 1: Plot of NIS for the DVL.

13.4 Wave frequency estimator

Figure 13.17 consist of three plots. The top plot is the frequency estimate over time. The middle plot is the AHRS pitch estimate which is used as input to the frequency estimator. The bottom plot is the normalized Fast Fourier Transform (FFT) of the signal in the middle plot. The top plot shows that the frequency estimate converges to a frequency of about 2 rad/s. By inspection of the FFT, this seems likely, although some lower frequencies make it difficult to distinguish the wave motion from the vehicle motion. The convergence is slow, using 50 seconds to converge which is considerable considering the application. The estimate is nice and stable once it has converged.

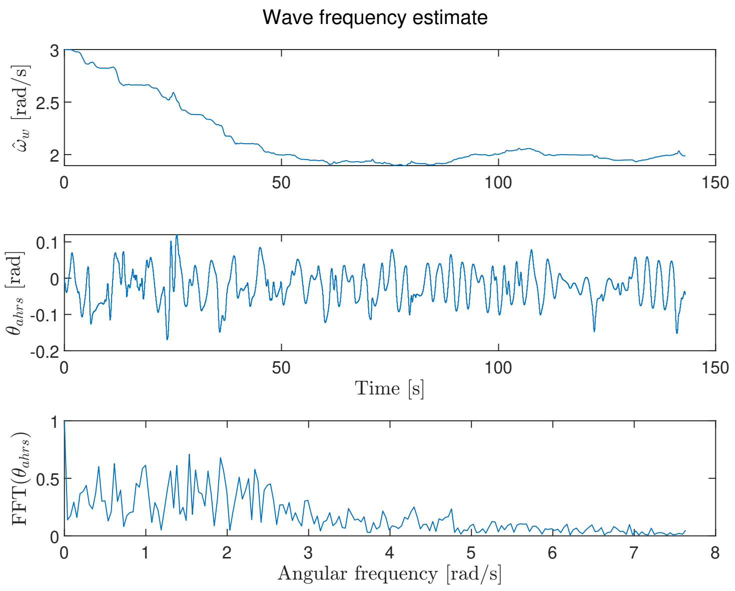

Figure 13.17: Dataset 2: Estimate of the wave spectrum peak frequency.

13.5 Pressure bias estimator

The pressure bias estimator consists of two parts; one low frequency bias (upper plots), and one bias that mainly consists of frequencies close to the wave frequency estimate (lower plots). Both of these estimates need a bit of time in order to converge. The low-frequency bias needs a period with USBL and pressure measurements in order to determine the difference between the two, and the wave-frequency bias estimate requires the wave-frequency estimator to converge before we can expect it to work. Until then it will tend to pick up other frequencies that may lead to unwanted effects.

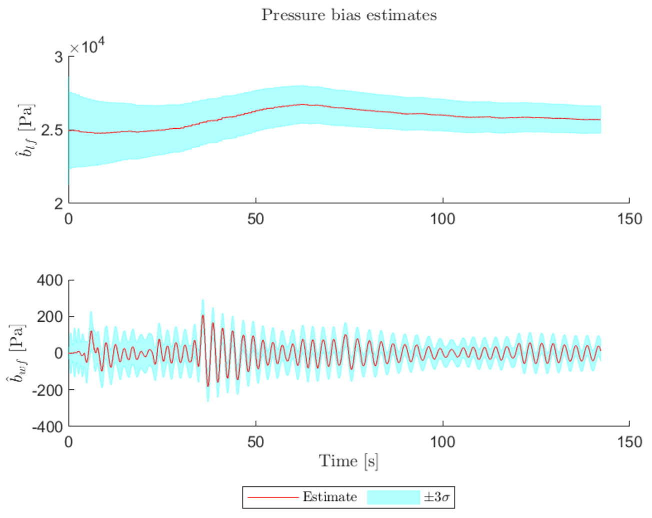

Figure 13.18: Dataset 1: Pressure bias estimates.

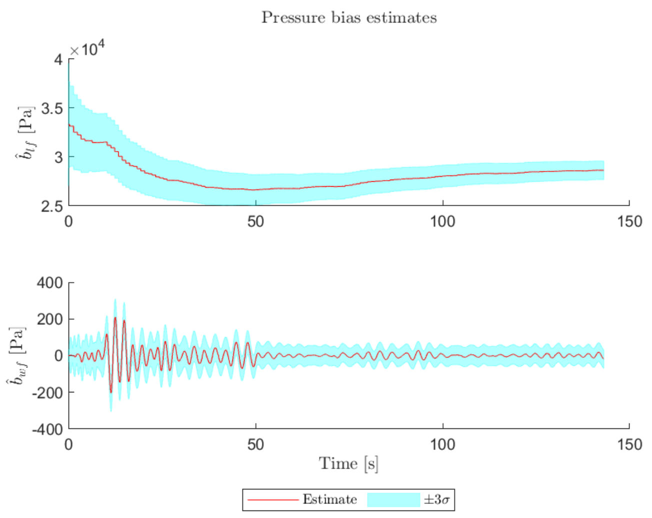

Figure 13.19: Dataset 2: Pressure bias estimates.

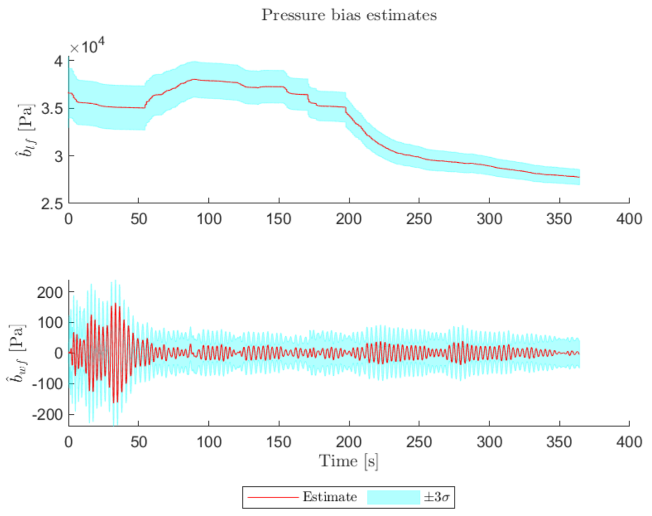

Figure 13.20: Dataset 3: Pressure bias estimates.

Figures 13.18, 13.19 and 13.20 show the convergence of the pressure bias estimates on Dataset 1,2 and 3 respectively. The low-frequency bias estimates are of a significant scale (meters of depth), indicating that modeling of the bias is indeed needed. The estimates remain fairly stable but are slowly changing. There is no way that tides and changes in atmospheric pressure would change the bias this quickly, so there are definitely imperfections here. The best bet would be that the pressure measurement scaling factor (water density and gravity) is off. These types of errors would naturally pile up on the bias and make it dependent on the measurements. On the other side; these types of errors will always be present and a bias term is probably the safest place for them to go (much worse if they influence the INS state estimates). The wave frequency estimates are always small, although generally larger before the frequency estimator converges at about 50-70 seconds. Once the frequency estimate converges, the bias estimate also stabilizes on all three datasets. Note that 100 Pa of pressure corresponds to about 1 cm of water depth, so the amplitude of these biases are in ranges of millimeters. This is negligible in this context and a lot less than expected. This can be due to a low-end accelerometer and INS not being able to capture the movement of the waves accurately enough, or, perhaps more likely, it can be that we have overestimated the impact of waves on the pressure measurement. Nevertheless, the estimate behaves in a robust manner and does not appear to introduce unwanted oscillations into the system, instead, it settles on a small amplitude close to zero whenever there is no/little bias for it to estimate. This tells us that the design and implementation of the wave pressure bias estimator itself are working as intended, although it may not be necessary for the application.

13.6 Wave filter

The performance of the wave filters will be evaluated on Dataset 2, which is a dataset where the observer estimation performance is descent enough to isolate the filter performance from the observer performance, while there are enough waves present to make the wave motion distinguishable. In general; keep in mind that the frequency estimator converges about halfway into the dataset (see Figure 13.17).

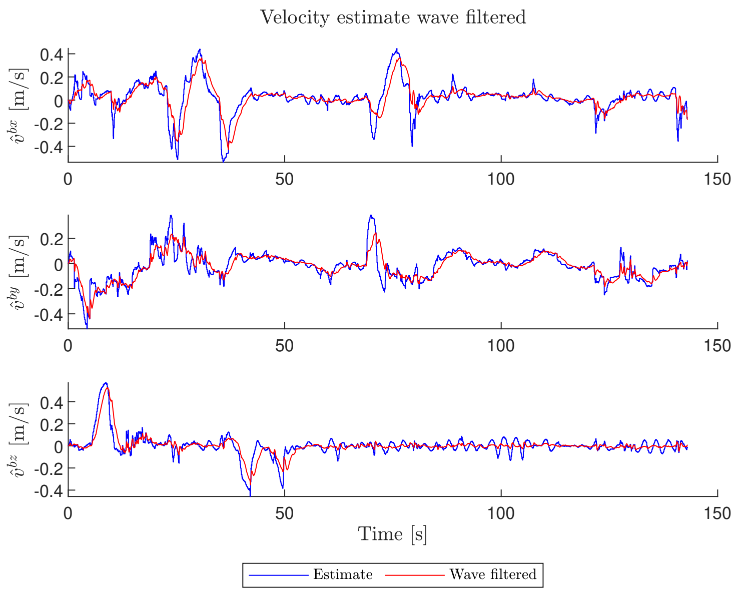

Figure 13.21: Dataset 2: Velocity estimate before and after filtering.

Figure 13.21 shows the result of wave filtering the velocity estimate. The wave-filtering behaves nicely from the beginning although the frequency estimate does not converge until about 80 seconds. After that it is clear that the filter removes a lot of the oscillations from the signal, keeping the low-frequency motion while being able to follow high-frequency spikes (for instance middle plot at about 130 seconds). It also becomes apparent that wave filtering comes with the price of phase lag, especially for the lower frequencies.

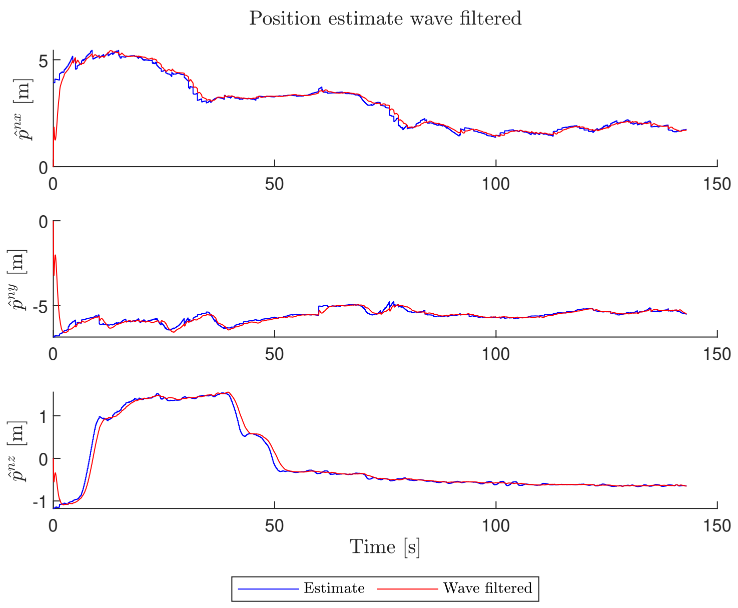

Figure 13.22: Dataset 2: Position estimate before and after filtering.

Figure 13.22 contains the result of wave filtering on the position estimate. The bottom plot (z-estimate) shows signs of the wave filtering actually being useful, the curve is very smooth after, and the result is definitely usable as control feedback. On the two upper plots, it is apparent that the wave filter follows the unfortunate saw-tooth estimates fairly well. It is very difficult to distinguish any wave motion in the original estimates so wave filtering has little use here (although this is not the wave filter's fault).

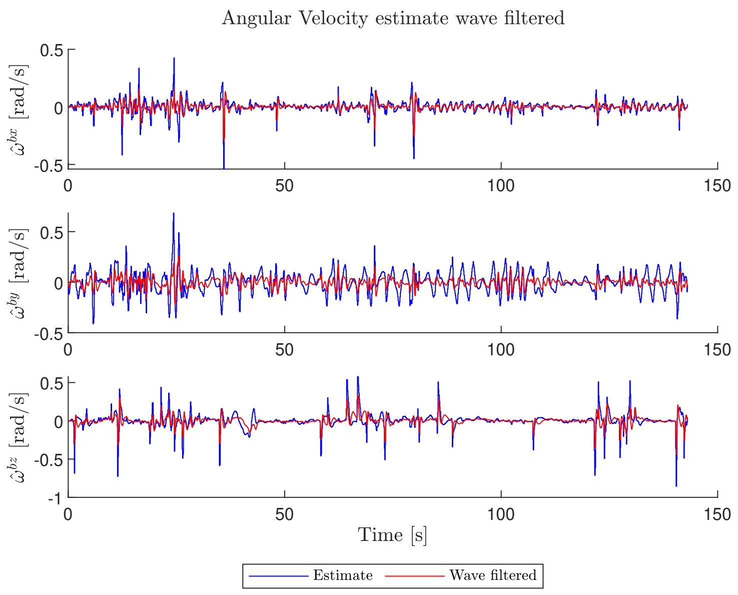

Figure 13.23: Dataset 2: Angular velocity estimate before and after filtering.

The angular velocity is also interesting to the wave filter since a Proportional-Integral-Derivative (PID)-controller for attitude would need the derivative of attitude to work. The results are shown in Figure 13.23. In the upper plot (angular velocity about the body frame x-axis), the filtering seems to have little effect, while the filtering works as intended in the middle plot. The oscillations in the top plot seem to have a significantly higher frequency, which is unexpected. It is difficult to say exactly what causes this, but the oscillations in the top plot are negligible anyways.

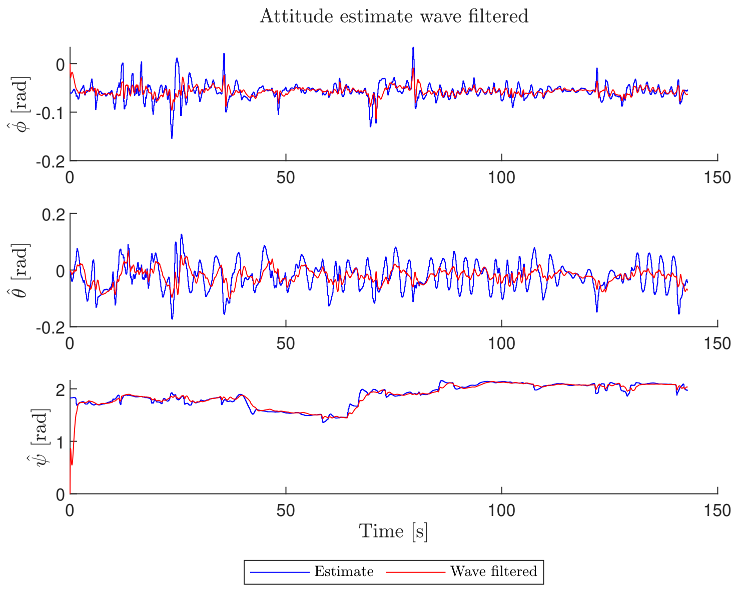

Figure 13.24: Dataset 2: Attitude estimate before and after filtering.

Wave filtering also seems to work as intended in Figure 13.24, especially the middle plot. This is expected since the measurement of this state is used as input to the frequency estimator.

In general, the wave filters appear to be working as intended. The discussion is more a matter of whether or not wave filtering is worth it on such an agile craft. The phase lag is considerable and it is not obvious that the wave filters would save the control system and actuators from unnecessary wear as the part that is filtered out is not very noticeable.

Chapter 14

Conclusion and Further Work

This chapter concludes on the results and discussion of the previous chapter in Section 14.1, and proposes changes and improvements to the design in Section 14.2.

14.1 Conclusion

In short, the results in Chapter 13 show some of the modules in the design performed well, while others (the most important) did not. We will start with what worked.

The wave frequency estimator (Section 13.4) worked well and converged to an estimate that gave good wave filtering performance for the wave filters in Section 13.6. The convergence is, however, a bit slow for the application but in general this style of frequency estimation seems robust and well-suited for the application.

The use of adaptive notch filters for wave filtering also seems like a good choice for the application. As discussed in Section 13.6, they clearly remove a lot of the wave motion from the observer output. They are simple, reliable, and guaranteed stable. The major concern should be that they add significant phase lag which can be a bit challenging for control systems on such a small craft.

The pressure bias estimator was the most experimental module in the design. The idea behind the estimator was to ensure that the pressure measurement became consistent with the USBL through a low-frequency bias and the accelerometer through a bias oscillating at wave frequency. The low-frequency estimate converges nicely, as discussed in 13.5. The bias estimate is of a significant magnitude, indicating that it is beneficial when using two depth sensors. The wave frequency bias estimate has, in general, a very small amplitude, and therefore little impact on the system performance, neither positively nor negatively. However, the design and implementation worked well.

The NIS outlier rejection seems to work well when tested in Section 13.3. The outlier rejection avoids the inclusion of false measurements into the observer algorithm. Unfortunately, some seemingly “good” measurements are lost along the way. Further tuning may be in order. In general, outlier rejection seems like a wise inclusion when using DVL in the environment of a fish farm.

The aided INS observer is the core of this design. The observer design itself works and runs in a robust manner, but the performance of the estimation itself is insufficient for the application. This is mainly because of the position estimates which are bad on "ideal" input and simply unacceptable when the USBL signals are lost (see Section 13.2). In short, this is due to the overall quality of the measurements and signals available being insufficient for the chosen observer design. Or, the other way around, the wrong observer type was selected for the measurements available. Simply put, the IMU is not able to keep the state estimates precise enough during periods between USBL measurements, and the USBL is not consistent enough to compensate for the IMU shortcomings.

14.2 Further work

Since the observer performed worse than desired, this section is dedicated to what should be done in order to achieve the desired observer performance. Section 14.2.1 discussed what should be kept from the design in this thesis, Section 14.2.2 what should be changed/redesigned, Section 14.2.3 what was unnecessary to include, and Section 14.2.4 what should be added to the design.

14.2.1 What should be kept and improved upon

Some parts of the observer design worked well and should be kept in a future improvement/redesign. The style of frequency estimator used here is well suited for the application and should be kept, while the initial convergence rate should be improved (for instance through an initial gain as in Belleter et al. (2015)). The adaptive wave filters are also a good fit and should be kept if wave filtering is considered necessary. The low-frequency bias (bias between USBL and pressure depth) is necessary to ensure filter consistency and should be included if both measurements are used (the alternative is to use only the pressure sensor and let the z-estimate be separate from the USBL x-y). Finally, DVL outlier rejection is useful for a complex environment like a fish farm. Further improvements and adaption to the redesign of the observer could improve performance even more.

14.2.2 What should be redesigned

The observer. There are two options here:

1. Retrofit the vehicle with a higher-end (and much more expensive) IMU allowing for a higher rate, more precise, and better-timed input to the INS. This allows keeping the observer design. This is probably the best option considering everything but the cost.

2. Switch to a model-based design.

Switching to a model-based design leads to a lot of other issues (see Section 7.2.1). It is by no means certain that this would work better than the design tested here (especially in rapidly changing environments). The main challenge would be modeling. Since the BlueROV2 already has an attitude observer (here referred to as AHRS), a 3-DOF model for the translational motion should suffice. Most modeling approaches for this shape of vehicle are unreasonable, but the sides of the vehicle are all of similar size, modeling the vehicle as a sphere is the simplest of the unreasonable alternatives.

14.2.3 What should be left out

In high-end pressure-aided inertial navigation, the wave pressure bias estimator would probably be useful. The example in Willumsen et al. (2007) is seabed mapping, where the precision of the vertical measurement is vital. For these applications, the inclusion of such an estimator is probably wise, and the design in Section 10.2 would work well. Considering that the estimated bias is in the range of millimeters, the impact on the system is negligible compared to the major error sources (IMU quality). The estimator also complicates the observer tuning with extra variables. It is therefore recommended to leave this out of future improvements/redesigns.

14.2.4 What should be added

If a better IMU becomes available, attempts at outlier rejection on the USBL measurements may improve performance. The requirement here is that the IMU measurements are so good that the INS maintains precise measurements during the period when the USBL is rejected. In this thesis, the INS estimates got worse during the period of rejecting measurements than the rejected measurements themselves.

References

Amundsen, H.B., 2020. *Robust nonlinear ROV motion control for autonomous inspections of aquaculture net pens.* Master's thesis. Trondheim: Norwegian University of Science and Technology.

Aranovskiy, S., Bobtsov, A., Kremlev, A. and Luk'yanova, G., 2007. A robust algorithm for identification of the frequency of a sinusoidal signal. *Journal of computer and systems sciences international*, 46, June.

Bar-Shalom, Y., Li, X.R. and Kirubarajan, T., 2001. *Estimation with applications to tracking and navigation.* New York: Wiley.

Belleter, D.J., Galeazzi, R. and Fossen, T.I., 2015. Experimental verification of a global exponential stable nonlinear wave encounter frequency estimator. *Ocean engineering*, 97, pp.48–56.

Bjelland, H.V., Føre, M., Lader, P., Kristiansen, D., Holmen, I.M., Fredheim, A., Grøtli, E.I., Fathi, D.E., Oppedal, F., Utne, I.B. and Schjølberg, I., 2015. Exposed aquaculture in norway. *Oceans 2015 - mts/ieee washington*, pp.1–10.

Blue Robotics, 2022. *BlueROV2 datasheet: the world's most affordable high-performance ROV* [Online]. Revision 03/22, March. Available from: `https://bluerobotics.com/store/rov/bluerov2/`.

Dukan, F., 2014. *ROV motion control systems.* PhD thesis. Trondheim: Norwegian University of Science and Technology.

FAO, 2018. *The state of world fisheries and aquaculture 2018 - meeting the sustainable development goals.* (technical report). Rome.

Farrell, J., 2008. *Aided navigation: GPS with high rate sensors.* New York: McGraw Hill Education.

Føre, M., Frank, K., Norton, T., Svendsen, E., Alfredsen, J.A., Dempster, T., Eguiraun, H., Watson, W., Stahl, A., Sunde, L.M., Schellewald, C., Skøien, K.R., Alver, M.O. and Berckmans, D., 2018. Precision fish farming: a new framework to improve production in aquaculture. *Biosystems engineering*, 173. Advances in the Engineering of Sensor-based Monitoring and Management Systems for Precision Livestock Farming, pp.176–193.

Fossen, T.I., 2021. *Handbook of marine craft hydrodynamics and motion control.* 2nd ed. Hoboken, NJ: Wiley.

Fossen, T.I. and Strand, J.P., 1999. Passive nonlinear observer design for ships using lyapunov methods: full-scale experiments with a supply vessel. *Automatica*, 35(1), pp.3–16.

Gade, K., 1997. *Integrering av treghetsnavigasjon i en autonom undervannsfarkost.* (technical report 97/03179). Kjeller: Forsvarets forskningsinstitutt.

Gradshteĭn, I.S., Ryzhik, I.M., Jeffrey, A. and Zwillinger, D., 2007. *Table of integrals, series and products.* Vol. 7. Elsevier Ltd.

Grip, H.F., Fossen, T.I., Johansen, T.A. and Saberi, A., 2017. Nonlinear observer for attitude, position, and velocity: theory and experiments. In: *Multisensor attitude estimation - fundamental concepts and applications.* Taylor & Francis, p.1.

Groves, P.D., 2008. *Principles of gnss, inertial, and multi-sensor integrated navigation systems.* Boston: Artech House.

Guennebaud, G., Jacob, B. et al., 2010. *Eigen v3* [Online]. Available from: `http://eigen.tuxfamily.org`.

Hval, M.N., 2012. *Modelling and control of underwater inspection vehicle for aquaculture sites.* Master's thesis. Trondheim: Norwegian University of Science and Technology.

Jaffre, F.M., Austin, T.C., Allen, B.G., Stokey, R. and von Alt, C.J., 2005. Ultra short baseline acoustic receiver/processor. July, 1382–1385 Vol. 2.

Kalman, R.E., 1960. A new approach to linear filtering and prediction problems. *Transactions of the asme–journal of basic engineering*, 82(Series D), pp.35–45.

Li, M. and Mourikis, A.I., 2012. Improving the accuracy of ekf-based visual-inertial odometry. *2012 ieee international conference on robotics and automation*, pp.828–835.

Madgwick, S.O.H., Harrison, A.J.L. and Vaidyanathan, R., 2011. Estimation of imu and marg orientation using a gradient descent algorithm. *2011 ieee international conference on rehabilitation robotics.* IEEE, pp.1–7.

Mahony, R., Hamel, T. and Pflimlin, J.-M., 2008. Nonlinear complementary filters on the special orthogonal group. *Ieee transactions on automatic control*, 53(5), pp.1203–1218.

MathWorks, 2023. *Lowpass filter orientation using quaternion slerp* [Online]. Available from: `https://www.mathworks.com/help/nav/ug/lowpass-filter-orientation-using-quaternion-slerp.html` [Accessed 2 May 2023].

Rundtop, P. and Frank, K., 2016. Experimental evaluation of hydroacoustic instruments for rov navigation along aquaculture net pens. *Aquacultural engineering*, 74, pp.143–156.

Solà, J., 2017. *Quaternion kinematics for the error-state kalman filter* [Online]. (Accessed: 19 August 2022). Available from: `http://www.iri.upc.edu/people/jsola/JoanSola/objectes/notes/kinematics.pdf`.

ST Microelectronics, 2013a. *L3GD20H: three-axis digital output gyroscope* [Online] L3GD20H. (Rev 2, March 2013, DocID023469), March. Available from: `https://www.st.com/en/mems-and-sensors/l3gd20h.html`.

ST Microelectronics, 2013b. *LSM303D: 3D accelerometer and 3d magnetometer* [Online] LSM303D. (Rev 2, November 2013, DocID023312), November. Available from: `https://www.st.com/en/mems-and-sensors/lsm303d.html#overview`.

Stenså, O., 2022. *Choice and design of a 6 degree-of-freedom observer for an UUV operating in fish farms in the presence of time-varying environmental disturbance.* Specialization project for MSc in Cybernetics. Unpublished.

TE Connectivity, 2019. *MS5837-30BA: ultra-small, gel-filled, pressure sensor with stainless steel cap* [Online] MS5837-30BA. (Rev C2, December 2019), December. Available from: `https://www.te.com/usa-en/product-MS583702BA01-50.html`.

The International Gravimetric Bureau, n.d. *Prediction of gravity value* [Online]. Available from: `https://bgi.obs-mip.fr/data-products/outils/prediction-of-gravity-value/` [Accessed 17 April 2023].

Thorvaldsen, T., Holmen, I.M. and Moe, H.K., 2015. The escape of fish from norwegian fish farms: causes, risks and the influence of organisational aspects. *Marine policy*, 55, pp.33–38.

Trawny, N. and Roumeliotis, S., 2005. *Jacobian for conversion from euler angles to quaternions.* (technical report TR-2005-004). Department of Computer Science & Engineering University of Minnesota, November.

Van Loan, C., 1978. Computing integrals involving the matrix exponential. *Ieee transactions on automatic control*, 23(3), pp.395–404.

Waterlinked, 2018. *Dvl documentation* [Online]. Accessed: 12 December 2022. Available from: `https://waterlinked.github.io/`.

Willumsen, A.B., Hagen, O.K. and Boge, P.N., 2007. Filtering depth measurements in underwater vehicles for improved seabed imaging. *Oceans 2007 - europe*, pp.1–6.

Appendices

Appendix A

Derivations

A.1 Process model

A.1.1 Angular velocity error

The gyro measurement equation is:

$$\boldsymbol{y}^b_{gyro} = \boldsymbol{\omega}^b + \boldsymbol{b}^b_{gyro} + \boldsymbol{w}^b_{gyro} \tag{A.1}$$

The INS angular velocity model is:

$$\hat{\boldsymbol{\omega}}^b = \boldsymbol{y}^b_{gyro} - \hat{\boldsymbol{b}}^b_{gyro} \tag{A.2}$$

The angular velocity error becomes:

$$\begin{aligned} \boldsymbol{\delta\omega}^b &= \boldsymbol{\omega}^b - \hat{\boldsymbol{\omega}}^b & \text{(A.3)} \\ &= -\boldsymbol{\delta b}^b_{gyro} - \boldsymbol{w}^b_{gyro} & \text{(A.4)} \end{aligned}$$

A.1.2 Attitude error

The quaternion differential equation is:

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes \boldsymbol{\omega}^b \tag{A.5}$$

The INS differential equation is:

$$\dot{\hat{\boldsymbol{q}}} = \frac{1}{2}\hat{\boldsymbol{q}} \otimes \hat{\boldsymbol{\omega}}^b \tag{A.6}$$

Due to the multiplicative nature of the quaternion and the product rule in differentiation, the quaternion error becomes:

$$\begin{aligned}
\dot{\boldsymbol{\delta q}} &= \dot{\boldsymbol{q}} \otimes \hat{\boldsymbol{q}}^* + \boldsymbol{q} \otimes \dot{\hat{\boldsymbol{q}}}^* \\
&= \left(\frac{1}{2}\boldsymbol{q} \otimes \boldsymbol{\omega}^b\right) \otimes \hat{\boldsymbol{q}}^* + \boldsymbol{q} \otimes \left(\frac{1}{2}\hat{\boldsymbol{q}} \otimes \hat{\boldsymbol{\omega}}^b\right)^* \\
&= \frac{1}{2}\boldsymbol{q} \otimes \boldsymbol{\omega}^b \otimes \hat{\boldsymbol{q}}^* + \frac{1}{2}\boldsymbol{q} \otimes (-\hat{\boldsymbol{\omega}}^b) \otimes \hat{\boldsymbol{q}}^* \\
&= \frac{1}{2}\boldsymbol{q} \otimes \boldsymbol{\delta\omega} \otimes \hat{\boldsymbol{q}}^* \\
&= \frac{1}{2}\boldsymbol{\delta q} \otimes \hat{\boldsymbol{q}} \otimes \boldsymbol{\delta\omega} \otimes \hat{\boldsymbol{q}}^* \\
&= \frac{1}{2}\boldsymbol{\delta q} \otimes \mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{\delta\omega}
\end{aligned} \tag{A.7}$$

Using the first order approximation: $\boldsymbol{\delta q} = \begin{bmatrix}1 & \frac{1}{2}\boldsymbol{\delta\alpha}^n\end{bmatrix}^\top$, and: $\dot{\boldsymbol{\delta q}} = \begin{bmatrix}0 & \frac{1}{2}\dot{\boldsymbol{\delta\alpha}}^n\end{bmatrix}^\top$:

$$\begin{bmatrix}0 \\ \dot{\boldsymbol{\delta\alpha}}^n\end{bmatrix} = \begin{bmatrix}1 \\ \frac{1}{2}\boldsymbol{\delta\alpha}^n\end{bmatrix} \otimes \mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{\delta\omega}^b$$

Carrying out the quaternion product and neglecting the upper row, we get:

$$\dot{\boldsymbol{\delta\alpha}}^n = \mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{\delta\omega} - \frac{1}{2}\mathbf{S}(\mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{\delta\omega})\boldsymbol{\delta\alpha}^n$$

Neglecting the second order product between $\boldsymbol{\delta\omega}$ and $\boldsymbol{\delta\alpha}^n$:

$$\dot{\boldsymbol{\delta\alpha}}^n = \mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{\delta\omega}$$

Inserting: $\boldsymbol{\delta\omega} = -\boldsymbol{\delta b}^b_{gyro} - \boldsymbol{w}^b_{gyro}$ (found in Appendix A.1.1)

$$\dot{\boldsymbol{\delta\alpha}}^n = -\mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{\delta b}^b_{gyro} - \mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{w}^b_{gyro} \tag{A.8}$$

A.1.3 Acceleration error

The measurement equation for the accelerometer is:

$$\boldsymbol{y}^b_{acc} = \boldsymbol{a}^b + \boldsymbol{b}^b_{acc} - \mathbf{R}(\boldsymbol{q})^\top\boldsymbol{g}^n + \boldsymbol{w}^b_{acc} \tag{A.9}$$

The INS acceleration model is:

$$\hat{\boldsymbol{a}}^b = \boldsymbol{y}^b_{acc} - \hat{\boldsymbol{b}}^b_{acc} + \mathbf{R}(\hat{\boldsymbol{q}})^\top\boldsymbol{g}^n \tag{A.10}$$

The linearized acceleration error thus becomes:

$$\begin{aligned}
\boldsymbol{\delta a}^b &= \boldsymbol{a}^b - \hat{\boldsymbol{a}}^b \\
&= -\boldsymbol{\delta b}^b_{acc} + (\mathbf{R}(\boldsymbol{q})^\top - \mathbf{R}(\hat{\boldsymbol{q}})^\top)\boldsymbol{g}^n - \boldsymbol{w}^b_{acc}
\end{aligned} \tag{A.11}$$

Using the approximation that: $\mathbf{R}(\boldsymbol{q})^\top \approx \mathbf{R}(\hat{\boldsymbol{q}})^\top(\boldsymbol{I}_3 - \mathbf{S}(\boldsymbol{\delta\alpha}^n))$, the acceleration error becomes:

$$\begin{aligned}
\boldsymbol{\delta a}^b &= -\boldsymbol{\delta b}^b_{acc} - \mathbf{R}(\hat{\boldsymbol{q}})^\top\mathbf{S}(\boldsymbol{\delta\alpha}^n)\boldsymbol{g}^n - \boldsymbol{w}^b_{acc} \\
&\approx \mathbf{R}(\hat{\boldsymbol{q}})^\top\mathbf{S}(\boldsymbol{g}^n)\boldsymbol{\delta\alpha}^n - \boldsymbol{\delta b}^b_{acc} - \boldsymbol{w}^b_{acc}
\end{aligned} \tag{A.12}$$

A.1.4 Velocity error

The velocity differential equation is:

$$\dot{\boldsymbol{v}}^b = \boldsymbol{a}^b - \mathbf{S}(\boldsymbol{\omega}^b)\boldsymbol{v}^b \tag{A.13}$$

The INS differential equation is:

$$\dot{\hat{\boldsymbol{v}}}^b = \hat{\boldsymbol{a}}^b - \mathbf{S}(\hat{\boldsymbol{\omega}}^b)\hat{\boldsymbol{v}}^b \tag{A.14}$$

The linearized velocity error dynamics become:

$$\begin{aligned}
\dot{\boldsymbol{\delta v}}^b &= \dot{\boldsymbol{v}}^b - \dot{\hat{\boldsymbol{v}}}^b \qquad\qquad (A.15)\\
&= \boldsymbol{\delta a}^b + \mathbf{S}(\hat{\boldsymbol{\omega}}^b)\hat{\boldsymbol{v}}^b - \mathbf{S}(\boldsymbol{\omega}^b)\boldsymbol{v}^b\\
&= \boldsymbol{\delta a}^b + \mathbf{S}(\hat{\boldsymbol{\omega}}^b)\hat{\boldsymbol{v}}^b - \mathbf{S}(\hat{\boldsymbol{\omega}}^b + \boldsymbol{\delta\omega}^b)(\hat{\boldsymbol{v}}^b + \boldsymbol{\delta v}^b)\\
&= \boldsymbol{\delta a}^b + \mathbf{S}(\hat{\boldsymbol{\omega}}^b)\hat{\boldsymbol{v}}^b - \mathbf{S}(\hat{\boldsymbol{\omega}}^b)\hat{\boldsymbol{v}}^b - \mathbf{S}(\hat{\boldsymbol{\omega}}^b)\boldsymbol{\delta v}^b - \mathbf{S}(\boldsymbol{\delta\omega}^b)\hat{\boldsymbol{v}}^b - \mathbf{S}(\hat{\boldsymbol{\delta\omega}}^b)\boldsymbol{\delta v}^b
\end{aligned}$$

Neglecting the second order product $\mathbf{S}(\hat{\boldsymbol{\delta\omega}}^b)\boldsymbol{\delta v}^b$ and flipping some cross products we get:

$$\dot{\boldsymbol{\delta v}}^b = -\mathbf{S}(\hat{\boldsymbol{\omega}}^b)\boldsymbol{\delta v}^b + \boldsymbol{\delta a}^b + \mathbf{S}(\hat{\boldsymbol{v}}^b)\boldsymbol{\delta\omega}^b$$

Inserting $\boldsymbol{\delta a}^b = \mathbf{R}(\hat{\boldsymbol{q}})^\top \mathbf{S}(\boldsymbol{g}^n)\boldsymbol{\delta\alpha}^n - \boldsymbol{\delta b}^b_{acc} - \boldsymbol{w}^b_{acc}$ (found in Appendix A.1.3), and $\boldsymbol{\delta\omega}^b = -\boldsymbol{\delta b}^b_{gyro} - \boldsymbol{w}^b_{gyro}$ (found in subsection A.1.1), we get:

$$\dot{\boldsymbol{\delta v}}^b = -\mathbf{S}(\hat{\boldsymbol{\omega}}^b)\boldsymbol{\delta v}^b + \mathbf{R}(\hat{\boldsymbol{q}})^\top \mathbf{S}(\boldsymbol{g}^n)\boldsymbol{\delta\alpha}^n - \boldsymbol{\delta b}^b_{acc} - \mathbf{S}(\hat{\boldsymbol{v}}^b)\boldsymbol{\delta b}^b_{gyro} - \boldsymbol{w}^b_{acc} - \mathbf{S}(\hat{\boldsymbol{v}}^b)\boldsymbol{w}^b_{gyro} \tag{A.16}$$

A.1.5 Position error

The position kinematics model is:

$$\dot{\boldsymbol{p}}^n = \mathbf{R}(\boldsymbol{q})\boldsymbol{v}^b \tag{A.17}$$

The INS equation is:

$$\dot{\hat{\boldsymbol{p}}}^n = \mathbf{R}(\hat{\boldsymbol{q}})\hat{\boldsymbol{v}}^b \tag{A.18}$$

The linearized position error dynamics become:

$$\begin{aligned}
\dot{\hat{\boldsymbol{\delta p}}}^n &= \dot{\boldsymbol{p}}^n - \dot{\hat{\boldsymbol{p}}}^n \qquad\qquad (A.19)\\
&= \mathbf{R}(\boldsymbol{q})\boldsymbol{v}^b - \mathbf{R}(\hat{\boldsymbol{q}})\hat{\boldsymbol{v}}^b
\end{aligned}$$

Using the approximation that $\mathbf{R}(\boldsymbol{q}) \approx (\boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{\delta\alpha}^n))\,\mathbf{R}(\hat{\boldsymbol{q}})$, we get:

$$\begin{aligned}
\dot{\hat{\boldsymbol{\delta p}}}^n &\approx \mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{\delta v}^b + \mathbf{S}(\boldsymbol{\delta\alpha}^n)\,\mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{v}^b\\
&= \mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{\delta v}^b - \mathbf{S}(\mathbf{R}(\hat{\boldsymbol{q}})(\hat{\boldsymbol{v}}^b + \boldsymbol{\delta v}^b))\boldsymbol{\delta\alpha}^n
\end{aligned}$$

Neglecting the crossproduct $\mathbf{S}(\mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{\delta v}^b)\boldsymbol{\delta\alpha}^n$, we get:

$$\dot{\hat{\boldsymbol{\delta p}}}^n = \mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{\delta v}^b - \mathbf{S}(\mathbf{R}(\hat{\boldsymbol{q}})\hat{\boldsymbol{v}}^b)\boldsymbol{\delta\alpha}^n \tag{A.20}$$

A.1.6 Pressure bias error

The pressure biases are modeled as a linear system on the form:

$$\dot{\boldsymbol{b}}_{pbe} = \boldsymbol{A}_{pbe}\boldsymbol{b}_{pbe} + \boldsymbol{G}_{pbe}\boldsymbol{w}_{pbe} \tag{A.21}$$

The pressure bias estimator is defined as:

$$\dot{\hat{\boldsymbol{b}}}_{pbe} = \hat{\boldsymbol{A}}_{pbe}\hat{\boldsymbol{b}}_{pbe} \tag{A.22}$$

The pressure bias error becomes:

$$\begin{aligned}\dot{\boldsymbol{\delta b}}_{pbe} &= \dot{\boldsymbol{b}}_{pbe} - \dot{\hat{\boldsymbol{b}}}_{pbe} &\text{(A.23)}\\ &\approx \hat{\boldsymbol{A}}_{pbe}\boldsymbol{\delta b}_{pbe} + \hat{\boldsymbol{G}}_{pbe}\boldsymbol{w}_{pbe} &\text{(A.24)}\end{aligned}$$

, where the matrices $\boldsymbol{A}_{pbe}$ and $\boldsymbol{G}_{pbe}$ are replaced by $\hat{\boldsymbol{A}}_{pbe}$ and $\hat{\boldsymbol{G}}_{pbe}$ to include the estimate of the peak frequency.

A.2 Measurement innovations and jacobians

A.2.1 USBL

The USBL measurement equation is:

$$\boldsymbol{y}^n_{usbl} = \boldsymbol{p}^n + \mathbf{R}(\boldsymbol{q})\boldsymbol{r}^b_{tp/b} + \boldsymbol{\eta}^n_{usbl} \tag{A.25}$$

The predicted measurement is:

$$\hat{\boldsymbol{y}}^n_{usbl} = \hat{\boldsymbol{p}}^n + \mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{r}^b_{tp/b} \tag{A.26}$$

The innovation becomes:

$$\begin{aligned}\boldsymbol{\delta y}^n_{usbl} &= \boldsymbol{y}^n_{usbl} - \hat{\boldsymbol{y}}^n_{usbl} &\text{(A.27)}\\ &= \boldsymbol{\delta p}^n + (\mathbf{R}(\boldsymbol{q}) - \mathbf{R}(\hat{\boldsymbol{q}}))\boldsymbol{r}^b_{tp/b} + \boldsymbol{\eta}^n_{usbl}\end{aligned}$$

Using the approximation: $\mathbf{R}(\boldsymbol{q}) \approx (\boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{\delta\alpha}^n))\,\mathbf{R}(\hat{\boldsymbol{q}})$

$$\begin{aligned}\boldsymbol{\delta y}^n_{usbl} &\approx \boldsymbol{\delta p}^n + \mathbf{S}(\boldsymbol{\delta\alpha}^n)\,\mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{r}^b_{tp/b} + \boldsymbol{\eta}^n_{usbl}\\ &= \boldsymbol{\delta p}^n - \mathbf{S}(\mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{r}^b_{tp/b})\boldsymbol{\delta\alpha}^n + \boldsymbol{\eta}^n_{usbl}\\ &= \underbrace{\begin{bmatrix}\boldsymbol{I}_3 & \mathbf{0}_3 & -\mathbf{S}(\mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{r}^b_{tp/b}) & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3\end{bmatrix}}_{\boldsymbol{H}_{usbl}(\hat{\boldsymbol{x}})}\boldsymbol{\delta x} + \boldsymbol{\eta}^n_{usbl} &\text{(A.28)}\end{aligned}$$

A.2.2 Pressure sensor

The pressure sensor measurement is modeled as:

$$y_{ps} = b_{atm} + b_{lf} + b_{wf} + \rho(\boldsymbol{g}^n)^\top(\boldsymbol{p}^n_{b/n} + \mathbf{R}(\boldsymbol{q})\boldsymbol{r}^b_{ps/b}) + \eta_{ps} \tag{A.29}$$

The predicted measurement becomes:

$$\hat{y}_{ps} = b_{atm} + \hat{b}_{lf} + \hat{b}_{wf} + \rho \boldsymbol{g}^{n\top}(\hat{\boldsymbol{p}}^n_{b/n} + \mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{r}^b_{ps/b}) \tag{A.30}$$

The innovation is:

$$\begin{aligned} \delta y_{ps} &= y_{ps} - \hat{y}_{ps} \\ &= \delta b_{lf} + \delta b_{wf} + \rho(\boldsymbol{g}^n)^\top \boldsymbol{\delta p}^n + \rho(\boldsymbol{g}^n)^\top(\mathbf{R}(\boldsymbol{q}) - \mathbf{R}(\hat{\boldsymbol{q}}))\boldsymbol{r}^b_{ps/b} + \eta_{ps} \end{aligned} \tag{A.31}$$

Using the approximation: $\mathbf{R}(\boldsymbol{q}) \approx (\boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{\delta\alpha}^n))\,\mathbf{R}(\hat{\boldsymbol{q}})$:

$$\begin{aligned} \delta y_{ps} &\approx \delta b_{lf} + \delta b_{wf} + \rho(\boldsymbol{g}^n)^\top \boldsymbol{\delta p}^n + \rho(\boldsymbol{g}^n)^\top \mathbf{S}(\boldsymbol{\delta\alpha}^n)\,\mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{r}^b_{ps/b} + \eta_{ps} \\ &= \rho(\boldsymbol{g}^n)^\top \boldsymbol{\delta p}^n - \rho(\boldsymbol{g}^n)^\top \mathbf{S}(\mathbf{R}(\hat{\boldsymbol{q}})\boldsymbol{r}^b_{ps/b})\boldsymbol{\delta\alpha}^n + \delta b_{lf} + \delta b_{wf} + \eta_{ps} \\ &= \underbrace{\begin{bmatrix}\rho(\boldsymbol{g}^n)^\top & \mathbf{0_{1x3}} & -\rho(\boldsymbol{g}^n)^\top \mathbf{S}(\mathbf{R}(\hat{\boldsymbol{q}})\bar{\boldsymbol{r}}^b_{ps}) & \mathbf{0_{1x3}} & \mathbf{0_{1x3}} & \begin{bmatrix}1 & 1 & 0\end{bmatrix}\end{bmatrix}}_{\boldsymbol{H}_{ps}(\hat{\boldsymbol{x}})} \boldsymbol{\delta x} + \eta^n_{ps} \end{aligned} \tag{A.32}$$

A.2.3 DVL

The DVL measurement equation is:

$$\boldsymbol{D}\boldsymbol{y}^d_{dvl} = \boldsymbol{v}^b_{b/n} - \mathbf{S}(\boldsymbol{r}^b_{d/b})\boldsymbol{\omega}^b + \boldsymbol{D}\boldsymbol{\eta}^d_{dvl} \tag{A.33}$$

The predicted measurement is:

$$\hat{\boldsymbol{y}}^b_{dvl} = \hat{\boldsymbol{v}}^b_{b/n} - \mathbf{S}(\boldsymbol{r}^b_{d/b})\hat{\boldsymbol{\omega}}^b_{b/n} \tag{A.34}$$

The innovation becomes:

$$\begin{aligned} \boldsymbol{\delta y}^b_{dvl} &= \boldsymbol{D}\boldsymbol{y}^d_{dvl} - \hat{\boldsymbol{y}}^b_{dvl} \\ &= \boldsymbol{\delta v}^b - \mathbf{S}(\boldsymbol{r}^b_{d/b})\boldsymbol{\delta\omega}^b + \boldsymbol{D}\boldsymbol{\eta}^d_{dvl} \end{aligned} \tag{A.35}$$

Inserting the angular velocity error $\boldsymbol{\delta\omega}^b = -\boldsymbol{\delta b}^b_{gyro} - \boldsymbol{w}^b_{gyro}$ (found in Appendix A.1.1):

$$\begin{aligned} \boldsymbol{\delta y}^b_{dvl} &= \boldsymbol{\delta v}^b + \mathbf{S}(\boldsymbol{r}^b_{d/b})\boldsymbol{\delta b}^b_{gyro} + \boldsymbol{D}\boldsymbol{\eta}^d_{dvl} + \mathbf{S}(\boldsymbol{r}^b_{d/b})\boldsymbol{\eta}^b_{gyro} \\ &= \underbrace{\begin{bmatrix}\mathbf{0}_3 & \boldsymbol{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{S}(\boldsymbol{r}^b_{d/b}) & \mathbf{0}_3\end{bmatrix}}_{\boldsymbol{H}_{dvl}} \boldsymbol{\delta x} + \boldsymbol{D}\boldsymbol{\eta}^d_{dvl} + \mathbf{S}(\boldsymbol{r}^b_{d/b})\boldsymbol{\eta}^b_{gyro} \end{aligned} \tag{A.36}$$

A.2.4 AHRS

The AHRS measurement equation is:

$$\boldsymbol{Y}_{ahrs} \approx (\boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{\eta}^n_{ahrs}))\,\mathbf{R}(\boldsymbol{q}) \tag{A.37}$$

The predicted measurement is:

$$\hat{\boldsymbol{Y}}_{ahrs} = \mathbf{R}(\hat{\boldsymbol{q}}) \tag{A.38}$$

Since rotation matrices are multiplicative, the innovation must become the measurement multiplied by the inverse of the predicted measurement:

$$\begin{aligned} \boldsymbol{\delta Y}_{ahrs} &= \boldsymbol{Y}_{ahrs}\hat{\boldsymbol{Y}}^\top_{ahrs} \\ &= (\boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{\eta}^n_{ahrs}))\,\mathbf{R}(\boldsymbol{q})\,\mathbf{R}(\hat{\boldsymbol{q}})^\top \end{aligned}$$

Using the approximation that: $\mathbf{R}(\hat{\boldsymbol{q}})^\top \approx \mathbf{R}(\boldsymbol{q})^\top(\boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{\delta\alpha}))$, we get:

$$\begin{aligned}\boldsymbol{\delta Y}_{ahrs} &\approx (\boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{\eta}^n_{ahrs}))\,\mathbf{R}(\boldsymbol{q})\,\mathbf{R}(\boldsymbol{q})^\top(\boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{\delta\alpha})) \\ &= \boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{\delta\alpha}) + \mathbf{S}(\boldsymbol{\eta}_{ahrs}) + \mathbf{S}(\boldsymbol{\delta\alpha})\,\mathbf{S}(\boldsymbol{\eta}^n_{ahrs})\end{aligned}$$

Assuming that the product $\mathbf{S}(\boldsymbol{\delta\alpha})\,\mathbf{S}(\boldsymbol{\eta}^n_{ahrs})$ is negligible:

$$\boldsymbol{\delta Y}_{ahrs} \approx \boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{\delta\alpha} + \boldsymbol{\eta}^n_{ahrs})$$

Realizing that the information in $\boldsymbol{\delta Y}_{ahrs}$ is approximately skew-symmetric, we can simplify further by:

$$\begin{aligned}\boldsymbol{\delta y}_{ahrs} &= \mathrm{vex}\,(\boldsymbol{\delta Y}_{ahrs}) && \text{(A.39)} \\ &= \boldsymbol{\delta\alpha} + \boldsymbol{\eta}^n_{ahrs} \\ &= \underbrace{\begin{bmatrix}\mathbf{0}_3 & \mathbf{0}_3 & \boldsymbol{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3\end{bmatrix}}_{\boldsymbol{H}_{ahrs}}\boldsymbol{\delta x} + \boldsymbol{\eta}^n_{ahrs} && \text{(A.40)}\end{aligned}$$

A.3 Attitude jacobian

The goal of this section is to derive an attitude Jacobian that can be used to convert attitude errors between Euler angles and rotation vectors represented in NED. The approach is inspired by Trawny and Roumeliotis (2005). Let $\boldsymbol{R}$ be a rotation matrix describing the true rotation and $\hat{\boldsymbol{R}}$ be our estimate of that rotation. When the error between these matrices is suffiently small, it can be approximated using a rotation vector in NED, $\boldsymbol{\delta\alpha}^n$ like:

$$\boldsymbol{R} \approx \underbrace{(\boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{\delta\alpha}^n))}_{\triangleq \boldsymbol{\delta R}_\alpha}\hat{\boldsymbol{R}}$$

When using Euler angle representation, the same approximation can be used for the individual rotations:

$$\begin{aligned}\mathbf{R}(\phi) &\approx \mathbf{R}(\hat{\phi})\underbrace{(\boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{e}_x\delta\phi))}_{\triangleq \boldsymbol{\delta R}_\phi} \\ \mathbf{R}(\theta) &\approx \mathbf{R}(\hat{\theta})\underbrace{(\boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{e}_y\delta\theta))}_{\triangleq \boldsymbol{\delta R}_\theta} \\ \mathbf{R}(\psi) &\approx \mathbf{R}(\hat{\psi})\underbrace{(\boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{e}_z\delta\psi))}_{\triangleq \boldsymbol{\delta R}_\psi}\end{aligned}$$

, where $\boldsymbol{e}_x$, $\boldsymbol{e}_y$, $\boldsymbol{e}_z$ are the x, y, and z unit vectors. The Euler angles are defined such that:

$$\boldsymbol{R} = \mathbf{R}(\psi)\,\mathbf{R}(\theta)\,\mathbf{R}(\phi)$$

Inserting the previous equations:

$$\boldsymbol{\delta R}_\alpha\hat{\boldsymbol{R}} = \mathbf{R}(\hat{\psi})\boldsymbol{\delta R}_\psi\,\mathbf{R}(\hat{\theta})\boldsymbol{\delta R}_\theta\,\mathbf{R}(\hat{\phi})\boldsymbol{\delta R}_\phi$$

Rearranging:

$$\boldsymbol{\delta R}_\alpha = \mathbf{R}(\hat{\psi})\boldsymbol{\delta R}_\psi\,\mathbf{R}(\hat{\theta})\boldsymbol{\delta R}_\theta\,\mathbf{R}(\hat{\phi})\boldsymbol{\delta R}_\phi\hat{\boldsymbol{R}}^\top$$

Inserting: $\hat{\boldsymbol{R}} = \mathbf{R}(\hat{\psi})\,\mathbf{R}(\hat{\theta})\,\mathbf{R}(\hat{\phi})$, we get:

$$\boldsymbol{\delta R}_\alpha = \mathbf{R}(\hat{\psi})\boldsymbol{\delta R}_\psi\,\mathbf{R}(\hat{\theta})\boldsymbol{\delta R}_\theta\,\mathbf{R}(\hat{\phi})\boldsymbol{\delta R}_\phi\,\mathbf{R}(\hat{\phi})^\top\mathbf{R}(\hat{\theta})^\top\mathbf{R}(\hat{\psi})^\top$$

Expanding using identity matrices:

$$\begin{aligned}\boldsymbol{\delta R}_\alpha &= \mathbf{R}(\hat{\psi})\boldsymbol{\delta R}_\psi\,\underbrace{\mathbf{R}(\hat{\psi})^\top\mathbf{R}(\hat{\psi})}_{\boldsymbol{I}_3}\,\mathbf{R}(\hat{\theta})\boldsymbol{\delta R}_\theta\,\underbrace{\mathbf{R}(\hat{\theta})^\top\mathbf{R}(\hat{\psi})^\top\mathbf{R}(\hat{\psi})\,\mathbf{R}(\hat{\theta})}_{\boldsymbol{I}_3}\,\mathbf{R}(\hat{\phi})\boldsymbol{\delta R}_\phi\,\mathbf{R}(\hat{\phi})^\top\mathbf{R}(\hat{\theta})^\top\mathbf{R}(\hat{\psi})^\top \\ &= \underbrace{\mathbf{R}(\hat{\psi})\boldsymbol{\delta R}_\psi\,\mathbf{R}(\hat{\psi})^\top}_{\boldsymbol{\Psi}}\underbrace{\mathbf{R}(\hat{\psi})\,\mathbf{R}(\hat{\theta})\boldsymbol{\delta R}_\theta\,\mathbf{R}(\hat{\theta})^\top\mathbf{R}(\hat{\psi})^\top}_{\boldsymbol{\Theta}}\underbrace{\mathbf{R}(\hat{\psi})\,\mathbf{R}(\hat{\theta})\,\mathbf{R}(\hat{\phi})\boldsymbol{\delta R}_\phi\,\mathbf{R}(\hat{\phi})^\top\mathbf{R}(\hat{\theta})^\top\mathbf{R}(\hat{\psi})^\top}_{\boldsymbol{\Phi}}\end{aligned}$$

The individual factors can be simplified after inserting the $\boldsymbol{\delta}$ factors:

$$\begin{aligned}\boldsymbol{\Phi} &= \mathbf{R}(\hat{\psi})\,\mathbf{R}(\hat{\theta})\,\mathbf{R}(\hat{\phi})(\boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{e}_x\delta\phi))\,\mathbf{R}(\hat{\phi})^\top\mathbf{R}(\hat{\theta})^\top\mathbf{R}(\hat{\psi})^\top \\ &= \boldsymbol{I}_3 + \mathbf{S}(\mathbf{R}(\hat{\psi})\,\mathbf{R}(\hat{\theta})\,\mathbf{R}(\hat{\phi})\boldsymbol{e}_x\delta\phi) \\ &= \boldsymbol{I}_3 + \mathbf{S}(\mathbf{R}(\hat{\psi})\,\mathbf{R}(\hat{\theta})\boldsymbol{e}_x\delta\phi) \\ \boldsymbol{\Theta} &= \mathbf{R}(\hat{\psi})\,\mathbf{R}(\hat{\theta})(\boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{e}_y\delta\theta))\,\mathbf{R}(\hat{\theta})^\top\mathbf{R}(\hat{\psi})^\top \\ &= \boldsymbol{I}_3 + \mathbf{S}(\mathbf{R}(\hat{\psi})\,\mathbf{R}(\hat{\theta})\boldsymbol{e}_y\delta\theta) \\ &= \boldsymbol{I}_3 + \mathbf{S}(\mathbf{R}(\hat{\psi})\boldsymbol{e}_y\delta\theta) \\ \boldsymbol{\Psi} &= \mathbf{R}(\hat{\psi})(\boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{e}_z\delta\psi))\,\mathbf{R}(\hat{\psi}) \\ &= \boldsymbol{I}_3 + \mathbf{S}(\mathbf{R}(\hat{\psi})\boldsymbol{e}_z\delta\psi) \\ &= \boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{e}_z\delta\psi)\end{aligned}$$

Returning to $\boldsymbol{\delta R}_\alpha$:

$$\boldsymbol{\delta R}_\alpha = \boldsymbol{\Psi\Theta\Phi}$$

Inserting:

$$\boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{\delta\alpha}^n) = (\boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{e}_z\delta\psi))(\boldsymbol{I}_3 + \mathbf{S}(\mathbf{R}(\hat{\psi})\boldsymbol{e}_y\delta\theta))(\boldsymbol{I}_3 + \mathbf{S}(\mathbf{R}(\hat{\psi})\,\mathbf{R}(\hat{\theta})\boldsymbol{e}_x\delta\phi))$$

Multiplying and only keeping first order terms:

$$\begin{aligned}\boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{\delta\alpha}^n) &\approx \boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{e}_z\delta\psi) + \mathbf{S}(\mathbf{R}(\hat{\psi})\boldsymbol{e}_y\delta\theta) + \mathbf{S}(\mathbf{R}(\hat{\psi})\,\mathbf{R}(\hat{\theta})\boldsymbol{e}_x\delta\phi) \\ &= \boldsymbol{I}_3 + \mathbf{S}(\boldsymbol{e}_z\delta\psi + \mathbf{R}(\hat{\psi})\boldsymbol{e}_y\delta\theta + \mathbf{R}(\hat{\psi})\,\mathbf{R}(\hat{\theta})\boldsymbol{e}_x\delta\phi)\end{aligned}$$

It is now clear that:

$$\begin{aligned}\boldsymbol{\delta\alpha}^n &= \boldsymbol{e}_z\delta\psi + \mathbf{R}(\hat{\psi})\boldsymbol{e}_y\delta\theta + \mathbf{R}(\hat{\psi})\,\mathbf{R}(\hat{\theta})\boldsymbol{e}_x\delta\phi \\ &= \begin{bmatrix}\mathbf{R}(\hat{\psi})\,\mathbf{R}(\hat{\theta})\boldsymbol{e}_x & \mathbf{R}(\hat{\psi})\boldsymbol{e}_y & \boldsymbol{e}_z\end{bmatrix}\begin{bmatrix}\delta\phi \\ \delta\theta \\ \delta\psi\end{bmatrix}\end{aligned}$$

$$\boldsymbol{\delta\alpha}^n = \underbrace{\begin{bmatrix}\cos\hat{\psi}\cos\hat{\theta} & -\sin\hat{\psi} & 0 \\ \sin\hat{\psi}\cos\hat{\theta} & \cos\hat{\psi} & 0 \\ -\sin\hat{\theta} & 0 & 1\end{bmatrix}}_{\boldsymbol{T}_{att}(\hat{\theta},\hat{\psi})}\begin{bmatrix}\delta\phi \\ \delta\theta \\ \delta\psi\end{bmatrix} \tag{A.41}$$

The jacobian $\boldsymbol{T}_{att}$ can also be inverted for the inverse transformation:

$$\begin{bmatrix}\delta\phi \\ \delta\theta \\ \delta\psi\end{bmatrix} = \underbrace{\begin{bmatrix}\frac{\cos\hat{\psi}}{\cos\hat{\theta}} & \frac{\sin\hat{\psi}}{\cos\hat{\theta}} & 0 \\ -\sin\hat{\psi} & \cos\hat{\psi} & 0 \\ \frac{\cos\hat{\psi}\sin\hat{\theta}}{\cos\hat{\theta}} & \frac{\sin\hat{\psi}\sin\hat{\theta}}{\cos\hat{\theta}} & 1\end{bmatrix}}_{\boldsymbol{T}_{att}^{-1}(\hat{\theta},\hat{\psi})}\boldsymbol{\delta\alpha}^n \tag{A.42}$$

As you would expect when working with Euler angles, the inverse Jacobian is not defined for $\hat{\theta} = \pm 90^\circ$