NTNU
Norwegian University of
Science and Technology

Proposal and comparison of an eXogenous Kalman Filter and a Particle Filter for use with ROV thruster models

Herman Øen Gustavsen

Marine Technology
Submission date: June 2018
Supervisor: Ingrid Schjølberg, IMT

Norwegian University of Science and Technology
Department of Marine Technology

NTNU Trondheim
Norwegian University of Science and Technology
Department of Marine Technology

MASTER THESIS IN MARINE CYBERNETICS
SPRING 2018
FOR
STUD. TECH. HERMAN ØEN GUSTAVSEN

Proposal and comparison of an eXogenous Kalman Filter and a Particle Filter for use with ROV thruster models

Work description

The field of underwater robotics are moving towards more autonomous and automatic operations. With human interference kept at a minimum, strict requirements are put on the underwater vehicles ability to interpret its surroundings. In such setting it is crucial that the vehicle is able to determine its current position, attitude and velocity at any given time. These states can be observed either directly or indirectly through a given set of measurements. A common method is to use a mathematical model of the vehicle to predict the future states, and then correct the predictions by comparing them to other measurement sources.

This thesis aims to further investigate model based state estimators, with emphasis on methods that are both more robust and more accurate than the methods used for this purpose today. On basis of theoretical findings, two model based state estimators are proposed in this thesis. The proposed state estimators are tested, compared and verified in simulations. Experimental data are collected from experiments conducted in the NTNU Marine Cybernetics Laboratory. These data are used for offline testing.

Scope of work

1. Literature review of topics relevant for the thesis work. The main focus is observer design and state estimation for underwater vehicles.
   (a) Investigate and analyse different methods for implementation of the recursive Bayesian estimator.
   (b) Investigate how the methods in a) have been applied for underwater applications.
2. Re-design and propose your own observer designs on basis of the findings in the literature review. The proposed designs should be based on ROV thruster models.
3. Develop a simulator for testing of the estimation filters.

4. Perform simulations on real data.
5. Discuss and compare results from simulations and experimental work.
6. Write thesis on basis of theoretical findings and results.

Supervisor: Ingrid Schjølberg

Preface

This thesis is written by Herman Øen Gustavsen as part of a Master's degree at the Department of Marine Technology, at the Norwegian University of Science and Technology (NTNU) in Trondheim. The research summarized in this report is conducted in the time span from January to June 2018. The main motivation for the thesis have been to look further into state estimation of underwater vehicles. This is a field of study that is still not fully explored and understood. The reader of the thesis is assumed to have knowledge of both control engineering and marine technology.

I would like to express my gratitude to Professor Ingrid Schjølberg for her encouragement, and for always being available for questions and guidance. I would also like to thank Bård Nagy Stovner and Stian Skaalvik Sandøy for helping me out with various theoretical questions that have appeared when working on the thesis. A special thanks goes to Mikkel Cornelius Nielsen for his time spent preparing the equipment in the MC-lab. Further, I would also like to thank Bent O. Arnesen for providing me with the ROV-model used in Simulink.

Finally, I would like to thank friends and family for always being supportive. Last, but not least, thank you Guro for your patience and for always making my days complete.

Summary

Robust and accurate observer design is a prerequisite for safe and efficient underwater navigation. Especially with today's increased focus on automation and autonomy. In this thesis two model based observer designs for underwater vehicles are proposed and compared. One is based on the eXogenous Kalman Filter (XKF) and the other is based on the Particle Filter (PF). The designs are also compared to the Extended Kalman Filter (EKF) which is one of the standard observers used today. The drawback of the EKF is the lack of global stability due to a destabilising feedback loop providing linearization points. In the XKF this problem is solved by instead providing linearization points from a globally stable auxiliary state estimator . In the proposed XKF design, a Nonlinear passive Observer (NLO) is implemented for this purpose. The PF however uses the mathematical model directly to generate a number of plausible state hypotheses. Only the most plausible hypotheses makes up the final state estimates. Using the model directly, i.e. without linearization, theoretically makes the PF a more accurate observer than the EKF and the XKF. Albeit global stability cannot be proven for the PF either.

A simulator was developed in Matlab/Simulink to evaluate and compare the filters. In simulations it was assumed that noisy measurements of thrust were available for use with a mathematical ROV model. In addition, it was assumed that noisy measurements of position and heading were available from other measurement sources. Experimental data were also gathered from physical experiments using a ROV in the NTNU MC-lab. While the proposed observers were designed with the intention of using measured thrust as input, such measurements were not available with the equipment in the MC-lab. Hence, measurements from an Inertial Measurement Unit (IMU) were used as a replacement of the thruster model. As a consequence only the PF was tested using experimental data.

Simulations shows that the EKF and XKF performs relatively similar when both filters are given correct initial values. In the case of inaccurate initial values the differences are larger. As expected, the EKF diverges if the initial values are too erroneous, while the XKF is always able to converge. In cases where both filters are able to converge, the XKF has much faster convergence rates. The XKF gives slightly longer computational times, but its great stability properties makes it the preferable filter compared to the EKF. The PF proves to be the most accurate observer, and it has the fastest convergence rates when the initial values are erroneous. Using the PF with experimental data verifies that the implementation suggested in the thesis works as intended also outside the simulator. The big drawback of the PF is however that it uses approximately 20 times longer time than the Kalman filters. In addition the PF may diverge if not tuned properly. Taken everything into consideration the XKF is considered the best observer for autonomous vehicles.

Sammendrag

En robust og presis tilstandsestimator er en forutsetning for sikker og effektiv undervannsnavigering. Spesielt med tanke på det stadig økende fokuset på autonomi og automatisering. I denne oppgaven er det derfor foreslått to modellbaserte tilstandsestimatorer for undervannsfartøyer. Den ene estimatoren baserer seg på teorien bak eksogent Kalman filter (eXogenous Kalman Filter / XKF), mens den andre baserer seg på partikkelfilteret (PF). Begge estimatorene er sammenliknet med hverandre, og de er også sammenliknet med det utvidede Kalman filteret (Extended Kalman Filter / EKF). EKF er en av mest brukte tilstandsestimatorene i dag, men ulempen med denne tilstandsestimatoren er at den har en tilbakekoblingssløyfe som genererer lineariseringspunkter for filteret. Tilbakekoblingen gjør at global stabilitet ikke er mulig å oppnå. Dette problemet er løst i XKF ved at lineariseringspunktet i stedet kommer fra en hjelpeestimator. Denne hjelpeestimatoren er designet med global stabilitet. I XKF-designet som er foreslått i denne oppgaven, brukes et passivfilter som hjelpeestimator. I PF brukes den matematiske modellen direkte for å generere et sett med mulige tilstandshypoteser. Kun de mest sannsynlige tilstandshypotesene utgjør det endelige tilstandsestimatet. Da PF bruker den matematiske modellen direkte er det i teorien mer presist en EKF og XKF, men global stabilitet kan heller ikke oppnås med dette filteret.

For å evaluere filtrene ble det utviklet en simulator i Matlab/Simulink. I simuleringene ble det antatt at undervannsfartøyets posisjon og kurs var kjent fra støyete målinger. Det var også antatt at målinger av pådraget fra undervannsfartøyets propeller var kjent. Eksperimentelle data ble også samlet fra forsøk utført med et undervannsfartøy i NTNU sin MC-lab. I laben var det ikke mulig å måle pådraget, slik som filtrene baserer seg på. Det ble derfor samlet data fra et akselerometer og et gyroskop i stedet. Som en konsekvens av dette var det kun mulig å bruke eksperimentelle data med PF.

Simuleringene viser at EKF og XKF har relativt lik ytelse når begge filterne er gitt korrekte initialverdier. Ved ukorrekte initialverdier er forskjellene derimot større. Som forventet divergerer EKF hvis feilen i initialverdier blir for stor. Dette gjelder ikke for XKF som alltid konvergerer, samme hvor stor feilen er. I tilfeller hvor begge filtrene konvergerer gjør også XKF dette mye raskere enn EKF. Det viser seg at PF er den mest nøyaktige estimatoren og det har også den raskeste konvergenstiden ved feilaktige initialverdier. Den store ulempen med dette filteret er at det bruker nesten 20 ganger lenger tid en Kalman-filtrene. Og dersom man ikke stiller inn PF riktig er det en viss sjans for at estimatene vil divergere. Alt tatt i betraktning er XKF ansett som det beste filteret for fremtidige autonome undervannsfartøyer.

Table of Contents

List of Tables

List of Figures

Abbreviations

AUV	Autonomous Underwater Vehicle
DKF	Discrete Kalman Filter
DOF	Degree Of Freedom
EKF	Extended Kalman Filter
GAS	Global Asymptotic Stability
GES	Global Exponential Stability
GNC	Guidance, Navigation and Control
IMR	Inspection, Maintenance and Repair
IMU	Inertial Measurement Unit
IS	Importance Sampling
KF	Kalman Filter
LKF	Linear Kalman Filter
LTV	Linear Time Varying
NED	North East Down
NLO	NonLinear Observer
PID	Proportional Integral Derivative
PDF	Probability Density Function
QMC	Qualisys Motion Capture
QTC	Qualisys Track Manager
ROV	Remotely Operated Vehicle
SIR	Sampling Importance Resampling
UUV	Unmanned Underwater Vehicle
XKF	eXogenous Kalman Filter

Chapter 1

Introduction

This thesis considers topics on nonlinear state estimation for underwater vehicles. This chapter presents the motivation for further exploration of this topic, recent advances of this topic, and how the thesis is organized.

1.1 Motivation and background

The field of underwater robotics has expanded at fast rates the latter years as technology has become both cheaper and better. The result is unmanned underwater vehicles (UUVs) designed for a large variety of applications. The main driving force being the UUVs capability of efficiently operating in the harsh environments of the ocean space. Both industry, military and researchers are amongst others that benefit from this technology.

Subsea Inspection, Maintenance and Repair (IMR) operations are frequently carried out in the oil and gas industry by use of Remotely Operated Vehicles (ROVs) (Henriksen et al., 2016). ROVs are a subclass of UUVs and are characterized of being tethered (Candeloro, 2016). The physical link between the ROV and the surface vessel allows for power- and data transfer. And it is essential for communication with the human operator controlling the ROV with a joystick. However, having a human in the control loop comes with certain disadvantages. A human needs lots of training to gain experience, there is a need for rest at certain time intervals, having a human on-site is expensive, and human errors are unavoidable. Huge costs are also associated with the support vessel which can cost hundreds of thousands of dollars each day depending on vessel size and zone of operation (Schjølberg and Utne, 2015). In addition the current weather condition may prevent the support vessel from operation.

Due to the mentioned reasons, the focus on automation and autonomy in underwater applications are increasing. This imposes strict requirements to the systems implemented on the UUV in terms of safety, robustness and accuracy. In general, the motion control system consists of three independent blocks denoted *Guidance*, *Navigation* and *Control* (GNC) (Fossen, 2011). Figure 1.1 shows the signal flow between these blocks.

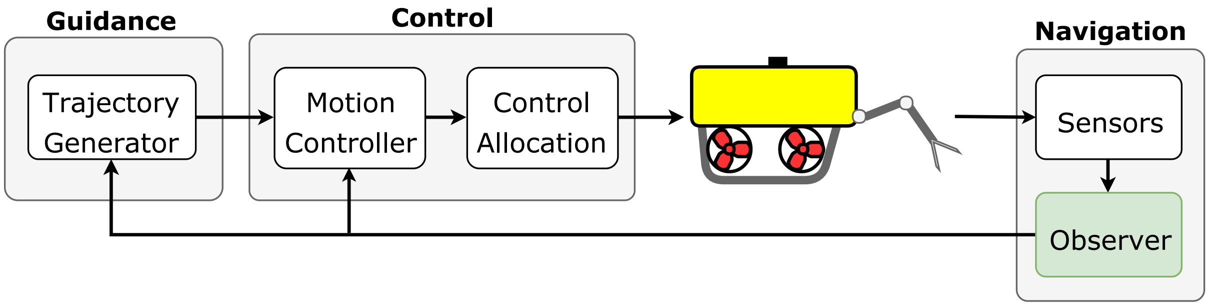


Figure 1.1: GNC

Each block in the GNC system serves a specific task and a short summary of their function is provided below. A more detailed explanation is found in (Fossen, 2011).

- **Navigation**: The task of this block is to estimate the current state of the vehicle based on measurements from different sensors. For underwater vehicles the main states of interest are position, attitude and velocity. However, in many cases states cannot be measured directly, and all measurements are in addition contaminated with noise. The purpose of the observer is to combine different measurement sources, and ultimately suppress noise and estimate the states that are not directly observable.

- **Guidance**: With the *current* state of the vessel known from the navigation block, this block calculates how to bring the vehicle to the *desired* state. Hence, desired positions, velocities and accelerations are calculated in this block.

- **Control**: The motion controller calculates the necessary control forces and moments necessary to bring the vessel to the desired positions, velocities and accelerations found in the guidance block. The control allocation decides how the control forces are best distributed between the thrusters on the vessel.

Based on the GNC system it is evident that every decision and movement made by an autonomous vehicle are based on the systems own perception of its surroundings. Hence, both safety and the degree of accuracy are strongly dependent of how well the true system state is known. The main focus in this thesis is therefore to look further into the observer part of the navigation system. This is labeled with colour in figure 1.1.

The work in this thesis assumes position and attitude are known directly through noisy measurements, while velocities are estimated using a mathematical model together with noisy measurements of the thruster output. Regardless of what method is chosen to estimate the states, the common factor is that all filters have access to the exact same information. Hence, the performance of the observer is a result of how well it is able to extract useful information from raw data. In addition, constraints in form of time consumption and available computer power are always present. One obvious constraint for filters to be used in Autonomous Underwater Vehicles (AUV), and many other applications, is the ability to run in real-time. For this reason only recursive algorithms are considered in this thesis. The goal is to investigate methods that are both more robust and more accurate than the industry standard today, which is the Extended Kalman Filter (EKF).

1.2 Previous work

Due to the importance of having accurate state estimates of the UUV, the field of observer design is widely investigated. Traditional estimation methods for the nonlinear nature of UUVs are often based on the EKF. This method approximate nonlinear models using linearization, and being a stochastic estimator makes it handle noise well. However, it comes without guarantees on stability. Another method that is widely developed is the NonLinear passive Observer (NLO) which avoids linearization and usually comes with great stability properties. These estimators typically assumes a deterministic relation between measurements and states. Hence, no analysis of noise in sense of minimum variance etc. can be made. See (Candeloro et al., 2012a) for more details on NLOs for ROVs. In (Candeloro et al., 2012b) both methods were implemented and tested using a ROV. The results reviled that *the EKF gave the best performance in terms of estimation and control error, while the passive filter resulted to be highly sensible on the parameters settings and less accurate*.

Recent publications suggests that the two methods can be combined in order to obtain the best qualities from both. This method is referred to as the eXogenous Kalman Filter (XKF) (Johansen and Fossen, 2017). Using this implementation, the lack of stability in the EKF is solved by making the EKF inherit the stability properties of an auxiliary estimator. The main focus when designing the auxiliary estimator is to make sure it has great convergence capabilities, preferably global. No specific requirements are put on the optimally or sensitivity with respect to noise (Johansen and Fossen, 2017). This endorse for use of the NLO as an auxiliary estimator. Implementations of the XKF for use with underwater vehicles using measurements from Inertial Measurement Units and hydroacoustic sensor networks may be found in (Stovner, 2018).

With the growth in computational power the last decades, more computationally expensive statistical methods has been made possible. This entails that the estimation problem can be solved directly by use of numerical Monte Carlo methods. Monte Carlo methods uses sample means to estimate population means (Dunn and Shultis, 2011). The most popular Monte Carlo method for state estimation is the Particle Filter (PF). Recent advances using the PF for underwater state estimation is found in (Zhao et al., 2012). Here, a PF is implemented using a pure kinematic model of a ROV with hydroacoustic position and a doppler speed logger measurements. It is found from the conclusion of this paper that *Simulations based on real data showed the estimation from the PF to outperform that of traditional Kalman filter approaches. Handling of transients when the ROV motion went from stationary to manoeuvring was also found better with the particle filter.*

1.3 Objectives and limitations

The main objective of this thesis is to explore and investigate different model based observer designs for underwater vehicles. With the industry standard today being the EKF, the focus in this thesis is investigate observer designs that have both better stability properties and are more accurate. This involves a thorough theoretical analysis in addition to practical implementation of the theory. Relevant filters are derived and compared in simulations. The PF is also used with experimental data to verify its performance in more realistic cases.

Observers rely on data from different sensors, but in this thesis less attention is put on the source of the sensor data itself. Thus, some input data are only assumed to be available through a linear relation between measurements and states. More specific implementations may involve instruments with given characteristics that must be taken into consideration.

1.4 Contributions

The contributions of the work in this thesis is given in the list below.

- A suggestion of an XKF for use with ROV thruster models is derived.
- A suggestion of a PF for use with ROV thruster models is derived.
- The observer designs are verified using a simulator developed in Matlab/Simulink.
- The implementation of the PF is verified using experimental data.

1.5 Outline of thesis

The thesis is comprised by a total of 9 chapters. A presentation of the rest of the chapters is given below.

Chapter 2 introduces the general theory behind recursive state estimation. This includes a presentation of different recursive estimators, where some of these will be further analysed later in the thesis. Theory on how some of these state estimators have been applied in underwater applications is also included.

Chapter 3 presents all relevant mathematical models used for later simulations and evaluation of experimental data.

Chapter 4 builds upon the theory presented in chapter 2, but the focus of this chapter is general implementation and design of the EKF, XKF, and PF. This forms the basis of the observer implementation later derived for ROVs.

Chapter 5 is a continuation of the previous chapter where the specific case of EKF-, XKF-, and PF-implementation for ROVs are considered.

Chapter 6 presents relevant plots and tables from simulated scenarios.

Chapter 7 presents relevant plots and tables from the PF using experimental data.

Chapter 8 compares and discusses the results obtained in chapter 6 and 7.

Chapter 9 makes a conclusion on basis of the results and the discussions. This chapter also contains a suggestion for further work on this topic.

Chapter 2

Background theory on state estimation

State estimators can be implemented in many different ways, and the prerequisites for using one or the other depends on several factors. This chapter presents the underlying theory of state estimation and gives a introduction on different implementations of the recursive Bayesian estimator. Relevant applications for underwater applications are also presented and discussed.

2.1 Characterization of nonlinear state estimation

The goal of nonlinear filtering is to estimate the states of a dynamic system using the information available at any given time. This information comes from observations using either internal on-board sensors measuring inertial motion and relative positions, or from external sensors measuring bearing and range to the target for instance (Gustafsson, 2012). Perfect observations of system states are rarely available in practice due to uncertainty in sensor measurements and mathematical modelling. Highly accurate sensors are also associated with high costs and technical difficulties. This motivates for combination of different information sources to produce a state estimate that is more accurate than each of the information sources individually.

To be able to estimate the system states it is necessary to write the dynamical system on state space form. This is done using a set of differential equations on the form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}) \tag{2.1}$$

where t is the time, $\boldsymbol{x} \in \mathbb{R}^n$ is the state vector, $\boldsymbol{u} \in \mathbb{R}^m$ is the input to the system and $\boldsymbol{w} \in \mathbb{R}^n$ is the process noise. Using this procedure, the vector $\boldsymbol{x}$ contains values for all important properties of the system. In navigation cases for example, like in this thesis, the relation between position, velocity and acceleration is typically described by Eq. (2.1).

The mathematical model in Eq. (2.1) is not impeccable as it is only an approximation of the physics of the real system. Hence, this model alone is not enough to determine the true states of the system. In addition it is required a measurement model that relates states and observations. In its general form the measurement model is expressed as

$$\boldsymbol{y} = \boldsymbol{h}(t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}) \tag{2.2}$$

where $\boldsymbol{y} \in \mathbb{R}^p$ is the measurement vector and $\boldsymbol{v} \in \mathbb{R}^p$ is the measurement noise. Notice that the measurement vector do not necessarily have the same dimension as the state vector. This implies that some of the states are not directly measured.

With both models available, the idea of state estimation is to guess the current state of the system with the dynamic model. Then use the measurement model to correct this guess. Using this approach one seeks to find the best estimate $\hat{\boldsymbol{x}}$ of the real state of the system. As both models are prone to errors and noise, the challenge of estimation is to decide how much to rely on either model.

2.2 Stability properties of non-linear systems

The stability properties of a system plays an important role as this defines the practicable usefulness of the given system. There are many different types of stability, but generally speaking stability implies the system stays within known boundaries. This section serves to present the different stability classifications used for non-linear systems. This is based on (Khalil, 2015), and the reader is also referred to this book for an introduction on *how* to perform an stability analysis as this will not be covered here.

Consider $f(x,t)$ where $f\colon\ [0,\infty) \times D \to R^n$ is piecewise continuous in t and locally Lipschitz in x on $[0,\infty) \times D$. Here $D \subset R^n$ is a domain that contains the origin $x = 0$. The equilibrium point $x = 0$ of $\dot{x} = f(x,t)$ is

- *Stable* if for each $\epsilon > 0$, and any $t_0 \geq 0$ there is a $\delta = \delta(\epsilon, t_0) > 0$ such that

$$\|x(t_0)\| < \delta \Rightarrow \|x(t)\| < \epsilon, \quad \forall\, t \geq t_0 \tag{2.3}$$

Hence, *stability* only guarantees that the states stays bounded for all time, given any bounded initial conditions. Stronger requirements on stability must therefore be set in order to guarantee convergence to the equilibrium point. The rate of which convergence to the equilibrium occur is divided in two main classes.

- *Asymptotic stability* is obtained if the system is stable and there exists a constant $c = c(t_0)$ such that $x(t) \to 0$ as $t \to \infty$, $\forall \; \|x(t_0)\| < c$
- *Exponential stability* is obtained if if there exists positive constants c, k and λ such that

$$\|x(t)\| \leq k \, \|x(t_0)\| \, e^{-\lambda(t-t_0)}, \quad \forall \; \|x(t_0)\| < c \tag{2.4}$$

From the above definitions one sees that both asymptotic- and exponential stability guarantee convergence to the equilibrium point. Being bounded by an exponential function implies faster convergence rates. With c being a positive constant, convergence to the equilibrium point can only be guaranteed for a *local* region of attraction. If convergence can be guaranteed for any initial conditions, the region of attraction is said to be *global*. In addition to this, the stability of time-varying systems is said to be *uniform* if it is independent of t_0.

2.3 Recursive Bayesian estimation

In statistics, Bayesian theory makes it possible to use prior knowledge and observed data to more accurately model the probability of an event. This is also referred to as *conditional probability*. The idea behind this concept is to calculate the probability of an event being true, *given* that another event has occurred. Instead of events it is often convenient to talk about the probability of a hypothesis H being true, given an evidence E. In terms of Bayes' theorem this is stated mathematically as

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} \tag{2.5}$$

Here, $P(H)$ is the *prior probability*, i.e. the probability of the hypothesis being true, before the evidence is evaluated. Hence, $P(H|E)$ is the *posterior probability* as this takes the evidence into account. The term $P(E)$ is the probability of observing the evidence, while $P(E|H)$ is the probability of observing the evidence, given that hypothesis H is true.

In the case of state estimation, all observations up to and including time k are available from the set of observations $\boldsymbol{y}_{1:k} = \{y_1, y_2, \ldots, y_k\}$. In Bayes' theorem this is the evidence used to evaluate a state hypothesis. In *filtering* applications this information is used to extract information about states at time k. While in *prediction* applications this information is used to derive information about states some time $k + \tau$ in the future. To obtain a state estimate the intention is to recursively compute the filtering density and the prediction density of the system (Gustafsson, 2012). The basis of these calculations are the dynamical model and the measurement model presented in the previous section. To implement these models in the Bayesian framework, it is convenient to rewrite the models on a more general form. Thus, the dynamical model is equivalent to $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ and the measurement model is equivalent to $p(\boldsymbol{y}_k|\boldsymbol{x}_k)$.

Using Bayes rule, the recursive Bayesian filter can now be derived under two assumptions. First assumption is that states follow a first-order Markov process, i.e. future states depends on the current state only. Secondly, it is assumed that the observations are independent of the given states (Chen, 2003). The filtering/posterior density from the previous timestep $k-1$, denoted $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})$, is used as input at the current timestep k. In the first step of the recursive algorithm this is used togheter with the dynamical model to obtain the predicted density

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1}) = \int_{\mathbb{R}^n} \underbrace{p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})}_{\substack{\text{dynamic}\\ \text{model}}} \underbrace{p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{1:k-1})}_{\substack{\text{previous}\\ \text{posterior}}} d\boldsymbol{x}_{k-1} \tag{2.6}$$

The measurement model is then used to obtain the new filtering/posterior density

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) = \frac{\overbrace{p(\boldsymbol{y}_k|\boldsymbol{x}_k)}^{\substack{\text{measurement}\\ \text{model}}} \overbrace{p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1})}^{\substack{\text{prediction}\\ \text{density}}}}{\underbrace{p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1})}_{\text{normalization}}} \tag{2.7}$$

where the normalization term is given by

$$p(\boldsymbol{y}_k|\boldsymbol{y}_{1:k-1}) = \int_{\mathbb{R}^n} p(\boldsymbol{y}_k|\boldsymbol{x}_k) p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k-1}) d\boldsymbol{x}_k \tag{2.8}$$

As can be seen in the above equations, the filtering density updates the model based on measurements. This distribution is also referred to as the posterior distribution of the state vector. This distribution depends on the prediction density which is calculated from the posterior density of the previous timestep. Hence, at the start of the recursion an initial guess of $p(\boldsymbol{x}_1|\boldsymbol{y}_0)$ must be made. At each timestep the filtering density can be used to calculate standard measures such as the minimum variance estimate

$$\boldsymbol{x}_k^{MV} = \int_{\mathbb{R}^n} \boldsymbol{x}_k p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) d\boldsymbol{x}_k \tag{2.9}$$

and the corresponding covariance of this measure

$$\boldsymbol{P}_k = \int_{\mathbb{R}^n} (\boldsymbol{x}_k - \boldsymbol{x}_k^{MV})(\boldsymbol{x}_k - \boldsymbol{x}_k^{MV})^T p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) d\boldsymbol{x}_k \tag{2.10}$$

2.4 Realisation of the recursive Bayesian filter

The recursive Bayesian filter provides a general solution on state estimation from a dynamical model, when the states are partially observed through a measurement model. However, the nature of the dynamic model and the measurement model determines how to implement the Bayesian framework in practice. For this reason there are an extensive amount of different implementations to choose from. All having benefits and disadvantages. The methods range from analytical solutions when certain restrictions are fulfilled, to numerical solutions in the more general cases. In this section some of the most common methods are presented, starting with the most restricted one. By loosen the restrictions it is shown how the implementations are changed to account for this.

2.4.1 Linear Kalman Filter

The Linear Kalman Filter (LKF) was introduced in (Kalman, 1960) and laid the foundation for the most known implementation of the recursive Bayesian filter today. Due to large progress in digital computing last decades, the Kalman filter has been subject to extensive research. This has resulted in numerous extensions to the original LKF and some of these will be presented later.

As the name suggests, the LKF requires both the dynamic model $\boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w})$ and the measurement model $\boldsymbol{h}(t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v})$ to be linear functions. In addition, the LKF assumes the posterior density is Gaussian distributed at every timestep. In other words, the posterior filtering distribution is fully described through its mean and variance. Under these assumptions it is possible to benefit from a huge advantage illustrated in figure 2.1; a linear transformation of a Gaussian is also a Gaussian.

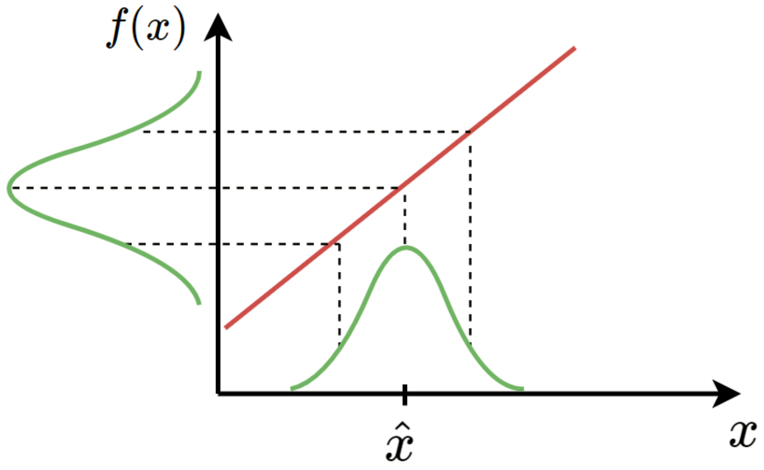


Figure 2.1: Linear transformation of Gaussian

In practice this means that when the posterior distribution from the last timestep is processed through the linear dynamic- and measurement models, the new posterior distribution is also Gaussian. Having this knowledge about the posterior is essentially what makes it possible to derive an analytical solution to this problem. It is shown that the linear Kalman filter is globally stable, and provides the optimal solution to the minimum variance estimate in Eq. (2.9) (Gustafsson, 2012). Hence, under these assumptions the LKF is not only a plausible solution, it is also the best possible solution.

For underwater vehicles the LKF is generally not applicable due to the mathematical models being non-linear. However, in (Candeloro et al., 2012b) it is suggested that the ROV system model could be divided in several modules with a proper switching strategy. By dividing the $360°$ heading interval in sectors of $10°$ it is possible to approximate the vehicle dynamics by 36 linearized system models. A benefit of this method is that each of the 36 linearized models can be calculated and stored offline. Field tests using the ROV Minerva showed that this method gave satisfactory performance.

2.4.2 Extended Kalman Filter

Linear modelling of dynamics and measurements are rarely a good approximation of real systems. Ergo, the LKF is not applicable in many cases. Fortunately this problem can be solved by assuming linear behaviour of the system dynamics around the current system state. The strategy of the EKF is therefore to make a linearization of the system model around the state estimate from the previous timestep. See (Fossen, 2011) for instance, where relevant examples for marine vessels are also included. The overall structure of the EKF is illustrated in figure 2.2.

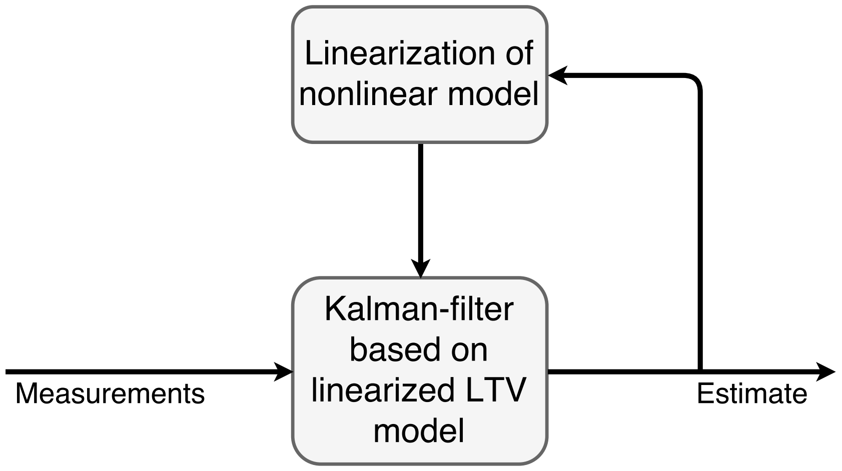

Linearization of nonlinear model
Kalman-filter based on linearized LTV model
Measurements
Estimate

Figure 2.2: The Extended Kalman Filter

By linearizing the mathematical models it is possible to use the LKF equations with nonlinear systems. However, the assumption of Gaussian noise distribution still applies. Figure 2.3 illustrates how a nonlinear function is linearized around the state estimate $\hat{x}$. In the same way as the all linear case seen in figure 2.1 earlier, the linearization ensures the Gaussian assumption is not violated.

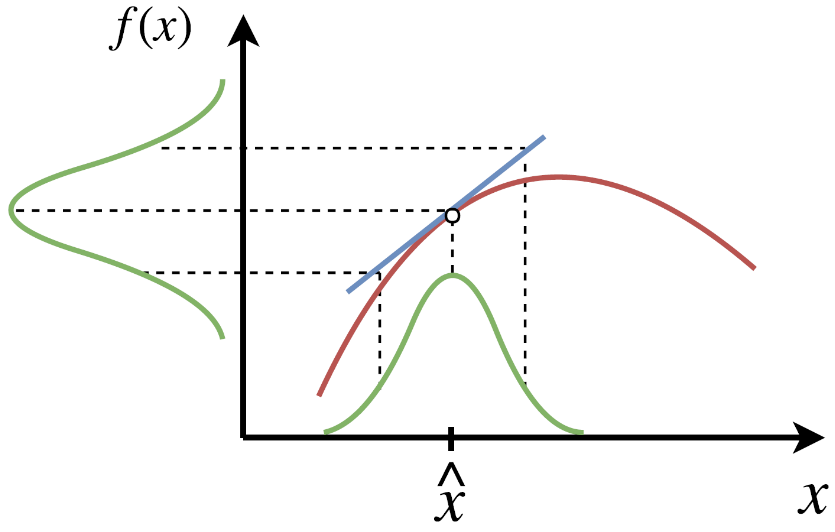


Figure 2.3: Gaussian transformation with linearized function

The EKF is probably the most used state estimator for underwater vehicles due to its simplicity and versatility. This observer was also tested in (Candeloro et al., 2012b) where it proved to give better results than the 36 sector implementation of the LKF. In this case however, all calculations must be done on-line as it is inconvenient/impossible to store system models for all possible linearization points.

2.4.3 Difficulties and drawbacks using EKF

The EKF makes the Kalman filter much more versatile by allowing use of nonlinear system models. The linearization do however come with limitations and the EKF is thus not a complete solution for nonlinear systems with Gaussian noise.

The linearization requires the Jacobian of the models to be calculated. Hence, the EKF will not work for non-linear models if it is not differentiable at every possible linearization point. Another drawback is related to the calculation of the Jacobian itself. Finding this analytically can be difficult as the derivatives may be complicated. An alternative is to calculate the Jacobian numerically, but this might lead to high computational times. The degree of nonlinearity is also essential for the success of the EKF. With severe nonlinearities a linearization is not a good approximation of the real system. This is clearly shown in figure 2.4 on the next page.

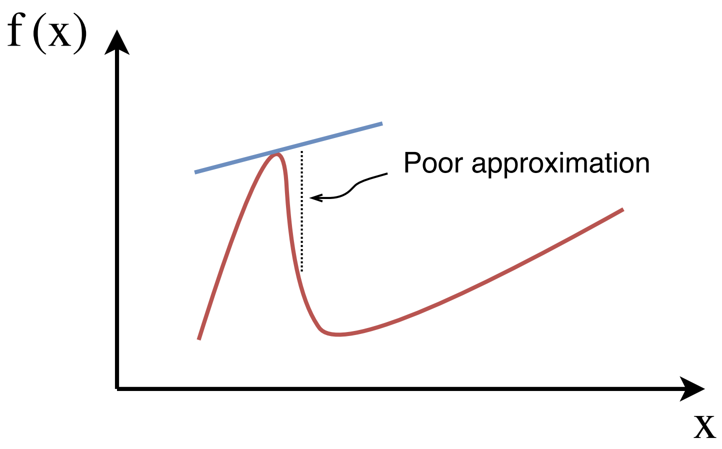


Figure 2.4: Poor linear approximation

While the LKF is shown to be the optimal state estimator, the same is not true for the EKF. This is because the linearization is only an approximation of the mathematical models and thus erroneous. The EKF is therefore said to be *near-optimal*.

When it comes to stability the EKF can only be proven to be locally stable, while the LKF is globally stable. This implies that the convergence of the EKF is sensitive to the initial state, while the LKF will converge regardless of the initial state. The sensitivity in initial conditions is due to the feedback loop seen in figure 2.2. Choosing a poor initial condition leads to a poor linearizaion of the system, which again leads to a poor new state estimate. Hence, the feedback loop is unstable and the EKF may have convergence problems due to this. How to solve this problem is discussed in the next section.

2.4.4 The eXogenous Kalman Filter

With the EKF linearization is only possible around the previous state estimate as there is no other source to feasible linearization points. This might lead to instability and for this reason the EKF comes without guarantees on stability. Avoiding the destabilising feedback-loop would ergo be an great advantage, and this problem is solved using the XKF.

The structure of the XKF is shown i figure 2.5. Compared to the structure of the EKF shown in figure 2.2 one sees that the only difference is how the estimates to linearize about are generated. While the EKF uses its own estimates to linearize about, the XKF uses estimates from an *auxiliary state estimator*. Hence, the XKF is a cascaded system where the feedback-loop is avoided.

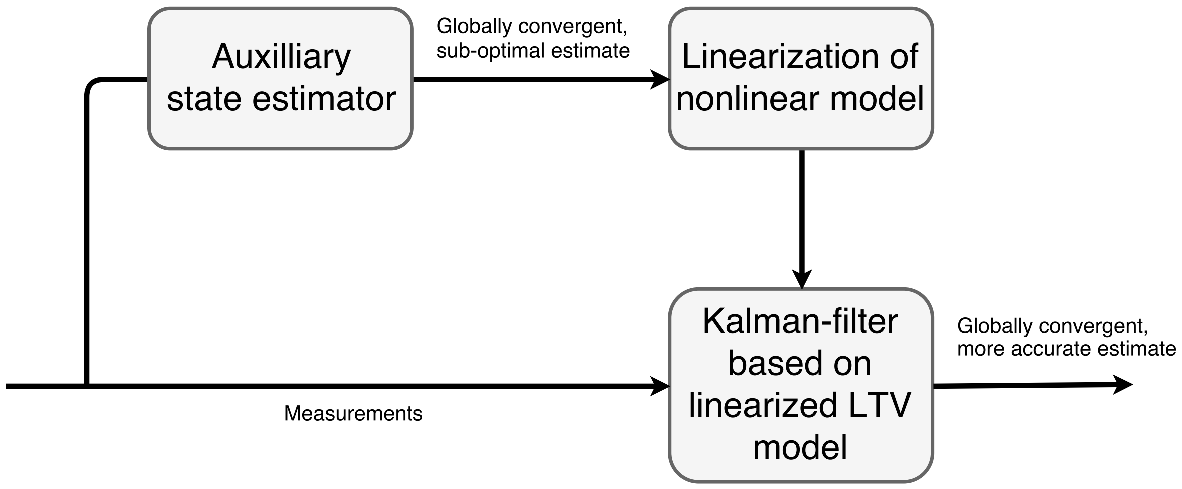

Auxilliary state estimator
Globally convergent, sub-optimal estimate
Linearization of nonlinear model
Kalman-filter based on linearized LTV model
Globally convergent, more accurate estimate
Measurements

Figure 2.5: The eXogenous Kalman Filter

It is shown that an XKF consisting of an auxiliary state estimator and a LKF in cascade inherits the stability properties of the auxiliary state estimator. See (Johansen and Fossen, 2017) or (Johansen and Fossen, 2016) for the full proof. This is an important property of the cascaded system as it allows the designer to make an auxiliary state estimator with global or near global stability properties.

It is evident that the auxiliary state estimator is able to make a globally convergent state estimate on its own. So why is it necessary to make the cascaded structure with the Kalman-filter as in figure 2.5? The answer to this question is that the auxiliary state estimator is only designed with respect to stability. This means that the noise in the system is typically not considered at this stage. Instead the noise is considered in the KF at the next stage. As a result the cascaded system yields both global stability and great noise suppression.

2.4.5 Designing the auxiliary state estimator

The main focus when designing an auxiliary state estimator is strong stability properties. From section 2.2 it is clear that an estimator with GES or GAS characteristics is preferred. This allows for many different approaches, but for many systems it is not trivial and maybe impossible to design an estimator with these properties.

In some cases it may be possible to design a *nonlinear passive observer*. This category of observers assumes a deterministic relation between measurements and states. The benefit of these observers are that they usually take asymptotic or exponential stability as the starting point for the design (Johansen and Fossen, 2017). Based on for instance Lyapunov-stability, proper tuning parameters can be chosen in order to achieve the desired performance.

Another approach may be to use a LKF as this is known to have global stability properties. This method is for instance used in (Johansen et al., 2016) where pseudorange measurements yield a nonlinear relation between measurements and the true range. By introducing new variables it is possible to rewrite the nonlinear pseudorange measurement function to a *quasi-linear* form. This allows for use with the LKF. If noise is not present, the new quasi-linear time varying (LTV) measurement function is free from approximations and the second stage of this XKF is thus not necessary. Noise is however always present in real applications and in this case the LTV measurement function only provides sub-optimal estimates. This is due to prior information about nonlinearities being eliminated in the transformation. For this reason the LTV measurement model is only used for generation of the linearization point about which the true nonlinear measurement function is linearized. This implementation is called the *Three-Stage Filter* and the overall structure is illustrated in figure 2.6. This filter is highly relevant for underwater applications as hydroacoustic sensor networks yields pseudorange measurements.

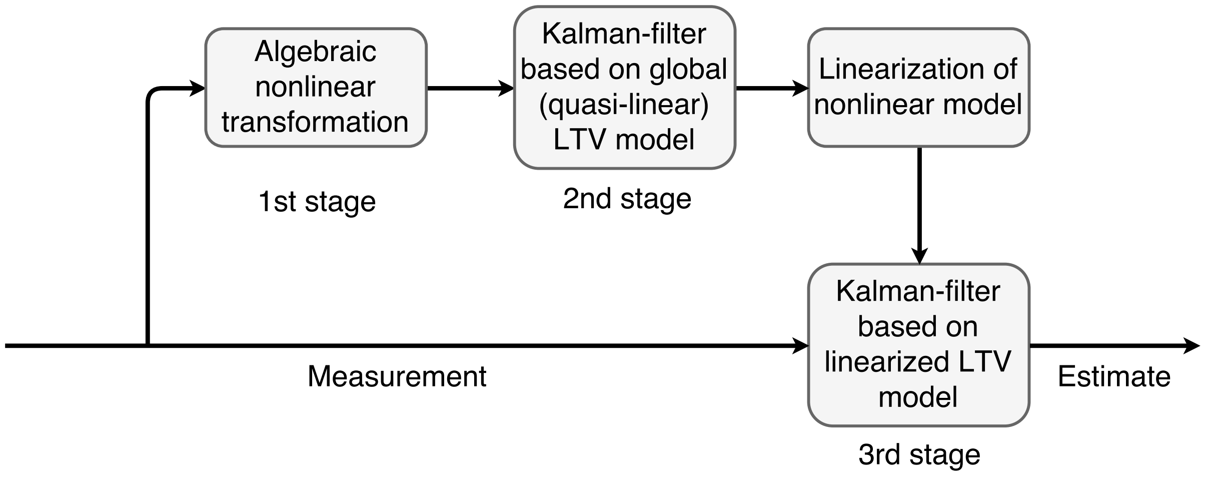

Algebraic nonlinear transformation
1st stage
Kalman-filter based on global (quasi-linear) LTV model
2nd stage
Linearization of nonlinear model
Kalman-filter based on linearized LTV model
3rd stage
Measurement
Estimate

Figure 2.6: Three-Stage Filter

2.4.6 Unscented Kalman Filter

The Unscented Kalman Filter (UKF) is another improvement of the EKF where the posterior Gaussian distribution is determined with higher precision. In the EKF the dynamical model and the measurement model are approximated using linearizaton. The UKF does not approximate the nonlinear models. Instead it approximates the posterior Gaussian distribution by evaluating a set of sample points using the nonlinear models directly. These points are carefully selected using an unscented transformation and are named sigma points (Wan and Merwe, 2000). Each sigma point is associated with a given weight and this forms the basis of the reconstruction of the posterior Gaussian distribution. The concept of this method is illustrated in figure 2.7.

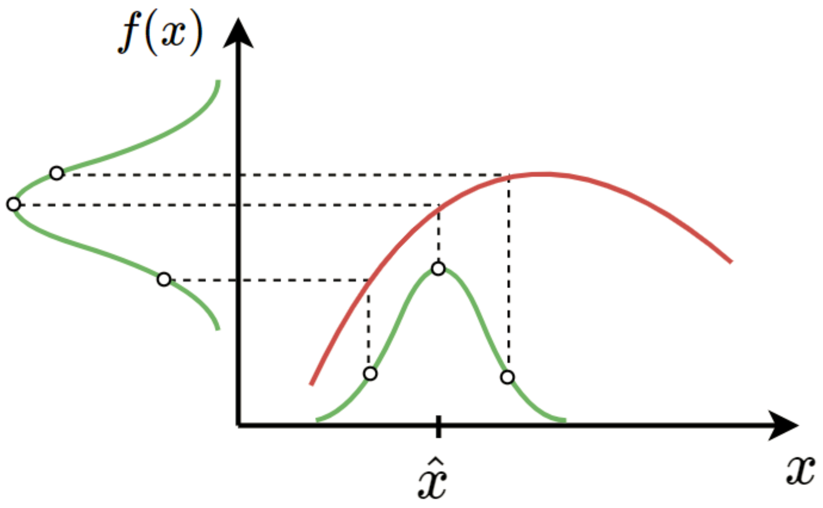


Figure 2.7: Gaussian construction from sigma points

This method captures the the posterior mean and covariance of the Gaussian accurately to the third order Taylor expansion (Wan and Merwe, 2000). This makes the UKF more applicable to systems with severe nonlinearities compared to the EKF. As the UKF evaluates the system model directly and not its Jacobian, it solves the problem of the Jacobian might being hard to derive. Like the EKF, global stability cannot be proven for the UKF either.

In (Allotta et al., 2016) the performance of the UKF was investigated using experimental data from an AUV. Sensor data were gathered from an Inertial Measurement Unit (IMU), a pressure sensor, and a global position system during periodic resurfing. In this paper it is concluded that the UKF was more accurate than the EKF. In (D'Alfonso et al., 2015) however, the two filters were implemented on a ground robot for estimation of position and orientation using range measurements. In this case the results showed that neither filter was better than the other in terms of accuracy. A plausible explanation to this result is that the nonlinearities in the system were not to severe. Hence, it is not guaranteed that the UKF will perform better than the EKF.

2.4.7 Particle Filter

The filtering methods presented above are limited to systems where the nonlinearities are not too severe, and the noise characteristics are assumed to be Gaussian. Under these assumptions the state estimation problem is all about transforming one Gaussian distribution into another Gaussian distribution at the next timestep. Ergo, the update laws only consider the mean and variance at each timestep. For non-Gaussian systems however, it is generally not possible to know the posterior Probability Density Function (PDF) on forehand as it can have any arbitrary shape. As a consequence no analytical solution can be derived for this problem.

The PF solves this by use of a Monte-Carlo method where the posterior PDF is approximated using a set of weighted particles (Doucet et al., 2000). According to (Gustafsson, 2010) the first PF methods dates back to the 1950s. However, due to limited computing power, it was not before later the PF as it is known today originated. The approach used in (Gordon et al., 1993) is therefore regarded as the beginning of the particle filter era. As mentioned in the *related work* section, recent advances of the PF for underwater vehicles indicates that this filter is able to provide better state estimates than the traditional Kalman filter methods. In (Zhao et al., 2014) it is also shown that the PF can be implemented to effectively detect faults on underwater vehicles. This results in the PF being able to efficiently estimate states even in case of sensor drop outs etc. However, just like most of the other filters presented earlier, global stability cannot be proven.

In the PF, the known posterior PDF from the last timestep $k-1$, i.e. $p(\boldsymbol{x}_{0:k-1}|\boldsymbol{y}_{1:k-1})$, is used to approximate the unknown posterior PDF $p(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{1:k})$ at the current time step. At each iteration a finite number of N state hypothesises, also called *particles* are drawn from $p(\boldsymbol{x}_{0:k-1}|\boldsymbol{y}_{1:k-1})$. Then all particles are evaluated using the mathematical model of the system, much like the UKF. However one important difference is that the particles in the PF are drawn randomly and not according to a set of rules as in the UKF. Next, on basis of the measurements, each particle is assigned a weight which tells how sufficient the given particle is describing for $p(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{1:k})$. Finally, $p(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{1:k})$ is constructed by relying more on the high-weighted particles. When the posterior PDF at the current timestep is constructed, it can be evaluated to find its mean and variance for instance.

To understand the workings of the PF it is an advantage to know the principle of *importance sampling* (IS). Hence, a somewhat detailed presentation of this principle, based on (Owen, 2013), is given next. This makes it easier to understand the implementation in later chapters.

Importance sampling

To demonstrate the principle of IS, assume that the expected value of a function $f(x)$ is to be calculated when the PDF of this function is given by $p(x)$. The expected value is then found from the integral

$$\mathbb{E}\left\{f(x)\right\} = \int_D f(x)p(x)dx \tag{2.11}$$

The domain that is to be evaluated is denoted D, hence $p(x) = 0 \ \forall \ x \notin D$. It is often difficult to evaluate the integral in Eq. (2.11) directly. Instead it can be approximated using

$$\mathbb{E}\left\{f(x)\right\} \approx \frac{1}{N}\sum_{i=1}^{N} f(X_i) \tag{2.12}$$

where X_i are samples drawn independently from $p(x)$.

Unfortunately it is not always possible to draw samples efficiently from $p(x)$, even though it might be trivial to evaluate $p(x)$. In such case one is forced to use a *proposal distribution* $q(x)$ to approximate the true *target distribution* $p(x)$. The proposal distribution should be easy to sample from, and must fulfil $p(x) > 0 \rightarrow q(x) > 0$ (Thrun et al., 2001). This is to ensure that all possible samples in $p(x)$ can be reconstructed using $q(x)$. By introducing the proposal distribution, Eq. (2.11) is rewritten

$$\mathbb{E}\{f(x)\} = \int_D \frac{f(x)p(x)}{q(x)} q(x)dx = \int_D f(x)w(x)q(x)dx \tag{2.13}$$

where $w(x) = \frac{p(x)}{q(x)}$ is a *weighting function*. The value of this function indicates how well the proposal distribution describes the target distribution at the given x. Similar to Eq. (2.12), the last integral in (2.13) is approximated using

$$\mathbb{E}\{f(x)\} \approx \frac{1}{N} \sum_{j=1}^{N} f(X_j)w(X_j) \tag{2.14}$$

where X_j are samples drawn independently from $q(x)$. The expected value of $f(x)$ is now obtained by sampling from $q(x)$ instead of $p(x)$. This solves the problem of $p(x)$ being difficult to sample from. Only evaluation of $p(x)$ at the given samples are necessary.

The idea of IS is visualised in figure 2.8 on the next page. The grey bars at the top of the illustration represents samples from the proposal distribution $q(x)$. All bars are equally important and have the same height, but one can see there are more bars where the value of $q(x)$ is larger. The bars at the bottom of the illustration are the exact same bars as the ones at the top. However, the bottom bars are weighted according to how well they describe the target density $p(x)$. For instance, the bars to the right have little importance as $p(x) \approx 0$ in this area. This is illustrated by the short height. Consequently these samples have little influence when calculating the expected value using Eq. (2.14). Notice that the proposal distribution in the illustration is not the best one. If $q(x)$ had a shape more similar to $p(x)$, more samples with high weights would have been obtained.

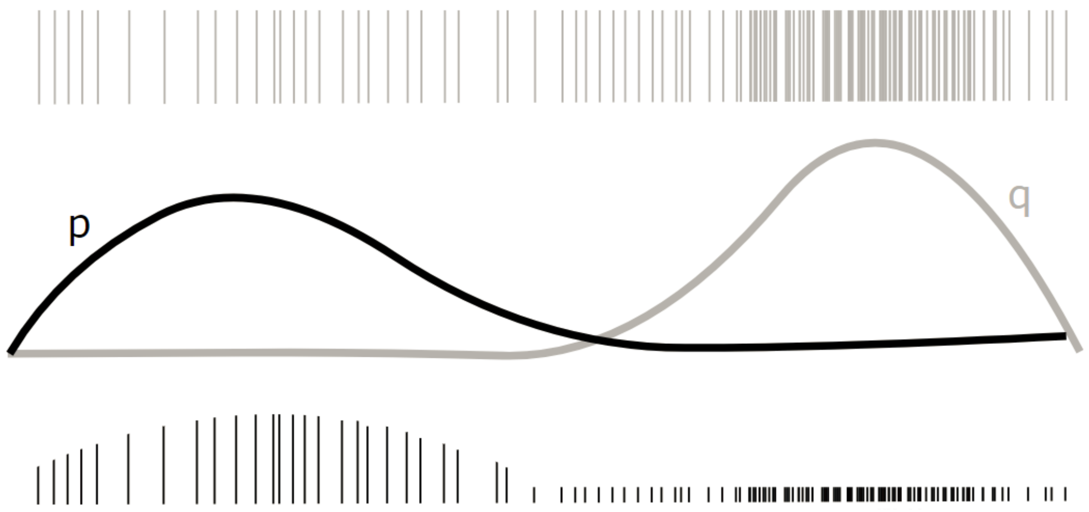


Figure 2.8: Visulization of importance resampling

The relative importance of each sample is found by normalizing the weights.

$$\tilde{w}(x^i) = \frac{w(x^i)}{\sum_{j=1}^{N} w(x^j)} \tag{2.15}$$

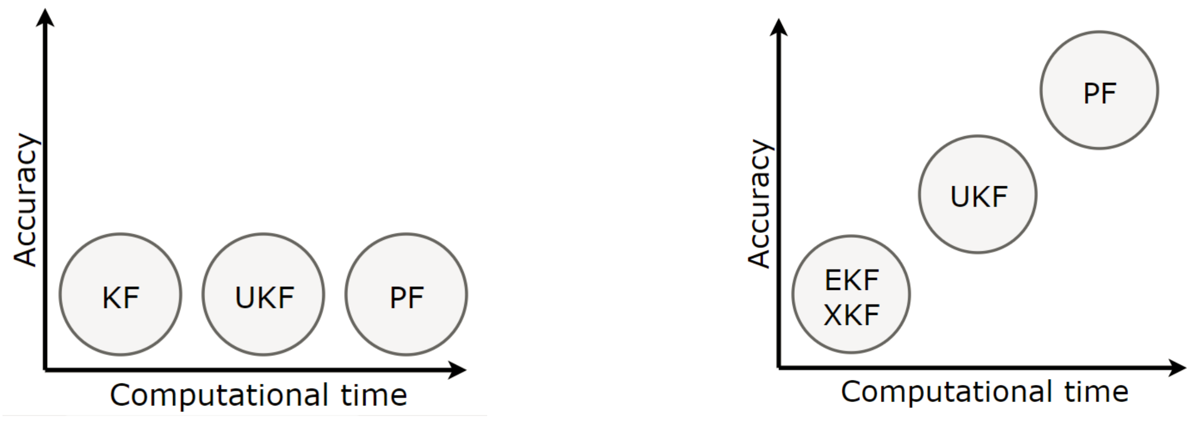
The target distribution can then be approximated as

$$p(x) \approx \sum_{i=1}^{N} \tilde{w}(x^i)\delta(x - x^i) \tag{2.16}$$

where $\delta(\cdot)$ is the Dirac-delta function.

2.5 Summary of filtering methods

The choice of a state estimator is essentially a trade-off between accuracy, computational effort and stability. In the case of linear and Gaussian systems the choice is obvious. In this case the LKF is proven to be the optimal solution and in addition it has global stability properties. Using a UKF or a PF will only contribute to higher computational times. This is illustrated in figure 2.9a.



(a) Linear and Gaussian

(b) Nonlinear and/or non-Gaussian

Figure 2.9: Comparison of Bayesian estimators

For non-linear systems it may be harder to determine what filter to use. Of the prior mentioned methods, only the PF is able to handle non-Gaussian noise characteristics. However it is not trivial to know which method yields the best results if the system is nonlinear and the Gaussian assumption holds. In such case the EKF probably gives the fastest solution considering computation time. And if it is possible to design an auxilliary estimator, an XKF with great stability properties can be made. However, the accuracy of these methods may not be at an acceptable level. The UKF might perform better as the linearization in the EKF/XKF is a source to errors. Further improvements in accuracy may be obtained by using the PF. By increasing the number of particles one gets closer to the real posterior distribution of the system, but this also yields higher computational times. Figure 2.9b shows how the different methods compare to each other in theory when the nonlinearities are severe and noise deviates from the Gaussian assumption.

Chapter 3

Modeling

Relevant models for marine vessels and state estimation are presented in this chapter. The models and notation presented here will be frequently used throughout the thesis.

3.1 Reference frames

When describing the position and orientation of a rigid body in space, there are three possible translations: *surge*, *sway*, *heave*, and three possible rotations: *roll*, *pitch*, *yaw*. Together they constitute six degrees of freedom (DOF). Figure 3.1 below shows how the different degrees of freedom are related. The outline of a ship is also drawn to see how this is applies to marine vessels. The parameters are used according to the SNAME standard in table 3.1.

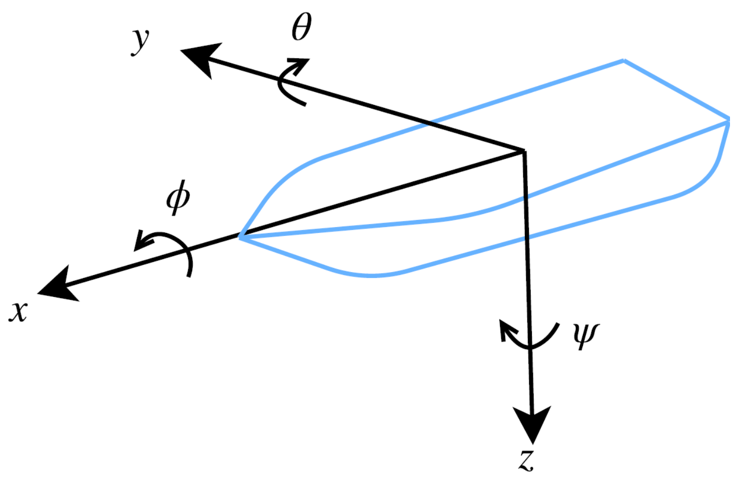


Figure 3.1: Six degree of freedom representation

Table 3.1: SNAME (1950) notation from (Fossen, 2011)

DOF		Forces and moments	Linear and angluar velocities	Position and Euler angles
1	Surge	X	u	x
2	Sway	Y	v	y
3	Heave	Z	w	z
4	Roll	K	p	ϕ
5	Pitch	M	q	θ
6	Yaw	N	r	ψ

In order to relate the orientation of the vessel relative to the surface of the earth, two different coordinate systems or *reference frames* must be used. There are many different reference frames to choose from, but for marine applications two of the most widely used are the NED- and BODY reference frames which are in the category of *geographic reference frames* (Fossen, 2011).

NED: The *North-East-Down* coordinate system $\{n\} = (x_n, y_n, z_n)$ with origin o_n is usually defined as a moving tangent plane on the surface of earth. The x-axis points to the true north, the y-axis points towards east, and the z-axis points downwards normal to the surface of earth. For local navigation an earth-fixed tangent plane of the surface is used. In this case one can assume that $\{n\}$ is inertial and Newton's laws apply.

BODY: The body-fixed reference frame $\{b\} = (x_b, y_b, z_b)$ is moving with the vessel with the origin o_b fixed to the craft. Usually x_b points forwards, y_b towards starboard and z_b downwards. Linear and angular velocities of a vessel are expressed in this frame, while position and orientation should be expressed relative to an inertial frame.

The inertial frame is often approximated by $\{n\}$, but other alternatives exists. In global navigation for instance there are other more convenient reference frames where the origin of the frames are fixed to the center of the earth.

Figure 3.2 shows how $\{n\}$ and $\{b\}$ are related. The square field is the tangent plane in the NED-frame, while the box is represented using BODY-frame. How to mathematically relate these frames will be shown in the next section.

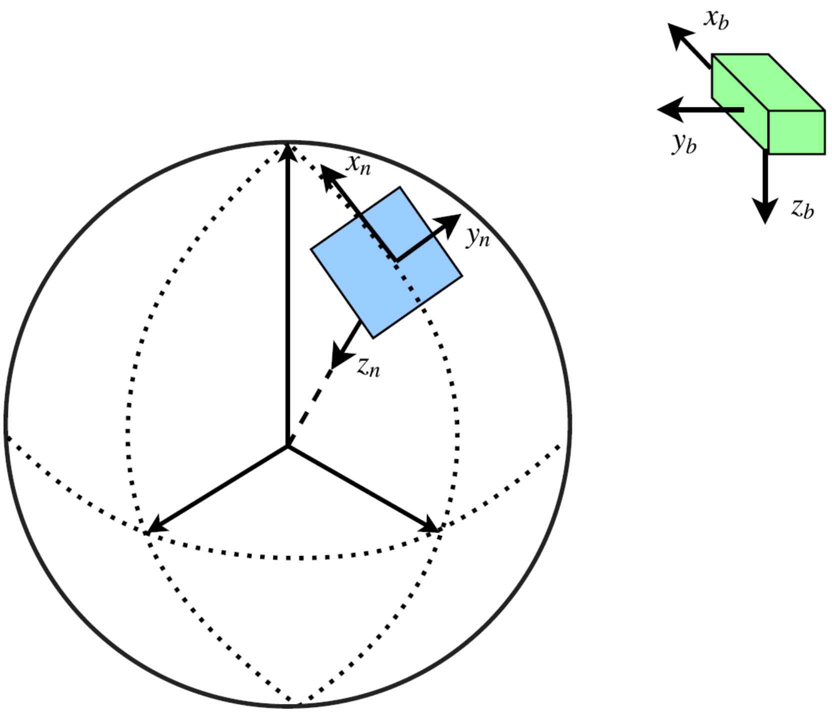


Figure 3.2: NED-frame and BODY-frame

3.2 Vessel modeling

In the development of a mathematical model for a vessel, one distinguish between the *kinematics* which deals with the pure geometrical aspects of motion. And the *kinetics* which is the analysis of the forces that causes the motion (Fossen, 2011). This chapter serves to introduce how these two topics together make up the total vessel model and will be described according to (Fossen, 2011).

3.2.1 Kinematics

To fully describe the orientation of a body in space a six-DOF model is needed. Using the SNAME notation from table 3.1, the positions in $\{n\}$-frame and the velocities in $\{b\}$-frame can be represented as

$$\boldsymbol{\eta} = [\boldsymbol{p}^n_{b/n}, \boldsymbol{\Theta}_{nb}]^T = [x\ y\ z\ \phi\ \theta\ \psi]^T \tag{3.1}$$

$$\boldsymbol{\nu} = [\boldsymbol{\nu}^b_{b/n}, \boldsymbol{\omega}^b_{b/n}]^T = [u\ v\ w\ p\ q\ r]^T \tag{3.2}$$

These can be related using a rotation matrix transforming $\boldsymbol{\nu}$ to $\{n\}$-coordinates

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{3.3}$$

where

$$\boldsymbol{J}(\boldsymbol{\eta}) = \begin{bmatrix} \boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb}) & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & \boldsymbol{T}_\Theta(\boldsymbol{\Theta}_{nb}) \end{bmatrix} \tag{3.4}$$

The sub-matrices of $\boldsymbol{J}(\boldsymbol{\eta})$ are a result of the principal rotation matrices presented below

$$\boldsymbol{R}_{x,\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix}, \boldsymbol{R}_{y,\theta} = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix}, \boldsymbol{R}_{z,\psi} = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

where $s\cdot = sin(\cdot)$ and $c\cdot = cos(\cdot)$ and x, y, z corresponds to the axis the rotation is about.

The linear velocity transformation matrix can then be defined as

$$\boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb}) := \boldsymbol{R}_{z,\psi}\boldsymbol{R}_{y,\theta}\boldsymbol{R}_{x,\phi} \tag{3.6}$$

And the angular velocity transformation matrix $\boldsymbol{T}_\Theta(\boldsymbol{\Theta}_{nb})$ can be found from the relation

$$\boldsymbol{\omega}_{b/n}^b = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \boldsymbol{R}_{x,\phi}^T \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \boldsymbol{R}_{y,\theta}^T\boldsymbol{R}_{x,\phi}^T \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} := \boldsymbol{T}_\Theta^{-1}(\boldsymbol{\Theta}_{nb})\dot{\boldsymbol{\Theta}}_{nb} \tag{3.7}$$

Expanding Eq. (3.7) yields

$$T_\Theta(\boldsymbol{\Theta}_{nb}) = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \end{bmatrix} \tag{3.8}$$

A disadvantage of using the Euler angles $\boldsymbol{\Theta}_{nb} = [\phi\ \theta\ \psi]$ to represent attitude is that $T_\Theta(\boldsymbol{\Theta}_{nb})$ is singular for pitch angles of $\theta = \pm 90°$. An alternative to the Euler angles, that are free from singularities, is the quaternion representation (Fossen, 2011).

3.2.2 Kinetics

The kinetics of a vessel are comprised of rigid-body forces, hydrodynamic forces and hydrostatic forces according to the manoeuvring theory of (Fossen, 2011). In presence of irrotational currents the hydrodynamic forces are dependent on the relative velocity vector $\boldsymbol{\nu}_r = \boldsymbol{\nu} - \boldsymbol{\nu}_c$, while the rigid body forces depends on $\boldsymbol{\nu}$ only. Hence,

$$\begin{aligned}&\underbrace{\boldsymbol{M}_{RB}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu}}_{\text{rigid-body forces}} + \underbrace{\boldsymbol{M}_A\dot{\boldsymbol{\nu}}_r + \boldsymbol{C}_A(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r}_{\text{hydrodynamic forces}} + \\ &\cdots \underbrace{\boldsymbol{g}(\boldsymbol{\eta}) + \boldsymbol{g}_0}_{\text{hydrostatic forces}} = \boldsymbol{\tau} + \boldsymbol{\tau}_{env}\end{aligned} \tag{3.9}$$

It is however shown that this equation can be represented using relative velocities only if currents are irrational and constant in $\{n\}$. Using this assumption, and introducing $\boldsymbol{M} = \boldsymbol{M}_{RB} + \boldsymbol{M}_A$ and $\boldsymbol{C}(\boldsymbol{\nu}_r) = \boldsymbol{C}_{RB}(\boldsymbol{\nu}_r) + \boldsymbol{C}_A(\boldsymbol{\nu}_r)$ for simplicity of notation, vessel kinetics are represented as

$$\boldsymbol{M}\dot{\boldsymbol{\nu}}_r + \boldsymbol{C}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{g}(\boldsymbol{\eta}) + \boldsymbol{g}_0 = \boldsymbol{\tau} + \boldsymbol{\tau}_{env} \tag{3.10}$$

Where $\boldsymbol{M}$ is the rigid body inertia and added mass matrix, $\boldsymbol{C}$ is the rigid body and added mass Coriolis matrix, $\boldsymbol{D}$ is the damping matrix, $\boldsymbol{g}(\boldsymbol{\eta})$ and $\boldsymbol{g}_0$ are the restoring forces, and $\boldsymbol{\tau}$ and $\boldsymbol{\tau}_{env}$ are the control forces and environmental forces respectively.

3.2.3 Complete vessel state space model

The kinematic and the kinetic models are put together in a state space formulation to represent the full mathematical model of a vessel. Combining Eq. 3.3 and Eq. 3.10 in a state space model yields

$$\begin{bmatrix}\dot{\boldsymbol{\eta}} \\ \dot{\boldsymbol{\nu}}_r\end{bmatrix} = \begin{bmatrix}\boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu}_r \\ \boldsymbol{M}^{-1}[-\boldsymbol{C}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r - \boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r - \boldsymbol{g}(\boldsymbol{\eta}) - \boldsymbol{g}_0 + \boldsymbol{\tau} + \boldsymbol{\tau}_{env}]\end{bmatrix} \tag{3.11}$$

When all 6 DOFs are represented this model consists of 12 states, i.e. $\boldsymbol{\eta} = [x\ y\ z\ \phi\ \theta\ \psi]^T$ and $\boldsymbol{\nu} = [u\ v\ w\ p\ q\ r]^T$. Depending on the application, the model is often reduced to represent less DOFs. For a regular surface ship for instance, it is often not necessary to model heave, pitch and roll. The model is thus reduced to $\boldsymbol{\eta} = [x\ y\ \psi]^T$ and $\boldsymbol{\nu} = [u\ v\ r]^T$ in such case.

3.3 Bias models

A bias model is often used in observers to better account for slowly-varying forces and unmodelled effects. A properly designed bias model will in turn lead to no steady-state offset in the velocity estimates (Sørensen, 2013). According to (Fossen, 2011) a common source to slowly varying forces and moments in marine applications are second order wave forces, ocean currents and wind forces. The bias estimate is therefore composed of a wide range of components and is nonphysical. There are two mathematical models frequently used to determine the bias term according to (Fossen, 2011). The first is the *first order Markov model*

$$\dot{b} = -T_b^{-1} b + w_b \tag{3.12}$$

where w_b is a zero mean Gaussian white noise vector and T_b is a diagonal matrix with positive bias time constants. The other alternative is to model the bias as a *Wiener process*

$$\dot{b} = w_b \tag{3.13}$$

This method is also referred to as *random walk* and is simply the integrated of the Gaussian white noise.

Chapter 4

Filter design

In chapter 2 several implementations of the recursive Bayesian filter were presented and discussed. In this chapter equations and mathematical foundation needed to implement the LKF, EKF, XKF and PF are presented. The first three filters have a very similar structure as the EKF builds upon the LKF, and the XKF further builds upon the EKF. It is therefore natural to first present the LKF even tough it is not further used in the thesis. All implementations and concepts presented in this chapter are general and may be applied to any dynamical system.

Recall from the introduction that the general structure of the dynamical model and the measurement model is expressed as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}) \tag{4.1a}$$

$$\boldsymbol{y} = \boldsymbol{h}(t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}) \tag{4.1b}$$

4.1 The Linear Kalman Filter

In the special case of linear systems Eq. (4.1a) and Eq. (4.1b) are rewritten on the form

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{E}\boldsymbol{w} \tag{4.2a}$$

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} + \boldsymbol{v} \tag{4.2b}$$

where $\boldsymbol{A}$ is the system matrix, $\boldsymbol{B}$ is the input matrix, $\boldsymbol{E}$ is the noise matrix, $\boldsymbol{H}$ is the measurement matrix, and $\boldsymbol{D}$ is the feedforward matrix. The matrix $\boldsymbol{D}$ is set to zero when there is no feedforward in the system. The variables $\boldsymbol{w}$ and $\boldsymbol{v}$ represents the process noise and measurement noise respectively. Both are assumed to be normally distributed with zero mean, i.e.

$p(\boldsymbol{w}) \sim N(0, \boldsymbol{Q})$ and $p(\boldsymbol{v}) \sim N(0, \boldsymbol{R})$. The two noise contributions are also assumed to be independent of each other.

4.1.1 Discretization

The state space model in Eq. (4.2) is continuous and not applicable with the discrete LKF before it is discretized. The discretized state space model is obtained by applying simple changes to the continuous state space model. Following the notation of (Fossen, 2011), Eq. (4.2a) and (4.2b) are rewritten to discretized form as

$$\boldsymbol{x}_{k+1} = \boldsymbol{\Phi}\boldsymbol{x}_k + \boldsymbol{\Delta}\boldsymbol{u}_k + \boldsymbol{\Gamma}\boldsymbol{w}_k \tag{4.3a}$$

$$\boldsymbol{y}_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{D}\boldsymbol{u}_k + \boldsymbol{v}_k \tag{4.3b}$$

where the discretized matrices are found using Euler integration with time step h

$$\begin{aligned}\boldsymbol{\Phi} &= exp(\boldsymbol{A}h)\\ &\approx \boldsymbol{I} + \boldsymbol{A}h + \frac{1}{2}(\boldsymbol{A}h)^2 + \cdots + \frac{1}{N!}(\boldsymbol{A}h)^N\\ &\approx \boldsymbol{I} + \boldsymbol{A}h\end{aligned} \tag{4.4}$$

$$\boldsymbol{\Delta} = \boldsymbol{A}^{-1}(\boldsymbol{\Phi} - \boldsymbol{I})\boldsymbol{B} \tag{4.5}$$

$$\boldsymbol{\Gamma} = \boldsymbol{A}^{-1}(\boldsymbol{\Phi} - \boldsymbol{I})\boldsymbol{E} \tag{4.6}$$

4.1.2 General implementation

For the very first iteration there are no state estimate or covariance of the state estimate available. A prior estimate $\bar{\boldsymbol{x}}_0$ and a corresponding error covariance $\bar{\boldsymbol{P}}_0$ must therefore be given as input to the filter at the first iteration. The Kalman gain matrix which minimizes the mean-square estimation error can then be found as

$$\boldsymbol{K}_k = \bar{\boldsymbol{P}}_k\boldsymbol{H}_k^T(\boldsymbol{H}_k\bar{\boldsymbol{P}}_k\boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1} \tag{4.7}$$

Using the Kalman gain and the measurements, the predicted states are updated.

$$\hat{\boldsymbol{x}}_k = \bar{\boldsymbol{x}}_k + \boldsymbol{K}_k\left[\boldsymbol{y}_k - \boldsymbol{H}_k\bar{\boldsymbol{x}}_k\right] \tag{4.8}$$

The error covariance are calculated in the next step.

$$\hat{\boldsymbol{P}}_k = [\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k]\bar{\boldsymbol{P}}_k[\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k]^T + \boldsymbol{K}_k \boldsymbol{R}_k \boldsymbol{K}_k^T \tag{4.9}$$

Finally, the propagation of states and covariances to be used in the next step are calculated.

$$\bar{\boldsymbol{x}}_{k+1} = \boldsymbol{\Phi}_k \hat{\boldsymbol{x}}_k + \boldsymbol{\Delta}_k \boldsymbol{u}_k \tag{4.10}$$

$$\bar{\boldsymbol{P}}_{k+1} = \boldsymbol{\Phi}_k \hat{\boldsymbol{P}}_k \boldsymbol{\Phi}_k^T + \boldsymbol{\Gamma}_k \boldsymbol{Q}_k \boldsymbol{\Gamma}_k^T \tag{4.11}$$

The matrices $\boldsymbol{Q}_k = \boldsymbol{Q}_k^T > 0$ and $\boldsymbol{R}_k = \boldsymbol{R}_k^T > 0$ are tuning matrices chosen by the designer of the filter. More about their function is discussed in section 4.2.3.

4.2 The Extended Kalman Filter

The EKF is based on linearization of the state space model in Eq. (4.1). This is done by calculating the Jacobians of $\boldsymbol{f}$ and $\boldsymbol{h}$. The Jacobians are then evaluated at the previous state estimate $\hat{\boldsymbol{x}}$, and the current input $\boldsymbol{u}$. Denote the Jacobians $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{H}$ and $\boldsymbol{D}$ as

$$\boldsymbol{A}_{ij} = \left.\frac{\partial \boldsymbol{f}_i}{\partial \boldsymbol{x}_j}\right|_{\hat{\boldsymbol{x}}} \quad \text{and} \quad \boldsymbol{B}_{ij} = \left.\frac{\partial \boldsymbol{f}_i}{\partial \boldsymbol{u}_j}\right|_{\boldsymbol{u}} \quad \text{and} \quad \boldsymbol{H}_{ij} = \left.\frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{x}_j}\right|_{\hat{\boldsymbol{x}}} \quad \text{and} \quad \boldsymbol{D}_{ij} = \left.\frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{u}_j}\right|_{\boldsymbol{u}} \tag{4.12}$$

The linearized system in a small interval Δ around $\hat{\boldsymbol{x}}$ and $\boldsymbol{u}$ is then given by

$$\Delta \boldsymbol{x}_{k+1} \approx \boldsymbol{A} \cdot \Delta \boldsymbol{x}_k + \boldsymbol{B} \cdot \Delta \boldsymbol{u}_k + \boldsymbol{E} \cdot \boldsymbol{w}_k \tag{4.13a}$$

$$\Delta \boldsymbol{y}_k \approx \boldsymbol{H} \cdot \Delta \boldsymbol{x}_k + \boldsymbol{D} \cdot \Delta \boldsymbol{u}_k + \boldsymbol{v}_k \tag{4.13b}$$

where $\boldsymbol{E}$ is assumed to be independent of $\boldsymbol{x}$ and $\boldsymbol{u}$ and thus a matrix with constant values. It is apparent that the system in Eq. (4.13) has a structure identical to the linear state space.

4.2.1 General implementation and discretization

With the linearized system in Eq. (4.13) it is possible to use the Kalman filter equations for linear systems presented in section 4.1.2. Some modifications must be done first. The state propagation in Eq. (4.10) is now calculated using the original nonlinear function. Hence,

$$\bar{\boldsymbol{x}}_{k+1} = \mathcal{F}(\hat{\boldsymbol{x}}_k, \boldsymbol{u}_k) \tag{4.14}$$

where

$$\boldsymbol{\mathcal{F}}(\hat{\boldsymbol{x}}_k, \boldsymbol{u}_k) = \hat{\boldsymbol{x}}_k + h\boldsymbol{f}(\hat{\boldsymbol{x}}_k, \boldsymbol{u}_k) \tag{4.15}$$

is obtained using Euler integration (Fossen, 2011).

Likewise, if the measurement equation is nonlinear, the state update estimate in Eq. (4.8) must be implemented as

$$\hat{\boldsymbol{x}}_k = \bar{\boldsymbol{x}}_k + \boldsymbol{K}_k\left[\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k\right] \tag{4.16}$$

where

$$\hat{\boldsymbol{y}}_k = \boldsymbol{h}(\bar{\boldsymbol{x}}_k, \boldsymbol{u}_k) \tag{4.17}$$

Discretization is done as in section 4.1.1, using the matrices from Eq. (4.13).

4.2.2 Implementation of an auxiliary state estimator for XKF

Assuming a state estimate $\breve{\boldsymbol{x}}$ is available from an auxiliary state estimator, the implementation of an XKF is done with simple changes to the EKF. If the dynamical model is nonlinear it must be linearized around $\breve{\boldsymbol{x}}$ according to

$$\hat{\boldsymbol{x}}_{k+1} = \boldsymbol{f}(\breve{\boldsymbol{x}}_k, \boldsymbol{u}_k) + \left.\frac{\partial \boldsymbol{f}_i}{\partial \boldsymbol{x}_j}\right|_{\breve{\boldsymbol{x}}_k, \boldsymbol{u}_k}(\hat{\boldsymbol{x}}_k - \breve{\boldsymbol{x}}_k) \tag{4.18}$$

where $\hat{\boldsymbol{x}}$ is the estimate from the complete XKF. To make the original EKF applicable with the new linearization point $\breve{\boldsymbol{x}}$, the right hand side of Eq. (4.18) replaces $\boldsymbol{f}(\hat{\boldsymbol{x}}_k, \boldsymbol{u}_k)$ in the estimate propagation equation in Eq. (4.15).

If the measurement function is nonlinear, it must be linearized around $\breve{\boldsymbol{x}}$ in the same manner. Hence, the measurement function becomes

$$\hat{\boldsymbol{y}}_k = \boldsymbol{h}(\breve{\boldsymbol{x}}_k, \boldsymbol{u}_k) + \left.\frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{x}_j}\right|_{\breve{\boldsymbol{x}}_k, \boldsymbol{u}_k}(\hat{\boldsymbol{x}}_k - \breve{\boldsymbol{x}}_k) \tag{4.19}$$

Using the same notation as earlier this implies (4.16) should be implemented with

$$\hat{\boldsymbol{y}}_k = \boldsymbol{h}(\breve{\boldsymbol{x}}_k, \boldsymbol{u}_k) + \left.\frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{x}_j}\right|_{\breve{\boldsymbol{x}}_k, \boldsymbol{u}_k}(\bar{\boldsymbol{x}}_k - \breve{\boldsymbol{x}}_k) \tag{4.20}$$

as the measurement function instead of Eq. (4.17).

Discretization is done in the same way as for the EKF, using the matrices linearized around $\breve{\boldsymbol{x}}$.

4.2.3 Tuning of Kalman filters

In the Kalman filter equations there are two dedicated tuning matrices. The noise covariance matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ represents the process noise and measurement noise respectively. Usually the measurement noise is known from the characteristics of the sensors, while the process noise is unknown and must be determined by the designer of the filter. High values in these matrices corresponds to large uncertainty, and vica verca. The ratio between these matrices is thus important. If $\boldsymbol{Q}$ is large compared to $\boldsymbol{R}$ it implies the measurements should be trusted more than the mathematical model. Similarly, if $\boldsymbol{Q}$ is small compared to $\boldsymbol{R}$ it implies the mathematical model should be trusted more than the measurements.

4.3 Particle Filter

As previously discussed there are no analytical solution to the PF. This implies no exact equations exists for the general case of particle filtering. A detailed and general approach supported by theory is therefore presented in this section. A more specific implementation for state estimation of ROVs are presented in the next chapter.

4.3.1 Sequential importance sampling

The principle of importance sampling introduced in section 2.4.7 should now be implemented to fit with the recursive Bayesian estimator. This is done using the principle of *sequential importance sampling*. At every iteration the aim is to approximate the true posterior PDF of the states $p(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{1:k})$ by a set of samples. Notice that $\boldsymbol{x}_{0:k}$ and $\boldsymbol{y}_{1:k}$ are the whole trajectory of states and measurements. This implies the PF not only considers the current state $\boldsymbol{x}_k$, but in the general case it generates and evaluates a set $\{\boldsymbol{x}^i_{0:k}\}^N_{i=1}$ of N different trajectories. Where N is the number of samples used in the iterations.

The true posterior PDF is unknown and samples cannot be drawn from this distribution directly. Using the principle of importance sampling, samples are drawn from a known proposal distribution $q(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{1:k})$ instead. This makes it possible to augment the trajectories $\boldsymbol{x}^i_{0:k-1}$ from the previous iteration with new state samples $\boldsymbol{x}^i_k$ drawn from $q(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}, \boldsymbol{y}_{1:k})$. This gives the current set of trajectories $\boldsymbol{x}^i_{0:k} = \left\{\boldsymbol{x}^i_{0:k-1}, \boldsymbol{x}^i_k\right\}$. From the theory of importance sampling, the weight of trajectory i can now be expressed as

$$w^i_k \propto \frac{p(\boldsymbol{x}^i_{0:k}|\boldsymbol{y}_{1:k})}{q(\boldsymbol{x}^i_{0:k}|\boldsymbol{y}_{1:k})} \tag{4.21}$$

It is shown, see for instance (Arulampalam et al., 2002), that the weight updates in Eq. (4.21) can be calculated recursively from

$$w_k^i = w_{k-1}^i \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k^i)p(\boldsymbol{x}_k^i|\boldsymbol{x}_{k-1}^i)}{q(\boldsymbol{x}_k^i|\boldsymbol{x}_{0:k-1}^i, \boldsymbol{y}_{1:k})} \tag{4.22}$$

This equation is straightforward to evaluate as $p(\boldsymbol{y}_k|\boldsymbol{x}_k^i)$ is obtained from the measurement equation $\boldsymbol{h}(\cdot)$ and the known PDF of the measurement noise $\boldsymbol{v}_k$. Similarly, $p(\boldsymbol{x}_k^i|\boldsymbol{x}_{k-1}^i)$ is obtained from the system equation $\boldsymbol{f}(\cdot)$ and the known PDF of the system noise $\boldsymbol{w}_k$. The denominator is the proposal density. Using the weights from Eq. (4.22), the posterior density can now be approximated as

$$p(\boldsymbol{x}_{0:k}|\boldsymbol{y}_{1:k}) \approx \sum_{i=1}^{N} w_k^i \delta(\boldsymbol{x}_{0:k} - \boldsymbol{x}_{0:k}^i) \tag{4.23}$$

Estimation of the filtering stage only
In many cases only the filtering posterior PDF $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ is of interest. A proposal distribution that only depends on $\boldsymbol{x}_{k-1}$ and $\boldsymbol{y}_k$, i.e. $q(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{y}_k)$ is then used. This means that only the most recent state estimates have to be stored for the next iteration. Eq. (4.22) then changes to

$$w_k^i \propto w_{k-1}^i \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k^i)p(\boldsymbol{x}_k^i|\boldsymbol{x}_{k-1}^i)}{q(\boldsymbol{x}_k^i|\boldsymbol{x}_{k-1}^i, \boldsymbol{y}_k)} \tag{4.24}$$

Using the weights in Eq. (4.24) the filtering posterior PDF can be approximated according to

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) \approx \sum_{i=1}^{N} w_k^i \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^i) \tag{4.25}$$

and this is the method which is used in the implementation described in the next section.

4.3.2 General implementation

The theory presented above is often implemented as what is known as the *Sampling Importance Resampling* (SIR) filter. Here, the proposal distribution $q(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{y}_k)$ is set equal to the prior density $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}^i)$ which is the system model. In addition, a crucial resampling step is added to avoid particles ending up in areas with low posterior probability. The approach and terminology used in this section is mainly based on the work of (Arulampalam et al., 2002) and (Gustafsson, 2012). When following the steps it is a good idea to look at figure 4.1 simultaneously as this provides a visual interpretation of the PF.

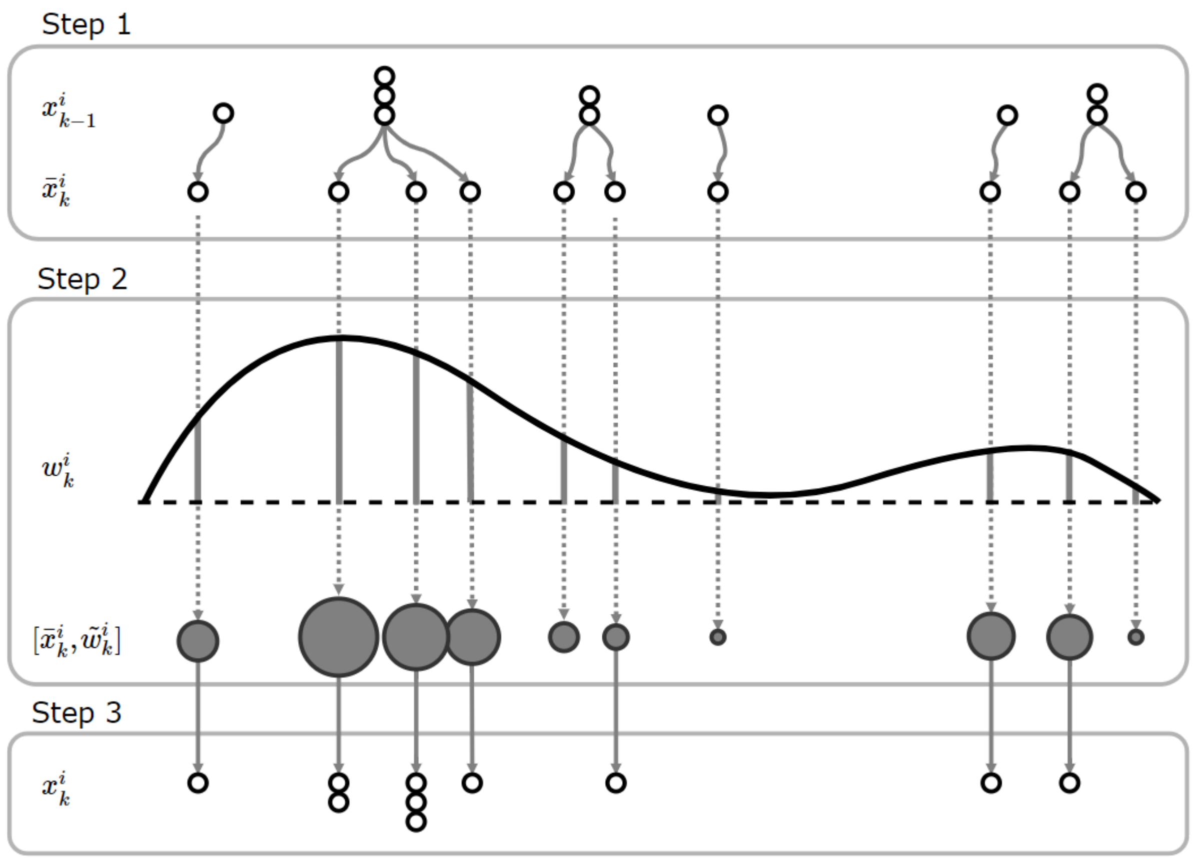


Figure 4.1: Illustration of steps in SIR-filter

Step 1: Prediction using system model

The set of particles $\chi_{k-1} = \{\boldsymbol{x}^i_{k-1}\}^N_{i=1}$ from the previous iteration is used as input to the next iteration. Said differently, the posterior PDF of the last iteration is the proposal distribution for the current iteration. Based on this the state prediction $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ is calculated. This is done using the system model

$$\bar{\boldsymbol{x}}^i_k = \boldsymbol{f}(\boldsymbol{x}^i_{k-1}, \boldsymbol{e}^i_{k-1}) \quad , i = 1, \cdots, N \tag{4.26}$$

Here $\boldsymbol{e}^i_{k-1}$ is generated randomly using the known PDF of the noise contribution in the system model. This is done to spread the state hypotheses in the particles and thus ensure diversity. In step 1 of figure 4.1 it is seen how the particles are spread due to the noise added to the system model. Notice that $\boldsymbol{u}_k$ is omitted from the state prediction notation. This is done to simplify notation only. The state predictions are stored as a temporarily set of particles

$$\bar{\chi}_k = \{\bar{\boldsymbol{x}}^i_k\}^N_{i=1} \tag{4.27}$$

Step 2: Assign weights based on measurements

Next step is to assign weights to each of the N particles. By setting the proposal distribution equal to the system model, i.e. $q(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{y}_k) = p(\boldsymbol{x}_k|\boldsymbol{x}^i_{k-1})$, the weights from Eq. (4.24) reduces to

$$w_k^i \propto p(\boldsymbol{y}_k|\boldsymbol{x}_k^i) \tag{4.28}$$

The term w_{k-1}^i is disregarded from the weight calculations due to the weight of all particles being set to $1/N$ after resampling. This is further discussed in the next step. On basis of Eq. (4.28) the weight of particle i is found from

$$w_k^i = p(\boldsymbol{y}_k|\bar{\boldsymbol{x}}_k^i) \tag{4.29}$$

This entails the weights being equal to the probability of the measurement being true, given the state prediction $\bar{\boldsymbol{x}}_k^i$ of particle i being true. The calculation is done using the measurement equation $\boldsymbol{h}(\cdot)$ and the known PDF of the measurement noise v_k.

The weight of a particle itself is not particularly interesting. It is the weight relative to other particles that provides useful information. Hence, all weights are normalized according to

$$\tilde{w}_k^i = \frac{w_k^i}{\sum_{j=1}^{N} w_k^j} \tag{4.30}$$

Step 3: Resampling

Unless the proposal distribution is the true posterior distribution itself, the variance of the importance weights will increase with time when using sequential importance sampling (Doucet et al., 2000). This will in turn lead to all but one particle having normalized weights close to zero after some iterations. This does not mean that this one particle is a good guess of the true state. It only means that this particle is more plausible than the others. Consequently, huge computational effort is put into particles that do not contribute to the final estimate.

Resampling is introduced as a solution to this problem. The idea behind this technique is to get rid of the particles with low weight in the temporarily set of particles $\bar{\chi}_k$. A new set of particles χ_k is then obtained by drawing particles from $\bar{\chi}_k$, where the probability of drawing a random particle is proportional to its normalized weight. This process is visualised in figure 4.2.

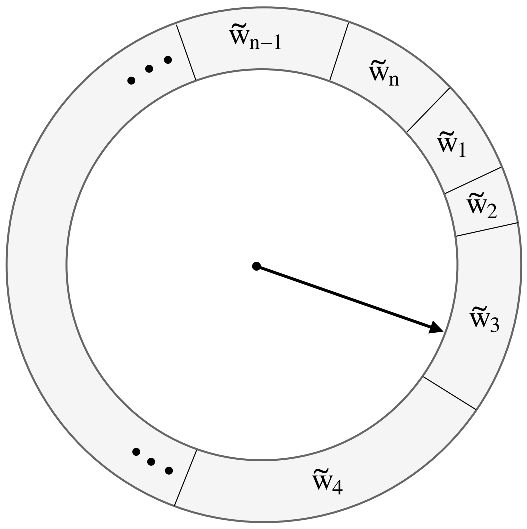


Figure 4.2: Sampling based on particle weight

Imagine that the arrow in the figure will stop randomly at any location if it is given a spin. The particle that corresponds to the field where the arrow points when it stops will be added to the set $\boldsymbol{\chi}_k$. When a temporally state hypothesis $\bar{\boldsymbol{x}}_k$ is added to $\boldsymbol{\chi}_k$, it is denoted $\boldsymbol{x}_k$. By repeating this procedure N times, the final set of particles $\boldsymbol{\chi}_k$ is obtained. From the illustration it is straightforward to see that particles with low weights rarely makes it to $\boldsymbol{\chi}_k$. Likewise will particles with high weights probably get sampled multiple times. In this way only the most plausible state hypothesises survive for the next iteration, without changing the original PDF. The final set of particles to be used as input in the next iteration is thus

$$\boldsymbol{\chi}_k = \left\{\boldsymbol{x}_k^i\right\}_{i=1}^N \tag{4.31}$$

As mentioned in step 2, all weights are set equal to $1/N$ after resampling. This is because the importance of a particle is now indirectly reflected by how many times it got sampled in this step. Put differently, the posterior PDF is described through the density of the resampled particles in $\boldsymbol{\chi}_k$.

Step 4: Evaluate the posterior PDF

The posterior PDF is now represented by the resampled particles. The PDF can then be evaluated using standard statistical measures. The expected value is typically the most interesting value in estimation problems. This is found by calculating the mean of each respective state at the given timestep

$$\boldsymbol{\mu} = E(\boldsymbol{x}_k|\boldsymbol{y}_k) \approx \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_k^i \tag{4.32}$$

The variance of the states may also be interesting as this tells the degree of uncertainty in the expected values. This is calculated from

$$Var(\boldsymbol{x}_k|\boldsymbol{y}_k) \approx \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_k^i - \boldsymbol{\mu})^2 \tag{4.33}$$

4.3.3 Effective resampling

As mentioned in the previous section, the resampling stage is necessary to make sure the particles maintain high weights. If not the PF will suffer from *degeneracy*, which means most of the particles gets negligible weights after a few iterations. This is illustrated in figure 4.3. At every iteration the spread of the particles becomes larger and the result is poor state estimates.

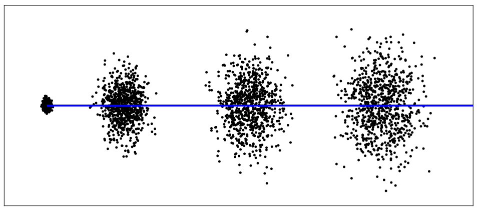


Figure 4.3: Particle evolution without resampling

In the SIR-implementation presented in the previous section, resampling is done in step 3 at every iteration. This step is computationally expensive and a proper choice of resampling algorithm is thus important. If a poor resampling algorithm is chosen computing power will be wasted unnecessary. In (Gustafsson, 2012) an algorithm named Ripley's method is recommended. This algorithm has a complexity linear to the number of particles N. If implementation at a sufficiently low level this complexity level cannot be beaten.

Another measure that will reduce computational time is to resample only when it is necessary. Even if the variance of the particle weights increases between iterations without resampling, it may stay below an acceptable limit some iterations forward in time. If this is the case, resampling is only necessary when the effective number of samples becomes to low. This is referred to as *sampling importance sampling*. As stated in (Arulampalam et al., 2002) and (Gustafsson, 2012) the effective number of samples can be approximated from the relation

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N}(w_k^i)^2} \tag{4.34}$$

The value of this function lies in the interval $1 \leq \hat{N}_{eff} \leq N$. A small value implies severe degeneracy, while $\hat{N}_{eff} = N$ is obtained directly after resampling. A threshold value N_T is chosen and resampling is done whenever $\hat{N}_{eff} \leq N_T$.

4.3.4 Tuning of the particle filter

The tuning of the PF is rather intuitive and consists of tuning parameters for the process noise and the measurements noise, just like the Kalman filters. In addition, the number of particles used in the filter is also considered as a tuning parameter.

When specifying the process noise, one is determining how much the particles should be spread between iterations. With a high degree of uncertainty in the mathematical model it is necessary with a widespread particle cloud. This makes sure at least some of the particles ends up in areas of interest. Likewise, if the model is very accurate, the particle cloud can be more concentrated.

The choice of tuning parameters for the measurement noise depends on what measurement distribution is chosen to evaluate the particles. On a general basis the tuning parameters must be chosen to shape the measurement distribution as desired. In the case of a Gaussian distribution for instance, the tuning parameter is the variance.

The last tuning parameter is the number of particles N. This parameter is significant both for the accuracy of the filter, and the computational effort needed to run the filter. It is conspicuous that a larger number of particles leads to more particles in the areas of interest, and thus yield better estimates. It is also evident that more particles leads to more calculations as each particle is evaluated individually. The designer must therefore choose a number of particles such that the balance between accuracy and computation time is satisfactory.

Some pitfalls might occur when tuning the PF. If the particles are not spread sufficiently between each iteration, all particles will be very similar to each other. Having an infinite number of identical particles is just as effective as having only one particle, due to all providing the same information. With low diversity in the particles one ends up with a poor representation of the posterior distribution of the estimated states. In worst case low spread of the particles might lead to the particles loosing track of the measurements and diverge. Imagine having only one particle (or many identical/low spreading). In such situation it does not matter what weight the particle is given as the filter is only able to resample this one particle anyways. Hence, the measurement becomes irrelevant and the filter runs on the mathematical model only. This will eventually lead to divergence of the state estimates.

Chapter 5

Implementation on a ROV

In the previous chapters the general theory behind the various Kalman filters and the PF were presented. In this chapter the theory is applied to design model based state estimators for a ROV. It is further shown how the Matlab/Simulink simulator was assembled, and how the experimental data was gathered.

5.1 The eXogenous Kalman Filter

The proposed XKF design is comprised by a KF at the second stage, having a NLO as the auxiliary estimator. This section presents how both these estimators can be derived using the mathematical ROV model in section 3.2. A bias term is also added to better handle imperfections in the model and for estimation of slowly varying forces.

5.1.1 Derivation of matrices for the second stage KF

The bias state estimates are implemented as a first order Markov model. It is further assumed that the ROV is passively stable in pitch and roll. Hence, these DOFs are not modeled in the filter. The resulting state space model of this system is given in Eq. (5.1).

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{\nu} \tag{5.1a}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} = -\boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{R}(\psi)^T\boldsymbol{b} + \boldsymbol{\tau} + \boldsymbol{w}_\nu \tag{5.1b}$$

$$\dot{\boldsymbol{b}} = -\boldsymbol{T}_b^{-1}\boldsymbol{b} + \boldsymbol{w}_b \tag{5.1c}$$

where $\boldsymbol{\eta} = [x\ y\ z\ \psi]^T \in \mathbb{R}^{4\times1}$, $\boldsymbol{\nu} = [u\ v\ w\ r]^T \in \mathbb{R}^{4\times1}$ and $\boldsymbol{b} = [b_x\ b_y\ b_z\ b_\psi]^T \in \mathbb{R}^{4\times1}$.

With the state space model in (5.1), the EKF design in (Candeloro et al., 2012b) can be used to derive the matrices for the second stage KF. From this model it is evident that the system matrix contains nonlinearities due to several rotation matrices relating $\{b\}$ and $\{n\}$ frame. In addition, both the damping and coriolis matrices may contain nonlinear functions with respect to the velocities of the ROV. This motivates for use of the EKF. The state space of the ROV model in 5.1 has the following structure

$$\dot{\boldsymbol{x}} = \boldsymbol{F}(\boldsymbol{x}) + \boldsymbol{B}\boldsymbol{\tau} + \boldsymbol{E}\boldsymbol{w} \tag{5.2}$$

where $\boldsymbol{x} = [\boldsymbol{\eta}\ \boldsymbol{\nu}\ \boldsymbol{b}]^T \in \mathbb{R}^{12\times1}$. Expanding the matrices in Eq. 5.2 yields

$$\dot{\boldsymbol{x}} = \underbrace{\begin{bmatrix} \boldsymbol{R}(\psi)\boldsymbol{\nu} \\ \boldsymbol{M}^{-1}(-\boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{R}^T(\psi)\boldsymbol{b}) \\ -\boldsymbol{T}_b^{-1}\boldsymbol{b} \end{bmatrix}}_{F(x)\in\mathbb{R}^{12\times1}} + \underbrace{\begin{bmatrix} 0 \\ \boldsymbol{M}^{-1} \\ 0 \end{bmatrix}}_{B\in\mathbb{R}^{12\times4}} \boldsymbol{\tau} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & \boldsymbol{M}^{-1} & 0 \\ 0 & 0 & \mathbf{1}_{4\times4} \end{bmatrix}}_{E\in\mathbb{R}^{12\times12}} \boldsymbol{w} \tag{5.3}$$

Noisy measurements of x, y, z and ψ are assumed available. It is also assumed a linear relation between measurements and states. The measurements model can thus be written on the form

$$\boldsymbol{y} = \boldsymbol{H}(\boldsymbol{x}) + \boldsymbol{v} \tag{5.4}$$

where expansion of this model yields

$$\boldsymbol{y} = \underbrace{\begin{bmatrix} \mathbf{1}_{4\times4} & \mathbf{0}_{4\times8} \end{bmatrix}}_{H\in\mathbb{R}^{4\times12}} \boldsymbol{x} + \boldsymbol{v} \tag{5.5}$$

After discretization the above matrices are ready for use in the second stage KF. Linearization of this model is done as described in section 4.2.2, where the linearization point is given by the auxiliary state estimator. Alternatively the EKF is implemented by instead linearize around the state estimates from the previous iteration. See appendix B.3 for the Matlab implementation of this observer.

5.1.2 The auxilliary state estimator

The auxilliary state estimator is implemented as a NLO. In (Refsnes, 2007) it is shown how such estimators can be implemented on underwater vehicles with GES stability properties. The derivation builds upon the same state space as in Eq. (5.1), but without the Coriolis matrix. The most important property of the auxilliary estimator is however the stability properties. Hence, neglecting the Coriolis matrix should still give a feasible linearization point at sufficiently low speeds. In addition, the system is assumed to be noise-free when designing the NLO. For this reason there are no terms in the NLO equations that are dedicated to evaluation of noise. The NLO derived in (Refsnes, 2007) which is used as an auxiliary estimator in the XKF is given in Eq. (5.6) below.

$$\dot{\hat{\boldsymbol{\eta}}} = \boldsymbol{R}(\psi)\hat{\boldsymbol{\nu}} + \boldsymbol{L}_1\tilde{\boldsymbol{\eta}} \tag{5.6a}$$

$$\boldsymbol{M}\dot{\hat{\boldsymbol{\nu}}} = -\boldsymbol{D}(\hat{\boldsymbol{\nu}})\hat{\boldsymbol{\nu}} + \boldsymbol{R}^T(\psi)(\boldsymbol{L}_2\tilde{\boldsymbol{\eta}} + \hat{\boldsymbol{b}}) + \boldsymbol{\tau} \tag{5.6b}$$

$$\dot{\hat{\boldsymbol{b}}} = -\boldsymbol{T}_b^{-1}\hat{\boldsymbol{b}} + \boldsymbol{L}_3\tilde{\boldsymbol{\eta}} \tag{5.6c}$$

where $\boldsymbol{\eta} = [x\ y\ z\ \psi]^T \in \mathbb{R}^{4\times1}$, $\boldsymbol{\nu} = [u\ v\ w\ r]^T \in \mathbb{R}^{4\times1}$ and $\boldsymbol{b} = [b_x\ b_y\ b_z\ b_\psi]^T \in \mathbb{R}^{4\times1}$. The matrices $\boldsymbol{L}_1 = diag[l_{11}\ l_{12}\ l_{13}\ l_{14}]^T \in \mathbb{R}^{4\times1}$, $\boldsymbol{L}_2 = diag[l_{21}\ l_{22}\ l_{23}\ l_{24}]^T \in \mathbb{R}^{4\times1}$ and $\boldsymbol{L}_3 = diag[l_{31}\ l_{32}\ l_{33}\ l_{34}]^T \in \mathbb{R}^{4\times1}$ are tuning matrices specified by the designer.

The idea behind the NLO is to use the mathematical model of the ROV together with an injection term to correct for deviations in the model. The injection term is comprised by the measured states and their corresponding estimates. With measurements available on $\boldsymbol{\eta}$, the injection term becomes $\tilde{\boldsymbol{\eta}} = \boldsymbol{\eta}_{meas} - \hat{\boldsymbol{\eta}}$ multiplied with the tuning matrices. Higher values in the tuning matrices thus imply more aggressive convergence of the states. The Simulink-diagram of the NLO is attached in appendix A.2.1.

5.2 Particle Filter

A proposal of how the general implementation of the PF from chapter 4 could be implemented with the mathematical model of a ROV is given in this section. The implementation is limited to the three DOFs surge, sway and yaw. This implies that each particle in the filter contains a hypothesis of $\boldsymbol{\eta}_{hyp} = [x\ y\ \psi]^T$ and $\boldsymbol{\nu}_{hyp} = [u\ v\ r]^T$. The number of states are kept at a minimum due to the high computational cost associated with the PF. See appendix B.1 for the Matlab code derived on basis of the equations presented in this section.

Input to new iteration

Input to every new iteration are the measured thrust $\boldsymbol{\tau}_k = [\tau_{x,k}\ \tau_{y,k}\ \tau_{\psi,k}]$, the pose measurement $\boldsymbol{y}_k = [x_{meas,k}\ y_{meas,k}\ \psi_{meas,k}]$, and the set of N resampled particles $\boldsymbol{\chi}_{k-1}$ from the previous iteration. Each of the particles in $\boldsymbol{\chi}_{k-1}$ contains a state hypothesis for each of the six states in the filter. Hence, particle i in this set is expressed as $\boldsymbol{\chi}_{k-1}^i = [\boldsymbol{\eta}_{hyp,k-1}^i\ \boldsymbol{\nu}_{hyp,k-1}^i]$.

Step 1: Guess states using mathematical model

In the first step of the PF the six state hypotheses within each particle are augmented using the mathematical model of the ROV. Keep in mind that a mathematical model will never fully capture the true vehicle dynamics, and the values in $\boldsymbol{\tau}_k$ comes from noisy measurements. For this reason every particle is updated using slightly different values of $\boldsymbol{\tau}_k$ by adding a random parameter $\boldsymbol{\epsilon}_\tau$. In this way diversity in the particles are ensured.

The velocity rate of change used for calculations in particle i, denoted $\boldsymbol{\nu}^i_{rate,k}$, can be found from the recursive implementation of the kinetic equation of the ROV. This is expressed in Eq. (5.7) below.

$$\boldsymbol{\nu}^i_{rate,k} = \boldsymbol{M}^{-1}[\boldsymbol{\tau}_k + \boldsymbol{\epsilon}^i_\tau - \boldsymbol{D}(\boldsymbol{\nu}^i_{hyp,k-1})\boldsymbol{\nu}^i_{hyp,k-1} - \boldsymbol{C}(\boldsymbol{\nu}^i_{hyp,k-1})\boldsymbol{\nu}^i_{hyp,k-1}] \tag{5.7}$$

Here, $\boldsymbol{\nu}^i_{hyp,k-1}$ is the velocity hypothesis within particle i from the previous iteration. In the implementation, $\boldsymbol{\epsilon}^i_\tau$ takes random values from a Gaussian distribution with variance $\boldsymbol{\sigma}_\tau = [\sigma_x \ \sigma_y \ \sigma_\psi]^T$. The size of the intervals where $\boldsymbol{\epsilon}^i_\tau$ takes random values determines the spreading of the particles. Hence, $\boldsymbol{\sigma}_\tau$ is a tuning parameter which corresponds to the uncertainty in the mathematical model.

The velocity hypotheses contained in particle i are now augmented by integrating $\boldsymbol{\nu}^i_{rate,k}$ with the previous velocity hypotheses $\boldsymbol{\nu}^i_{hyp,k-1}$ of the given particle. This yields the following numerical integration

$$\boldsymbol{\nu}^i_{hyp,k} = \boldsymbol{\nu}^i_{hyp,k-1} + h \cdot \boldsymbol{\nu}^i_{rate,k} \tag{5.8}$$

The new velocity hypothesis from Eq. (5.8) can then be used with the kinematic equation of the ROV to augment the position and heading hypothesis $\boldsymbol{\eta}^i_{hyp,k-1}$. This is done using using Eq. (5.9) below.

$$\boldsymbol{\eta}^i_{hyp,k} = \boldsymbol{\eta}^i_{hyp,k-1} + h \cdot \boldsymbol{R}^n_b(\psi^i_{hyp,k-1}) \cdot \boldsymbol{\nu}^i_{hyp,k} \tag{5.9}$$

where $\boldsymbol{R}^n_b(\psi^i_{hyp,k-1})$ is a rotation matrix and $\psi^i_{hyp,k-1}$ is the yaw hypothesis of particle i from previous iteration.

Repeating this procedure for all N particles gives the temporarily set of particles

$$\bar{\mathcal{X}}_k = \{\boldsymbol{\eta}^i_{hyp,k} \ \ \boldsymbol{\nu}^i_{hyp,k}\}^N_{i=1} \tag{5.10}$$

Step 2: Weight the particles

From step 1 are now N temporarily particles, each with their own state hypothesis $[\boldsymbol{\eta}^i_{hyp,k} \ \boldsymbol{\nu}^i_{hyp,k}]$, available. Pose measurements given by $\boldsymbol{y}_k = [x_{meas,k} \ y_{meas,k} \ \psi_{meas,k}]$ are also available. Only $\boldsymbol{\eta}^i_{hyp,k}$ can be directly compared to $\boldsymbol{y}_k$, but recall that $\boldsymbol{\eta}^i_{hyp,k}$ is calculated on basis of the values in $\boldsymbol{\nu}^i_{hyp,k}$. This implies that if $\boldsymbol{\eta}^i_{hyp,k}$ fits well with the pose measurement $\boldsymbol{y}_k$, it is because the corresponding velocity hypotheses $\boldsymbol{\nu}^i_{hyp,k}$ is also close to the true velocities. In this way the velocity estimates are indirectly compared to $\boldsymbol{y}_k$. Each particle can thus be imagined to exists in the three dimensional space illustrated in figure 5.1.

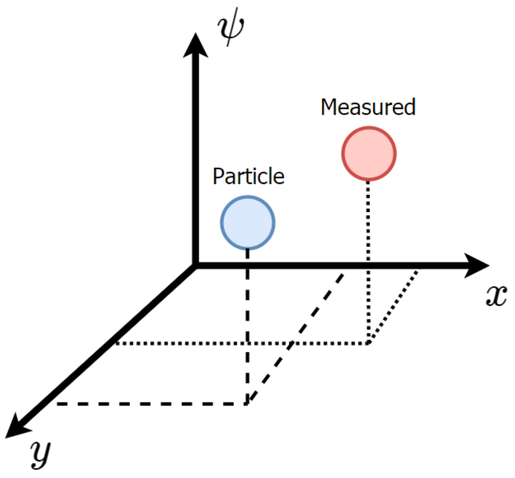


Figure 5.1: Measurement and particle

The weights assigned in this step is therefore calculated on basis of how well the particle fits with the measurement in all three measured dimensions. Assuming the noise of the measurements are normally distributed, the multivariate normal distribution in Eq. (5.11) can be used to calculate the weight of each particle.

$$p(\boldsymbol{\eta}^i_{hyp,k}|\boldsymbol{y}_k,\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}} exp\left(-\frac{1}{2}(\boldsymbol{\eta}^i_{hyp,k}-\boldsymbol{y}_k)\boldsymbol{\Sigma}^{-1}(\boldsymbol{\eta}^i_{hyp,k}-\boldsymbol{y}_k)^T\right) \tag{5.11}$$

The co-variance matrix is denoted $\boldsymbol{\Sigma}$, and $|\boldsymbol{\Sigma}|$ is the determinant of this. The number of variables n is in this case equal to three. Assuming there are no co-variance in the noise of the pose measurements, the co-variance matrix is written on the form

$$\Sigma = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\psi^2 \end{bmatrix} \tag{5.12}$$

where σ_x^2, σ_y^2 and σ_ψ^2 reflects the uncertainty of each measurement.

As discussed earlier the PF can also handle other distributions. The only requirement is that the distribution of the measurement is known. In case of another distribution of the measurements, one simply replaces Eq. (5.11) with the desired evaluation function.

Step 3 and 4: Resampling and evaluation

Step 3 and 4 are similar to the corresponding steps in the general implementation. By resampling only the high-weighted particles in the temporarily set $\bar{\chi}_k$ survives. The resampled and high-weighted particles are stored in a final set of particles χ_k. The mean of all the state hypotheses in χ_k are used as output from the filter. Then χ_k are used as input for next iteration.

5.3 Simulator

To evaluate the different filter designs, a simulator was developed in Matlab/Simulink. A fully functional simulator must contain all three GNC-blocks presented in the introduction. The navigation block is the foundation of the thesis and different implementations are presented in the previous sections. This section therefore briefly presents how the guidance-block and the controller-block were implemented in the simulator. The ROV-model used in simulations are also presented.

5.3.1 ROV model

The ROV model used in the simulator is based on the Videoray Pro 4 ROV and is developed by Bent O. Arnesen. No changes has been made to this model in this thesis. Details and assumptions made, beyond what is presented in this section, can be found in (Arnesen and Schjølberg, 2016) where the model is also compared to real experiments. The mathematical model used in the implementation follows the general structure of vessel modeling presented in chapter 3.2. It is a 4 DOF model as roll and pitch is passively stable for this ROV. The state vector of the ROV is thus comprised by $\boldsymbol{\eta} = [x\ y\ z\ \psi]^T$ and $\boldsymbol{\nu} = [u\ v\ w\ r]^T$.

Some general assumptions the model builds upon: center of mass is zero in x- and y-direction, density is evenly distributed, off-diagonal terms in $\boldsymbol{M}$ and $\boldsymbol{D}$ are assumed to be small compared to diagonal terms, low velocities, and the center of gravity (CG) and the center of buoyancy (CB) coincides. All matrices used in the implementation are given next, and the parameter values are listed in appendix A.1.

- The $\{b\}$ to $\{n\}$ reference frame transformation matrix is given by

$$\boldsymbol{J}(\boldsymbol{\eta}) = \boldsymbol{J}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 & 0 \\ \sin(\psi) & \cos(\psi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.13}$$

- $\boldsymbol{M} = \boldsymbol{M}_{RB} + \boldsymbol{M}_A$ is the system inertia matrix comprised of the rigid-body and added mass matrices.

$$\boldsymbol{M}_{RB} = \begin{bmatrix} m & 0 & 0 & 0 \\ 0 & m & 0 & 0 \\ 0 & 0 & m & 0 \\ 0 & 0 & 0 & I_z \end{bmatrix}, \quad \boldsymbol{M}_A = \begin{bmatrix} -X_{\dot{u}} & 0 & 0 & 0 \\ 0 & -Y_{\dot{v}} & 0 & 0 \\ 0 & 0 & -Z_{\dot{w}} & 0 \\ 0 & 0 & 0 & -N_{\dot{r}} \end{bmatrix} \tag{5.14}$$

- $\boldsymbol{C}(\boldsymbol{\nu}) = \boldsymbol{C}_{RB}(\boldsymbol{\nu}) + \boldsymbol{C}_A(\boldsymbol{\nu})$ is the Coriolis-centripetal matrix comprised of the rigid-body and added mass effects when the body frame rotates about an inertial frame.

$$\boldsymbol{C}_{RB}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & 0 & -mv \\ 0 & 0 & 0 & mu \\ 0 & 0 & 0 & 0 \\ mv & -mu & 0 & 0 \end{bmatrix}, \quad \boldsymbol{C}_A(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & 0 & Y_{\dot{v}}v \\ 0 & 0 & 0 & -X_{\dot{u}}u \\ 0 & 0 & 0 & 0 \\ -Y_{\dot{v}}v & X_{\dot{u}}u & 0 & 0 \end{bmatrix} \tag{5.15}$$

- $\boldsymbol{D}(\boldsymbol{\nu}) = \boldsymbol{D}_L + \boldsymbol{D}_{NL}(\boldsymbol{\nu})$ is the linear and non-linear hydrodynamic damping. There are many sources to hydrodynamic damping, where the most important ones are potential damping, skin friction, wave drift damping and vortex shedding (Faltinsen, 1993).

$$\boldsymbol{D}_L = \begin{bmatrix} -X_u & 0 & 0 & 0 \\ 0 & -Y_v & 0 & 0 \\ 0 & 0 & -Z_w & 0 \\ 0 & 0 & 0 & -N_r \end{bmatrix} \tag{5.16}$$

$$\boldsymbol{D}_{NL}(\boldsymbol{\nu}) = \begin{bmatrix} -X_{|u|u}|u| & 0 & 0 & 0 \\ 0 & -Y_{|v|v}|v| & 0 & 0 \\ 0 & 0 & -Z_{|w|w}|w| & 0 \\ 0 & 0 & 0 & -N_{|r|r}|r| \end{bmatrix} \tag{5.17}$$

- $\boldsymbol{g}_0$ is the restoring forces that try to bring the system to its natural equilibrium. It is implemented as

$$\boldsymbol{g}_0 = \begin{bmatrix} 0 \\ 0 \\ -(W - B) \\ 0 \end{bmatrix} \tag{5.18}$$

  where W is the gravitational force and B is the buoyancy force.

- $\boldsymbol{\tau}$ is the control input to the ROV model. The model is actuated in surge, sway, heave and yaw which yields $\boldsymbol{\tau} = [\tau_x \ \tau_y \ \tau_z \ \tau_\psi]$.

5.3.2 Guidance system

To make the ROV follow a desired path a guidance system was implemented. One of the most convenient ways to make desired paths is by specifying waypoints the ROV should visit. Then make the ROV follow the lines connecting the waypoints. This can be achieved using the *line of sight* (LOS) guidance law (Breivik and Fossen, 2008). The idea behind this algorithm is to generate a desired heading ψ_d which makes the ROV converge towards the line segments between the waypoints. This also requires the ROV to maintain a desired surge velocity. In this context there are two main guidance objectives. If the ROV should be at a given location at a given time it is referred to as *path tracking*, where both temporal and spatial constraints should be met. In the other case only spatial constraints are considered and this is referred to as *path following* (Breivik et al., 2008). Only the latter is implemented in this thesis. The field of LOS-guidance is extensive and the method chosen in the implementation was the *Lookahead-Based Steering law*. The geometry of this implementation is illustrated in figure 5.2.

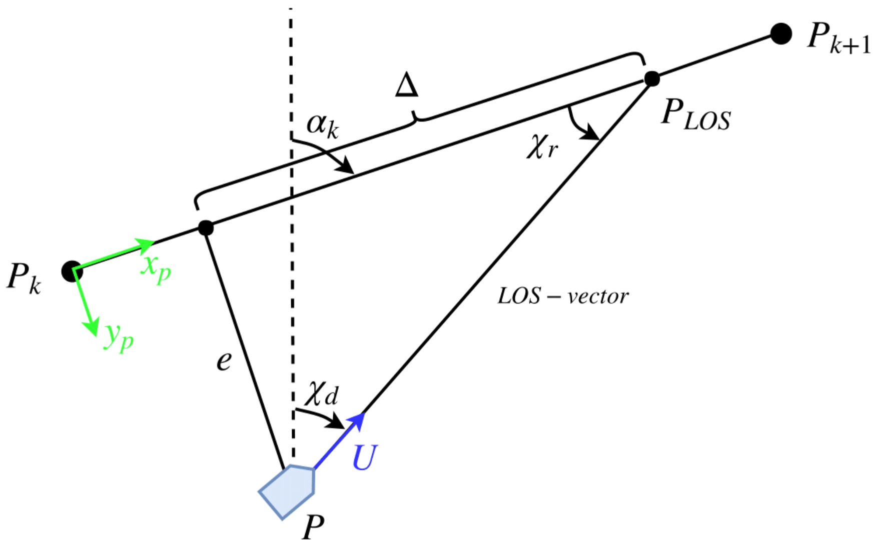


Figure 5.2: Lookahead-based straight line LOS

A vector denoted the *LOS-vector* goes from the vessel to a point $\boldsymbol{P}_{LOS} = [x_{LOS}, y_{LOS}]^T$ located at the path tangential line at lookahead distance $\Delta > 0$. The course angle is thus comprised by two parts.

$$\chi_d(e) = \alpha_k + \chi_r(e) \tag{5.19}$$

The first term is the path-tangential angle, i.e. the tilt angle of the path reference frame (x_p, y_p) relative to the global reference frame

$$\alpha_k = \tan^{-1}\left(\frac{y_{k+1} - y_k}{x_{k+1} - x_k}\right) \tag{5.20}$$

The second term is the velocity-path relative angle

$$\chi_r(e) = \tan^{-1}\left(\frac{-e}{\Delta}\right) \tag{5.21}$$

where e is the referred to as the *cross-track error*. This is the shortest distance between the vessel and the path, and may be found by simple geometric derivations.

The value of Δ greatly affects the manoeuvring characteristics of the vessel. A smaller value implies a more aggressive convergence towards the path compared to a larger value (Lekkas and Fossen, 2014). This is also intuitive by looking at figure 5.2. Further, it is clear that the velocity-path relative angle becomes zero when the vessel has converged fully to the path as this renders $e = 0$. This implies the heading is set equal to the path-tangential angle.

Switch way-point
Unless the vessel is going to follow a straight line from start to finish, it is necessary to specify the desired path by multiple straight lines when using straight line LOS. Hence, a method to determine when to follow the next line segment, i.e. switch way-point, must be implemented. In the two-dimensional plane this is solved by using a *circle of acceptance*. More specific, a circle with radius R is placed around the waypoint (x_{k+1}, y_{k+1}). If the position $(x(t), y(t))$ of the vessel satisfies

$$[x_{k+1} - x(t)]^2 + [y_{k+1} - y(t)]^2 \leq R_{k+1}^2 \tag{5.22}$$

the next waypoint is selected.

Alternatively, in three dimensions one could either implement a *cylinder of acceptance* or a *sphere of acceptance*. The principle is the same as the circle of acceptance, and what method to choose depends on the application. In the simulator developed in this thesis, a circle of acceptance was used. This means depth could have any arbitrary value inside the circle. This is equivalent to a cylinder of acceptance with infinite length.

5.3.3 Controller

Controllers for heading and surge speed were implemented in the simulator for the ROV to be able to follow the path from the LOS-guidance. Due to the ROV having actuators in sway, it is also possible to make a controller in this DOF to reduce the cross-track error even faster. In addition, a controller for depth may also be necessary. This was however not implemented.

Advanced controllers may take the system model into consideration. Examples on such controllers are the *feedback linearization controller* and the *backstepping controller* (Fossen, 2011). However, the Proportional-Integral-Derivative (PID) controller proved to work well in the simulator. Input to such a controller is the error value of a given state, and it is thus independent of the mathematical model of the system. That is, the state has a current value $x(t)$ and a desired value $x_d(t)$. The error term is then expressed as $e(t) = x(t) - x_d(t)$, and the PID-controller seeks to minimize this value by finding the optimal thrust value $u(t)$. As the name suggests, the PID-controller can be divided in three terms and a mathematical representation is given by Eq.

(5.23).

$$u(t) = K_p \cdot e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \cdot \frac{d}{dt} e(t) \tag{5.23}$$

The controller gains $K_p \geq 0$, $K_i \geq 0$ and $K_d \geq 0$ determines the contribution of each term on $u(t)$. The proportional term calculates a thrust value proportional to the error and the derivative term calculates a thrust value proportional to the rate of change in the error. The integral term sums up the errors over time which eventually will lead to no stationary error. The gains can also take the value zero, resulting in a PD-controller or a PI-controller for instance. See (Ang et al., 2005) for more details on the PID-controller.

In the simulator, PID-controllers were implemented for surge velocity using $e(t) = u(t) - u_d$. Where $u(t)$ is current surge speed and u_d is the constant desired surge speed. For heading the error $e(t) = \psi(t) - \psi_d(t)$ was used, where $\psi(t)$ and $\psi_d(t)$ are current and desired heading angles respectively.

5.3.4 Putting all modules together

With the ROV-model and all three GNC-blocks being specified, what remains is to connect all modules to render the simulator complete. In a real system the modules are connected as presented in illustration 1.1 in the introduction, where all modules are part of a closed loop. In the simulator however, not all modules were part of the closed loop as can be seen from the draft of the Simulink diagram in figure 5.3.

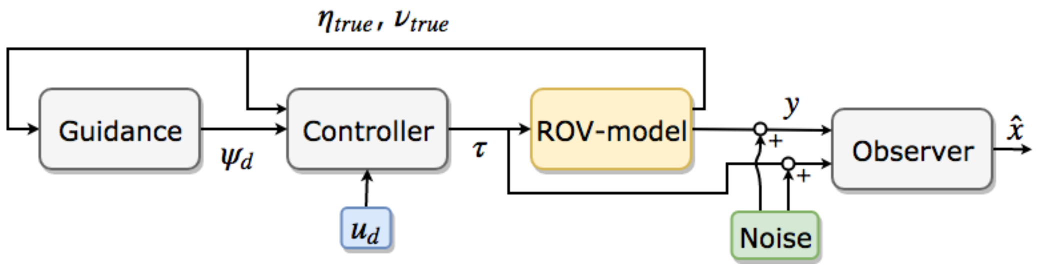


Figure 5.3: Draft of Simulink diagram

In this implementation the observer is placed outside the closed loop and the guidance-module, controller-module and ROV-model runs on noise-free/true values. This is because it is a well-known fact that observers and controllers influence each other in a closed loop. As the thesis aims to compare observers, it is desirable to eliminate any sources to disturbance.

As the closed loop runs on noise-free estimates it is necessary to add noise to the signals sent to the observer. Hence, Gaussian white noise is added to $\boldsymbol{\tau} = [\tau_x\ \tau_y\ \tau_z\ \tau_\psi]$ and $\boldsymbol{y} = [x\ y\ z\ \psi]$ and is illustrated by the noise block in figure 5.3. Further it is seen that the desired surge speed

u_d is set as an external input to the controller. This implies that this value is constant and not determined by the guidance block. See appendix A for the real Simulink diagrams.

5.4 Experimental setup

In order to verify the implementation of the PF, an experiment to gather sensor data was conducted in the NTNU MC-lab. This is a 40 meter long, 6.45 meter wide and 1.5 meter deep basin designed for cybernetic purposes. One of the core features of this lab is the Qualisys motion capturing system. This is a camera system comprised by both surface- and underwater cameras that are able to track a set of markers with high accuracy. Placing these markers on a vessel makes it possible to keep track of the vessels true position and attitude in the basin.

All sensor data were logged in the MC-lab and later processed using the PF offline. This follows the same analogy as in the simulator implementation. I.e. the observers put to the test are not part of the closed loop.

5.4.1 Available sensor data in MC-lab

The observers developed in previous sections essentially combines two sensor measurements in each DOF. First sensor data is the measured pose $\boldsymbol{\eta} = [x \ y \ z \ \psi]^T$ that is assumed to be available from noisy mesaurements. These measurements are generated by adding noise to the measurements from the Qualisys camera system in the MC-lab. Secondly is the the measured thrust $\boldsymbol{\tau} = [\tau_x \ \tau_y \ \tau_z \ \tau_\psi]$ that is used to approximate the velocities of the ROV by transforming $\boldsymbol{\tau}$ to the velocity vector $\boldsymbol{\nu}$ using a mathematical model of the ROV. For this reason it would have been desirable to measure the thrust in the experiments to investigate how all the the proposed observer designs work with real data. However, due to the available equipment not being able to measure thruster RPM, this was unfortunately not possible.

Due to this only the PF is tested with experimental data, using measurements obtained from an Inertial Measurement Unit (IMU) instead of the thruster model. Even if this is not directly comparable with the simulations, it will still show if the proposed PF design works in a more realistic setting. It will indeed also show if the PF is able to handle IMU-data well. While the PF can easily be altered to fit with IMU-data, the same is not true for the XKF proposed in this thesis. The NLO which ensures GES estimates requires measurement of thrust. Implementations of the XKF that fits with IMU-data may instead be found in (Seel and Schjølberg, 2016) and (Stovner, 2018).

5.4.2 BlueROV

Due to the experimental data not being based on the thruster model it did not matter what ROV was used to gather sensor data in the MC-lab. Rather than using the Videoray Pro 4 as in the simulations, a BlueROV 2 was used in the MC-lab.

The BlueROV 2 is a small observation class ROV where a total of six thrusters provide actuation in surge, sway, heave, roll and yaw. Pitch is thus the only DOF where this ROV is not actuated. Communication with the ROV is done using a tether connected to the MC-lab network, while power is supplied from an on-board battery-pack. A picture of NTNU's BlueROV is shown i figure 5.4.

IMU-data comprised by 3-DOF gyroscope, accelerometer and magnetometer are together with pressure/depth measurement available from the ROV. In addition it is possible to get video-feed from the on-board camera. It is also worth mention that the BlueROV is based on open-source electronics and software. Hence, this system is well-suited for configurations by the user. More information about the BlueROV is available at the website (BlueRobotics).

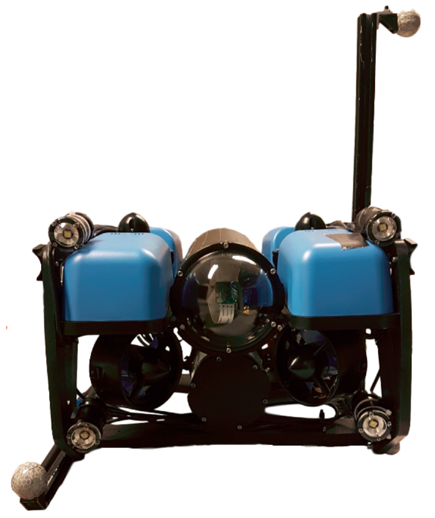


Figure 5.4: BlueROV with markers for Qualisys camera system

5.4.3 The Qualisys Motion Capture system

As previously mentioned the Qualisys Motion Capture (QMC) system was used to get the position and attitude of the ROV in the basin. Due to the high accuracy of this system the measurements from the QMC is regarded as the true state values. A set of minimum four reflective markers are placed on the object that is to be tracked by the QMC. Then a *rigid body* is made from the marker configuration in the Qualisys Track Manager (QTM) software. The rigid body is essentially an unique pattern of markers, and whenever the QTM recognise this pattern it knows it is the object of interest.

A total of six underwater cameras are available for underwater tracking in the NTNU MC-lab. Each camera emits infrared light. This light is reflected by the markers on the ROV and then caught by the camera lenses. This makes the QMC sensitive to various disturbances. It is essential that the cameras are able to recognise the rigid body, if not the signal drops out. The markers must therefore be placed clearly visible on the ROV. Placing the markers on long aluminium bars makes it easier for more cameras to see the markers simultaneously. Due to marker visibility the rather large volume of the basin is reduced to a much smaller operational space when using the QMC. Too close to either side of the basin makes the cameras loose track of the rigid body. It is also desirable to have as little light disturbance as possible. Both ceiling lights and internal lights on the ROV may disturb the cameras. Getting long measurement series without drop-outs is therefore sometimes challenging.

5.4.4 IMU sensor characteristics

As previously mentioned IMU-data were used as a replacement of the thruster model. An typical IMU provides measurements from 3-DOF gyroscopes, accelerometers and magnetometers. For the objectives of this thesis, the magnetometer is not necessary and will not be further discussed. The accelerometer measurements are integrated once to obtain velocities, and twice to obtain positions. Similarly, the gyro measures angular velocity and is integrated once to obtain attitude. Hence, the augmentation of the velocity hypotheses in Eq. (5.8) of the PF is now based on the integrated linear accelerations, and the angular velocities from the IMU. See appendix B.2 for the Matlab code of the PF using IMU data.

The mathematical representation of the measurements from an acceleromter and a rate gyro, as represented in (Fossen, 2011), is given below. These equations assumes the axis of the measurement-frame and the body-frame are parallel. This is also known as the *strapdown* assumption.

$$\boldsymbol{a}^b_{imu} = \boldsymbol{R}^b_n(\Theta)(\dot{\boldsymbol{\nu}}^n + \boldsymbol{g}^n) + \boldsymbol{b}^b_{acc} + \boldsymbol{w}^b_{acc} \tag{5.24}$$

$$\boldsymbol{w}^b_{imu} = \boldsymbol{w}^b + \boldsymbol{b}^b_{gyro} + \boldsymbol{w}^b_{gyro} \tag{5.25}$$

Here, $\boldsymbol{\Theta} = [\phi\ \theta\ \psi]^T$ is a vector with the Euler angles and $\boldsymbol{R}^b_n(\Theta)$ is the rotation matrix from $\{n\}$ to $\{b\}$. The gravitational force is represented as $\boldsymbol{g}^n = [0\ 0\ g]$, where $g \approx 9.81$ [m/s^2]. Both the accelerometer and the gyro are prone to biases and zero mean measurement noise, here denoted $\boldsymbol{b}^b_{acc}, \boldsymbol{b}^b_{gyro}$ and $\boldsymbol{w}^b_{acc}, \boldsymbol{w}^b_{gyro}$ respectively.

5.4.5 Sensor fusion

Integrating IMU-data to obtain velocities, positions and attitudes will quickly lead to large offsets from the true states. Sensor biases, misalignments and temperature variations are all sources to sensor drift (Fossen, 2011). Hence, estimation based on IMU-data alone is only applicable for short time intervals. The measurements from the QMC, where noise is added, do not suffer from sensor drift over time. Even if bias is present in these measurements, it would not get amplified from integration as position and heading are measured directly.

The two measurement sources complement each other well. The IMU-data are used to estimate velocities and positions some iterations forward in time. At approximately every 5th output from the IMU, new sensor data from the QMC are also available. This is due to the much lower sampling rate of the QMC. The QMC-data which is free from offsets, but noisy, is then used to correct any offsets that have built up from the integration of IMU-data.

5.4.6 Controlling the BlueROV

An Xbox controller was used to steer the BlueROV in the basin. The two joysticks on the Xbox controller was used to determine desired thrust in surge, sway, heave and yaw. In this way the user was free to steer the ROV independently in each of these 4 DOFs. In heave however, a PID-controller was implemented to make the ROV maintain a desired depth automatically. All these features, plus the communication with the QMC, required lots of software implementation. Everything regarding the implementation was organized by Mikkel Cornelius Nielsen. The resulting control loop for the user is seen i figure 5.5.

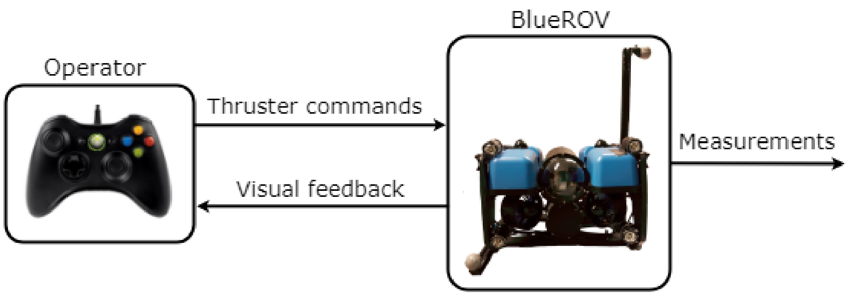


Figure 5.5: Human in-the-loop ROV control

5.4.7 Challenges with setup

One of the main challenges with the setup used in the lab was the underactuation of the BlueROV. Without actuation in pitch the ROV suffered from severe pitching when accelerating. As can be seen from Eq. 5.24, an accelerometer alone cannot determine if acceleration is due to velocity changes or gravity. Hence, the pitching introduced large uncertainties in the measured accelerations.

Chapter 6

Simulation results

In this chapter the results from the Matlab/Simulink implementation described in chapter 5 are presented. Computer simulations offers the opportunity of conducting multiple test-runs of the ROV-system at a relatively short amount of time. In addition it does not require access to the ROV-hardware or lab-facilities. Hence, when evaluating the state estimator designs, simulations makes it simpler to test out new ideas.

In simulations one can also benefit from the fact that all information about the system is available. With the real states being directly observable allows for effective evaluation of the performance of the state estimators. The Root-Mean-Square (RMS) error estimate is used as a measure of performance of the simulations in this thesis, and are calculated according to

$$RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_{est}^{i} - x_{true}^{i})^2} \tag{6.1}$$

where N is the number of samples. A lower RMS value implies there is less offset and fluctuation from the true state estimates.

Albeit simulations have many advantages, it is also important to know the limitations of a simulator. Ultimately the simulator is only a mathematical approximation of the real ROV-system, and simulations are only as good as the model they are based upon. For this reason one cannot expect the simulator to behave identical to the real system. This is also confirmed in (Arnesen and Schjølberg, 2016) where the ROV-model used for simulations in this thesis is compared with real experiments. Here it is seen that both simulations and experiments behave quite similar, but not identical. However, the goal of this thesis is not to examine how well simulations mimics a real Videoray Pro 4 ROV. It is rather to examine which filter is best suited to evaluate information from a given mathematical model and a number of given measurements. Hence, this comparison is to some degree independent of the accuracy of the simulator as it will still show which filter works best using this particular model. However, some filters may perform better than others if unmodeled effects are accounted for. An accurate model is therefore anyways desirable.

In the next subsections follows a presentation of each simulation case. Then relevant plots from each case are presented, before the RMS values of positions and velocities are listed. Different simulation runs yields slightly different RMS values due to the random characteristics of the noise. For this reason the RMS values presented later are comprised of the mean from 10 different simulations.

6.1 Simulation specifications

All particle filter simulations assumes linear and noisy measurements are available in surge, sway and yaw. The same is true for the Kalman filter simulations, but here it is also assumed that a linear and noisy measurement is available in heave. Five different cases are conducted in order to compare the performance of the filters described in chapter 5. These simulations are designed with the goal of revealing benefits and weaknesses of each filter.

The ROV follows the same lawn mower pattern in all simulations, where the LOS-algorithm has a lookahed distance of $\Delta = 0.5$ [m] and a circle of acceptance of $R = 1$ [m]. The small value of Δ means the ROV will converge aggressively to the desired path. The rov follows the path with a speed of $u = 0.3$ $[m/s]$.

The simulations are run for 120 seconds with a timestep of $h = 0.05$. The noise added to the simulations are Gaussian with a variance specified in table 6.1.

Table 6.1: Noise added to simulations

	Surge	Sway	Heave	Yaw
σ_y^2	0.02	0.02	0.02	0.01
σ_τ^2	0.02	0.02	0.02	0.02

6.1.1 Simulation 1: Steady state behaviour of EKF and XKF

The first simulation seeks to compare the steady state behaviour of the EKF and the XKF. This will in turn tell which of the two filters work best under normal operation. Hence, both filters are provided with the correct initial values $\boldsymbol{p}_0 = [0\ 0\ \ -10\ 90°]$, $\boldsymbol{v}_0 = [0\ 0\ 0\ 0]$ and $\boldsymbol{b}_0 = [0\ 0\ 0\ 0]$. The initial covariance is set to $P_0 = 0.1$ for all states. The observer gains are summarized in table 6.2.

Table 6.2: Simulation 1: Observer gains

EKF	XKF
$\boldsymbol{R}$ = diag[0.2, 0.2, 0.2, 0.08]	$\boldsymbol{R}$ = diag[0.2, 0.2, 0.2, 0.08]
$\boldsymbol{Q}$ = diag[0, 0, 0, 0, 5, 5, 5, 0.01, 1, 1, 1, 0.8]	$\boldsymbol{Q}$ = diag[0, 0, 0, 0, 5, 5, 5, 0.01, 1, 1, 1, 0.8]
	$\boldsymbol{L}_1$ = diag[1.2, 1.2, 1.2, 5]
	$\boldsymbol{L}_2$ = diag[10, 10, 10, 1]
	$\boldsymbol{L}_3$ = diag[3, 3, 3, 2.5]

The mathematical model of the ROV implemented in the filters is rather accurate in this simulation. The mass is set to 7.5 [kg] which is $\sim 7\%$ offsett from the real value. This leads to both the mass matrix and the Coriolis matrix being slightly different from the real ones. The damping parameters are also reduced to 95% of the original values.

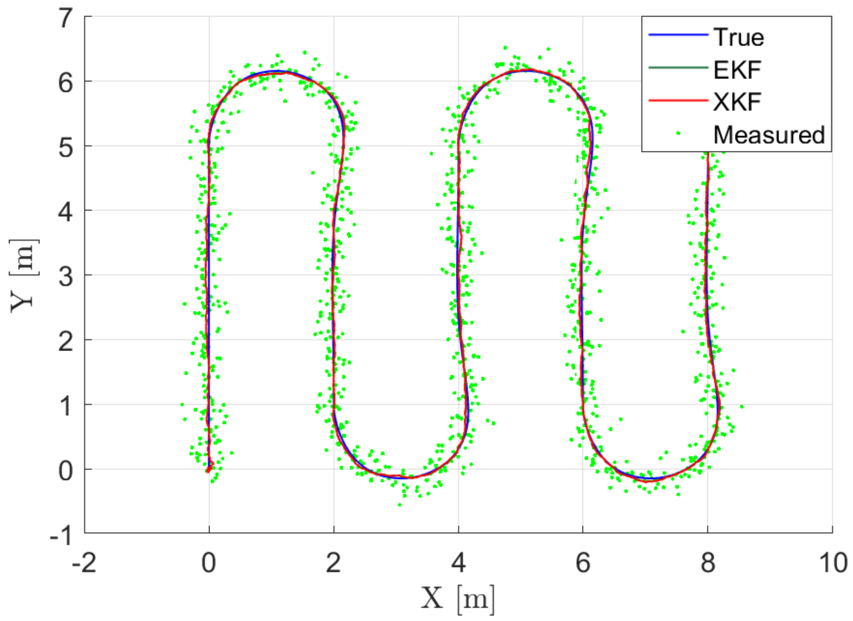


Figure 6.1: Simulation 1: XY-plot

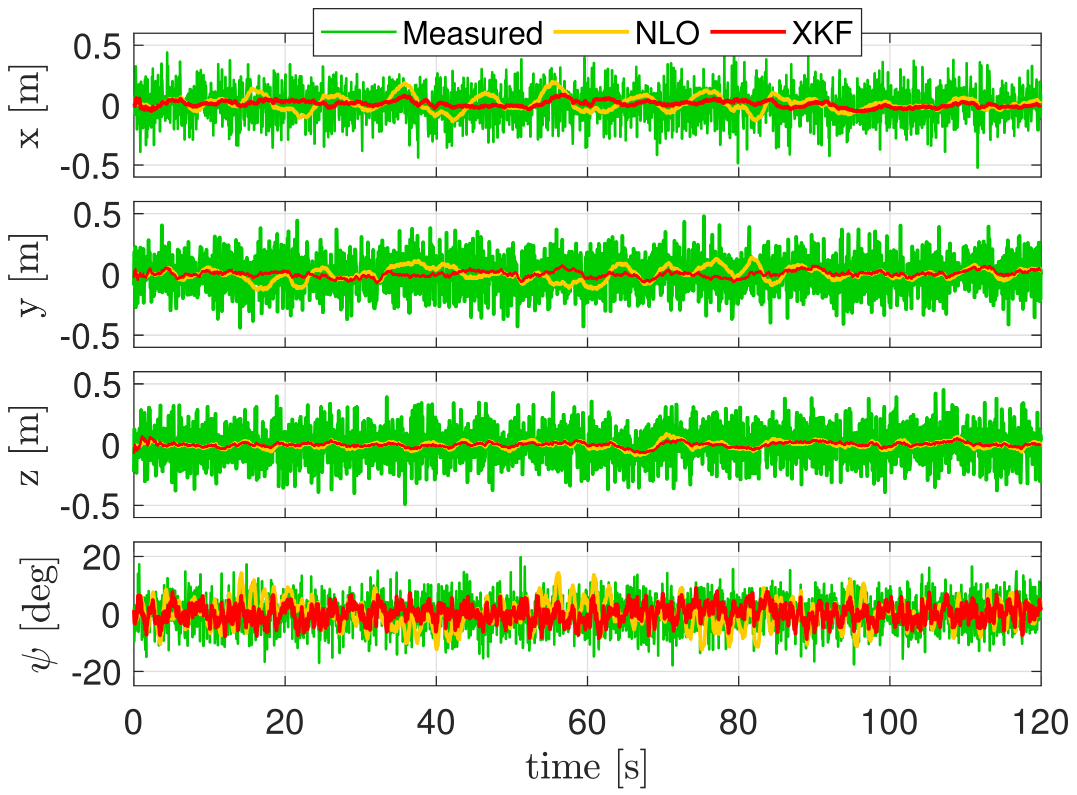


Figure 6.2: Simulation 1: From measurement to final XKF estimate, error plot

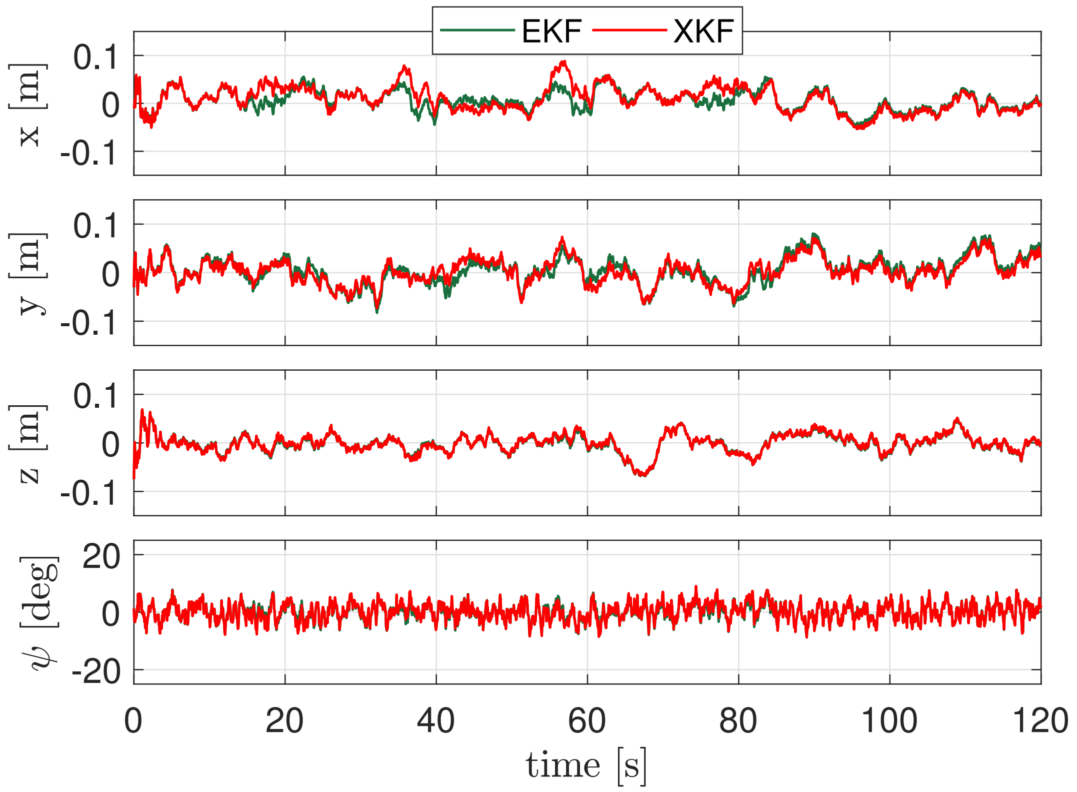


Figure 6.3: Simulation 1: Error in position estimates

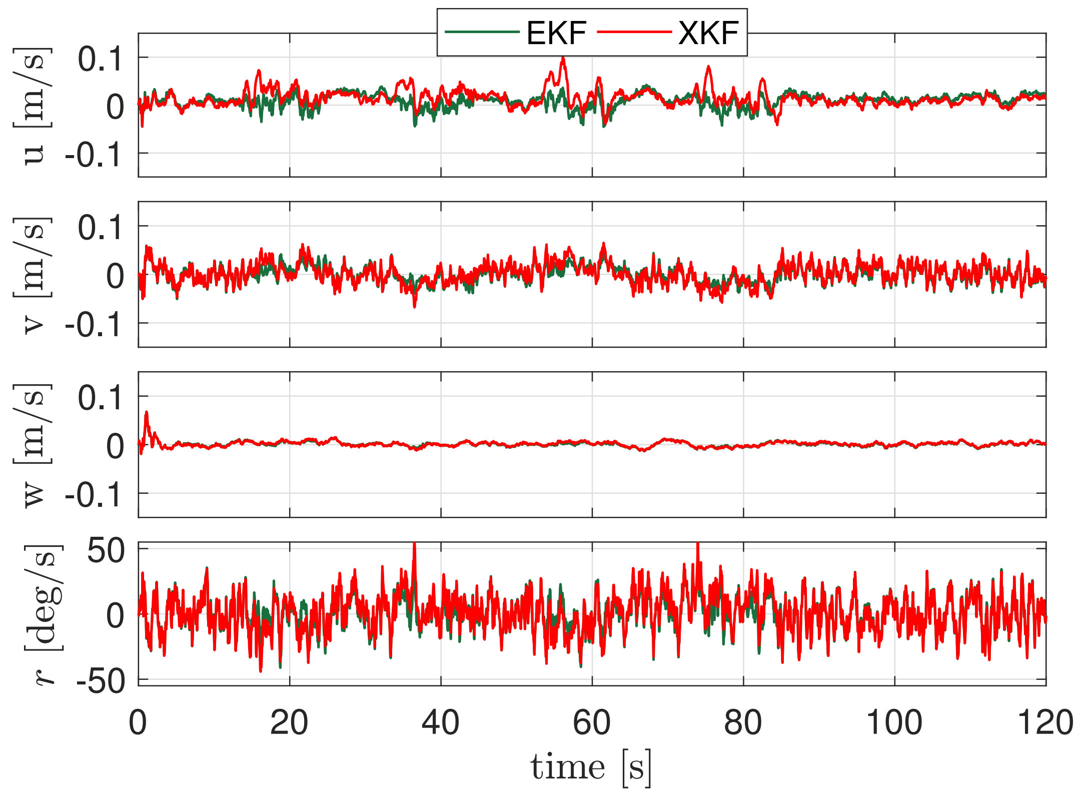


Figure 6.4: Simulation 1: Error in velocity estimates

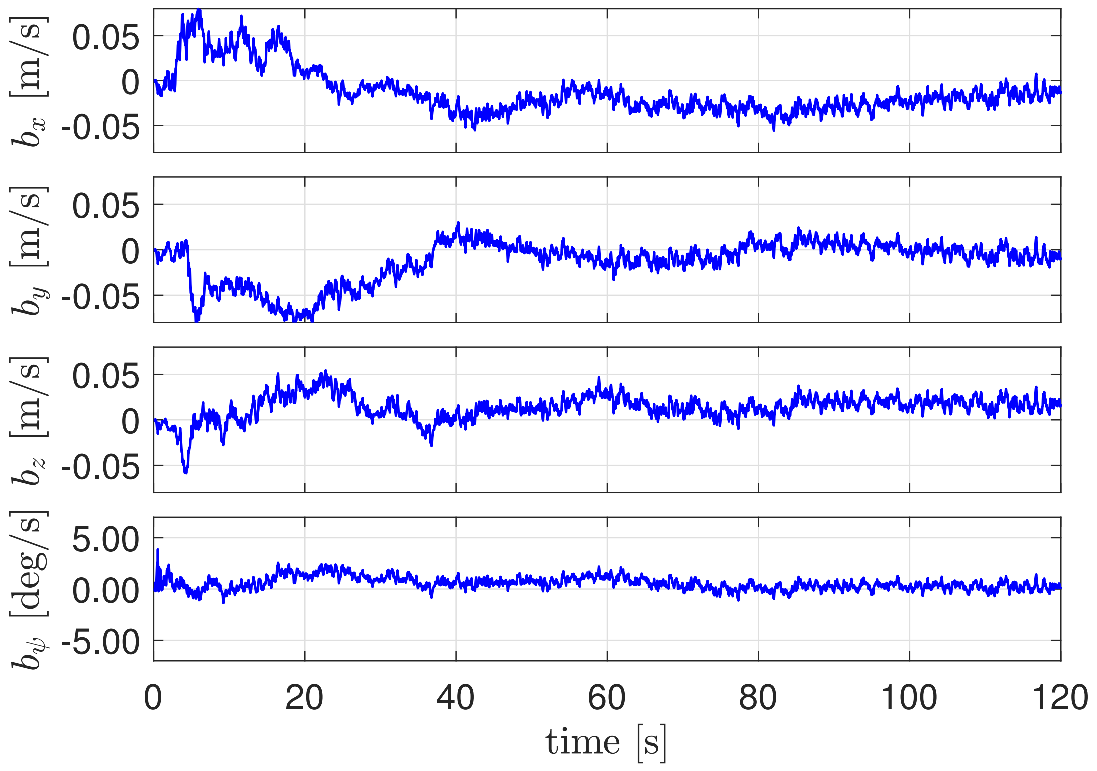


Figure 6.5: Simulation 1: Bias estimates in XKF

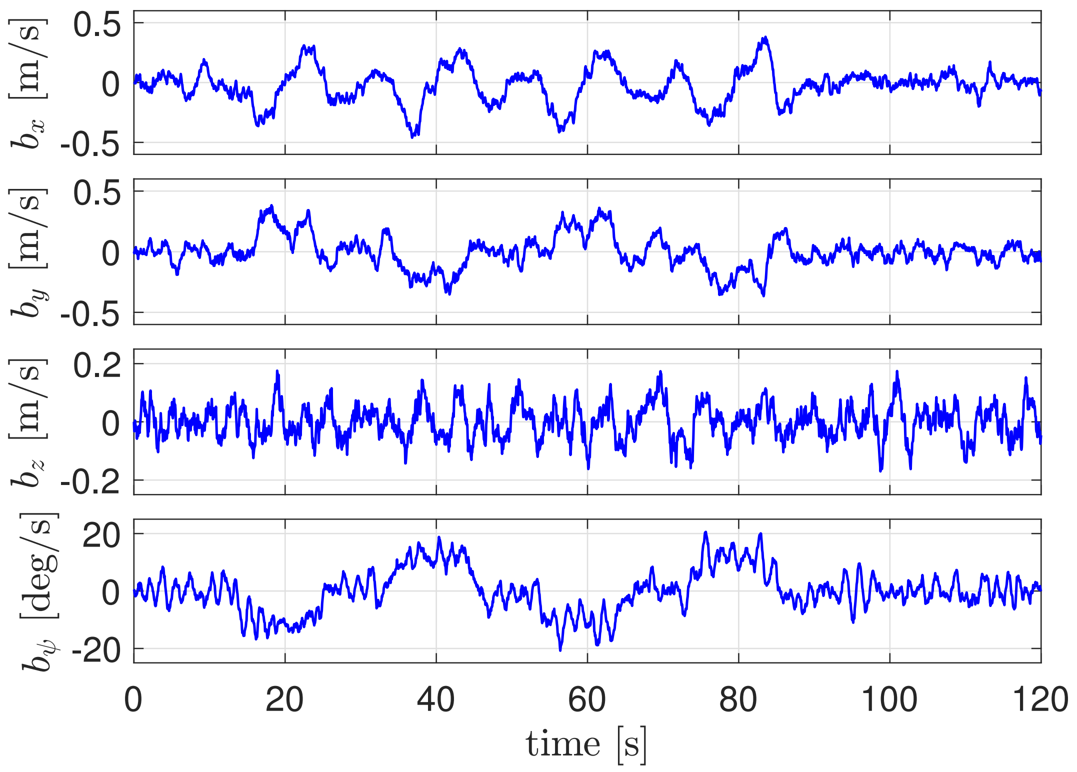


Figure 6.6: Simulation 1: Bias estimates in NLO

6.1.2 Simulation 2: Transient behaviour of EKF and XKF

This simulation is similar to the first one, but the initial conditions are erroneous. This will thus show if either filter has better transient behaviour. The initial values are chosen as $\boldsymbol{p}_0 = [10\ 3\ \ -11\ 0°]$, $\boldsymbol{v}_0 = [0.4\ 0.2\ 0.3\ 0.2]$ and $\boldsymbol{b}_0 = [0.06\ 0.08\ 0.1\ 0.03]$. The initial covariance is set to $P_0 = 0.1$ for all states. The observer gains are summarized in table 6.3. The mathematical model of the ROV is the same as in simulation 1.

Table 6.3: Simulation 2: Observer gains

EKF	XKF
$\boldsymbol{R}$ = diag[0.2, 0.2, 0.2, 0.08]	$\boldsymbol{R}$ = diag[0.2, 0.2, 0.2, 0.08]
$\boldsymbol{Q}$ = diag[0, 0, 0, 0, 50, 50, 50, 0.01, 1, 1, 1, 0.8]	$\boldsymbol{Q}$ = diag[0, 0, 0, 0, 5, 5, 5, 0.01, 1, 1, 1, 0.8]
	$\boldsymbol{L}_1$ = diag[5, 5, 5, 6]
	$\boldsymbol{L}_2$ = diag[10, 10, 10, 1]
	$\boldsymbol{L}_3$ = diag[3, 3, 3, 2.5]

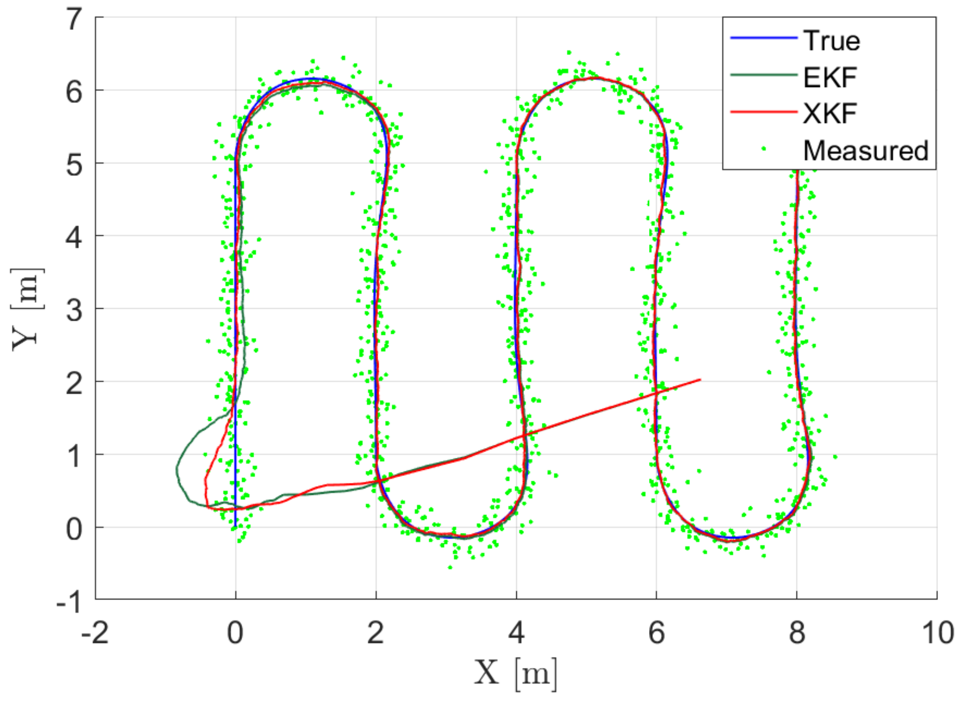


Figure 6.7: Simulation 2: XY-plot

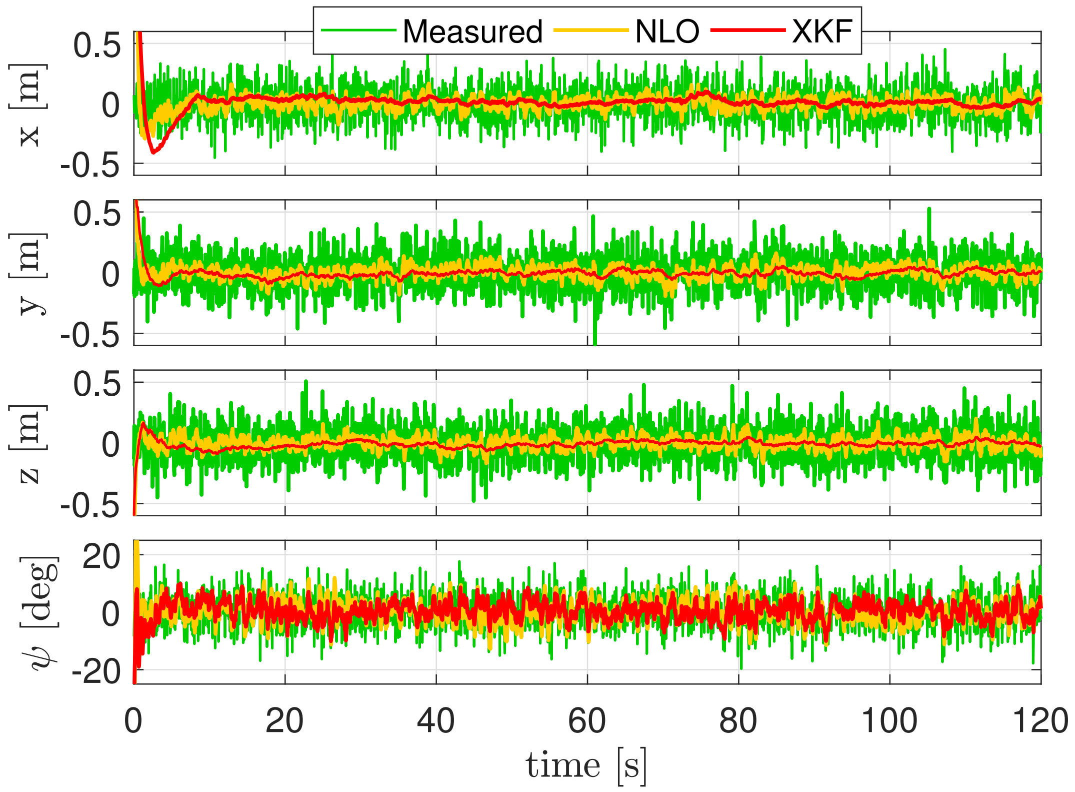


Figure 6.8: Simulation 2: From measurement to final XKF estimate, error plot

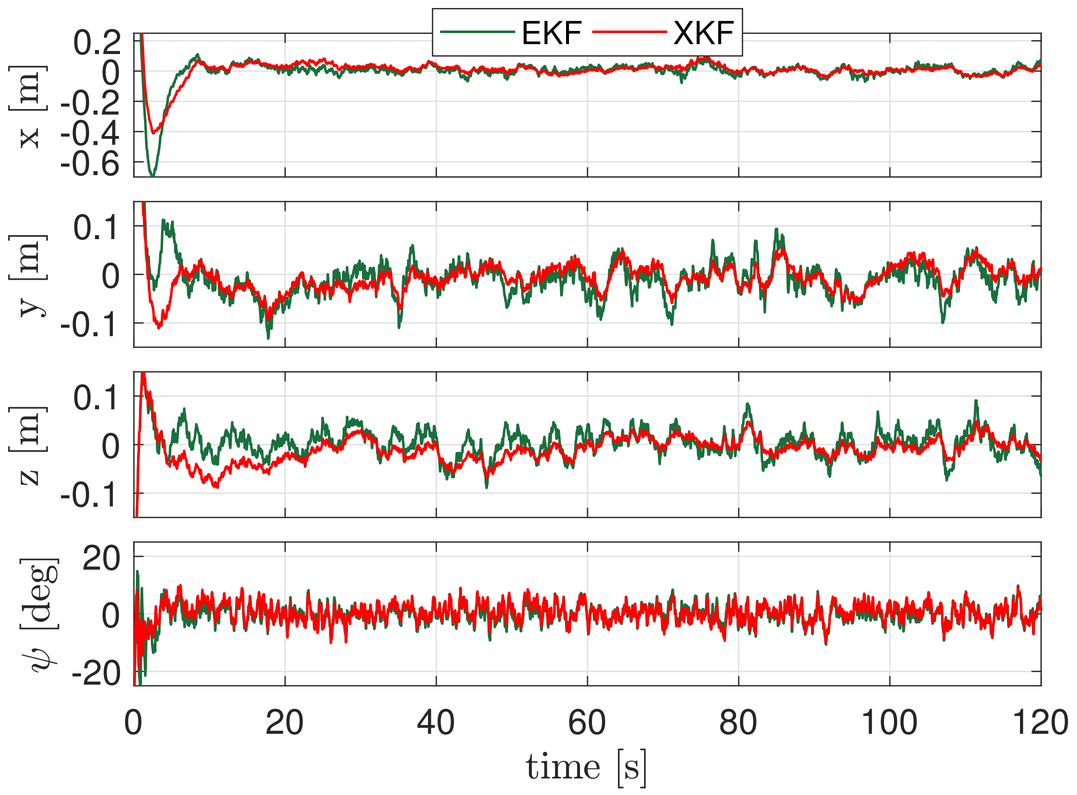


Figure 6.9: Simulation 2: Error in position estimates

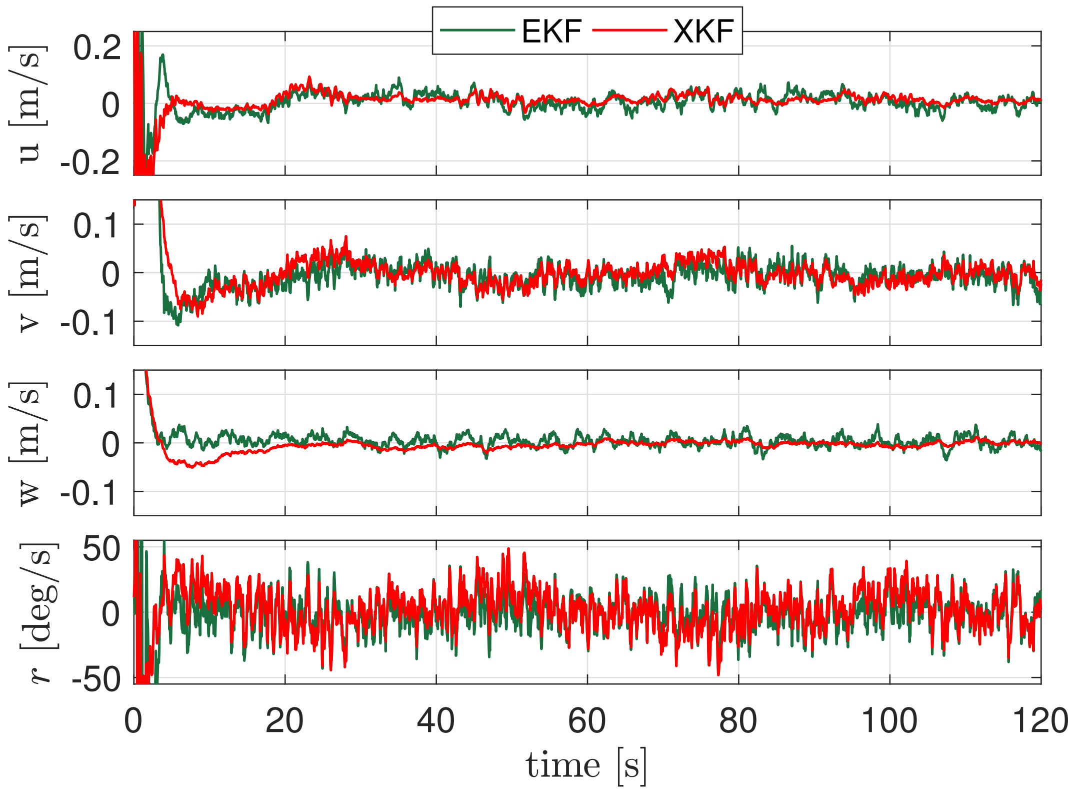


Figure 6.10: Simulation 2: Error in velocity estimates

6.1.3 Simulation 3: Steady state behaviour of PF

In simulation 3, the PF is put to the test using two different number of particles in the filter. The simulation with fewest particles uses a number of 500, while the simulation with most particles uses a number of 5000. These numbers are chosen as 500 seems to be the lower limit to obtain feasible results, and using more than 5000 seems to not provide noticeable better results. The filter is initialized using the correct values, i.e. $\boldsymbol{p}_0 = [0\ 0\ 90°]$ and $\boldsymbol{v}_0 = [0\ 0\ 0]$. The tuning matrices are equal in both cases and are summarised in table 6.4

Table 6.4: Simulation 3: Observer gains

Particle filter
$\boldsymbol{\sigma}_\tau^2$ = [0.9, 0.9, 0.6]
Σ = [0.045, 0.045, 0.017]

The mathematical model of the ROV used in the filter is implemented with the same parameters as in simulation 1. Simulation 1 and 3 therefore have access to exactly the same information about position and velocity. Later comparisons of RMS values will thus show which filter utilize this information best.

Comments to plots:
In figure 6.11 and 6.12 the particle clouds and corresponding measurements are only plotted for some iterations. This is done to make the plots easier to interpret.

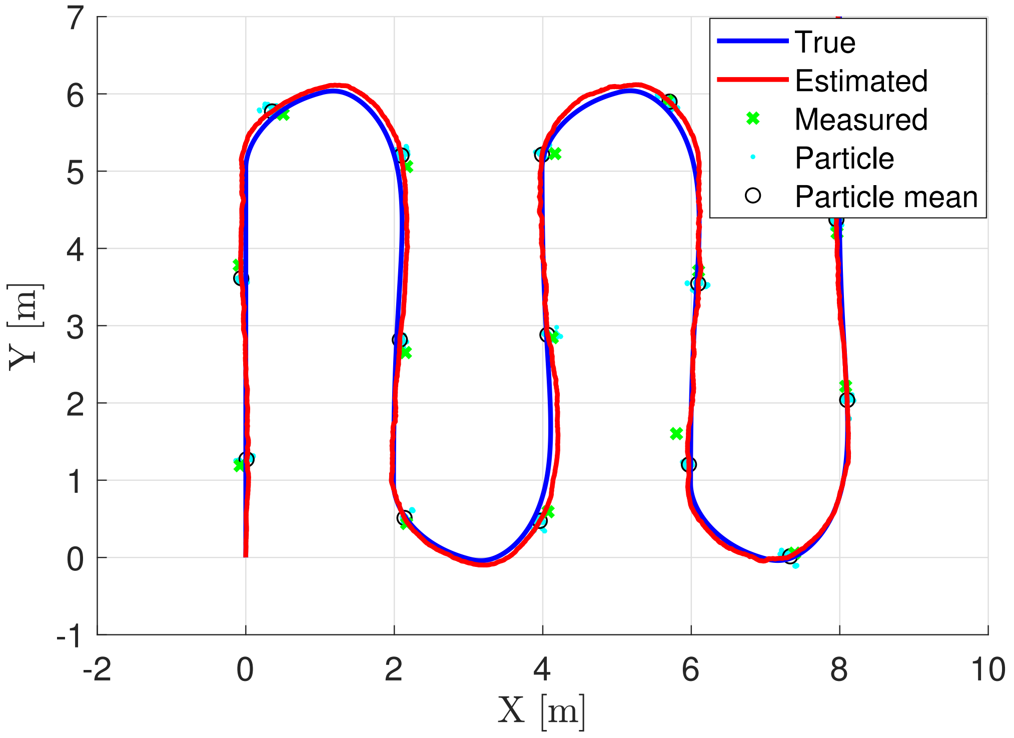


Figure 6.11: Simulation 3: XY-plot, N=500

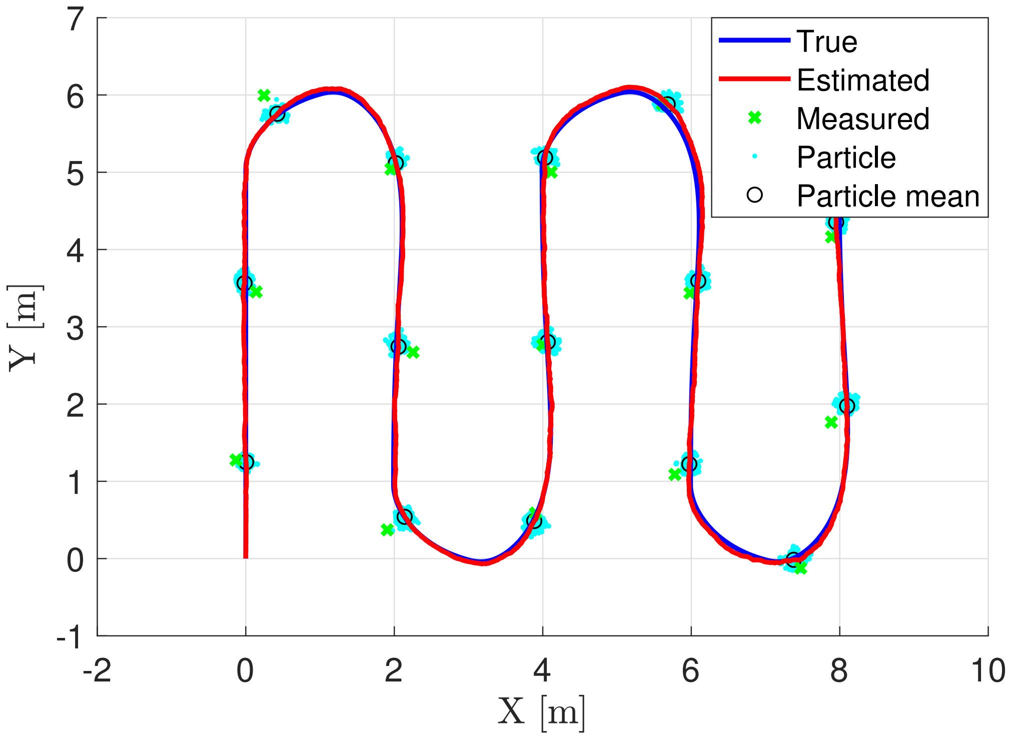


Figure 6.12: Simulation 3: XY-plot, N=5000

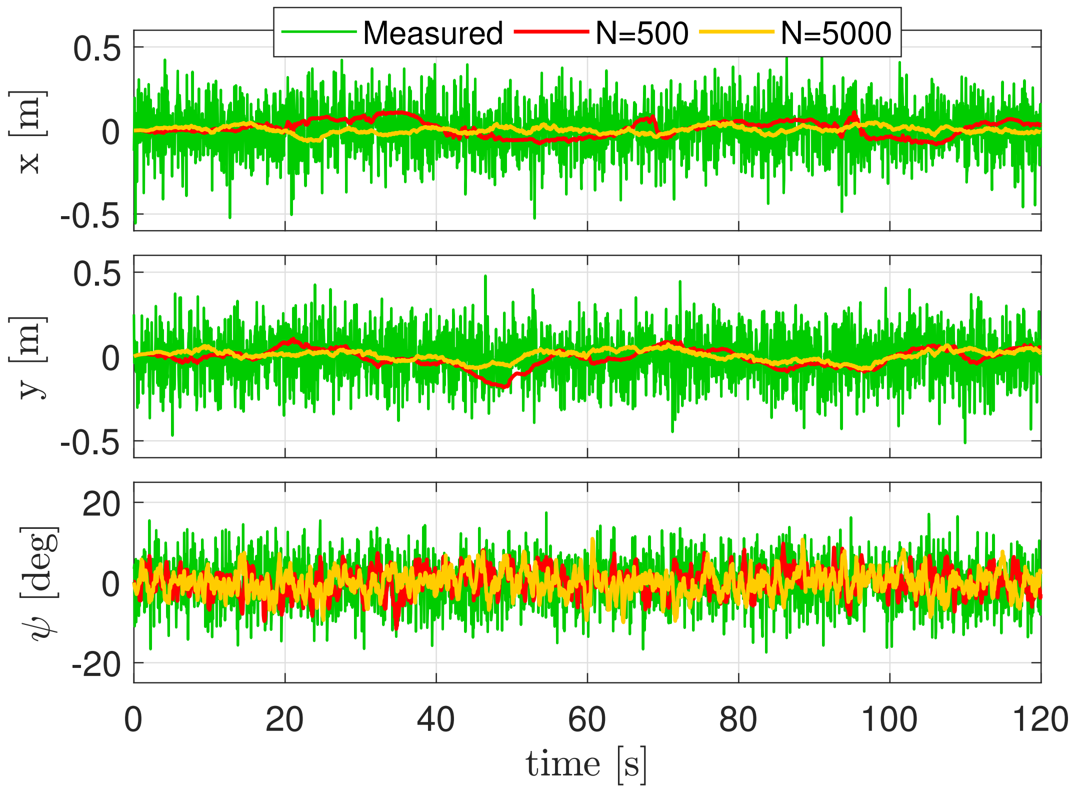


Figure 6.13: Simulation 3: Error in position estimates

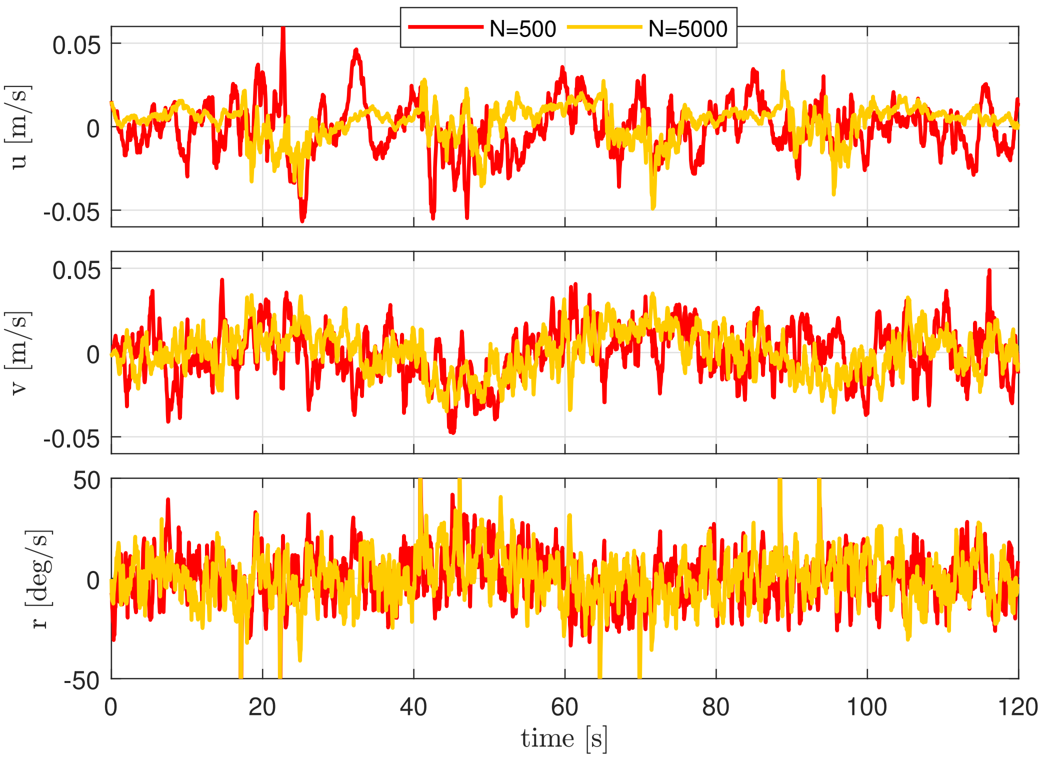


Figure 6.14: Simulation 3: Error in velocity estimates

6.1.4 Simulation 4: Transient behaviour of PF

In this simulation the transient behaviour of the PF is investigated. For the PF to be able to recover from a bad initial guess, it is necessary to spread the initial particles in the whole interval of state values where it is believed that the true state exists. It is assumed that the best knowledge about the true initial state lies somewhere in the intervals $x \in [-1, 10]$, $y \in [-2, 7]$ and $\psi \in [0, \pi]$. Hence, all particles are distributed randomly in these intervals. The same analogy goes for the velocities where each particle is given a velocity somewhere in the intervals $u \in [0,\ 0.5]$, $v \in [0,\ 0.5]$ and $r \in [0,\ 0.5]$. The intervals may off course be adjusted according to the prior knowledge, but using the intervals specified above requires a high number of initial particles. For this reason only the PF with 5000 particles are used in this simulation. This method requires more spreading in the particles compared to the steady state situation, and the new observer gains are summarised in table 6.5.

Table 6.5: Simulation 4: Observer gains

Particle filter
$\boldsymbol{\sigma}_\tau^2$ = [1.8, 1.8, 1]
$\boldsymbol{\Sigma}$ = [0.045, 0.045, 0.017]

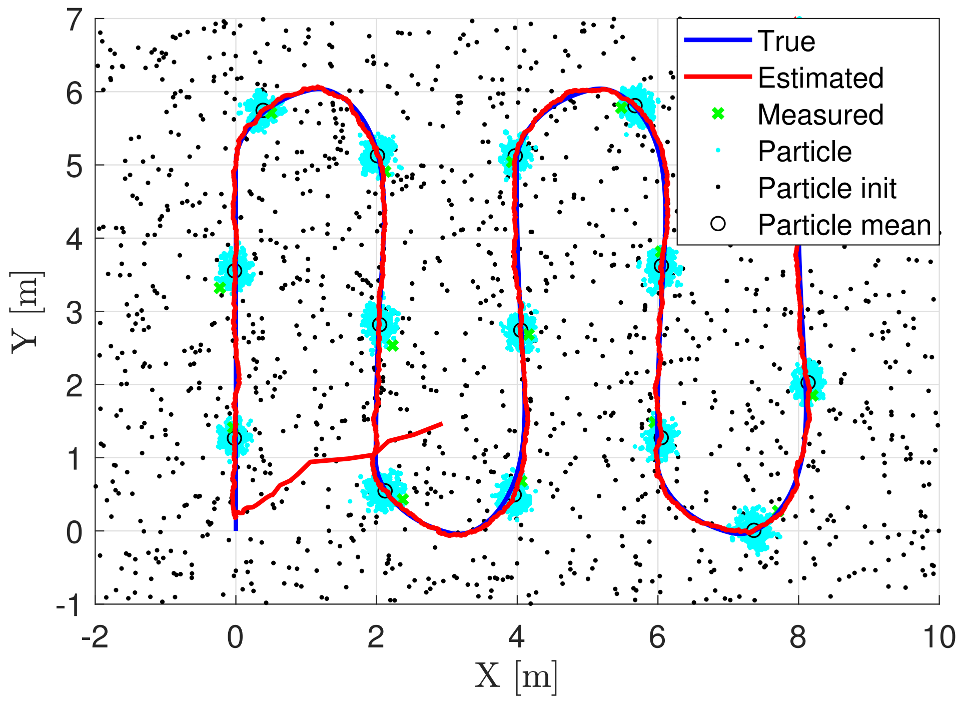


Figure 6.15: Simulation 4: XY-plot, N=5000

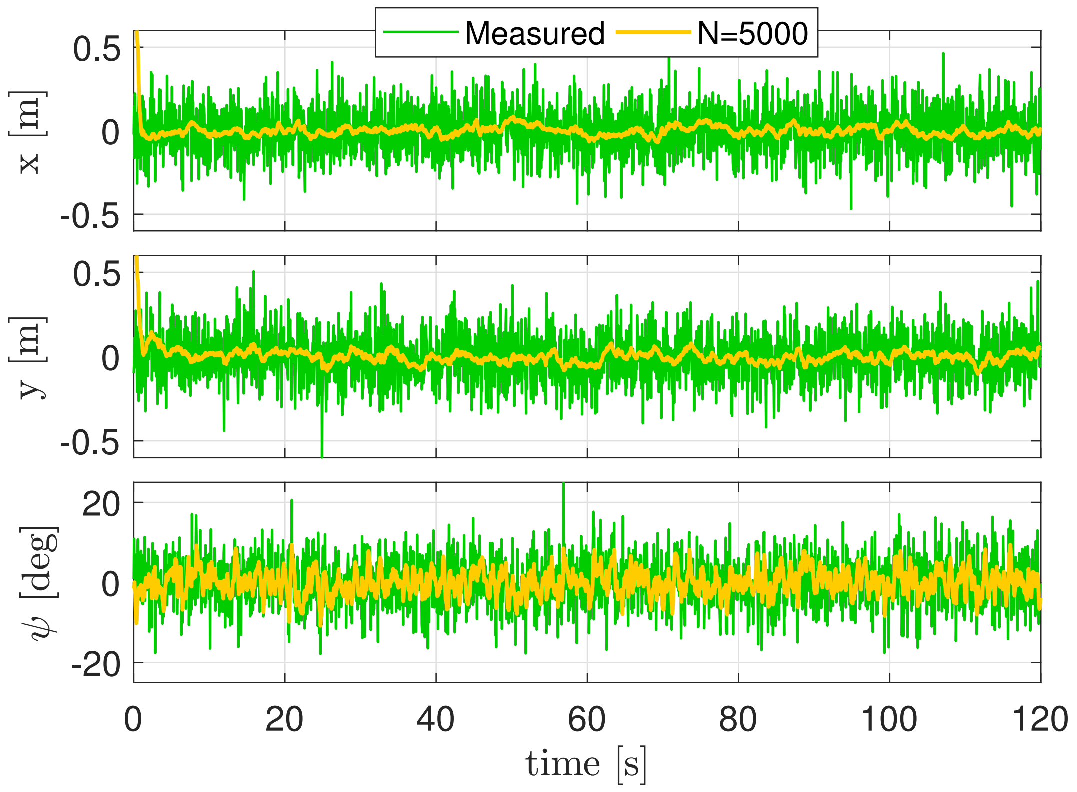


Figure 6.16: Simulation 4: Error in position estimates

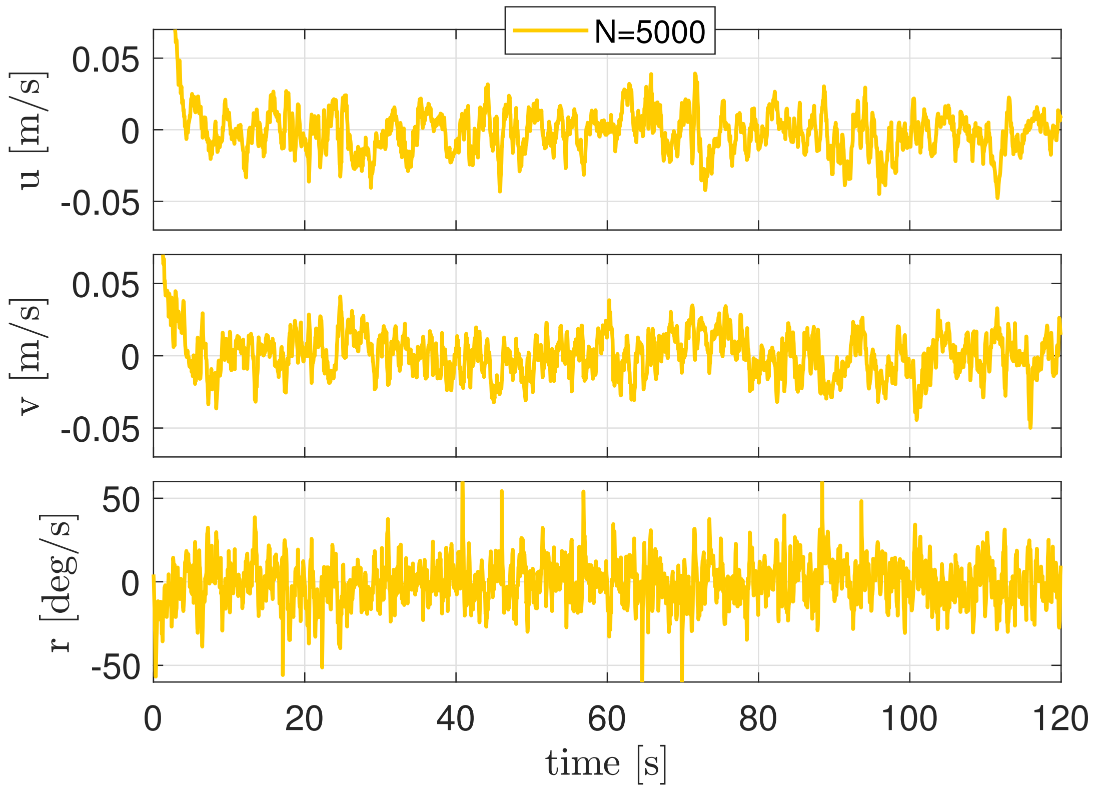


Figure 6.17: Simulation 4: Error in velocity estimates

6.1.5 Simulation 5: Less accurate mathematical model

In the previous simulations all filters were implemented with a rather precise mathematical model of the ROV. In this simulation the different filters are implemented using a less accurate model. The mass is set to 8.5 $[\text{kg}]$ which is $\sim 21\%$ more than the real value, and the damping parameters is set to 50% of the actual values. This will in turn show which filter is most robust against modelling errors. The filters are initialised with the true states $\boldsymbol{p}_0 = [0\ 0\ -10\ 90°]$ and $\boldsymbol{v}_0 = [0\ 0\ 0\ 0]$. The Kalman filters are also initialised with $b_0 = [0\ 0\ 0\ 0]$ and the initial covariance is set to $P_0 = 0.1$ for all states. The observer gains are summarised in table 6.6 and table 6.7 below.

Only RMS values are provided from this simulation. The figures are very similar to the figures from the steady state simulations, only with more noise in the estimates. Hence, the figures do not provide more information than the RMS values.

Table 6.6: Simulation 5: Observer gains Kalman filter

EKF	XKF
$\boldsymbol{R}$ = diag[0.2, 0.2, 0.2, 0.08]	$\boldsymbol{R}$ = diag[0.2, 0.2, 0.2, 0.08]
$\boldsymbol{Q}$ = diag[0, 0, 0, 0, 40, 40, 40, 0.01, 1, 1, 1, 0.8]	$\boldsymbol{Q}$ = diag[0, 0, 0, 0, 40, 40, 40, 0.01, 1, 1, 1, 0.8]
	$\boldsymbol{L}_1$ = diag[5, 5, 5, 6]
	$\boldsymbol{L}_2$ = diag[10, 10, 10, 1]
	$\boldsymbol{L}_3$ = diag[3, 3, 3, 2.5]

Table 6.7: Simulation 5: Observer gains particle filter

Particle filter
$\boldsymbol{\sigma}_\tau$ = [2.8, 2.8, 0.8]
Σ = [0.045, 0.045, 0.017]

6.2 RMS values simulation 1 and 3: Steady state behaviour

Table 6.8: Simulation 1 and 3: RMS values of position and heading estimates

	EKF	**XKF**	**PF, N=500**	**PF, N=5000**
Surge [m]	0.0226	0.0240	0.0397	0.0202
Sway [m]	0.0296	0.0289	0.0418	0.0233
Heave [m]	0.0163	0.0165	-	-
Yaw [deg]	3.0923	3.1799	3.5645	3.2449

Table 6.9: Simulation 1 and 3: RMS values of estimated velocities and angular rates

	EKF	**XKF**	**PF, N=500**	**PF, N=5000**
Surge [m/s]	0.0186	0.0210	0.0150	0.0101
Sway [m/s]	0.0173	0.0201	0.0145	0.0126
Heave [m/s]	0.0049	0.0054	-	-
Yaw [deg/s]	13.3176	13.8452	13.6243	13.0816

6.3 RMS values simulation 2 and 4: Transient behaviour

Table 6.10: Simulation 2 and 4: RMS values of position and heading estimates

	EKF	**XKF**	**PF, N=5000**
Surge [m]	0.2611	0.2378	0.1429
Sway [m]	0.0918	0.0803	0.0906
Heave [m]	0.0352	0.0344	-
Yaw [deg]	3.9224	3.4012	3.3460

Table 6.11: Simulation 2 and 4: RMS values of estimated velocities and angular rates

	EKF	**XKF**	**PF, N=5000**
Surge [m/s]	0.0681	0.0471	0.0209
Sway [m/s]	0.1493	0.1105	0.0261
Heave [m/s]	0.0338	0.0352	-
Yaw [deg/s]	30.0281	21.3439	13.8766

6.4 RMS values simulation 5: Less accurate model

Table 6.12: Simulation 5: RMS values of position and heading estimates

	EKF	**XKF**	**PF, N=500**	**PF, N=5000**
Surge [m]	0.0300	0.0318	0.0563	0.0326
Sway [m]	0.0415	0.0415	0.0597	0.0367
Heave [m]	0.0294	0.0294	-	-
Yaw [deg]	3.1419	3.2189	3.9902	3.4093

Table 6.13: Simulation 5: RMS values of estimated velocities and angular rates

	EKF	**XKF**	**PF, N=500**	**PF, N=5000**
Surge [m/s]	0.0525	0.0542	0.0473	0.0280
Sway [m/s]	0.0279	0.0297	0.0378	0.0284
Heave [m/s]	0.0147	0.0149	-	-
Yaw [deg/s]	15.1504	16.1782	16.3510	14.3529

6.5 Execution times

Table 6.14: Execution times

	EKF	**XKF**	**PF, N=500**	**PF, N=5000**
$\frac{\text{Total execution time}}{\text{Total simulated time}}$ $[-]$	0.025	0.033	0.058	0.57

Chapter 7

Experimental results

In this chapter the results obtained from using the PF with experimental data are presented. As discussed earlier, only measurements from the QMC system and from an IMU were available in the MC-lab. This makes it impossible to test the proposed XKF design as it relies on measurements of thrust. It is however possible to verify that the structure of the PF works as intended. Using the mathematical model or an IMU essentially builds upon the same principle. First the accelerations are found, then these are integrated to find the positions. If promising results can be obtained from using the IMU, it should also work using the mathematical model.

7.1 Experimental specifications

Both steady state and transient behaviour are tested using data from a ~ 27 $[s]$ long measurement series. The measurements from the QMC are regarded as true due to their low variance. Hence, these measurements are used to make the RMS values and error plots. The velocities are not directly measured by the QMC. These are instead found by numerical derivation of the position measurements. To make inputs for the observer however, the QMC measurements are added Gaussian noise with a variance of $\sigma^2 = 0.2$. This leads to the measurements having RMS values of ~ 0.2 $[m]$ in surge and sway position, and a RMS value of $\sim 11^\circ$ in heading. A successfully implementation of the PF will reduce these values noticeably, while also providing accurate velocity estimates. Both the steady state and transient case turned out to work best with the same gains. These gains are summarized in table 7.1.

Table 7.1: Experiment 1 and 2: Observer gains particle filter

Particle filter
$\boldsymbol{\sigma}_{imu}$ = [2.8, 2.8, 1.2]
Σ = [0.01, 0.01, 0.01]

In this case, $\boldsymbol{\sigma}_{imu}$ specifies the uncertainty in the IMU-data.

7.2 Experiment 1: Steady state

In the first experiment the states are initialised with the true values to see how the filter works in the steady state case. Hence, the initial states used in the filter are $[x\ y\ \psi] = [0.03,\ \ -0.92,\ \ 293°]$ and $[u\ v\ r] = [0.44,\ \ -0.24,\ \ 0.23]$.

Comment to plots: Due to the large spread in particles, only two particle clouds are plotted in figure 7.1. The smallest being the initial one.

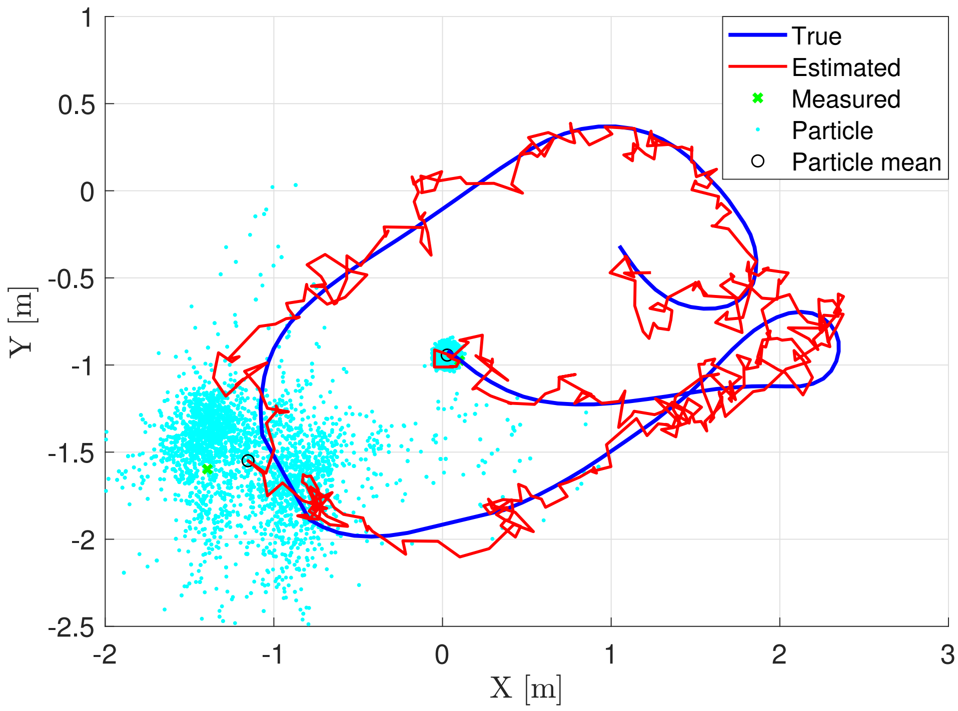


Figure 7.1: Experiment 1: XY-plot

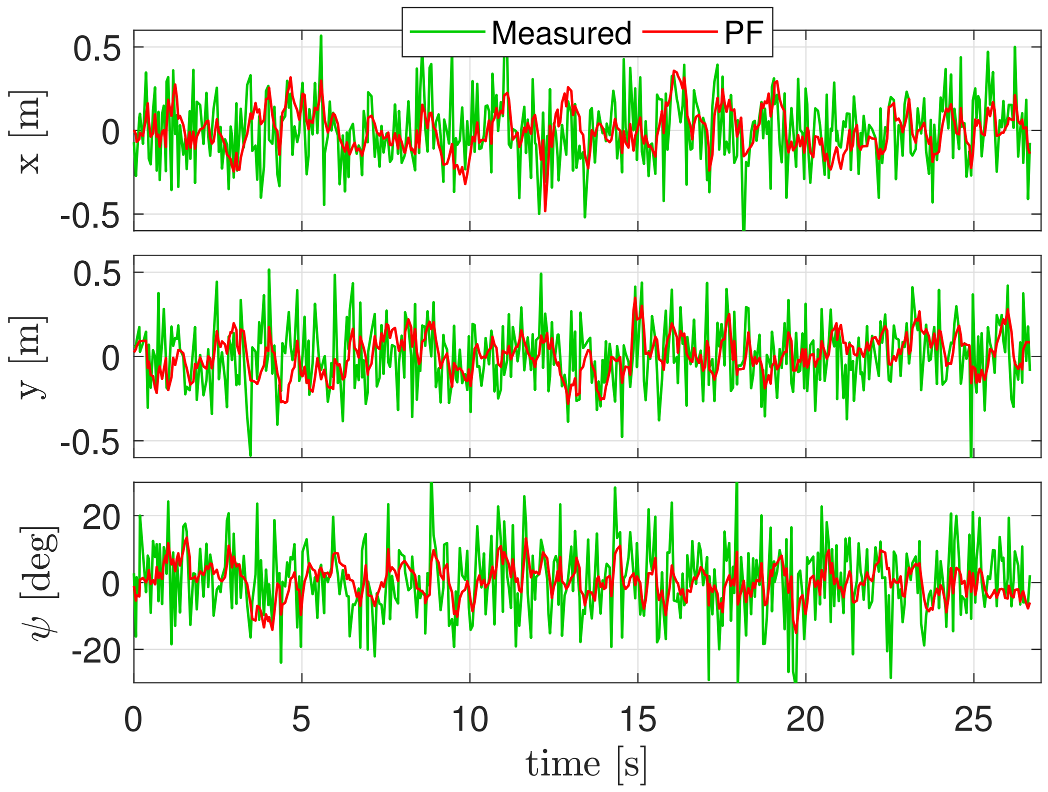


Figure 7.2: Experiment 1: Error in position estimates

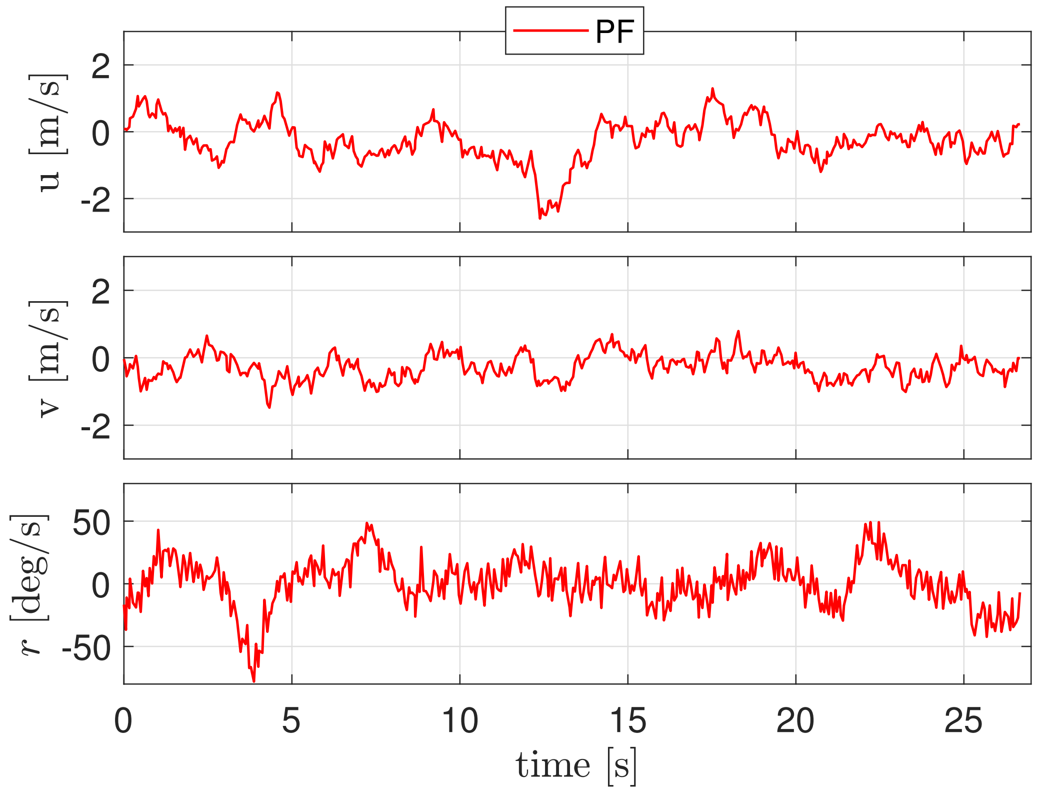


Figure 7.3: Experiment 1: Error in velocity estimates

7.3 Experiment 2: Transient

The second experiment is based on the same experimental data as the first one. This time however it is assumed that the initial values are not known. This will show how well the filter is able to converge using real sensor data. Remember from simulation 4 that uncertainty in the initial states must be specified as an interval of uncertainty in the PF. For experiments conducted in a basin, the uncertainty intervals of positions may be set equal to the dimensions of the basin. The intervals used in this experimental evaluation are set to cover the known operational space of the ROV. Hence, the intervals used are $x \in [-1, 10]$, $y \in [-2, 7]$ and $\psi \in [0, 360°]$ for position and heading. And for velocities the intervals $u \in [-2,\ 2]$, $v \in [-2,\ 2]$ and $r \in [-20,\ 20]$ are used. The gains used in this simulation are the same as the gains in table 7.1.

Comment to plots: Figure 7.4 and 7.5 are essentially the same plot, but the latter shows how the initial particle cloud evolve after the first resampling. Only the initial and one more particle cloud are plotted in order to avoid too many particle clouds overlapping each other.

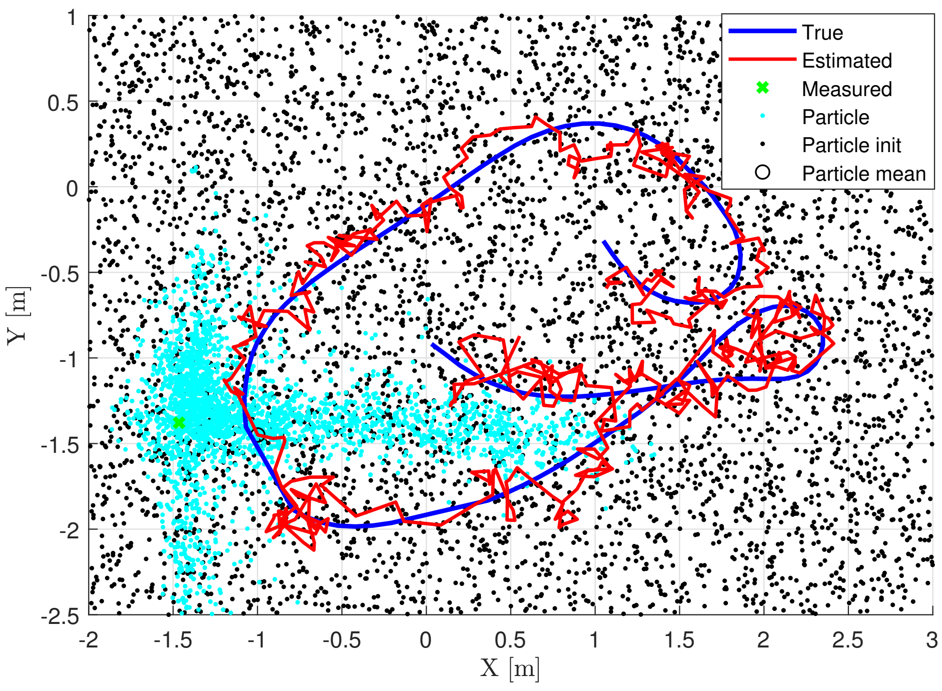


Figure 7.4: Experiment 2: XY-plot

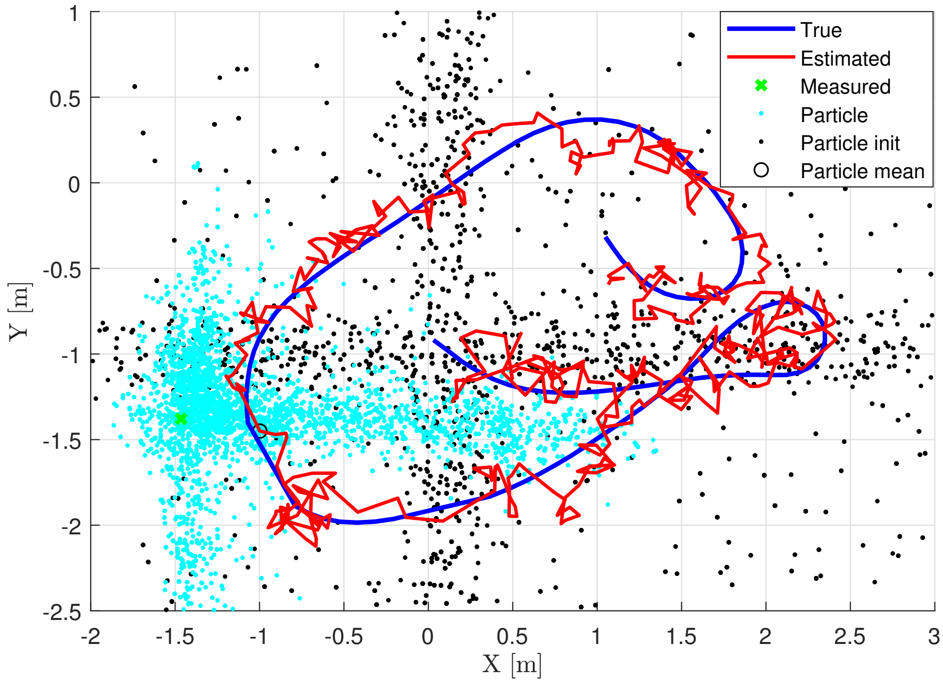


Figure 7.5: Experiment 2: XY-plot with resampled initial particles

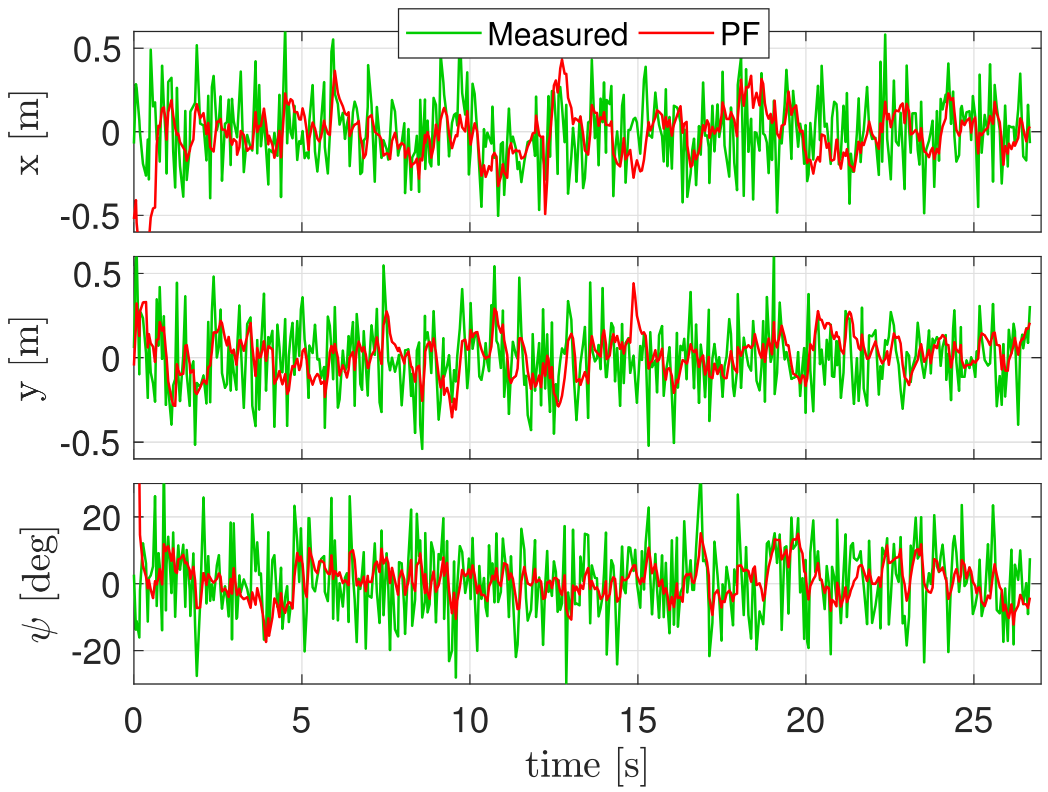


Figure 7.6: Experiment 2: Error in position estimates

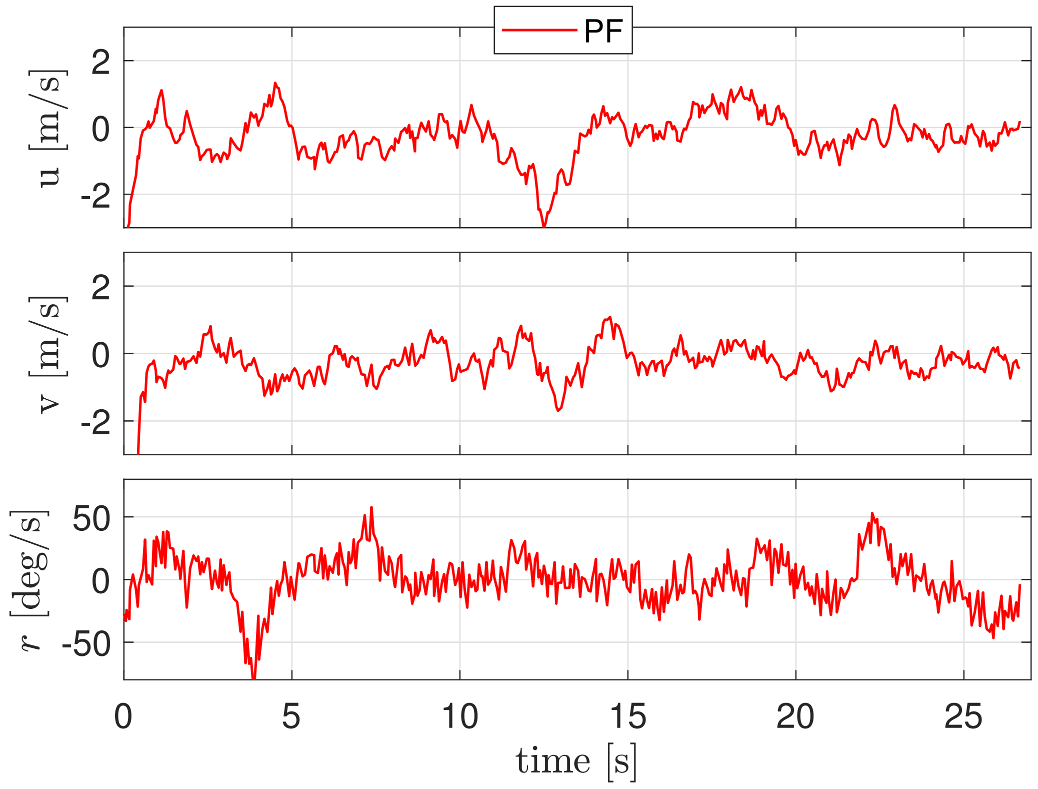


Figure 7.7: Experiment 2: Error in velocity estimates

7.4 RMS values and computational time

The RMS values from experiment 1, i.e. the steady state case are provided in table 7.2.

Table 7.2: RMS values from experiment 1

	Surge	Sway	Heave
$\boldsymbol{\eta}$	0.1255 $[m]$	0.1133 $[m]$	5.0071 $[deg]$
$\boldsymbol{\nu}$	0.6919 $[m/s]$	0.4712 $[m/s]$	19.448 $[deg/s]$

The computational time using experimental data yields a ratio of 0.49 when the total execution time is divided on the length of the measurement series. Unlike the simulations this ratio only considers the time used to run the filter. While the ratios found from the simulations also takes into account the time used to drive the simulator itself.

Chapter 8

Discussion of results

In this chapter the figures and RMS values are compared and discussed. Results from both simulations and experiments are covered in this chapter.

8.1 Simulation results

The simulated results are covered first. The focus of the discussion is how each filter behave in the steady state- and transient case, and how the filters compare to each other.

8.1.1 Steady state behaviour of EKF and XKF

In the case of correct initial conditions, i.e. simulation 1, there is not much that separates the two filters based on the results. From the XY-plot in figure 6.1 it is nearly impossible to separate the EKF estimate from the XKF estimate, and both are very close to the true trajectory. This figure also gives a visual impression of the measurement noise, and as can be seen the noise is greatly reduced by both filters. The reduction in measurement noise is further investigated for the XKF in figure 6.2. Here it is shown how the measurements are first transformed in the NLO, and how the final estimate of the XKF looks like. The NLO estimates fluctuates a lot in surge, sway and heave. In yaw one can also see that the error estimate is somewhat shifted away from the desired value of zero at certain time intervals. This is due to the slowly varying assumption of the mathematical model used in the NLO, and the shifts occurs at every turn as the velocities changes in order to make the ROV follow the desired path. However, the fluctuations and shifts are not reflected in the final estimates of the XKF. This proves the NLO estimates are feasible lineariazation points.

Figure 6.3 confirms that the position estimates of the EKF and the XKF are almost identical, only with some deviations at certain occasions. The RMS values of table 6.8 shows the same. Figure 6.4 shows that the velocity estimates of the two filters are also very similar. However, in this case the linearization point from the NLO is causing especially the surge estimate from the XKF to deviate slightly more than the estimate from the EKF when the ROV is turning. The bias estimates are also plotted for this simulation. It is seen that the bias estimates of the NLO in figure 6.6 have a much larger scale than the bias estimates in the final XKF estimates in figure 6.5. This is due to the bias estimates in the NLO are trying to compensate for the missing Coriolis matrix. Looking closer at the biases in the NLO it is evident, especially for the bias in yaw, that these estimates follows a periodically pattern in compliance with the turning of the ROV. The bias estimates of the XKF are small and have little influence on the velocity estimates.

8.1.2 Transient behaviour of EKF and XKF

In simulation 2, where the transient behaviour is put to the test, the differences between the two filters are larger. From the XY-plot in figure 6.7 one sees that the XKF converges faster and with less overshoot than the EKF. When both filters eventually reaches steady state their trajectories in the XY-plane coincides as earlier. The error plots of positions and velocities in figure 6.9 and 6.10 shows the same behaviour. I.e. faster convergence rates and less overshoots for the XKF in the transient stage, and similar behaviour in the steady state stage. From the RMS values of position and velocity in table 6.10 and 6.11 all values are high due to the offset in the transient stage, but the XKF has lower values because of the abovementioned reasons.

The fast convergence rates of the XKF is due to the great convergence capabilities of the NLO. As shown in the overview of the observer gain values in table 6.3, the NLO is tuned with higher values in $\boldsymbol{L}_1$ in the transient simulation. This leads to faster convergence rates, but as can be seen in figure 6.8 this also makes the NLO estimate being more noisy. As previously mentioned the main task of the NLO is not to reduce noise, but to make a reasonable linearization point. The noise in the NLO estimates is thus not a problem. Noise reduction is the primary objective of the Kalman filter in the second stage and as can be seen from the figure, the final estimates of the XKF has suppressed almost all noise.

Notice that the Kalman gain matrices in the XKF remains the same as in the steady state case. Only the NLO gain matrices are used to tune the transient behaviour. This is not the case with the EKF, where the gains are set higher to make it converge faster. As seen from the position- and velocity error plots in figure 6.9 and 6.10, this makes the EKF estimates slightly more noisy when the filter reaches steady state.

Finally, it must be mentioned that the initial values in simulation 2 were chosen such that the EKF was able to converge. Albeit not shown in the results, just slightly more erroneous initial conditions leads to the EKF estimates diverging and growing to infinity. This is not the case with the XKF. Here all states will converge, but it might take longer time if the initialisation is very inaccurate. Hence, the XKF has superior convergence capabilities compared to the EKF.

8.1.3 Steady state and transient behaviour of PF

Simulation 3 shows how the number of particles affects the performance of the steady state estimation of the PF. In the XY-plot of figure 6.11 a number of 500 particles are used. This yields an estimated trajectory which is relatively smooth, but with some clear deviations from the true path. Figure 6.12 shows the same case, but here it is used ten times as many particles. As can be seen from the figure this leads to much denser and more widespread particle clouds. The result is a more accurate approximation of the posterior distribution of the states, and the estimated trajectory is eminently close to the real trajectory. The degree of accuracy is best illustrated in figure 6.13 where it is seen that the position error of the PF with 5000 particles stays close to zero in surge and sway for all time instances. In the same figure one can see that the position error in surge and sway for the PF with 500 particles fluctuates much more around zero. This is also rooted in table 6.8 where the position RMS values in surge and sway are reduced significantly when using more particles. The yaw estimate is less affected by the increased number of particles.

Looking at figure 6.14 all velocity error estimates fluctuates around zero, and it is evident that the estimates using 5000 particles fluctuates slightly less than the estimates using 500 particles. The RMS values in table 6.9 confirms that the velocity estimates are more precise using 5000 particles. The price to pay for more accurate estimates is increased computational time. As seen from table 6.14 it takes approximately ten times longer to run the filter with ten times the number of particles.

From simulation 4 it is clear that the PF is capable to recover from erroneous initial conditions with very fast convergence rates and with little overshoot in the state estimates. In the XY-plot in figure 6.15 the particle cloud initially covers the whole operational space. As long as a sufficient number of particles are used, some of the initial particles will be close to the true state of the ROV. These particles gets a high weight and gets sampled more frequently in the next iteration. In this way the particle cloud quickly shrinks from covering the whole operational space to only cover the state space in close proximity to the ROV. From the position error estimates in figure 6.16, and the velocity error estimates in figure 6.17, it can be seen that the filter reaches the true states in just about a second.

8.1.4 Comparison of the EKF/XKF and PF

In this section the the Kalman filter implementations are compared against the PF. In the case of steady state and a accurate mathematical model, it is evident from the RMS values in table 6.8 and 6.9 that the PF with 5000 particles outperforms the Kalman filters in surge and sway. Both position and velocity estimates are more accurate. In yaw however, the Kalman filters estimate heading slightly more precise. The PF with 500 particles makes the least accurate position and heading estimates of all. On the other hand, its surge and sway velocity estimates are better than the estimates of the XKF and the EKF.

When using a less accurate mathematical model, the RMS values of table 6.12 and 6.13 yields different relations. The PF with 5000 particles still has the most accurate position estimate in sway, but the Kalman filters are more accurate in surge. The ratio between the heading estimates remains approximately the same. The PF with 500 particles still produces the most inaccurate position and heading estimates. Based on the RMS values of the velocities, the PF with 5000 particles produces a clearly better estimate of the surge velocity than the Kalman filters. It also produces a better yaw rate estimate. In sway the differences are smaller, but also less important as the main movement is in surge and yaw.

In the transient case, table 6.11 shows that the PF has significantly lower RMS values for all velocity estimates. This suggests that the velocity estimates of the PF converges much faster than in the EKF and XKF. Also the RMS value for position in surge from table 6.10 is much lower for the PF. For position in sway and heading, the differences are smaller. What is not reflected in the numbers is however the steady state performance of the filters after the transient stage. The XKF has one big advantage over the PF in this case. The NLO in the first step of the XKF can be tuned for the transient stage, while the KF in step two can be tuned for steady state. This is not possible with the PF, and as can be seen from table 6.5 the gains used for the PF in the transient simulation are higher than the gains used in the steady state simulation. This implies that the fast convergence of the PF is at expense of slightly less accurate estimates in the following steady state.

One of the biggest differences between the PF and the Kalman filters are the computational times. From table 6.14 it is evident that the PF with 5000 particles uses approximately 23 times longer time than the EKF, while The PF with 500 particles uses approximately 2.3 longer time. The XKF only uses about 1.3 longer time.

8.2 Experimental results

In this section the experimental results from chapter 7 are discussed. As previously mentioned only the PF is tested using experimental data.

8.2.1 Experiment 1

The results from experiment 1 shows that the PF is able to suppress the measurement noise. However, from the XY-plot of figure 7.1 it is apparent that even though noise is reduced, the position estimates are still fluctuating a lot around the true estimates. It is also seen from this figure that the spreading of the particles is rather large. This is due to the error sources discussed in section 5.4.7. It is thus necessary with large particle spreading to ensure that the true vehicle accelerations are captured by some of the particles. Figure 7.2 gives a visual interpretation of how well the PF is able to filter out noise from the measured states. The PF estimates fluctuates less and have lower peaks than the measurements. The RMS values in table 7.2 confirms this, where the estimates of surge and sway positions are reduced by $\sim 38\%$ and $\sim 43\%$ compared to the measurements. The RMS value of the heading is reduced by $\sim 55\%$.

Figure 7.3 shows that the velocity estimates are not very accurate. With the relatively low velocities of the ROV, the scale of the velocity error estimates are way above the acceptable limit. The error estimates are however fluctuating around zero. This suggest that the filter is working as intended, but the IMU-data are too inaccurate to determine velocities with higher precision. The reason for the large fluctuations is that many particles have reasonably high weights even if not all state hypothesis within the particle are accurate. For instance, a particle can contain inaccurate hypotheses of the heading angle and the velocities. Albeit these estimates being inaccurate, they may still produce a feasible position estimate together. Hence, the feasible position estimate gives the particle high weight even though the velocity and heading estimates are not accurate.

Running the filter offline with 5000 particles shows that the filter uses approximately 0.5 $[s]$ to evaluate 1 $[s]$ of measured data. This number depends on what software is used to implement the filter, and the specifications of the computer running this software. However, as this number is found using Matlab running on a standard up to date laptop, it confirms that the PF is highly computational expensive.

8.2.2 Experiment 2

The results from experiment 2 shows that the PF is able to locate the ROV in the basin with high convergence rates. The XY-plot in figure 7.4 is very similar to the XY-plot of the steady state XY-plot in experiment 1. However, here it is shown how the initial particles are spread over the whole operational space of the ROV. The XY-plot in figure 7.5 shows the reason for the fast convergence rate of the PF. After only one resampling the initial particles have transformed from a uniformly random distribution to a distinct cross. In the middle of the cross is the location of the ROV. This indicates that the resampling step efficiently removes low weighted particles. However, as can be seen there are still some particles outside the cross that seems to not contain accurate hypothesis. This is because the randomness in the resampling step leads to some low-weighted particles also being resampled. Another explanation is that these particles may contain very accurate yaw estimates that still gives a decent weight. The steady state behaviour of position errors in figure 7.6 and velocity errors in figure 7.7 is the same as in experiment 1. The interesting part of these plots are the first 1-2 seconds where one sees that the PF converges very fast. It is also evident that there are no overshoots in the estimates.

8.2.3 Simulations vs. experiments

The PF has been evaluated using both simulated data and experimental data. Even if the simulations uses a thruster model and the experiments uses IMU-data, there are still some comparisons that can be made. In both cases it seems to be an upper and a lower limit of how many particles are needed in the filter. More than 5000 particles only leads to higher computational times, and not more accurate estimates. The lower limit is around 500-1000 particles before the estimates becomes too inaccurate. Hence, both experimental and simulated data shows that an increase in particles is not always analogous with higher accuracy in the estimates. Albeit the velocity estimates are not very accurate in the experiment, they fluctuate around zero. This suggest that the implementation works, but that the sensor uncertainty is too large for accurate estimates. In the transient case both simulated and experimental data yields the same convergence rates of just a few seconds. There are very little overshoot in the transition to steady state. Hence, experiments confirms that the proposed PF design used in the simulations is working. The accuracy of the filter depends on the available information used as input source. In the simulations the input source is of high quality. These inputs are not affected by the pitch angle of the ROV or misalignment of the instruments etc. Hence, the input used in the simulations are much more predictable and accurate.

Chapter 9

Concluding remarks

In this section a conclusion is made on basis of the results and theory presented in the previous chapters. A section with suggestions for further work is also included.

9.1 Conclusion

In this thesis two observer designs for use with ROV thurster models has been proposed. The proposed designs are based on the theory behind the eXogenous Kalman filter and the particle filter. The observers have then been compared both with each other, and the Extended Kalman filter, which is one of the most widely used state observers for underwater vehicles today.

In the steady state case where all filters were given correct initial values, the EKF and the XKF performed relatively similar in simulations. Both filters were however outperformed by the PF in terms of estimation accuracy. In the transient case, where the filters were given erroneous initial values, the differences between the filters were larger. The XKF and the PF were both able to handle erroneous initial values well, but for the PF this was only possible if the initial uncertainty intervals were specified correctly. In addition, the cascaded structure of the XKF made it possible to tune the NLO for transient behaviour and the Kalman filter for steady state. In this way the XKF could provide the best possible estimates also after the transient stage. In terms of stability and transient behaviour the XKF can be concluded to have the best characteristics.

Experimental data were gathered from an ROV in the NTNU MC-lab. As thruster measurements were not available in the lab, IMU-data had to replace the thruster model in the PF. Due to the lack of thruster measurements, the EKF and the proposed XKF design could not be tested using experimental data. However, two conclusions can be made from using the PF with experimental data. The first is that the proposed PF, and the underlying method used in this thesis, works with other and more realistic measurement sources. The second is that the PF should be used with sensors having low update frequencies. The high update rates of the IMU gave long computational times even if no simulator had to be run simultaneously with the filter.

Due to its superior stability properties, and fast computational times it is concluded that the XKF is the most suitable filter for autonomous underwater applications. It increases computational times slightly compared to the EKF, but in return it is much more robust and the accuracy in estimates are approximately the same. It is also implemented with rather simple extensions to the original EKF. Even though the PF is more accurate and has the fastest convergence rates, it is not voted the best alternative due to two reasons. First reason is the high computational times. In simulations the best performing PF used approximately 20 times longer time than the Kalman filters. This leads to the PF demanding too much computational power in many situations. It is also the reason why only six states were estimated with the PF in this thesis. In comparison the Kalman filter implementations estimates twelve states in a fraction of the time. The second reason is that the PF estimates are at risk of diverging if particles are not handled correctly. Too little spread in particles may lead to the particles loosing track of the measurements and the estimates diverges. However, the PF is highly customizable and it can handle any systems or noise disturbances. For this reason there are certainly special cases where the PF could do a better job than the Kalman filters.

9.2 Further work

Further work on this topic should include experimental testing where measurement of thrust is available. This would make it possible to also test the proposed XKF design with more realistic data sources. More importantly it would make it possible to directly compare the simulated results with experimental results. A further step would be to conduct simulations and experiments with the filters running in the control-loop.

It would also have been interesting to investigate the use of a nonlinear measurement function in the filters. The linear assumption made in this thesis is not necessarily true for real measurement sources. Hence, further work should investigate how other and more specific measurement sources affect the different filters.

Including more states in the XKF, for instance states for estimation of wave motion would be a great feature for a more robust filter. In the PF this is probably not applicable due to the high computational cost.

Bibliography

Allotta, B., Caiti, A., Chisci, L., Costanzi, R., Corato, F. D., Fantacci, C., Fenucci, D., Meli, E., and Ridolfi, A. (2016). An unscented kalman filter based navigation algorithm for autonomous underwater vehicles. *Mechatronics*, pages 185 – 195. Volume 39.

Ang, K. H., Chong, G., and Li, Y. (2005). PID control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology*, pages 559–576. Volume 13, Issue 4.

Arnesen, B. O. and Schjølberg, I. (2016). *Motion Control Systems for ROVs, Underwater Path-Following for a Videoray Pro 4 ROV*. Norwegian University of Science and Technology, Faculty of Engineering, Science and Technology, Department of Marine Technology.

Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, pages 174–188. Volume 50, Issue 2.

BlueRobotics. https://www.bluerobotics.com/store/rov/bluerov2/, accessed: 23.05.2018.

Breivik, M., E. Hovstein, V., and Fossen, T. I. (2008). Straight-line target tracking for unmanned surface vehicles. *Modeling, Identification and Control (MIC)*, pages 131–149. Volume 29, Issue 4.

Breivik, M. and Fossen, T. I. (2008). Guidance laws for planar motion control. In *47th IEEE Conference on Decision and Control*, pages 570–577.

Candeloro, M. (2016). *Tools and Methods for Autonomous Operations on Seabed and Water Column using Underwater Vehicles*. Norwegian University of Science and Technology, Faculty of Engineering, Science and Technology, Department of Marine Technology.

Candeloro, M., Dezi, F., Sørensen, A. J., and Longhi, S. (2012a). Analysis of a multi-objective observer for UUVs. *3rd IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles*, pages 343 – 348. Volume 45, Issue 5.

Candeloro, M., Sørensen, A. J., Longhi, S., and Dukan, F. (2012b). Observers for dynamic positioning of ROVs with experimental results. *9th IFAC Conference on Manoeuvring and Control of Marine Craft*, pages 85 – 90. Volume 45, Issue 27.

Chen, Z. (2003). Bayesian filtering: From kalman filters to particle filters, and beyond. *Statistics: A Journal of Theoretical and Applied Statistics*. Volume 182, Issue 1.

D'Alfonso, L., Lucia, W., Muraca, P., and Pugliese, P. (2015). Mobile robot localization via EKF and UKF: A comparison based on real data. *Robotics and Autonomous Systems*, pages 122 – 127. Volume 74.

Doucet, A., Godsill, S., and Andrieu, C. (2000). On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, pages 197–208.

Dunn, W. and Shultis, J. (2011). *Exploring Monte Carlo Methods*. Elsevier Science.

Faltinsen, O. M. (1993). *Sea Loads on Ships and Offshore Structures*. Cambridge University Press.

Fossen, T. I. (2011). *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 1st edition.

Gordon, N. J., Salmond, D. J., and Smith, A. F. M. (1993). Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F - Radar and Signal Processing*, pages 107–113. Volume 140, Issue 2.

Gustafsson, F. (2010). Particle filter theory and practice with positioning applications. *IEEE Aerospace and Electronic Systems Magazine*, pages 53–82. Volume 25, Issue 7.

Gustafsson, F. (2012). *Statistical Sensor Fusion*. Studentlitteratur.

Henriksen, E. H., Schjølberg, I., and Gjersvik, T. B. (2016). Adaptable joystick control system for underwater remotely operated vehicles. *10th IFAC Conference on Control Applications in Marine Systems, CAMS 2016*, pages 167 – 172. Volume 49, Issue 23.

Johansen, T. A. and Fossen, T. I. (2016). Nonlinear filtering with exogenous kalman filter and double kalman filter. In *2016 European Control Conference (ECC)*, pages 1722–1727.

Johansen, T. A. and Fossen, T. I. (2017). The exogenous kalman filter (XKF). *International Journal of Control*, pages 161–167. Volume 90, Issue 2.

Johansen, T. A., Fossen, T. I., and Goodwin, G. C. (2016). Three-stage filter for position estimation using pseudorange measurements. *IEEE Transactions on Aerospace and Electronic Systems*, pages 1631–1643. Volume 52, Issue 4.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, pages 35–45. Volume 82.

Khalil, H. (2015). *Nonlinear Control, Global Edition*. Pearson Education Limited.

Lekkas, A. M. and Fossen, T. I. (2014). Minimization of cross-track and along-track errors for path tracking of marine underactuated vehicles. In *2014 European Control Conference (ECC)*, pages 3004–3010.

Owen, A. B. (2013). *Monte Carlo theory, methods and examples*.

Refsnes, J. E. G. (2007). *Nonlinear Model-Based Control of Slender Body AUVs*. Norwegian University of Science and Technology Faculty of Engineering Science and Technology, Department of Marine Technology.

Schjølberg, I. and Utne, I. B. (2015). Towards autonomy in ROV operations. *4th IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles, NGCUV 2015*, pages 183 – 188. Volume 48, Issue 2.

Seel, K. and Schjølberg, I. (2016). *Autonomous navigation of an ROV using tightly coupled integration of inertial and pseudo-range measurements*. Norwegian University of Science and Technology, Faculty of Engineering, Science and Technology, Department of Marine Technology.

Sørensen, A. J. (2013). *Marine control systems, propulsion and motion control of ships and ocean structures*. Norwegian University of Science and Technology, Department of Marine Technology.

Stovner, B. N. (2018). *Aided Inertial Navigation of Underwater Vehicles*. Norwegian University of Science and Technology, Faculty of Engineering, Science and Technology, Department of Marine Technology.

Thrun, S., Burgard, W., and Fox, D. (2001). *Probabilistic Robotics*. Intelligent robotics and autonomous agents. MIT Press.

Wan, E. A. and Merwe, R. V. D. (2000). The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pages 153–158.

Zhao, B., Blanke, M., and Skjetne, R. (2012). Particle filter ROV navigation using hydroacoustic position and speed log measurements. In *2012 American Control Conference (ACC)*, pages 6209–6215.

Zhao, B., Skjetne, R., Blanke, M., and Dukan, F. (2014). Particle filter for fault diagnosis and robust navigation of underwater robot. *IEEE Transactions on Control Systems Technology*, pages 2399–2407. Volume 22, Issue 6.

Appendices

Appendix A

Simulator specifics

A.1 Model parameters

The parameters used in the Videoray Pro 4 model in Simulink are summarized below. These parameters are the same parameters as found in (Arnesen and Schjølberg, 2016).

Table A.1: Hydrodynamic derivatives and damping coefficients

$X_{\dot{u}} = 2.5028\ kg$	$X_u = 2.31\ kg/s$	$X_{\|u\|u} = 18.5741\ kg/m$
$Y_{\dot{v}} = 7.1401\ kg$	$Y_v = 5.0326\ kg/s$	$Y_{\|v\|v} = 29.4535\ kg/m$
$Z_{\dot{w}} = 8.0125\ kg$	$Z_w = 6.6261\ kg/s$	$Z_{\|w\|w} = 60.139\ kg/m$
$N_{\dot{r}} = 0.0395\ kg\ m^2$	$N_r = 0.0288\ kg\ m/s$	$N_{\|r\|r} = 0.052\ kg\ m/s$

Table A.2: Weight and balance data

$x_g = 0.00\ m$	$x_b = 0.00\ m$	$m = 7\ kg$
$y_g = 0.00\ m$	$y_b = 0.00\ m$	$I_z = 0.0521\ kg\ m^2$
$z_g = 0.10\ m$	$z_b = 0.14\ m$	

A.2 Simulink diagram XKF

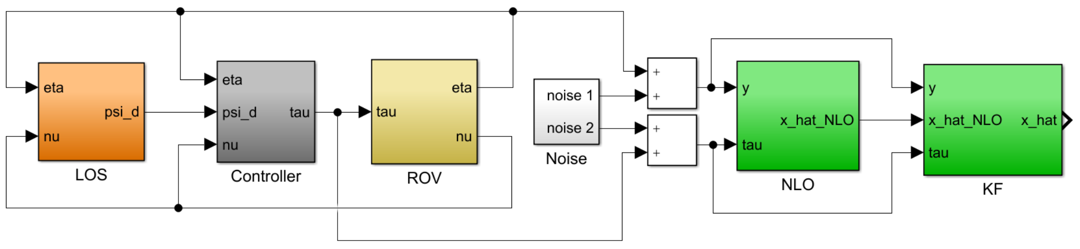


Figure A.1: Simulink diagram of simulator with XKF

A.2.1 Simulink diagram NLO

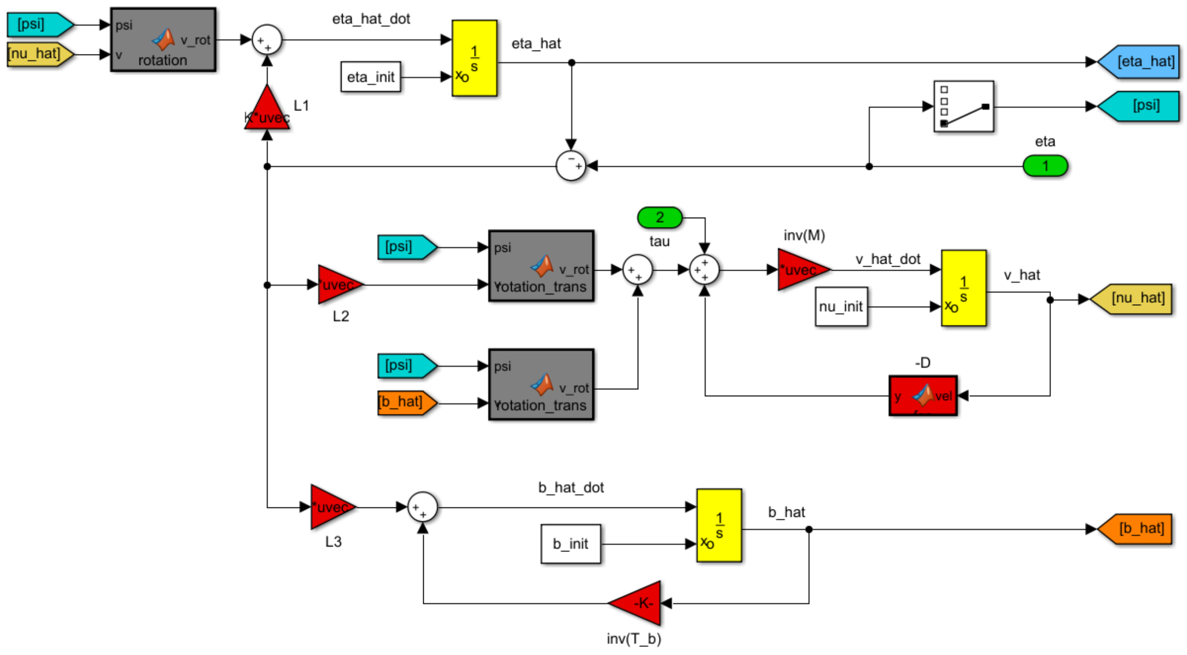


Figure A.2: Simulink diagram of the NLO used in the XKF

A.3 Simulink diagram PF

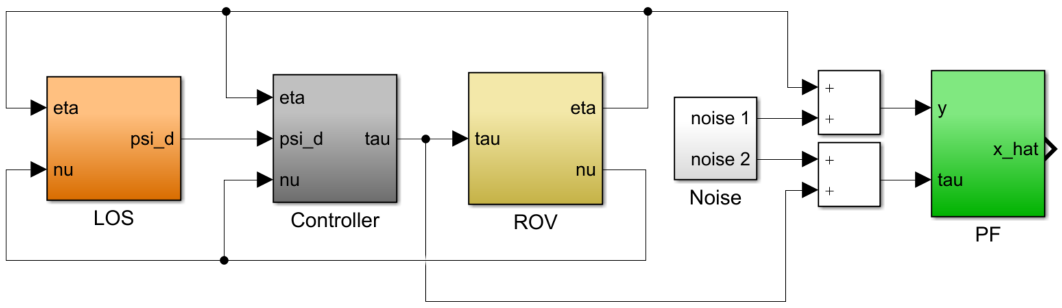


Figure A.3: Simulink diagram of simulator with PF

Appendix B

Program code

B.1 Particle filter using thruster model

This is the Matlab code used in the PF-block in the Simulink-diagram in appendix A.3.

B.1.1 Main

```
function x_hat = PF(y, init, noise, N, h, tau)
% y     —  measurement vector
% init  —  initial value vector
% noise —  system/measurement noise tuning parameters
% N     —  number of particles
% h     —  timestep
% tau   —  thruster measurements

% Initial particle values
persistent prevParticles
if isempty(prevParticles)
    prevParticles = zeros(N,6);
    prevParticles(:,1) = init(1)*ones(N,1);     % x
    prevParticles(:,2) = init(2)*ones(N,1);     % y
    prevParticles(:,3) = init(3)*ones(N,1);     % psi
    prevParticles(:,4) = init(4)*ones(N,1);     % u
    prevParticles(:,5) = init(5)*ones(N,1);     % v
    prevParticles(:,6) = init(6)*ones(N,1);     % r
```

```
end

% Noise in system (e) and measurement (v)
e = noise(1:3);
v = noise(4:6);

% Predict states based on ROV model
statePred = projectParticles(h,N,e,prevParticles,tau);

% Weight particles using measurement
Wmeas = weights(statePred,v,N,y);

% Resample based on weights
index = resample(Wmeas');

X =   statePred(index,1)';
Y =   statePred(index,2)';
Psi = statePred(index,3)';
U =   statePred(index,4)';
V =   statePred(index,5)';
R =   statePred(index,6)';

% Store particles for use in next iteration
prevParticles = [X ; Y ; Psi ; U ; V ; R ]';

% Final estimate
x_hat = [mean(X); mean(Y); mean(Psi); mean(U); mean(V); mean(R)];
end %PF
```

B.1.2 Sub-routines

```
function predictParticles = projectParticles( h,N,e,prevParticles,tau)

% Generate velocity hypothesises
tau_X = tau(1) + (e(1))^2*randn(N,1);
tau_Y = tau(2) + (e(2))^2*randn(N,1);
tau_Psi = tau(3) + (e(3))^2*randn(N,1);

```

```matlab
%% State projection
predictParticles = zeros(N,6);

for i=1:N
tau  = [tau_X(i), tau_Y(i), tau_Psi(i)]';
nu = [prevParticles(i,4), prevParticles(i,5), prevParticles(i,6)]';

% Velocity rates from mathematical model
nu_dot = VideorayPro4(tau, nu);

% Update velocity hypotheses
predictParticles(i,4) = prevParticles(i,4) + h*nu_dot(1);
predictParticles(i,5) = prevParticles(i,5) + h*nu_dot(2);
predictParticles(i,6) = prevParticles(i,6) + h*nu_dot(3);

% Update pose hypotheses
psi = prevParticles(i,3);
predictParticles(i,1) = prevParticles(i,1) + ...
                        h*(predictParticles(i,4)*cos(psi)- ...
                        predictParticles(i,5)*sin(psi));

predictParticles(i,2) = prevParticles(i,2) + ...
                        h*(predictParticles(i,4)*sin(psi) + ...
                        predictParticles(i,5)*cos(psi));

predictParticles(i,3) = psi + h*predictParticles(i,6);

end %projectParticles
```

```matlab
function nu_dot = VideorayPro4(tau, nu)

%Velocities
u = nu(1); % surge
v = nu(2); % sway
r = nu(3); % yaw

% Mass
m = 7; % Mass
Izz = 0.05205; % Yaw inertia
```

```
Xdu = −2.5028;   Ydv = −7.1401;   Ndr = −0.0395;
M = diag([m−Xdu, m−Ydv, Izz−Ndr]);

% Damping
Xu =  2.31+1;     Yv =  5.0326;   Nr =  0.0288; %D_Lin
Xuu = 18.5741+2;  Yvv = 29.4535;  Nrr = 0.0520; %D_Quad
D = diag([Xu + Xuu∗abs(u), Yv + Yvv∗abs(v), Nr + Nrr∗abs(r)]);

% Coriolis
CRB = [ 0 0 −m∗v; 0 0 m∗u; m∗v −m∗u 0];
CA  = [ 0 0  Ydv∗v; 0 0  −Xdu∗u; −Ydv∗v Xdu∗u  0];
C = CRB + CA;

% Velocity rates
a = inv(M)∗tau;
b = inv(M)∗D∗nu;
c = inv(M)∗C∗nu;

nu_dot = a(1:4,1)  − b(1:4,1) − c(1:4,1);
end % VideorayPro4
```

```
function W = weights(statePred,v,N,y)

W_x = zeros(1,N);
W_y = zeros(1,N);
W_psi = zeros(1,N);

% Assign weight to each particle in each measured dimension
for i=1:N
W_x(i) =   (1/sqrt(2∗pi∗v(1)))∗...
            exp(−(y(1)−statePred(i,1))^2/(2∗v(1)));

W_y(i) =   (1/sqrt(2∗pi∗v(2)))∗...
            exp(−(y(2)−statePred(i,2))^2/(2∗v(2)));

W_psi(i) = (1/sqrt(2∗pi∗v(3)))∗...
            exp(−(y(3) − statePred(i,3))^2/(2∗v(3)));
end

```

```

% Normalize weights
W_x = W_x./ sum(W_x);
W_y = W_y./ sum(W_y);
W_psi = W_psi./ sum(W_psi);

% Final weight is the mean of each normalized
% weight in measured dimensions
W = (W_x + W_y + W_psi)/3;

end %weights
```

```
function ind=resample(w)
% Return index of resampled particles
N=length(w);
u = rand(N,1);
wc = cumsum(w);
wc=wc/wc(N);
[dum,ind1]=sort([u;wc]);
ind2=find(ind1<=N);
ind=ind2-(0:N-1)';
end %resample
```

B.2 Particle filter using IMU data

B.2.1 Main

```
function x_hat = PF(y, init, noise, N, h, imu)
% y     -  measurment vector
% init  -  initial value vector
% noise -  system/measurement noise tuning parameters
% N     -  number of particles
% h     -  timestep
% imu   -  IMU measurements

% Initial particle values
if isempty(prevParticles)
    prevParticles = zeros(N,6);
    prevParticles(:,1) = init(1)*ones(N,1);     % x
    prevParticles(:,2) = init(2)*ones(N,1);     % y
    prevParticles(:,3) = init(3)*ones(N,1);     % psi
    prevParticles(:,4) = init(4)*ones(N,1);     % u
    prevParticles(:,5) = init(5)*ones(N,1);     % v
    prevParticles(:,6) = init(6)*ones(N,1);     % r
end

% Noise in system (e) and measurement (v)
e = noise(1:3);
v = noise(4:6);

% Predict states based on IMU-data
statePred = projectParticles(h,N,e,prevParticles,imu);

% Weight particles using measurement
Wmeas = weights(statePred,v,N,y);

% Resample based on weights
index = resample(Wmeas');

X =   statePred(index,1)';
Y =   statePred(index,2)';
Psi = statePred(index,3)';
```

```
U =   statePred(index,4)';
V =   statePred(index,5)';
R =   statePred(index,6)';

% Store particles for use in next iteration
prevParticles = [X ; Y ; Psi ; U ; V ; R ]';

% Final estimate
x_hat = [mean(X); mean(Y); mean(Psi); mean(U); mean(V); mean(R)];
end %PF
```

B.2.2 Sub-routines

Only one subroutine is changed to use IMU-data. Hence, the rest of the sub-routines are the same as in appendix B.1.

```
function predictParticles = projectParticles( h,N,e,prevParticles,imu)

% Generate velocity hypothesises
a_x = imu(1) + (e(1))^2*randn(N,1);
a_y = imu(2) + (e(2))^2*randn(N,1);
r   = imu(3) + (e(3))^2*randn(N,1);

% State projection
predictParticles = zeros(N,6);

% Update velocity hypotheses
predictParticles(:,4) = prevParticles(:,4) + a_x*h;
predictParticles(:,5) = prevParticles(:,5) + a_y*h;
predictParticles(:,6) = r;


% Update pose hypotheses
psi = prevParticles(i,3);
predictParticles(i,1) = prevParticles(i,1) + ...
                        h*(predictParticles(i,4)*cos(psi)- ...
                        predictParticles(i,5)*sin(psi));

predictParticles(i,2) = prevParticles(i,2) + ...
                        h*(predictParticles(i,4)*sin(psi) + ...
                        predictParticles(i,5)*cos(psi));

predictParticles(i,3) = psi + h*predictParticles(i,6);

end %projectParticles
```

B.3 eXogenous Kalman filter

This is the Matlab code used in the KF-block in the Simulink-diagram in appendix A.1.

B.3.1 Main

```matlab
function [x_hat,x_bar_n,P_bar_n]  = EKF(y,tau,Q,R,h,x_hat_NLO,x_bar,P_bar)
% y          - measurement vector
% tau        - measured thrust
% Q          - system noise matrix
% R          - measurement noise matrix
% h          - timestep
% x_hat_NLO - lin. point from NLO
% x_bar      - state projection from last iteration
% P_bar      - covariance projection from last iteration

I = eye(12,12);

%% State space
% This is only the linear matrices of the state-space.
% Non-linear system matrix is found in function "fx"

m = 7; %Mass
Izz = 0.05205; %Yaw inertia
Xdu = -2.5028; Ydv = -7.1401; Zdw = -8.0125; Ndr = -0.0395;
M = diag([m-Xdu, m-Ydv, m-Zdw, Izz-Ndr])

%% State space matrices
% Input matrix
B = [zeros(4,4)
     inv(M)
     zeros(4,4) ];

% Noise matrix
E = [zeros(4,4) zeros(4,4) zeros(4,4)
     zeros(4,4) inv(M)     zeros(4,4)
     zeros(4,4) zeros(4,4) ones(4,4)];

```

```
% Measurement matrix
H = [1 0 0 0 0 0 0 0 0 0 0 0 0 0
     0 1 0 0 0 0 0 0 0 0 0 0 0 0
     0 0 1 0 0 0 0 0 0 0 0 0 0 0
     0 0 0 1 0 0 0 0 0 0 0 0 0 0];

%% Extended kalman filter with linearization point from NLO

% Kalman gain
K = (P_bar*(H.')*inv(H*P_bar*(H.') + R));

% Update state estimate
x_hat = x_bar + K*(y - H*x_bar);

% Update error covariance
P_hat = (I - K * H)*P_bar*(I - K * H)' + K*R*(K.');

% State projection
x_bar_n = x_hat + h* ((fx(x_hat_NLO)+...
          jacobian(x_hat_NLO)*(x_hat-x_hat_NLO)) + B*tau);

% Error covariance projection
 phi = I + h*jacobian(x_hat_NLO); % Discretize
 gamma = h*E;
 P_bar_n = phi*P_hat*(phi.') + gamma*Q*(gamma.');
end %EKF
```

B.3.2 Sub-routines

```matlab
function f = fx(x_hat)
%Nonlinear system model

psi = x_hat(4);    % Heading estimate
nu = x_hat(5:8)'; % Velocity estimates
b = x_hat(9:12);  % Bias estimates

% Mass
m = 7;
Izz = 0.05205;
Xdu = -2.5028;   Ydv = -7.1401;   Zdw = -8.0125;   Ndr = -0.0395;
M = diag([m-Xdu, m-Ydv, m-Zdw, Izz-Ndr]);

% Damping
Xu =  2.31+1;     Yv =  5.0326;   Zw =  6.6261;   Nr =  0.0288;
Xuu = 18.5741+2;  Yvv = 29.4535;  Zww = 60.139;   Nrr = 0.0520; %D_Quad
D = diag([Xu + Xuu*abs(u), Yv + Yvv*abs(v), Zw + Zww*abs(w), Nr + Nrr*abs(
    r)]);

% Coriolis
CRB = [ 0 0 0 -m*v;0 0 0 m*u; 0 0 0 0;m*v -m*u 0 0];
CA  = [ 0 0 0 Ydv*v;0 0 0 -Xdu*u;0 0 0 0;-Ydv*v Xdu*u 0 0];
C = CRB + CA;

% Rotation matrix
R = [cos(psi)   -sin(psi)      0     0
     sin(psi)    cos(psi)      0     0
     0           0             1     0
     0           0             0     1 ];

% Non-linear system-matrix
f = [R*nu
    -inv(M)*(C+D)*nu+inv(M)*R'*b
    -ones(4,4)*b];
end %fx
```

```
function [J]=jacobian(x)
% Numerical computation of the Jacobian of fx at x

n=length(x);
J=zeros(n);
f = fx(x);
eps=1.e-8;
xperturb=x;
for i=1:n
   xperturb(i)=xperturb(i)+eps;
   fperturb = fx(xperturb);
   J(:,i)=(fperturb-f)/eps;
   xperturb(i)=x(i);
end %jacobian
```