DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Design of an Extended Kalman Filter for an Autonomous Sailboat

Relatório submetido à Universidade Federal de Santa Catarina
como requisito para a aprovação da disciplina:
DAS 5511: Projeto de Fim de Curso

Miguel Budag Becker

Florianópolis, Julho de 2019

Design of an Extended Kalman Filter for an Autonomous Sailboat

Miguel Budag Becker

Esta monografia foi julgada no contexto da disciplina

DAS 5511: Projeto de Fim de Curso

e aprovada na sua forma final pelo

Curso de Engenharia de Controle e Automação

Prof. Hector Bessa Silveira da UFSC

Banca Examinadora:

Moritz Buehler
Orientador na Empresa

Prof. Hector Bessa Silveira
Orientador no Curso

Prof. Hector Bessa Silveira
Responsável pela disciplina

Alice Ferreira Branco, Avaliador

Gabriel Thaler, Debatedor

Michelle Moreira de Araújo, Debatedor

Dedicado à minha falecida mãe que,
sempre me deu confiança para seguir em frente,
e enchergar como a vida pode ser feliz.

Acknowledgements

My deepest thanks to all the people that helped this work come true, and in special, my loved ones.

To my Professor and academic advisor Hector Bessa Silveira for the guidance, patience through this journey.

To Professor J. Adamy and my supervisor Moritz Bühler for the opportunity to work in this project, and expand my knowledge and interests beyond what was available otherwise.

"Science is a differential equation.
Religion is a boundary condition.
(Alan Turing)

Resumo

O presente trabalho foi desenvolvido no Laboratório de Métodos de Controle e Robótica (RMR) na TU Darmstadt - Alemanha, em parceria com a Sailing Team Darmstadt (STDA). Seu objetivo é desenvolver e construir um veleiro que atravesse o Oceano Atlântico sem tripulação e seja autossuficiente em energia apenas com propulsão eólica. A travessia do Oceano Atlântico corresponde a uma distância de cerca de 7000 km e demora pelo menos dois meses a atravessar. Para o controle e planejamento de trajetórias de um veleiro autônomo, é necessário ser capaz de conhecer as condições do sistema com confiança (como posição, velocidade, orientação). Mesmo que um dos sensores individuais pare de funcionar, o barco tem que ser capaz de se manter em operação com segurança. Com isso em mente, um Filtro de Kalman pode ser projetado para combinar os vários dados dos sensores e estimar os estados do sistema (posição(x,y), velocidade, *yaw* e velocidade angular), obtendo a melhor estimação possível, mesmo na presença de ruídos de medição e pertubações externas. Além disso, o oceano é um ambiente hostil, com condições meteorológicas complicadas, de modo que a maioria dos dispositivos tendem a apresentar defeitos após certo tempo. Em barcos autônomos, esses defeitos são especialmente importantes devido à forte dependência que medições precisas têm para um bom funcionamento do sistema. Por isso, é importante analisar se e quais falhas do sensor podem ser compensadas. Uma análise da dinâmica de um veleiro autônomo e dos algoritmos de estimação propostos culminou na seleção e implementação de um modelo não-linear e de um Filtro de Kalman Estendido. Este filtro foi então testado em diferentes cenários para avaliar sua robustez em situações reais através de simulações numéricas. No final do trabalho, foi obtido um método capaz de estimar os estados do veleiro com bastante precisão, mesmo para casos com altas covariâncias do ruído de medição, além de se mostrar tolerante a falhas de sensores. Este trabalho fornece um sistema de grande precisão e robustez na estimativa dos estados de um veleiro autônomo, permite sua análise sob uma simulação controlada, contribuindo massivamente para a subsequente implementação do veleiro autônomo.

Palavras-chave: Veleiro Autônomo, Estimação, Filtro de Kalman, Filtro de Kalman Estendido, Modelagem de Veleiro, Falha de sensores.

Abstract

The present work was developed at the Control Methods and Robotics Lab (RMR) in TU Darmstadt - Germany, in partnership with the Sailing Team Darmstadt (STDA). Their aim is to develop and build a sailboat that will cross the Atlantic Ocean unmanned and energy self-sufficient with wind propulsion alone. The Atlantic crossing corresponds to a distance of about 7000 km and take at least two months to cross. For the control and path planning of a autonomous sailing boat, it's necessary to have a reliable knowledge of the conditions of the system. Even if one of the individual sensors stops working, the boat has to be able to keep safely operating. With this in mind, a Kalman Filter must be designed to combine the various data from the sensors, and estimate the states of the system (including position, velocity, yaw angle), obtaining the best estimation possible. Besides the sea is a harsh environment with complicated weather conditions so most devices like the GPS module antennas or wind sensors are prone to malfunction under long periods of time. In autonomous sailing boats this is specially important because of the strong dependence on accurate measurements to have a good operation. So it's important to analyze if and which sensor failures can be compensated. An analysis of the dynamics of a autonomous sailing boat and of the estimation algorithms proposed culminated in a selection and implementation of a nonlinear model, and of a Kalman Filter. This filter was then tested under different scenarios to assess its robustness under a real world scenario as best as it could be in a simulation. At the end of the work, a method was obtained able to estimate the states of the sailboat quite accurately even for cases with high measurement noise covariances, while also being resistant to sensor failures. This work provides a system of great accuracy and robustness in the estimation of the states of an autonomous sailboat, allows its analysis under a controlled simulation, contributing massively to the subsequent implementation of the autonomous sailboat.

Keywords: Autonomous sailing boat, Estimation, Kalman Filter, Extended Kalman Filter, Sailboat modeling, Sensor fault, Sensor failure.

List of Figures

List of Tables

List of abbreviations and acronyms

FCP	Final Course Project
TU	Techniche Universitat
RMR	Regelungsmethoden und Robotik
KF	Kalman Filter
EKF	Extended Kalman Filter
UKF	Unscented Kalman Filter
DOF	Degrees of freedom
IMU	Inertial measurement unit
GPS	Global Positioning System
DGPS	Differential Global Positioning System
RMS	Root Mean Square
CEP	Circular error probable

Contents

1 Introduction

Sailing is a point of great interest since ancient times, as it was the main form of moving across the vast oceans for hundreds of years. However, an actual in-depth study of the physics behind it is a much more recent phenomenon, and is dated back to the 1970s [5]. The Boeing engineer Arvel Gentry made an effort to apply his knowledge of fluid dynamics to sailing yachts and found that much of had been produced around the subject was very deceitful. In one of his articles about the subject, he described how sails really work (literally the name of the article), and came to the conclusion that all the explanations in the sailing books on the interaction between the jib and mainsail are wrong [6]. The possibility of applying the advancements made in other fields, specially regarding the ability to accurately obtain information that was often relegated to guesswork and conjectures, such as precise measurements, allowed for the eventual arrival of present day models. Based on the high understanding of the physics behind its dynamics [7], the creation of convenient and effective simulations are now possible.

The present work was developed at the Control Methods and Robotics Lab (RMR) in TU Darmstadt - Germany, in partnership with the Sailing Team Darmstadt (STDA). Their aim is to develop and build a sailboat that will cross the Atlantic Ocean unmanned and energy self-sufficient. The boat is meant to compete in the Microtransat Challenge[1], more specifically in the fully autonomous sailing class, which means crossing the Atlantic Ocean with wind propulsion alone.

The Atlantic crossing corresponds to a distance of about 7000 km and will take at least two months. The boat must be well prepared for all weather conditions. The expanded sensor system and the completely self-sufficient energy supply, present in boat built by the STDA team, figure 1, requires the use of state estimation techniques, such as the Kalman Filter. This necessity opened the opportunity for participation of students of the course of Control Engineering and Automation in the project, providing thus the development of the present Final Course Project (FCP).

1.1 Motivation

The aim of the present work is to contribute to autonomous missions with propulsion by sails alone over large periods of time and across long distances, having guarantees of reliable information of the systems conditions while in the presence of noisy measurements, also accounting for extreme situations like an individual component failure in the middle

[1] https://www.microtransat.org/

Figure 1 – Prototype II developed by the STDA, comprised of a 2.20m long fuselage, mainsail, jib, expanded sensor system, and a completely self-sufficient energy supply provided by a solar panel (Image included with permission from Sailing Team Darmstadt e.V.)

of the ocean.

The sailboat's trajectory is decided by the control system inside the sailboat, which in turn needs stable and reliable measurements for it to function. However, the data provided by the sensors have measurement noises imbued in them by default, making the raw measurement unreliable as a estimation. This is where the Kalman Filter is useful as it can provide estimations with less noise than just utilizing the measurement alone.

All the energy for propulsion has to be generated only by the sails, and other subsystems of the boat such as the controller, sensors and actuators need to be supplied by other energy sources in order to function. And as discussed in length by [8], the limited space for solar panels and batteries for power supply cause various limitations of the types of solutions that are possible to be implemented, specially considering the overall time they will need to function, plus the necessity of constant monitoring and input from the actuators to properly follow a trajectory.

The last point is that the sea is a harsh environment and even for manned expeditions the adverse weather conditions are a serious issue. Most devices can't deal very well with the salt water and intense wind, specially sensors that need a certain level of

exposure, like the GPS module antennas or wind sensors. They are prone to malfunction, loss of signal or plain shutdown. In autonomous sailing boats this problem is increased, where they have no mean to be repaired or faults may go unnoticed for longs periods of time.

A faulty sensor sometimes might be even worse than a completely disabled one. This being a very important topic in unmanned systems, where sensor faults, if not detected and reconfigured in time, can lead to closed-loop instability and unrecoverable conditions, by feeding wrong measurement data to the control system. Consequently, the ability to disable them if necessary is crucial not just for energy savings, but also for safety and robustness.

So the main challenges with autonomous sailboats and sailing in general can be summed up as:

- non-intuitive dependence on wind direction for generating propulsion;
- noisy measurements,
- limited power supply,
- highly susceptible to sensor failure.

With all the limitations imposed on the hardware, such as limited space and energy available, as well as harsh environmental conditions, it falls on the software side to improve upon the data provided by the use of algorithms, such as the Kalman filter (KF). It might even be able to loosen those limitations by allowing for solutions like operating sensors and actuators at lower sampling rates, and still maintaining a good operational status. Kalman filter might even allow for operating in those extreme situations, thus having the potential of being a incredible versatile and robust algorithm.

1.2 Objectives

The general objective of this work is to increase the accuracy in the estimation of the states of the sailboat. The estimation is necessary for the control system, which regulates the trajectory to be followed.

It is desired to obtain a more reliable system that uses all the autonomy that the energy storage system can provide, checking for it's robustness under extreme situations, like sensor unavailability or fault.

The study will be based on nonlinear Kalman filter techniques to verify the improvement that its inclusion has in the estimation based on the noisy measurements that come by default with the sensors utilized.

For the realization of the project, both in the theoretical and practical scope, the specific objectives are listed below:

- Review the literature on Kalman filters;
- Review the literature on sailboat models and choose the most appropriate one.
- Study the Kalman filter literature.
- Obtain a theoretical model, consolidated in the literature, to simulate the sailboat in continuous time.
- Choose and implement a Kalman filter algorithm with the simulated sailboat data, in order to validate it.
- Implement, analyze and apply solutions to sensor fault scenarios.

1.3 Methodology

This work was developed based on 2 main steps.

At first the examination of the autonomous sailboat behavior. Continuing to the selection of a simpler model that can generate fast simulations, while being easier to implement and understand, and still being complex enough to represent the actual system. This model is used as a basis for subsequent study of the work.

The second step is to study and apply the estimation algorithm to the simulated data in order to test and validate it under different scenarios. The scenarios include sensor failure, and its correction, which also have to be implemented in simulations. This validation of the system opens the door to the possibility of practical implementation in the future, while also giving some insight about how the mechanics of the sailboat affect its estimation.

1.4 Structure of the document

The document is organized as follows. The second chapter presents the theoretical foundations of the present work. One beings by providing a review of sailboats briefly explaining how they work and it's complex technical issues. Then one presents the most usual existing techniques for modeling these sailboats, as well as their advantages and disadvantages. Lastly, one presents the methods of state estimation, in specific the KF. This culminates in the choice of a sailboat model to be used in conjunction with chosen Kalman filter technique.

The third chapter applies the knowledge presented in the the second chapter and applies to the implementation steps. The system used to generate the simulated sailboat data and the estimation of the data are presented. It also details the choice of the tuning parameters of the KF utilized in the simulations, and the process in which the sensor fault was detected and considered.

The fourth chapter presents the evaluation of the estimation algorithm under different relevant scenarios, including position and orientation sensor faults, position measurements with higher and lower measurement noises, and the shutting down of some sensors. This is made to ensure that the Kalman filter developed could sustain operation under a practical environment. This chapter also presents the results from the sensor fault correction implemented.

The last chapter is the conclusion of the project, meant as a discussion of the overall results of the project. In this, it is expected to validate the project objectives, as well as to verify if improvements were achieved, difficulties of the project and future works to be developed.

2 Theoretical Background

This chapter will briefly present a review of the contents that are relevant to the understanding of the work. The theoretical bases on sailboats are presented in the section 2.1. The existing models are analyzed in section 2.2. The model chosen is detailed in section 2.3. Next, the state estimators will be discussed in section 2.4, with KF being discussed in section 2.5 and section 2.6 for non-linear approaches.

The level of understanding presented in the text is sufficient for understanding the following chapters and, if it is in the reader's interest, a vast bibliography for further depth on the subject is provided.

2.1 Sailing boats behavior

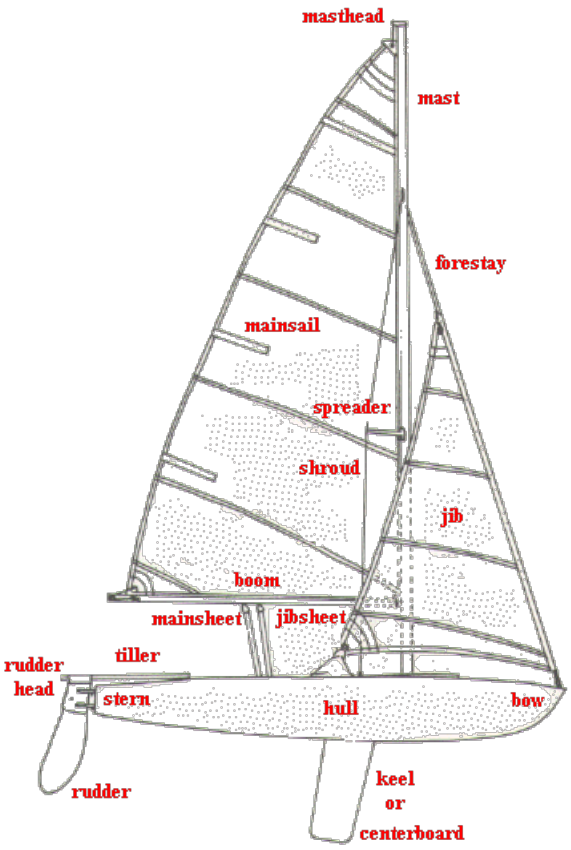


Figure 2 – Parts of a sailboat. In special note the hull, responsible for sustaining the boat and providing buoyancy, the mainsail and jib that provide the force from the wind, the keel that balances the aforementioned force resulting in the total force only in the forward direction, and rudder which controls the steering [1]

A sailing boat is in general a boat that is propelled by the wind using one or more sails. There are various different types and are normally composed of a mainsail, mast, hull, keel, and generally a headsail, as seen in the boat parts at figure 2. Depending on the size and shape of the headsail, it can be called a jib, genoa or spinnaker.

Thinking about examples of non-intuitive sailing boats behaviors, it's difficult to imagine that a boat could actually sail faster than the wind itself, but that's indeed possible. When sailing in a angle, the apparent speed of the wind on the sails can be greater than the actual speed of the wind, as in measured by a inertial observer. This relative wind creates a larger force on the sails that can push sailboats faster than the actual wind speed, as seen in figure 3.

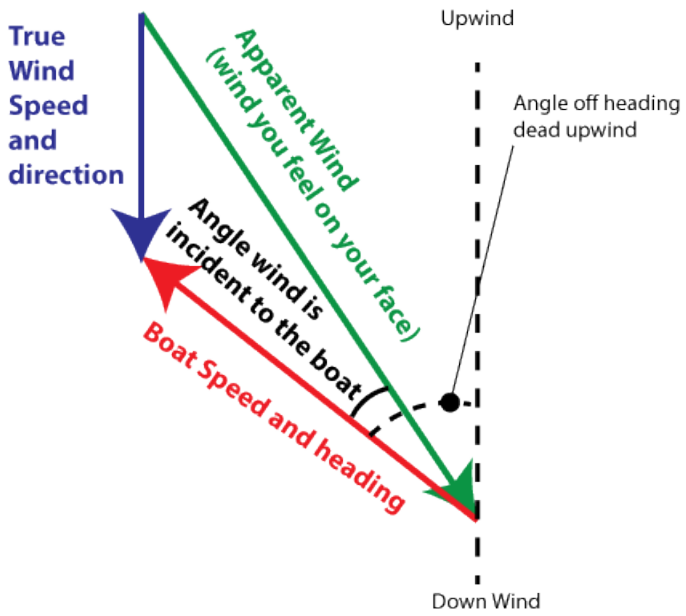


Figure 3 – true vs apparent wind, when moving from the right to left upwind [2]

Consequently it's no wonder that the models developed are incredibly nonlinear even at it's most simplified versions. Even bare minimum of dynamics introduced, like the notions of apparent wind and true wind as described above, proceed to introduce forces, vectors and subsequently trigonometrical functions into your model.

Also, it's important to remember that there is physical limitations imposed on the rudder and sail angles, when considering the extent of the complications included on the models.

2.2 Existing sailboat models

There exists quite a number of models dedicated to represent the navigation and control of ships, in special Fossen [9] which described the dynamics of motorized marine vehicles. However since sailboats are wind-powered, special mechanics have to be considered, and as such, different types of models were developed considering sailboats in specific [10], [11] and [12].

Originally this project was planned to work with the model developed Moritz et al [13], which outline a six degrees of freedom (6 DOF) model including wave effects while formulating the acting forces using fewer parameters in a effort to minimize the need of physical experimentation. Still, considering the general complexity of the model and the limited time available, a simpler comparable model was chosen. The selected model is a 3 DOF model developed by Melin [14], based on the model presented in Jaulin [15]. The simulation was made in continuous time, with the addition of continuous white noise as process and measurement noise.

Some of the simplifications compared to [13] include:

- Motion is confined to a horizontal plane at sea level disregarding roll and pitch. Therefore the model has only three DOF: surge, sway and yaw.
- Influence from waves and currents will be neglected.
- The mainsail and the headsail are combined into one effective sail.
- Lift and drag forces are combined into one force acting perpendicular to both sail and rudder.

Next sections display a brief explanation of the model presented in [14].

2.3 Sailing boat dynamics model

As mentioned in [14] the model presented is very close to the one developed in [15], so some relevant considerations regarding the difference between the two are:

(i) the control of the sail is made by the sail angle, being considered proportional to the length of the mainsheet, subsequently removing it from the equations;

(ii) The rudder force keeps the square on the boats velocity.

(iii) Sail sheet dynamics are simplified and accounted for by letting the model update the sail angle.

To start a reference system is needed, and the one utilized is the same of aforementioned paper, a North-East-Up reference frame, for simplicity. The origin of the system is the boat center of gravity. Since wind is defined as true wind and apparent wind, its referenced to the earth and the sailboat respectively.

2.3.1 Wind model

The wind is described using the vector $\mathbf{W}$, with the subscripts tw and aw regarding to the true and apparent wind. So being a_{tw} defined as the speed of the true wind and ψ_{tw} as the angle of the true wind, $\mathbf{W}_{tw}$ is defined in polar coordinates as,

$$\mathbf{W}_{p,tw} = \begin{bmatrix} a_{tw} \\ \psi_{tw} \end{bmatrix}. \tag{2.1}$$

Subsequently, $\mathbf{W}_{aw}$ is defined first in Cartesian coordinates as,

$$\mathbf{W}_{c,aw} = \begin{bmatrix} a_{tw}\cos(\psi_{tw} - \theta) - v \\ a_{tw}\sin(\psi_{tw} - \theta) \end{bmatrix}, \tag{2.2}$$

with the correspondent polar coordinates given by polar-cartesian conversion,

$$\mathbf{W}_{p,aw} = \begin{bmatrix} a_{aw} \\ \psi_{aw} \end{bmatrix} = \begin{bmatrix} |\mathbf{W}_{c,aw}| \\ \text{atan2}(\mathbf{W}_{c,aw}) \end{bmatrix}. \tag{2.3}$$

2.3.2 Nonlinear dynamics

The model is characterized by a set of nonlinear differential equations in state space. It contain five states, the position (x, y), orientation or yaw (θ), velocity (v) and angular velocity (w), as seen in figure 4. The control variables, δ_r and δ_s are correspondingly the rudder angle and sail angle. For general understanding the wind variables present in (2.2) are put together on the input vector $\mathbf{u}$.

Thus with the definitions of the state and input vectors respectively as

$$\mathbf{x} = \begin{bmatrix} x & y & \theta & v & w \end{bmatrix}^T, \tag{2.4}$$

$$\mathbf{u} = \begin{bmatrix} \delta_r & \delta_s & a_{tw} & \psi_{tw} \end{bmatrix}^T, \tag{2.5}$$

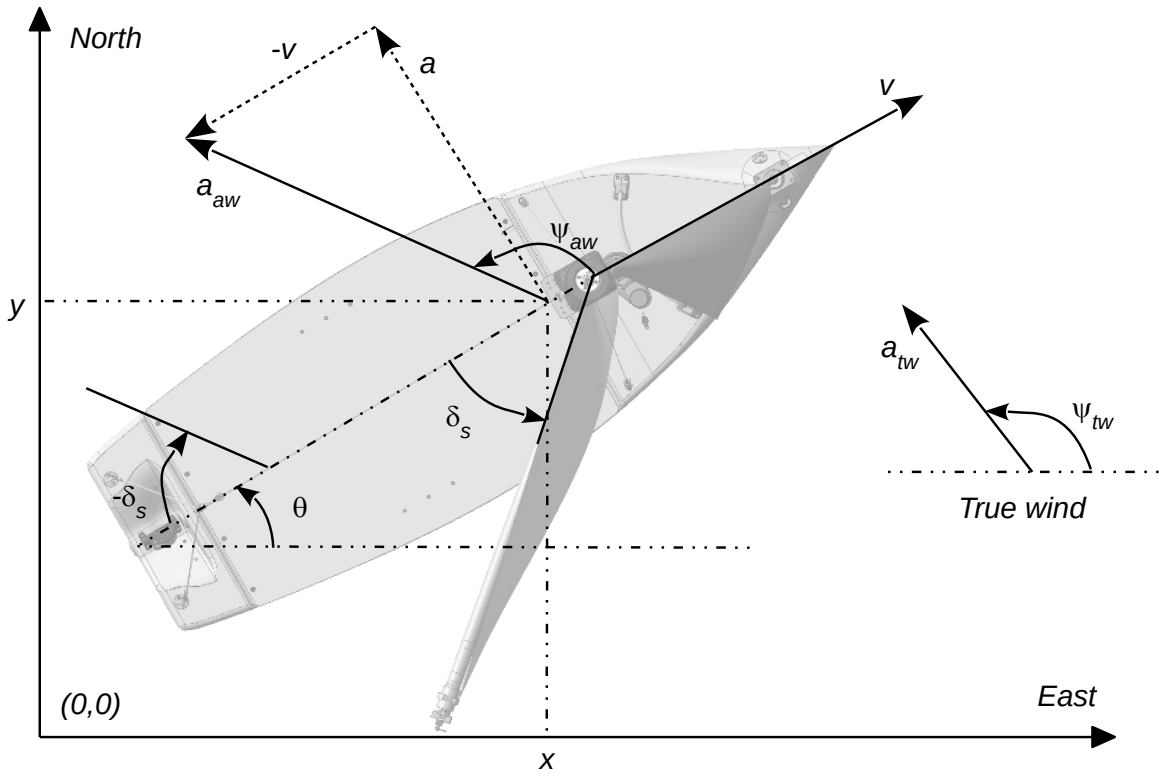


Figure 4 – Definition of the system, containing the states and input variables affecting the sailboat

a system of nonlinear dynamics is obtained

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) + \mathbf{q}, \tag{2.6}$$

$$\mathbf{z} = h(\mathbf{x}, \mathbf{u}) + \mathbf{r}, \tag{2.7}$$

with

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} v\cos\theta + p_1 a_{tw}\cos\psi_{tw} \\ v\sin\theta + p_1 a_{tw}\sin\psi_{tw} \\ w \\ \left(g_s\sin\delta_s - g_r p_{11}\sin\delta_r - p_2 v^2\right)/p_9 \\ \left(g_s\left(p_6 - p_7\cos\delta_s\right) - g_r p_8\cos\delta_r - p_3 wv\right)/p_{10} \end{bmatrix} + \mathbf{q}, \tag{2.8}$$

and the measurement equations as

$$\mathbf{z} = \mathbf{C}\mathbf{x} + \mathbf{r}, \ with \tag{2.9}$$

$$\mathbf{C} = \begin{bmatrix} H_x & 0 & 0 & 0 & 0 \\ 0 & H_y & 0 & 0 & 0 \\ 0 & 0 & H_\theta & 0 & 0 \\ 0 & 0 & 0 & H_v & 0 \\ 0 & 0 & 0 & 0 & H_w \end{bmatrix}. \tag{2.10}$$

The vectors $\mathbf{q}$ and $\mathbf{r}$ represent process and measurement noises respectively. The linear matrix C corresponds to the linear sensor functions, with it's main diagonal H_x, H_y, H_θ, H_v and $H_w \in \mathbb{R}$ being unitary when considering all measurements as being active.

Besides the change in position, the first two lines of equation (2.8) describe the boat's drift in function of the true wind speed and angle, multiplied by a drift coefficient. Drift occurs when wind interacts with the other boat surfaces besides the sail. The change in heading is described directly with the angular velocity state. The acceleration of the sailboat is directly proportional to the net force of the system, which is the relation between the force generated by the sail, and the resisting forces coming from the rudder and tangential friction, divided by the boat's mass. The angular acceleration, is described by the net torque and the damping term created by the rotational friction, divided by the boat's moment of inertia.

Other important factor of this model is the mainsheet tightness or flexibility against the wind, which is simulated by the updating of the sail angle inside the model itself,

$$\delta_s = -\text{sign}(\psi_{aw})\min\left(\ |\pi - |\psi_{aw}|| \, , \ |\delta_s| \ \right), \tag{2.11}$$

with the apparent wind angle ψ_{tw}, in radians, set between $-\pi \leq \psi_{tw} \leq \pi$.

2.3.3 Modeled forces

Regarding the forces generated by the sail and the rudder, they are simplified into a lift and drag force acting in a center of effort. The lift and drag forces are then combined into one force, g, acting perpendicular to the sail, g_s, and the rudder, g_r.

$$\begin{bmatrix} g_s \\ g_r \end{bmatrix} = \begin{bmatrix} p_4 a_{aw} \sin(\delta_s - \psi_{aw}) \\ p_5 v^2 \sin(\delta_r) \end{bmatrix}. \tag{2.12}$$

The resulting forces are approximated by the product of a constant, the velocity of incoming flow squared (air or water) and sinus of the attacking angles. The angles of attack

are determined by the direction of the apparent wind and the sail's angle, $\alpha_s = \psi_{aw} - \delta_s$, and the apparent velocity of the water near the rudder, $\alpha_r = -\delta_r$. The negative signs introduced by both angles are moved to the state equations (2.8) for simplicity. It's also important to notice that the square on the apparent wind velocity is neglected to account for the smaller effective area of the sail caused by boat's roll angle increase resulted of the higher apparent wind velocity.

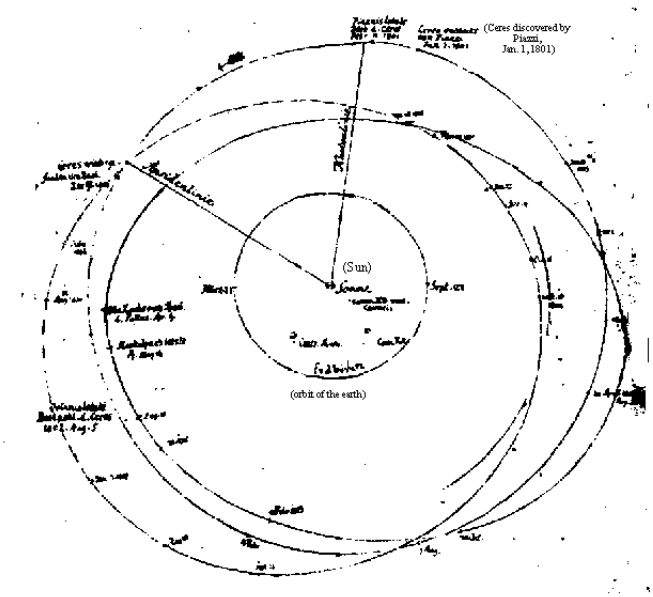
2.3.4 Model parameters

An explanation of all the parameters p_i can be found in Table 1. It also contains the comparison with the original values presented on [16] and a brief account on how some parameters were adapted to consider the different boat present in [13].

The true wind speed a_{tw} and ψ_{tw} could also be considered as parameters since they are constant on the simulations performed.

2.4 What are state estimators?

Measurements are everywhere, and have been so for longer than we normally think. At least since the time of Galileo Galilei (1564–1642), we already had a idea that these measurements come invariably coupled with some degree of noise, and with time, science only gave us more proof of that inevitability. As such a method to deal with those errors has been in constant research through the history of science.



Sketch of the orbits of Ceres and Pallas (nachlaß Gauß, Handb. 4). Courtesy of Universitätsbibliothek Göttingen.

Figure 5 – Sketch of the orbits of Ceres and Pallas, by Gauss [3]

Table 1 – Boat model parameters p_i, values utilized on this paper at the left, and original values for comparison at the right

parameter	value	unit	description	original value
p_1	0.03[a]	-	Drift coefficient	0.03
p_2	23[b]	kgs^{-1}	Tangential friction	40
p_3	213[c]	kgs^{-1}	Angular friction	6000
p_4	200[d]	kgs^{-1}	Sail lift	200
p_5	1500[d]	kgs^{-1}	Rudder lift	1500
p_6	0.43	m	distance to sail CoE (center of effort)	0.5
p_7	0.68	m	distance to mast	0.5
p_8	1.24	m	distance to rudder	2
p_9	350	kg	Boat mass	300
p_{10}	1066	kgm^2	Moment of inertia	400
p_{11}	0.2[a]	-	Rudder break coefficient	0.2

[a] Parameters p_1 and p_{11} didn't change because the nonlinear model present in [13] don't use them.
[b] Parameter p_2 is approximated based on the mass and damping terms of the velocity in the surge direction.
[c] Parameter p_3 is approximated based on the moment of inertia in the yaw axis and the yaw time constant.
[d] Parameters p_4 and p_5 are calculated in a differently being difficult to compare, thus are left the same.

The first to use such methods was Carl Friedrich Gauss (1777–1855) in 1795, with the *method of least squares*. Although nowadays it's mostly used for linear estimation, Gauss demonstrated its usefulness for non-linear problems when he used it to predict the future location of the newly discovered asteroid Ceres, figure 5, a feat considered incredibly hard using Kepler's complicated nonlinear equations of planetary motion. The possibility of estimating unknown parameters of theoretical models with such a practical and reliable method is, for anyone that even once tried a hand at modelling complex non-linear systems, incredibly important.

This ability to predict complex behaviors with simpler methods gave scientist a lot of power to extend theoretical models into models that can actually give results that fit reality.

2.5 The Kalman Filter

It's interesting to notice the similarities about how optimal estimation methods permitted Gauss to calculate the orbit of Ceres and how the Kalman filter was applied to navigation for the Apollo Project. It was used for estimating the trajectories of manned spacecraft going to the Moon and back, a few years after it was first published, an incredible feat for the time [17]. Another similarity is the dependency of Gaussian distributions of the errors to the functioning of the Kalman filter.

Generally a Kalman filter is a algorithm that makes the best use of both worlds, model-based predictions together with measurements that update those predictions.

Kalman filtering is a huge field whose depths are not fully explored in this project. An incredible amount of thesis and articles have been written on this subject since its publication, and much of the theory behind this section can be read in greater detail in books such as [18].

But to get a understanding of what Kalman Filters are, it's interesting to start with a discrete linear example. Consider the linear discrete equations,

$$\mathbf{x_k} = A\mathbf{x_{k-1}} + B\mathbf{u_k} + \mathbf{q_k}, \tag{2.13}$$

$$\mathbf{z_k} = C\mathbf{x_k} + \mathbf{r_k}, \tag{2.14}$$

where the the variables (state, observation, noise, control) are vectors, and the matrices represent the linear coefficients affecting the equations. The noise vectors $\mathbf{q_k}$ and $\mathbf{r_k}$ are uncorrelated white Gaussian noises.

The objective is to obtain the states $\mathbf{x}$ from the observations $\mathbf{z}$, considering the C matrix as a unit matrix, we could rewrite the second equation as:

$$\mathbf{x_k} = \mathbf{z_k} - \mathbf{r_k}. \tag{2.15}$$

The issue of course is that we don't know the current noise $\mathbf{r_k}$: since it's white noise, and random by definition. Fortunately, Kalman had the idea that it's possible to estimate the actual state if you take in consideration both the current observation and the previous estimated state. Engineers use a "hat" symbol over a state to show that it is estimated: so $\hat{\mathbf{x}}_\mathbf{k}$ is the estimate of the actual state. Then we can express the estimate as a relation between the past estimate and the actual observation:

$$\hat{\mathbf{x}}_\mathbf{k} = \hat{\mathbf{x}}_{\mathbf{k-1}} + \mathbf{G_k}\left(\mathbf{z_k} - C\,\hat{\mathbf{x}}_{\mathbf{k-1}}\right), \tag{2.16}$$

where $\mathbf{G_k}$ is a gain matrix expressing the relation.

To help understanding how the kalman gain influences the calculation of the estimated states, let's think of what happens for two edge values of $\mathbf{G_k}$. For $\mathbf{G_k} = 0$, we have

$$\hat{\mathbf{x}}_\mathbf{k} = \hat{\mathbf{x}}_{\mathbf{k-1}} + 0\left(\mathbf{z_k} - C\,\hat{\mathbf{x}}_{\mathbf{k-1}}\right) = \hat{\mathbf{x}}_{\mathbf{k-1}}. \tag{2.17}$$

Therefore, when the gains are zero, observation has no effect, and the current state gets equal to the previous (or as seen ahead, it's not updated). For G_k and C as a unit matrix (C being unitary means that all sensors are active) we get,

$$\hat{\mathbf{x}}_\mathbf{k} = \hat{\mathbf{x}}_{\mathbf{k-1}} + \mathbf{I}\ \left(\mathbf{z_k} - \mathbf{I}\,\hat{\mathbf{x}}_{\mathbf{k-1}}\right) = \hat{\mathbf{x}}_{\mathbf{k-1}} + \mathbf{z_k} - \hat{\mathbf{x}}_{\mathbf{k-1}} = \mathbf{z_k}. \tag{2.18}$$

So, when the gains are unitary, the previous state is disregarded, and we get the actual state estimation only based on the actual measurement.

Remembering that the actual gain values will not be zero or unitary and will fall somewhere between these two edges.

2.5.0.1 Calculating the Kalman gains

Now we have a way to compute the state estimate $\mathbf{x_k}$ based on the past estimate $\mathbf{x_{k-1}}$, the current measurement $\mathbf{z_k}$, and the gain $\mathbf{G_k}$ in (2.16).

What is left is a way to compute the gains, and for that we will use the covariance values from the measurement noises. Recall that each observation is associated with a particular noise value (2.15).

Recall that we can't know the actual noise values of the measurements, but we usually do know the covariances of the noises: as an example, the accuracies of the sensors present in the system, tells us approximately how noisy their outputs are. And this information can easily be found on the sensors datasheets. It's very important to remark that this handle of covariances only work because the transformations utilizing it are linear, otherwise its important Gaussian properties would be lost, and with it the ability to represent the noise with it.

This covariance values are called $\mathbf{R} \geq 0$; there is no subscript on it because it does not depend on time, but is instead a property of the sensors. So we can calculate the gains $\mathbf{G_k}$ in terms of $\mathbf{R}$:

$$\mathbf{G_k} = \mathbf{P_{k-1}C^T}\left(\mathbf{CP_{k-1}C^T} + \mathbf{R}\right)^{-1}, \tag{2.19}$$

where $\mathbf{P_k}$ are the estimation-error covariances, the averages of the squared errors of our estimations, and is computed recursively,

$$\mathbf{P_k} = (\mathbf{I} - \mathbf{G_kC})\,\mathbf{P_{k-1}}. \tag{2.20}$$

Again for better understanding this process, it's helpful to think what these two formulas are doing before we continue.

Assume that the covariances $\mathbf{P_{k-1}}$ on the previous estimation-error are zero, and for simplicity $\mathbf{C}$ is a unit matrix. Then the current gains $\mathbf{G_k}$ will be $0(0 + \mathbf{R})^{-1} = 0$, and the next state estimates will be equal to the current state estimates. Which makes sense, because the algorithm shouldn't adjust the state estimate if the prediction was correct. On the other hand, say the estimation-error covariances are a unit matrix. The gains will then be $\mathbf{I}/(\mathbf{I} + \mathbf{R})$. Considering that $\mathbf{R}$ is zero, which means that there is little noise in our system, the gains will be one, and the new state estimates $\mathbf{x_k}$ will be highly influenced by the measurements $\mathbf{z_k}$. A problem is that as $\mathbf{R}$ increases in value, the gains can quickly

become small. Meaning that when the system is too noisy, a bad estimation will have less influence on the overall gains. Noise disrupts the ability to rectify bad estimations.

About (2.20), it's useful to think of what happens for boundary values of the gains: when $\mathbf{G_k} = 0$, we have $\mathbf{P_k} = \mathbf{P_{k-1}}$. So, just as with (2.16), zero gains means no updates to the estimation-error covariances. When on the other hand $\mathbf{G_k} = \mathbf{I}$, we have $\mathbf{P_k} = 0$. Therefore the maximum gain corresponds to zero estimation-error covariances, with the only the current measurements being used to update the current states.

2.5.0.2 Prediction and Update

It's important to notice that the matrices in the state equations, (2.13), seems to not be acquainted anywhere in the equations for the state estimates, (2.16). The reason for that is that we need more equations to estimate the state, not only (2.16).

More precisely the Kalman filter needs a internal model of the state equations, to produce a estimate. To do that we use the state equation which represents how the states will be, and we rewrite it with a little hat on the x to indicate an estimate:

$$\hat{\mathbf{x}}_\mathbf{k} = \mathbf{A}\,\hat{\mathbf{x}}_{\mathbf{k-1}} + \mathbf{B}\mathbf{u_k}. \tag{2.21}$$

We can see that both (2.21) and (2.16) represent the estimates of the states, based on different kinds of information, and both of them are necessary. The first represents a prediction about what the estimate should be, and the second represents an update to this prediction, based on an observation.

To conclude, we use the matrix $\mathbf{A}$ to calculate a prediction of the estimation-error covariance.

$$\mathbf{P_k} = \mathbf{A}\mathbf{P_{k-1}}\mathbf{A^T}. \tag{2.22}$$

We still need to account for the process noise. For that we use $\mathbf{Q} \geq 0$ to represent the covariance of the process noise $\mathbf{q_k}$, and simply add it to the prediction of the estimation-error covariance above. This results in,

$$\mathbf{P_k} = \mathbf{A}\mathbf{P_{k-1}}\mathbf{A^T} + \mathbf{Q}. \tag{2.23}$$

Notice that the matrix $\mathbf{A}$ is multiplied twice, that's because $\mathbf{P_k}$ is by definition a squared error, and so, it is scaled by the square of the coefficients associated with the state values.

Together (2.21) and (2.23) represent the prediction phase of the Kalman Filter. With the other equations representing the update phase.

2.5.0.3 Linear Kalman Filter equations

A problem is that $\hat{\mathbf{x}}_\mathbf{k}$ and $\mathbf{P_k}$ have two values at the same time: the *a priori* value (before the measurement at the current time has been used to update the estimate) and the *a posteriori* value (after the current measurement has been used to update the estimate). To solve these distinctions they are indicated by a sign. The negative sign $(-)$ indicates the *a priori* value, and the positive sign $(+)$ indicates the *a posteriori* value. This is a better notation compared to the use of k and $k-1$ present in a lot of the literature, and will eliminate some confusion, specially in the update part of the algorithm.

Thus to conclude, here is the set of equations for the linear Kalman Filter,

- **Model:**

$$\mathbf{x_k} = A\mathbf{x_{k-1}} + B\mathbf{u_k} + \mathbf{q_k}, \tag{2.24}$$

$$\mathbf{z_k} = C\mathbf{x_k} + \mathbf{r_k}, \tag{2.25}$$

- **Predict:**

$$\hat{\mathbf{x}}_\mathbf{k}(-) = \mathbf{A}\,\hat{\mathbf{x}}_{\mathbf{k-1}}(+) + \mathbf{Bu_k}, \tag{2.26}$$

$$\mathbf{P_k}(-) = \mathbf{AP_{k-1}}(+)\mathbf{A^T} + \mathbf{Q}. \tag{2.27}$$

- **Update:**

$$\mathbf{G_k} = \mathbf{P_k}(-)\mathbf{C^T}\left(\mathbf{CP_k}(-)\mathbf{C^T} + \mathbf{R}\right)^{-1}, \tag{2.28}$$

$$\hat{\mathbf{x}}_\mathbf{k}(+) = \hat{\mathbf{x}}_\mathbf{k}(-) + \mathbf{G_k}\left(\mathbf{z_k} - C\,\hat{\mathbf{x}}_\mathbf{k}(-)\right), \tag{2.29}$$

$$\mathbf{P_k}(+) = \left(\mathbf{I} - \mathbf{G_k C}\right)\mathbf{P_k}(-). \tag{2.30}$$

The idea is that the cycle predict $\longrightarrow$ update, can be repeated for as many time steps as necessary.

2.6 Techniques For Kalman Filtering in Non-linear Systems

Unfortunately most of the systems in practice are not linear, we must create ways to adapt the Kalman filter technique to accommodate them. Among the alternatives [18], two will be described in this paper. The first is resolving the nonlinear problem through linearization, which the Extended Kalman filter is based of. The second is using statistical samples to approximate the nonlinear transformation, which gives basis to the Unscented Kalman Filter, not utilized in this project.

2.6.1 Linearization

The process of linearizing a system by the Jacobian, means essentially getting a approximation of the dynamics of the nonlinear system through its partial derivatives:

$$F \approx \frac{\partial f}{\partial x}, \tag{2.31}$$

$$H \approx \frac{\partial h}{\partial x}. \tag{2.32}$$

These partial derivatives can also be calculated numerically, as

$$\frac{\partial g}{\partial x_i} \approx \frac{g(x + \Delta x_i) - g(x)}{\Delta x_i} \tag{2.33}$$

for $g = f,\ h$.

This sort of method is known as a finite difference approximation and is can be used for analysis of quasilinear (i.e., near-linear) estimation problems by approximating the nonlinear state-transition and sensor functions. Linear approximations by default are not perfect and its quality depends directly on the quasilinearity metric of the system. Methods to test for quasilinearization exist, but are out of the scope of this project, more can be seen in [18].

Other options available include, such as complex step differentiation, that provide a more stable numerical approximation of the partial derivative, or using symbolic operations to calculate the Jacobian beforehand using MATLAB. This the reasons why they were not utilized will be discussed further in chapter 3.

2.6.2 Extended Kalman Filter

Extended Kalman filtering (EKF) was used in the very first application of Kalman filtering: the space navigation problem for the Apollo missions to the moon and back. The approach has been successfully applied to many nonlinear problems ever since. Success depends on the problem's being *quasilinear*, meaning that unmodeled errors due to linear approximation are insignificant compared to the modeled errors due to dynamic uncertainty (process noise) and measurement noise. Methods to verify the *quasilinearity* of this problem are beyond the scope of this thesis, and are left for future works.

In extended Kalman filtering, linearization is used as a mean of propagating information about the probability distribution of the internal model estimations. You need the linearization because to guarantee that the distribution keeps its Gaussian properties when propagated, the state-transitions and measurements need to be linear. And the state estimate probability distribution needs to be Gaussian, because otherwise, the means and covariances alone no longer characterize the distribution, losing its ability to represent the errors utilizing it (both estimation and measurement, process ones). Therefore, *nonlinear filtering cannot be implemented exactly by using only the means and covariances* [18]. And so all practical nonlinear filtering methods need approximations of some sort.

Still this means that the linearity is only necessary for the calculations involving the distributions, and so the the full nonlinear model can be used in propagation of the estimate and in computing predicted sensor outputs.

In conclusion, it's possible to generalize the linear Kalman equations using the nonlinear measurement/state models, provided that you linearize them for propagating the estimation error distributions. As an example, the linear model,

$$\mathbf{x_k} = A\mathbf{x_{k-1}} + B\mathbf{u_k} + \mathbf{q_k}, \tag{2.34}$$

becomes

$$\mathbf{x_{k+1}} = \mathbf{f}(\mathbf{x_k}, \mathbf{u_k}) + \mathbf{q_k}, \tag{2.35}$$

where $\mathbf{A}$ is replaced by the nonlinear state-transition function $\mathbf{f}$. But for the prediction of the estimation-error covariance $\mathbf{A}$ is replaced by the Jacobian of the state-transition function, $\mathbf{F_k}$ (2.31) (it has a index because it changes over time). The same for the measurement functions in the update phase, with $\mathbf{H_k}$ (2.32) representing the Jacobian of the nonlinear sensor function $\mathbf{h}$.

Thus the EKF equations can be defined as follows,

- **Model:**

$$\mathbf{x_k} = \mathbf{f}(\mathbf{x_{k-1}}, \mathbf{u_k}) + \mathbf{q_k}, \tag{2.36}$$

$$\mathbf{z_k} = \mathbf{h}(\mathbf{x_k}) + \mathbf{r_k}, \tag{2.37}$$

- **Predict:**

$$\hat{\mathbf{x}}_\mathbf{k}(-) = \mathbf{f}(\hat{\mathbf{x}}_\mathbf{k-1}(+), \mathbf{u_k}), \tag{2.38}$$

$$\mathbf{P_k}(-) = \mathbf{F_{k-1}}\mathbf{P_{k-1}}(+)\mathbf{F_{k-1}^T} + \mathbf{Q}. \tag{2.39}$$

$$\mathbf{F_{k-1}} = \frac{\partial f(x,u)}{\partial x}\Big|_{\hat{x}_{k-1}(+),u_k} \tag{2.40}$$

- **Update:**

$$\mathbf{G_k} = \mathbf{P_k}(-)\mathbf{H_k^T}\left(\mathbf{H_k}\mathbf{P_k}(-)\mathbf{H_k^T} + \mathbf{R}\right)^{-1}, \tag{2.41}$$

$$\hat{\mathbf{x}}_\mathbf{k}(+) = \hat{\mathbf{x}}_\mathbf{k}(-) + \mathbf{G_k}\left(\mathbf{z_k} - \mathbf{h}(\hat{\mathbf{x}}_\mathbf{k}(-))\right), \tag{2.42}$$

$$\mathbf{P_k}(+) = \left(\mathbf{I} - \mathbf{G_k}\mathbf{H_k}\right)\mathbf{P_k}(-). \tag{2.43}$$

$$\mathbf{H_k} = \frac{\partial h(x)}{\partial x}\Big|_{\hat{x}_{k-1}(-)} \tag{2.44}$$

2.6.3 Sampling methods

It's a method that use statistical sampling theory to estimate the nonlinear transformations of means and covariances. Take a probability distribution with mean μ, its associated covariance P, and a nonlinear transformation $h(.)$, a sampling method is used to select samples μ_i (called sigma points), transform them using $h(.)$, and estimate the new nonlinear μ and P from the transformed samples $h(\mu_i)$. These methods are special because they consider nonlinear effects on the mean and covariance that are disregarded by the approximations made in the linearization methods. However, some of these methods require considerably more computation than conventional Extended Kalman filter.

2.6.4 Unscented Kalman Filter

The Unscented Kalman Filter, first proposed by Julier and Uhlmann [19], is an efficient estimation method based on sigma point samplings and the unscented transformation, which is responsible for propagating the means and variances (of a random variable, like the errors) through the nonlinear transformations.

The UKF focus on the approximation issues of the EKF. The state distribution is still represented by a variable with a covariance associated to it, but is now specified by a set of calculated sigma points. These sigma points capture the mean and covariance of the variable, are then propagated through the non-linear system, and a new empirical Gaussian distribution is then computed, capturing the next mean and covariance without the need of any linear approximations.

The great advantage of the Unscented Kalman Filter is its relative equal computational efficiency compared with the EKF method, but with a clear increase in performance when considering greater nonlinearities [20].

The next chapter presents the estimation algorithm chosen for this project and present the necessary steps to apply the estimation theory, in a simulated environment.

3 Simulation Project and Implementation

In this chapter the necessary steps will be presented to apply the estimation theory, already presented, in a simulated environment. To do this, it is necessary to create the models to simulate, choose and implement the estimation method, and obtain the values of $\mathbf{R}$ and $\mathbf{Q}$ to tune the filter, so in the next chapter, being able to allow the analysis of the estimation algorithm.

3.0.1 Why EKF

Sampling methods have "extended" Kalman filtering to higher nonlinear problems, while still working as well as the EKF for quasilinear problems. This brought many to question the utility of the EKF, despite it's incredible success in many nonlinear problems ever since it's inception.

Provided of good quasilinear models the EKF is a reliable and accurate estimator, and its proven track record, specially in fields like navigation, is a test for the strength of the method. One can also consider that nowadays, the calculation of the partial derivatives is not such a computational issue as it was in the past.

The main issue is that the EKF cannot be applied to any nonlinear problem with guaranteed success. It's always a risk that the system might be too nonlinear and the approximations will introduce too much error into the estimations. But the same can be said for the UKF, even without using approximations for the nonlinearities, there is no guarantee that the estimation will work for every case, and it's accuracy must be verified before accepting its results.

In the end, EKF was chosen in the sense that's a more straightforward solution, a natural extension of the linear case, and it's comparable easiness of implementation. And it stands, a number of nonlinear Kalman filtering methods are still being discovered and refined, but no perfect solution to the estimation of nonlinear problems currently exists [18].

3.1 Sailboat Filter

3.1.1 Discretization of the continuous model

State estimators work in discrete time, and as such, for the model described in chapter 1 to be useful, it needs to be discretized.

The first option is to utilize the model directly, and apply an approximation to the continuous derivative,

$$\frac{d\mathbf{x}}{dt} = \frac{x_{k+1} - x_k}{T_s} \tag{3.1}$$

where T_s is the sampling time. This process results in the discrete equations below,

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \\ v_{k+1} \\ w_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + T_s \left(v_k \cos\theta_k + p_1 a_{tw} \cos\psi_{tw} \right) \\ y_k + T_s \left(v_k \sin\theta_k + p_1 a_{tw} \sin\psi_{tw} \right) \\ \theta_k + T_s w_k \\ v_k + T_s \left(g_s \sin\delta_s - g_r p_{11} \sin\delta_r - p_2 v_k^2 \right) / p_9 \\ w_k + T_s \left(g_s \left(p_6 - p_7 \cos\delta_s \right) - g_r p_8 \cos\delta_r - p_3 w_k v_k \right) / p_{10} \end{bmatrix}. \tag{3.2}$$

Notice that the fifth row is not simplified like the discrete equations presented at [14], the reason for that is the source of instability described on that thesis was resolved. More details about it is written in chapter 3.

3.2 Covariance analysis

A covariance refers to the measure of how random variables will change when they are compared to each other. They are necessary for generating the noise for the nonlinear system, and as parameters in the Kalman Filter. In this case **R** generates the measurement noise and represents it inside the update phase, and **Q** generates the process noise and represents it inside the predict phase of the Kalman filter. The methods utilized here are normally done for tuning of the KF but because of the necessity generating values for simulation, they were also utilized to generate the gaussian noise.

3.2.1 Obtaining the R and Q matrices

For the compiling of the measurement noise values we decided to use the BerryGPS-IMU module as basis, figure 6, for reasons of availability in probable future implementations.

The BerryGPS-IMU uses the CAM-M8 from uBlox, which is an advanced high quality GPS module, and also includes an inertial measurement unit, or IMU.

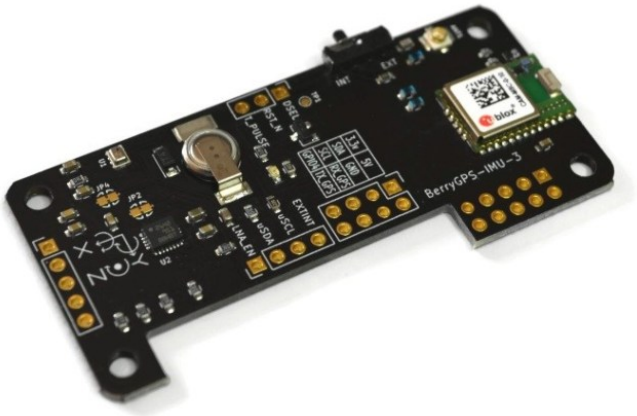


Figure 6 – Module BerryGPS-IMU, which provided the values for the measurement noises to generate more realistic simulations. It contains a GPS module, to the right and a IMU to the left

3.2.1.1 Position Accuracy

The horizontal position accuracy is < 2.5 m CEP (circular error probable) [21]. CEP values can be approximated to RMS (Root Mean Square) by multiplying it by a factor of 1.2 [22], so

$$\text{accuracy (RMS)} = 1.2 \times \text{accuracy (CEP)}$$
$$\text{accuracy (RMS)} = 1.2 \times 2.5 = 3 \text{ m}.$$

Since with most GPS or GPS/GLONASS systems, the mean errors (over a sufficiently long time interval) are close to zero, RMS may be considered essentially equivalent to one σ, or in other words the standard deviation. So,

$$\sigma_x = 3 \text{ m and}$$
$$\sigma_y = 3 \text{ m}.$$

However, this value was seen to be too high when utilized in simulations, as will be seen in the next chapter. So a method to improve this accuracy was researched and the solution was the utilization of a Differential GPS system.

3.2.1.2 Differential GPS

Differential GPS (DGPS) uses two GPS receivers noticing exactly the same satellites, with the installation located at a fixed location and a moving vessel, figure 7.

If they are receiving data from the same group of satellites, it can be assumed that they will take similar measurements and possibly introduce similar errors. And the known position of the base allows the constant comparison of the real position with the position given by the GPS system.

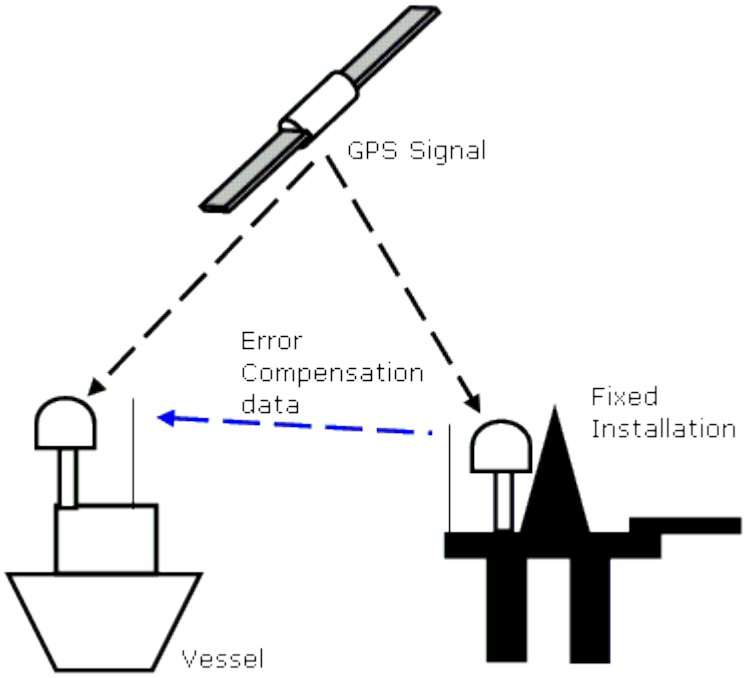


Figure 7 – Concept of Differential GPS for a moving vessel. Both vessel and installation have to receive the data from the same satellite [4]

The next step is to examine the values received comparing to the true known position and it's the difference between the two measurements that determines the error. The error identified is then used to produce corrections to the other receiver which is positioned in an unknown location.

Acquisition of data from the fixed and mobile stations can be done either in real time or through post-processing and can increase the position accuracy to centimeter levels.

This technique is nowadays widely available in most of the commercial GPS, including the CAM-M8. Meaning that the value reached earlier can be brought down depending on the implementation, being safe to assume that a accuracy close to values like 0.5 m should easily be achievable. Then the accuracy values from the position are

$$\sigma_x = 0.5 \text{ m and}$$
$$\sigma_y = 0.5 \text{ m}.$$

3.2.1.3 Velocity Accuracy

The velocity accuracy as informed by the CAM-M8 datasheet is 0.05 m/s at 30 m/s. For lower velocities such as a sailing boat in a 5 m/s true wind, the value may be different, but for that assumption, environment tests would be needed.

$$\sigma_v = 0.05 \text{ m/s}$$

3.2.1.4 Orientation and angular velocity accuracy

The IMU sensor available is the LSM9DS1 [23], including a gyroscope, accelerometer and a magnetometer.

- **Gyroscope**: the gyro angular zero-rate is ±30 dps and it was assumed the dynamic accuracy should fall into similar values. So the angular velocity accuracy was defined as approximately 0.52 rad/s.
- **Accelerometer**: the accelerometer offset accuracy is ±90 mg.
- **magnetometer**: the magnetometer has a Zero-gauss level of ±1 gauss.

As for the orientation measurement, as discussed by [24], the investigation of accuracy in orientation tracking of IMU devices, normally using the magnetometer, is not very common and rarely in great detail. So we utilized experimental results from other authors to have a reasonable approximate value as standard.

In Godwin et al. [25], is reported that the mean RMS error range was in between 1.9° and 3.5°, utilizing the oscillatory motion of a pendulum for the experiment. So the result in radians is between 0.0332 rad and 0.0611 rad. It was decided to take a value in between of 0.045 rad as the one sigma value for the orientation.

The values of the accuracies of the orientation and angular velocity are,

$$\sigma_\theta = 0.045 \text{ rad and}$$
$$\sigma_w = 0.52 \text{ rad/s.}$$

It's observed that the technical specification of commercial systems reported by vendors are presented with caveats and are poorly documented [24], so the assessments described here are at most good approximations and should not be taken as results of any experimentation conducted by the author.

Table 2 – Sensor accuracies results

symbol	value	unit	description
σ_x	0.5	m	x-axis position accuracy
σ_y	0.5	m	y-axis position accuracy
σ_θ	0.045	rad	orientation accuracy
σ_v	0.05	m/s	velocity accuracy
σ_w	0.52	rad/s	angular velocity accuracy

The compiled results from the sensor accuracies are in, Table 2. Thus the resulting covariance matrix R is as follows,

$$\mathbf{R} = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 & 0 \\ 0 & \sigma_y^2 & 0 & 0 & 0 \\ 0 & 0 & \sigma_\theta^2 & 0 & 0 \\ 0 & 0 & 0 & \sigma_v^2 & 0 \\ 0 & 0 & 0 & 0 & \sigma_w^2 \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} 0.5^2 & 0 & 0 & 0 & 0 \\ 0 & 0.5^2 & 0 & 0 & 0 \\ 0 & 0 & 0.045^2 & 0 & 0 \\ 0 & 0 & 0 & 0.05^2 & 0 \\ 0 & 0 & 0 & 0 & 0.52^2 \end{bmatrix} \tag{3.3}$$

3.2.1.5 process noise covariance matrix Q

The covariance matrix Q can be build based on assumptions about the dynamics of the system, it's generally constructed intuitively but things like unmodeled dynamics and parameter uncertainties are often modeled as process noise. According to [26], there are various ways to have reasonable results for the noise modelling, but experience and experiments are always needed.

For example, it's considered standard practice to pick a number, run simulations on data, and choose values based on that trial and error. It's also important, regarding the EKF, to not use covariance values that are too small, otherwise the filter may behave badly because of approximation errors originated from the linearization [27].

However one of the models presented at [26], it's said that a good rule of thumb is to set σ somewhere from $\frac{1}{2}\Delta s$ to Δs, where Δs is the maximum amount that the state

variable will change between sample periods. The results from the simulations performed with a 0.0005s sample time are

$$\Delta s_1 = 0.146,$$
$$\Delta s_2 = 0.148,$$
$$\Delta s_3 = 0.105,$$
$$\Delta s_4 = 0.037,$$
$$\Delta s_5 = 0.025.$$

So, the σ values for each state variable, considering $\sigma = \Delta s$ result as follows:

$$\sigma_1 = 0.146,$$
$$\sigma_2 = 0.148,$$
$$\sigma_3 = 0.105,$$
$$\sigma_4 = 0.037,$$
$$\sigma_5 = 0.025.$$

Giving process noise covariance matrix Q as follows,

$$\mathbf{Q} = \begin{bmatrix} 0.146^2 & 0 & 0 & 0 & 0 \\ 0 & 0.148^2 & 0 & 0 & 0 \\ 0 & 0 & 0.105^2 & 0 & 0 \\ 0 & 0 & 0 & 0.037^2 & 0 \\ 0 & 0 & 0 & 0 & 0.025^2 \end{bmatrix}.$$

It should be said that for simplicity we considered that there is no correlation between the state variables, which is definitely not the case, and should be adjusted in future works. Correlation would mean that there would be off-diagonal non-zero elements, and would considerably increase the complexity of the process noise covariance matrix.

3.3 Implementation

Initially the model proposed was developed in python, and was highly complex for development and testing of a EKF under the time available. That led to the necessity of a

simplified model, which allowed for a more flexible implementation and simulation under a different software, the one chosen being MATLAB and Simulink [28, 29].

3.4 Implementation in MATLAB

The software MATLAB is used for the setting up of the models and parameters, and evaluation of the results. One example is the setting of the sigma values for the process noise, based of results from simulations performed on the model in Simulink.

The nonlinear model is implemented as a MATLAB function, to allow for the evaluation on Simulink. This function describes the model equations and allows for the addiction of the white noise generated by the Simulink block. Including the limitations imposed on the value of the apparent wind angle, a conditional statement to wrap the angle in radians in between $[-\pi \ \ \pi]$.

Other implemented function is the function describing the approximated discrete version of the continuous model, described in the chapter 3.

3.4.1 Jacobian implementation

The EKF algorithm necessitates a value for the F_k at each new time step. This could be done symbolically to save computation effort by calculating the value symbolically once in MATLAB, and then just inputting the updated states estimates at each time step. But this can't be achieved because of the presence of logic statements, in the non-linear model, which otherwise would make the system unstable. One of such is the limitations imposed on the value of the apparent wind angle, as mentioned in section 3.4, that was implemented using *if* and *elseif* commands. Another example is the use of the *sign* function to implement the model of the mainsheet tightness or flexibility against the wind, as seen in chapter 2, which also don't allow for symbolical calculations in MATLAB.

The next solution is calculate the Jacobian values directly each time interval, using a numerical method such as finite difference or complex step differentiation.

So the MATLAB function, *lin*, was implemented. It contains the function *jacobianest* provided by [30], and can numerically approximate the value of the Jacobian based on the current state estimate provided. This function necessitates a function handle as input, this being the discrete model that it is utilized inside the Extended Kalman Filter.

Jacobianest utilizes the basic finite difference approximation to numerically calculate its partial derivatives, but other functions utilizing other approaches were considered. The method presented in [31] for example, utilizes complex step differentiation, which has many advantages in efficiency and accuracy over the finite difference approximations.

However it inadvertently introduces the definition of complex functions in functions like *atan2* [32], needed to calculate the apparent wind angle. The complex definition of *atan2* happens to not be supported by MATLAB [33], and in the end the easier solution was chosen for simplicity. Still this option is very important to consider in future works.

The functions implemented in MATLAB can be seen in listing 3.1.

Listing 3.1 – Important functions developed for this project.

```
function dx = nonlinear_simplified_wnoise(d)
function x = discretemodel(d)
function [F] = lin(d)
```

3.5 Implementation in Simulink

Even if the EKF itself is discrete, the main nonlinear model is simulated in continuous time. so Simulink was chosen for running the simulations because of the easiness of generating and incorporating the Gaussian white noise to the nonlinear continuous function. Besides the facility of debugging and visualizing the filter were very welcoming.

3.5.1 Gaussian white noise in continuous systems

The generation of gaussian white noise for the process and measurement noises were the main reason for the utilization of Simulink in this project, together with the difficulty of integrating it with the nonlinear model in continuous time.

As it stands this is not trivial in regular MATLAB code, and Simulink presents a block dedicated to it, called Band-Limited White Noise.

Figure 8 – Band-Limited White Noise block

As described in the Simulink documentation, the Band-Limited White Noise block, figure 8, generates normally distributed random numbers that are suitable for use in continuous or hybrid systems. It necessitates a very small sample time to accurately describe the effect of the white noise, called correlation time. It's recommended to use a correlation time t_c with a value much smaller than the fastest dynamics of the system.

About the noise power parameter, it requires a calculation be able to accurately reflect the intended covariance of the white noise. The documentation can be confusing on

this point so to simplify, with C_w as covariance of the white noise, N_p as noise power, the formulation can be described as,

$$C_w = \frac{N_p}{t_c},$$
$$N_p = C_w t_c.$$

Utilizing the values described for the process noise in chapter 2 for example, the process noise can be generated, figure 9 , and added in continuous form as a signal in Simulink. The same process is also done for the measurement noise.

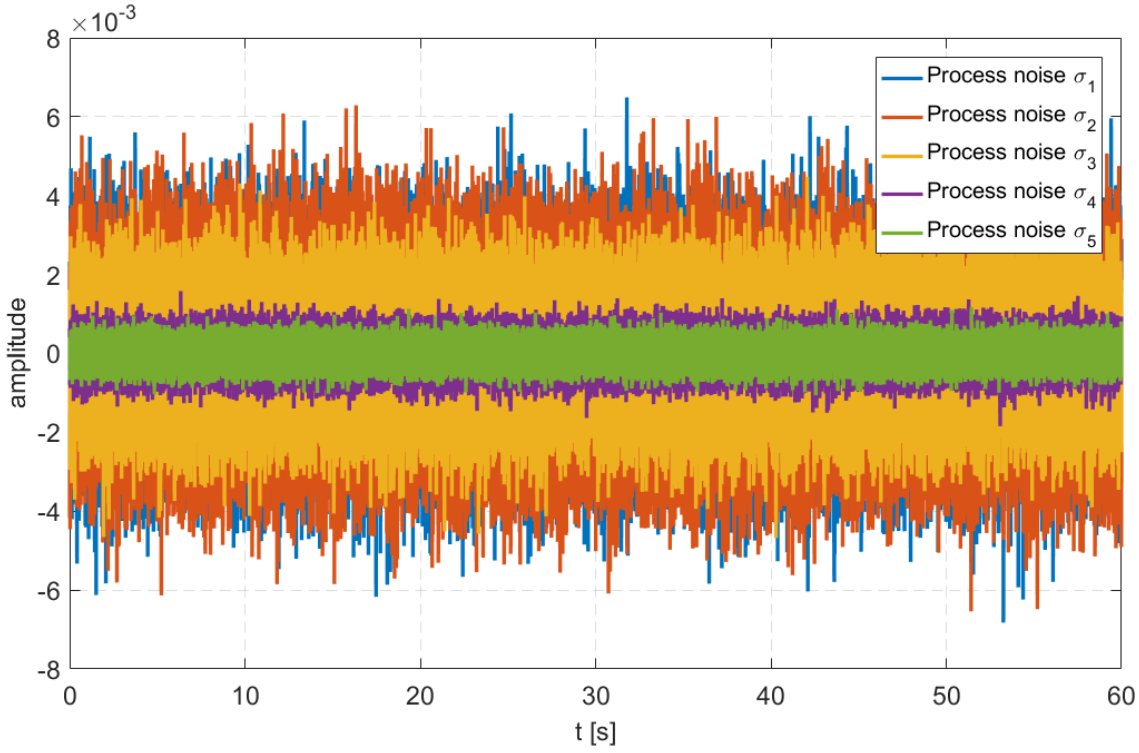


Figure 9 – Result of the generation of the process noise in Simulink for the five states of the system, for both positions, x and y, orientation θ, velocity v and angular velocity w, in order from 1 to 5 respectively

3.5.2 Description of the simulation

When the model function is given the current state of the sailboat (Position, velocity and angular velocity), current speed and angle of true wind and the control variables, the derivative of each sailboat state is returned. The value is then integrated and the resulting states are routed back as inputs, utilizing the solver ODE45 in continuous time steps, figure 10. The block Interpreted MATLAB function is utilized to run the function.

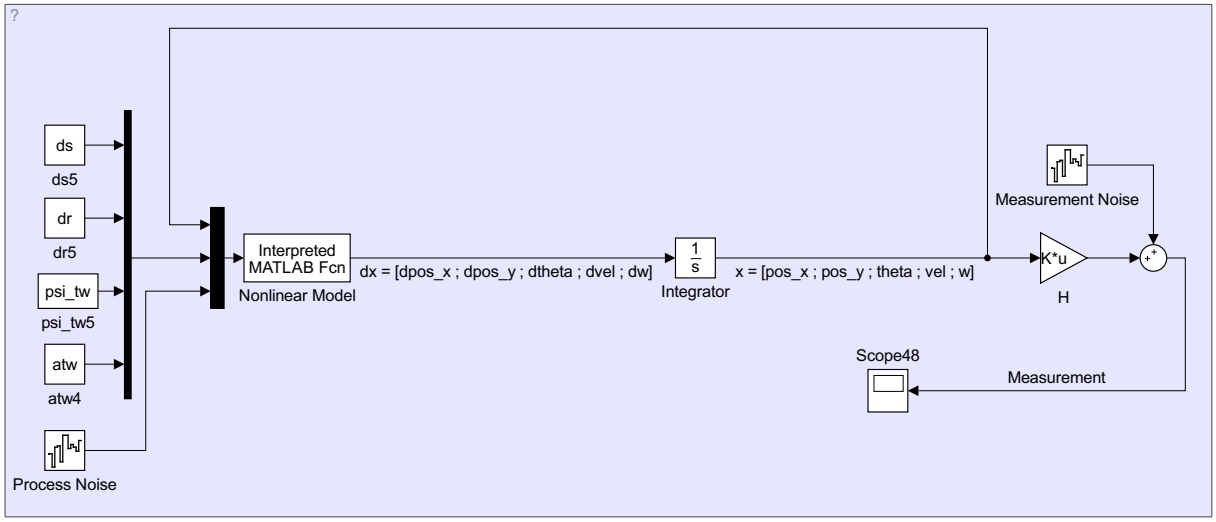


Figure 10 – Simulation of the nonlinear model in continuous time utilizing Simulink, with the added process and measurement noises

As for the simulation of the EKF, the algorithm described in chapter 2, was implemented using signals and basic blocks, figure 11. The exceptions being the Interpreted MATLAB function blocks utilized in both the linearization function and the discrete function model for the prediction part. Repeating that the notation used as $k|-$ means the value at time k before the measurements were accounted for. Two unit delays were utilized to break the loop between iterations.

3.6 Wind filter implementation

The scope of this project omits the implementation of the wind filter. The sailboat filter implemented was of greater relevance and the techniques implemented can easily be extended for the creation of the wind filter if desired.

Other reason is the lack of a wind model available, or wind data to filter, and so the estimator for obtaining the speed and angle of the true wind could not be tested. Some implementations such as a random walk for the generation of the wind signals can be implemented in the future, though the use of actual meteorological data may be preferable if easily available.

3.7 Sensor failure implementation

To test the efficiency of the EKF in simulations, a sensor failure fault needed to be implemented. The fault chosen was the stuck fault, for its prevalence in control systems, acting directly on the systems measurements [34].

Stuck fault is the fault where sensor gets stuck at its position and mistakenly gives

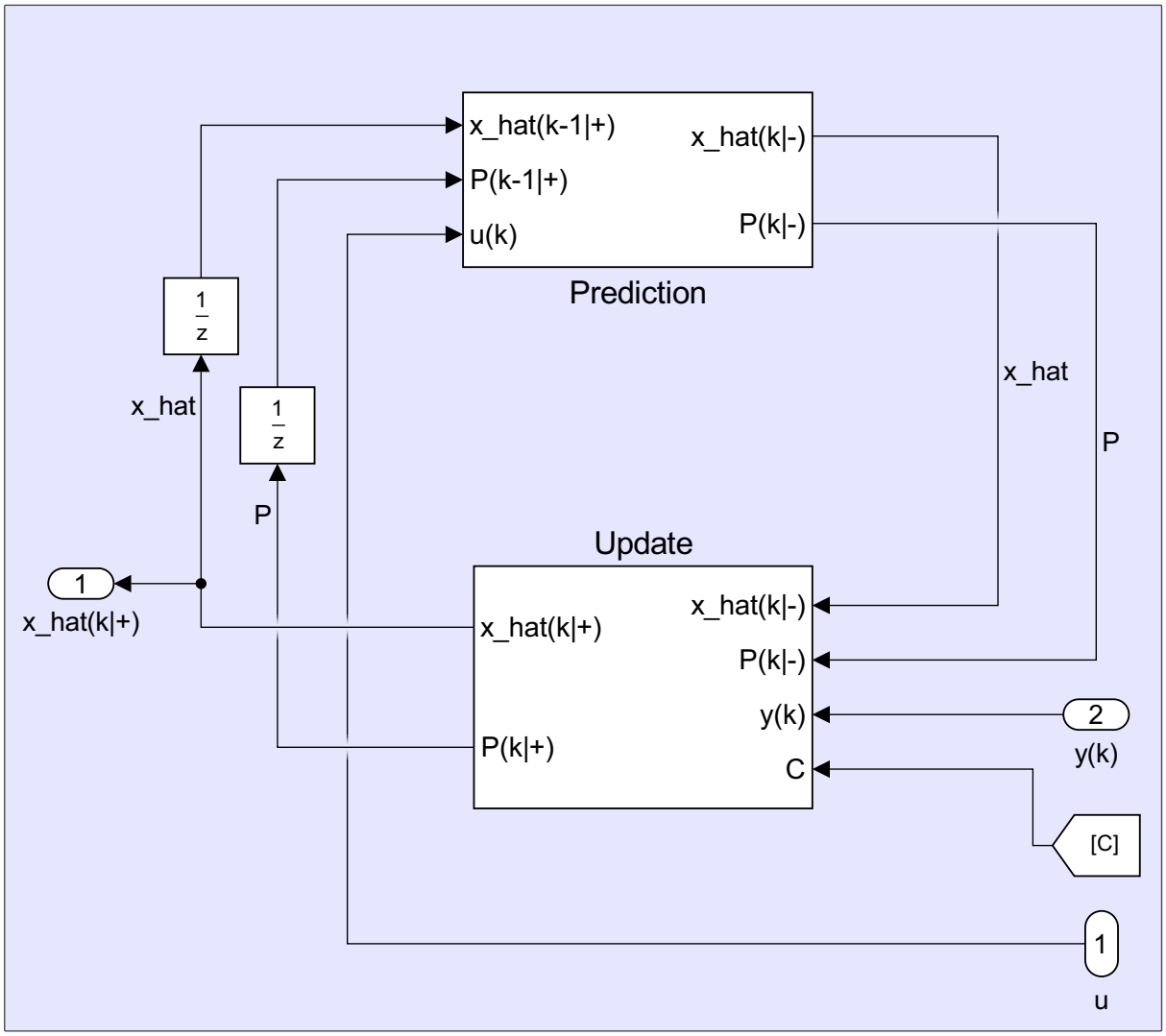


Figure 11 – Implementation of EKF algorithm in Simulink, presenting both Prediction and Update phases, with the arrows showing the flow the estimation and the error covariance matrix. The negative sign $(-)$ indicates the *a priori* value, and the positive sign $(+)$ indicates the *a posteriori* value. k and $k-1$ was utilized to demonstrate the values from the current and the past iterations of the filter. The sensor function matrix C was considered as an input for the implementation of the sensor faulure correction

a constant output value. This output is the last value of the sensor at the time instant when it got stuck and failed. It is implemented on Simulink, using the subsystem 'Sensor Failure', figure 12.

The subsystem is also capable of generating stuck faults for an limited amount of time, called intermittent stuck faults. This is meaningful because permanent and intermittent stuck faults have very different effects on a system. For example a system that might diverge if presented with a permanent stuck fault, might be stable in a intermittent scenario.

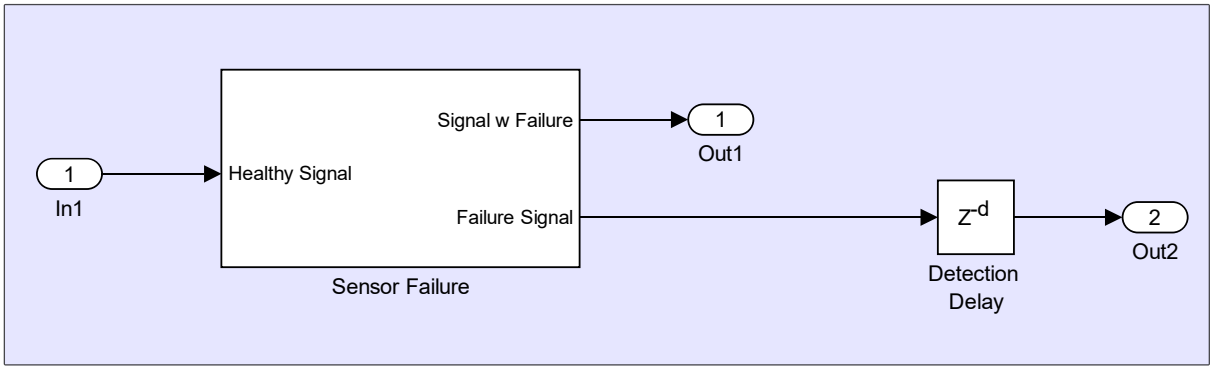


Figure 12 – Implementation of the sensor failure and failure signal in Simulink. The signal that triggers the failure is utilized as the signal that the failure detection would generate, incremented by the a delay block to model said detection

3.7.1 Fault detection

A subproduct of the implementation of the sensor failure, was the generation of a failure signal. This signal is the same one that triggers the fault on the subsystem. Of the existence of this signal arose the idea of having a fault detection system. This idea was based on the article presented by Seema Singh [35], which deals with stuck faults in flight control systems.

The fault detection is modelled as the failure signal coupled with a simple detection delay, figure 12, that simulates the time an actual detection algorithm might take.

3.7.2 Fault consideration inside the filter

Having a fault detection system, it is natural to try to implement a system to correct it. So a correction system is made utilizing the sensor function matrix to improve the EKF.

This system was not fully implemented because of time constraints, but the basic idea could be implemented in a way that permitted it to be simulated. It consists in utilizing the failure detection signal from the fault detection system to update the sensor function matrix while the EKF is still running, as seen in figure 11. This allows the filter to disregard the values of the faulty measurements, and consequently behave as if the sensor was disabled.

The effects of this correction will be explored in chapter 4.

3.8 Angular velocity instability

In [14] it's said that the angular velocity causes instability because of the damping term, present in its state-transition equation. Expanding on the reasons given it might be

that nonlinear internal models, can easily lead to computational errors, in special case, in the Jacobian calculation.

If it presents round-off errors, they can compound and generate asymmetries or non positive semi-definite estimation covariance matrices, making the estimation worse and if big enough unstable [18].

What we may be seeing is the kalman filter being able to correct these small roundoff errors for the simplified discrete model utilized in [14], but for the more complex model it falls apart.

Any probability distribution utilizing the covariance matrix must be symmetric. The symmetry comes from its definition. The covariance tells you how two variables are related and therefore if x variable is related to y variable, y will be related to x in the same way.

In the first implementation the discrete model described chapter 2, it was indeed detected a small asymmetry, and was corrected by utilizing the expression $P = 0.5\,(P + P')$ in the prediction step. And this correction was enough to rectify the angular velocity instability.

The next chapter presents the simulation goals and results of the implementations performed on this chapter.

4 Simulation and Observed Results

The simulations produced provide data from a nonlinear continuous model with process noise to test the developed EKF, through a linear measurement function with measurement noise. However, it's important to remind that empirical data is necessary to validate its performance, and these results should keep that in consideration.

4.1 Goals

The goals are to evaluate the efficiency of the EKF, considering the levels of noise utilized and the sensors available, under different scenarios. The intent is simulating edge cases to demonstrate proper kalman filter behavior, even under stressful situations.

- The first scenario describes the default case (all the sensors active) trajectory in the plane. It's serves to illustrate the impact of the EKF on estimating the position, relative to simply using the plain measurements.
- The second scenario explores how the EKF works with and without the use of Differential GPS, or to be more precise, with two different accuracies for the positions.
- The third scenario is important to describe how the EKF would function disregarding some sensors altogether. This important in cases where it's necessary to shut down some sensors for some reason, energy concerns being one of them.
- The fourth scenario analyses what the introduction of sensor faults in the position and orientation measurements generates on the behavior of the EKF. This sensor faults are the stuck faults described in chapter 3. It also explores the improvements made by the introduction of the fault correction, in both intermittent and permanent stuck fault cases.

All the scenarios consist in the boat acting without the usage of a controller, meaning that the values for the angle of the rudder and sail are kept constant in $\delta_s = 30°$ and $\delta_r = -5°$ respectively. The same for the true wind variables, with $psi_{tw} = 0°$ and $a_{tw} = 5\text{m/s}$.

The sample times for measurements and the functioning of the EKF are $T_s = 0.1\text{s}$. The values for the $\mathbf{R}$ and $\mathbf{Q}$ matrices are defined as in chapter 2. White noise is utilized in all scenarios with both process and measurement noises also defined by $\mathbf{R}$ and $\mathbf{Q}$, with it's implementation described in chapter 3. The exception being the second scenario, where the $\mathbf{R}$ values for the two system positions are increased.

Initial conditions for the states and the estimation error matrix are $x(0) = \hat{x}(0) = 0$ and $P(0) = I$ (identity) respectively. The initial value for P is considered a good first guess, and usually used to avoid errors by bad modeling, and considers that you have no no initial information of the system. That's justified as a unitary initial estimation error covariance, and means that the EKF will put more emphasis on the measurements for it's first estimations.

4.2 Scenario 1

The first scenario represents the operation of the EKF in the default situation, where all sensors and DGPS are available and working as intended.

The figure 13 shows that the EKF manages to follow the simulated real position of the sailboat with a relative good precision, through the simulated time of 120s. This is further noted by observing the first 10 seconds of the above figure, where we can have a closer look into it, and permit the inclusion of the measured position without cluttering the image with information, shown in the figure 14.

The EKF clearly manages a estimate with significant improvement over the measured position, even considering it's comparably chaotic behavior.

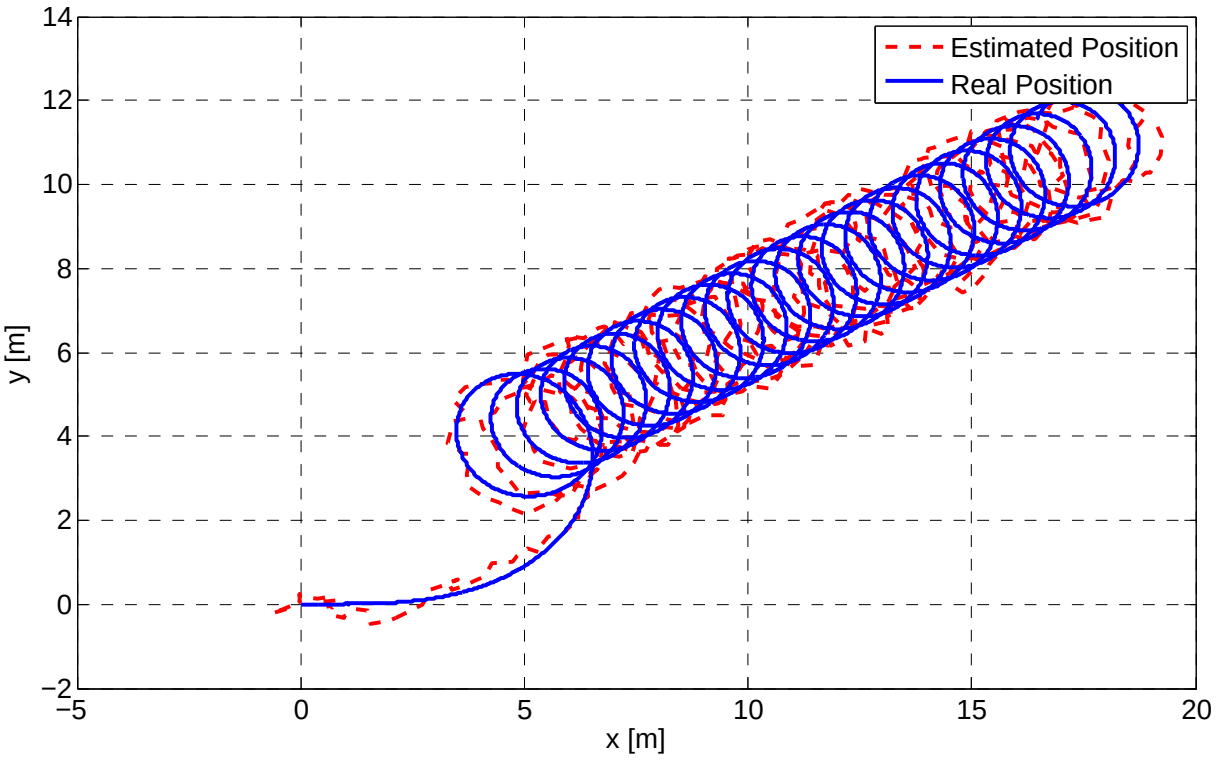


Figure 13 – Positions x and y for the default scenario, real vs estimated. Input and true wind variables are constant, with simulation time of 120s

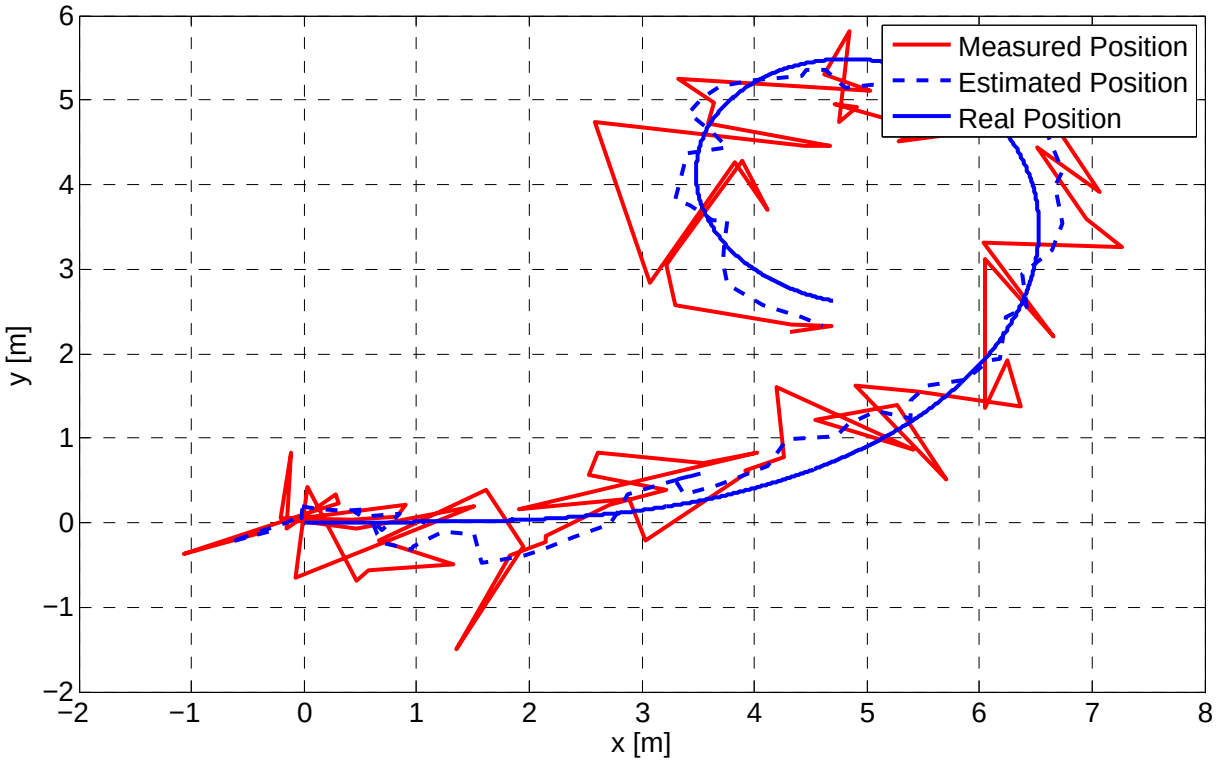


Figure 14 – Positions x and y for the default scenario, measured vs real vs estimated. Input and true wind variables are constant, with simulation time of 10s

4.2.1 State-space disturbance

This subsection includes simulation results from the first scenario with disturbances in the state-space. These can be originated from unmodeled behaviors, such as waves and other sudden fluctuations. The disturbance was modelled as a step of $Pd = [-3\ 3\ 1\ 1\ 0.25]$ being added to each state variable at $t = 27$s. The simulation can be observed at figure 15.The simulation time of 45s was chosen to better observe the results.

The EKF manages to return to normal values in a short time, even considering big disturbance values for the positions of -3 and 3m. The positions and velocity converged back with a settling time close to 0.5s, and the angular velocity took a bit longer with 1.5s. This is overall a very good result and show that the EKF is resilient against state-space disturbances.

These perturbations were also considered for the other scenarios and with good simulation results obtained in the states estimation, but not shown in the document.

4.3 Scenario 2

The second scenario also represents the default case but without the Differential GPS system working, meaning that the measurement error covariance of the measured positions are increased to 3m compared to the usual 0.5m.

The figure 16 shows that the EKF still manages to follow the simulated real position of the sailboat in some capacity, through the simulated time of 120s. Again looking at first 10 seconds of the above figure, figure 17, it's clear that the measurement of the position provided by the GPS alone is not very useful. The behavior seen can be explained because the slow sailboat velocity only causes small differences in position between the samples acquired, causing the high measurement error to be very apparent in the figure.

Still the EKF still manages to converge, even when fed with extremely inaccurate position measurements.

4.4 Scenario 3

The third scenario consists of simulations of the EKF considering four different cases of sensor shutdowns and the default cases for comparison, with and without the DGPS. The sensor shutdowns are modeled by the utilization of different values in the sensor function, as following,

$$\mathbf{C} = \begin{bmatrix} H_x & 0 & 0 & 0 & 0 \\ 0 & H_y & 0 & 0 & 0 \\ 0 & 0 & H_\theta & 0 & 0 \\ 0 & 0 & 0 & H_v & 0 \\ 0 & 0 & 0 & 0 & H_w \end{bmatrix} \text{ with,} \tag{4.1}$$

(i) Default cases, where $diag\,(C) = 1$, representing that the system has all the modelled sensors working;

(ii) $H_x = 0$ and $H_y = 0$ representing system without the position measurements;

(iii) $H_\theta = 0$ representing the system without the yaw measurement;

(iv) $H_v = 0$ representing the system without the velocity measurement;

(v) $H_w = 0$ representing the system without the angular velocity measurement;

The estimation errors showcased in this scenario give a very detailed look into the efficiency of the EKF. As discussed in the first two scenarios, both default cases result in a stable estimation, with a considerable reduction in the measurement noise, figure 18 and 19.

The second case, figure 20, the EKF without the GPS position measurements isn't able to stay stable for much time. After 120 seconds, a drift starts to appear, meaning

that the internal EKF model alone cant keep predicting the positions permanently, even with the other measurements available, without the updates from the measurements.

In the third case, figure 21, the orientation estimate keeps surprisingly stable values, which is interesting since it's one of the measurements with lower noises. This means that the model and the angular velocity measurement are greatly contributing to the estimation.

From the simulations without the velocity measurement in the fourth case, figure 22, the estimations manage stable results. This occurrence signify that the velocity measurements are being utilized as updates in very small amounts, this being a result of the Kalman gain calculated by the EKF. The high velocity estimation error, present at the beginning of the simulation, is caused because the initial state uncertainty. That uncertainty would be corrected by the measurement, but since it's not available, the EKF needs some iterations for the internal model apply the correction, based on the other measurements.

The fifth case demonstrates that the EKF can function perfectly without the angular velocity measurement, figure 23. This is also expected, being that its measurement noise is the highest of all the measurements, and compared to it's small values, is really high. And so much of it's estimate was already being introduced by the predictions.

4.5 Scenario 4

As mentioned in chapter 3, stuck faults are a very important topic when deliberating about estimators, and as we can see by these simulations, that's also the case for the EKF. The Stuck fault happens at $t = 20$s for all the cases and has a duration of 5 seconds for the intermittent cases.

The effect of the the permanent stuck fault on the estimation errors is visibly high for both position and orientation faults, figure 24 and 25. The EKF can't discern by itself that the value being fed to it by the measurements is wrong, and will try update the estimations utilizing it depending on the information contained in the $\mathbf{R}$ matrix. This causes the value of the estimation to keep following the measurement even in the fault scenarios. Still, the EKF manages to keep good estimations of the other 3 states for the GPS fault, and only causes some drift in the positions and velocity for the orientation fault.

The intermittent faults causes equal problems as the permanent faults, figure 26 and 27, while the faults are occurring, but it shows that the EKF manages to return to the correct estimation values very quickly after the sensors come to normal. Again this is expected given that the estimations in both cases are keep incorporating the measurement

no matter what, so if the measurement comes back to it's expected value, the estimates will follow.

Still this is a relative good result, in the sense that if the stuck faults are not causing instability on the EKF for the time periods tested. Still more simulations with different time values are needed to provide insight of the stability of the EKF to this kind of fault, both in the permanent and intermittent cases.

4.5.1 Sensor function correction

Regarding the sensor function correction, the fault signal detection being modeled as a 5 seconds delay, the simulations showed impressive results.

The influence of the the permanent stuck fault on the estimation errors is very reduced for both position and orientation faults, if compared to its measurements, figure 28 and 29. The EKF is able to correct its internal sensor function and so ignore the values given by the faulty measurements. The EKF will then act with behavior similar to the second and third cases of the scenario 3. This means that some drift in the positions will begin to happen after some time, but it's a much improved position estimate regardless.

The simulations results for the corrected EKF in the intermittent cases are even better, with complete correction of the stuck fault in both cases, figure 30 and 31.

4.6 Overall Comparison/Analysis

An very important observation from these simulations is the effect of the measurement on the system is always significant for the EKF, no matter the noise values. This is best seen comparing the results from the second scenario to the second case from the third scenario, and permanent GPS stuck fault without correction, in the fourth scenario. In the case missing the DGPS measurements, the sensor function is still complete, and so the bad measurements are included in the calculations of the EKF, but they are stable. In case where the GPS measurements are disabled, the sensor function is still incomplete, so no position measurements are taken, and because the internal EKF model can't predict alone the real positions and the estimate begins to drift. And in the permanent GPS stuck fault case, the sensor function is complete, but this time the measurements included are faulted ones, and the EKF can't correct it, so the drift in the estimates also occur. The conclusion is, no matter the value of the covariance of the measurement error, if the mean is zero, the EKF can utilize it to a better effect, than if the measurement is disabled or faulted.

It's also interesting to notice that the positions estimation drift in the second case of the second scenario, have a correlation to the fact that the process noise, introduced in

the main nonlinear state equations, affect the ability of the internal EKF model to predict the estimates. This indicates that the worse the internal model is, the worse the estimates in those cases will turn out.

The drift caused by the uncorrected orientation stuck fault may give hints that the influence of the orientation estimate on the EKF is very considerable. This is is logically consistent with the rationale that the orientation changes contribute nonlinearity in the position equations, and the higher the nonlinearities, the worse the EKF performs. This leads to the conclusion that bigger sailboats may benefit of different estimation techniques that don't involve linearization, as the UKF for example.

Still the EKF implemented in this chapter show very good results, managing to decrease the noise being fed to it, resilience to sensor faults and failures. This allows for a great basis for the further development of the boat's control system, and eventual physical implementation for the Microtransat Challenge.

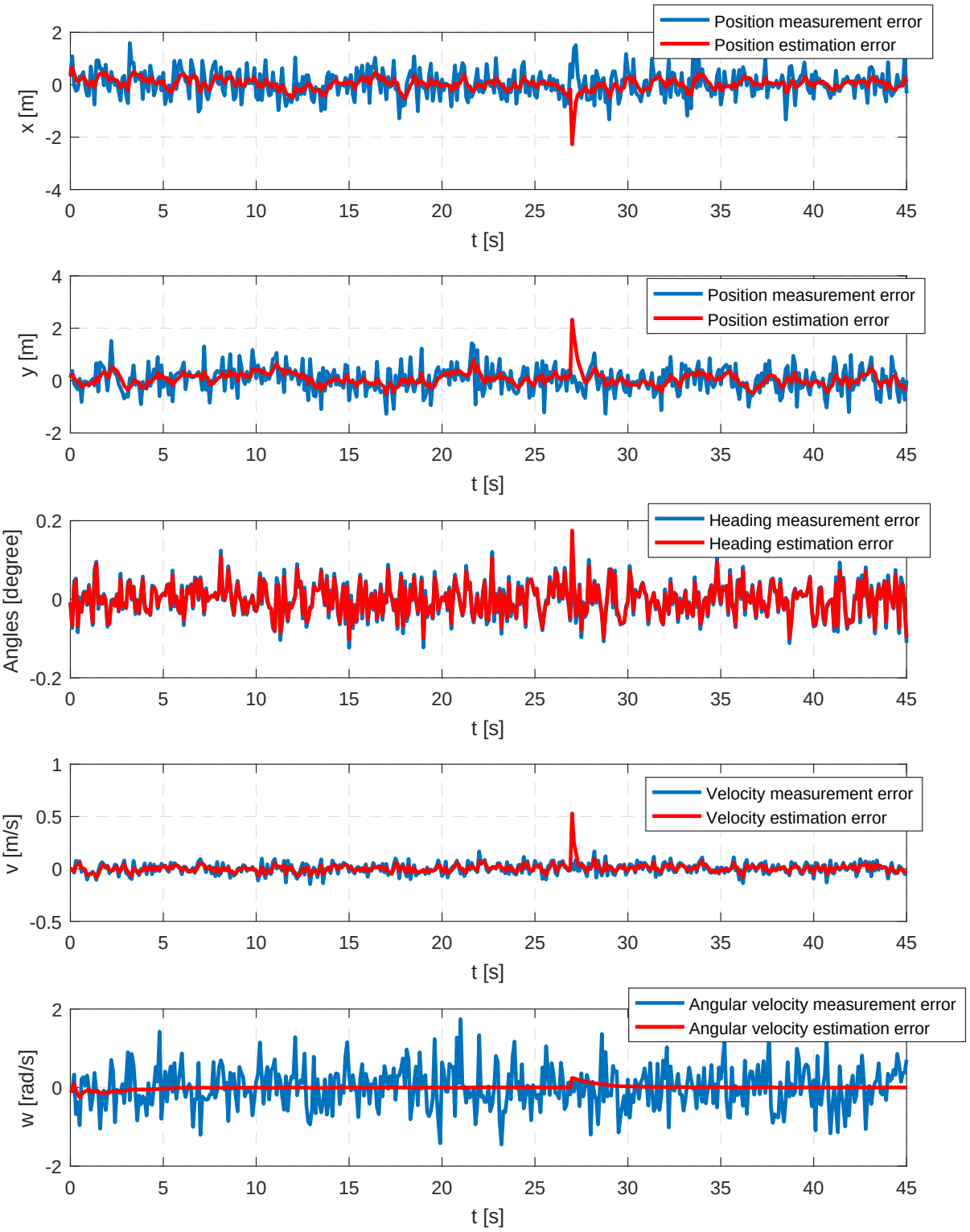


Figure 15 – Estimation errors on the default case with state-space disturbances, measured vs estimated. The disturbance occurs at $t = 27$s, as a step in all states. Input and true wind variables are constant, with simulation time of 45s

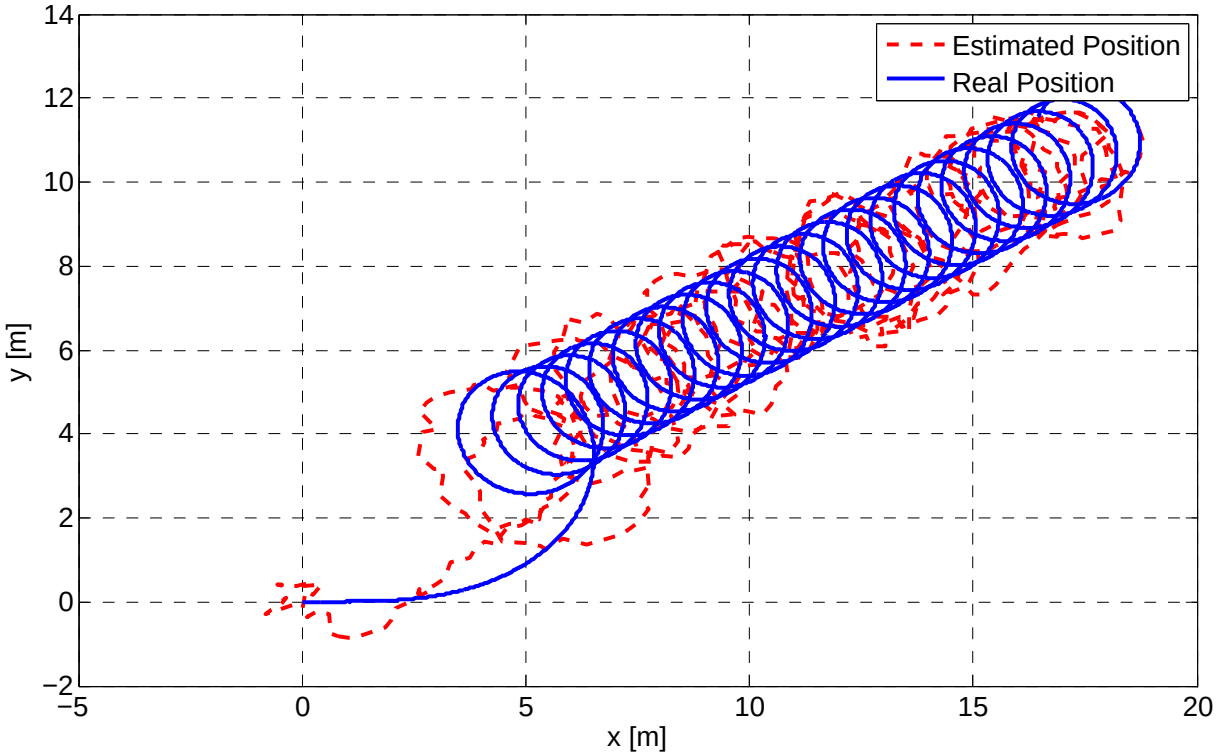


Figure 16 – Positions x and y for the default case without DGPS, real vs estimated. Input and true wind variables are constant, with simulation time of 120s

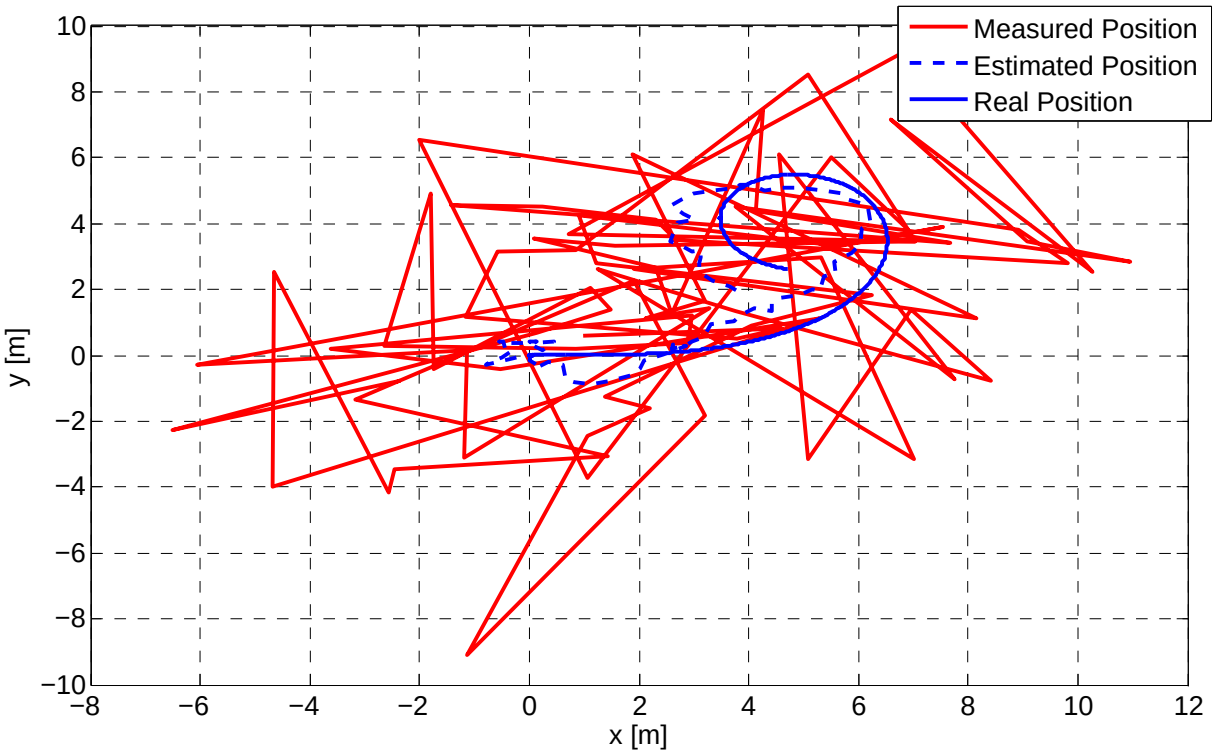


Figure 17 – Positions x and y for the default case without DGPS, measured vs real vs estimated. Input and true wind variables are constant, with simulation time of 10s

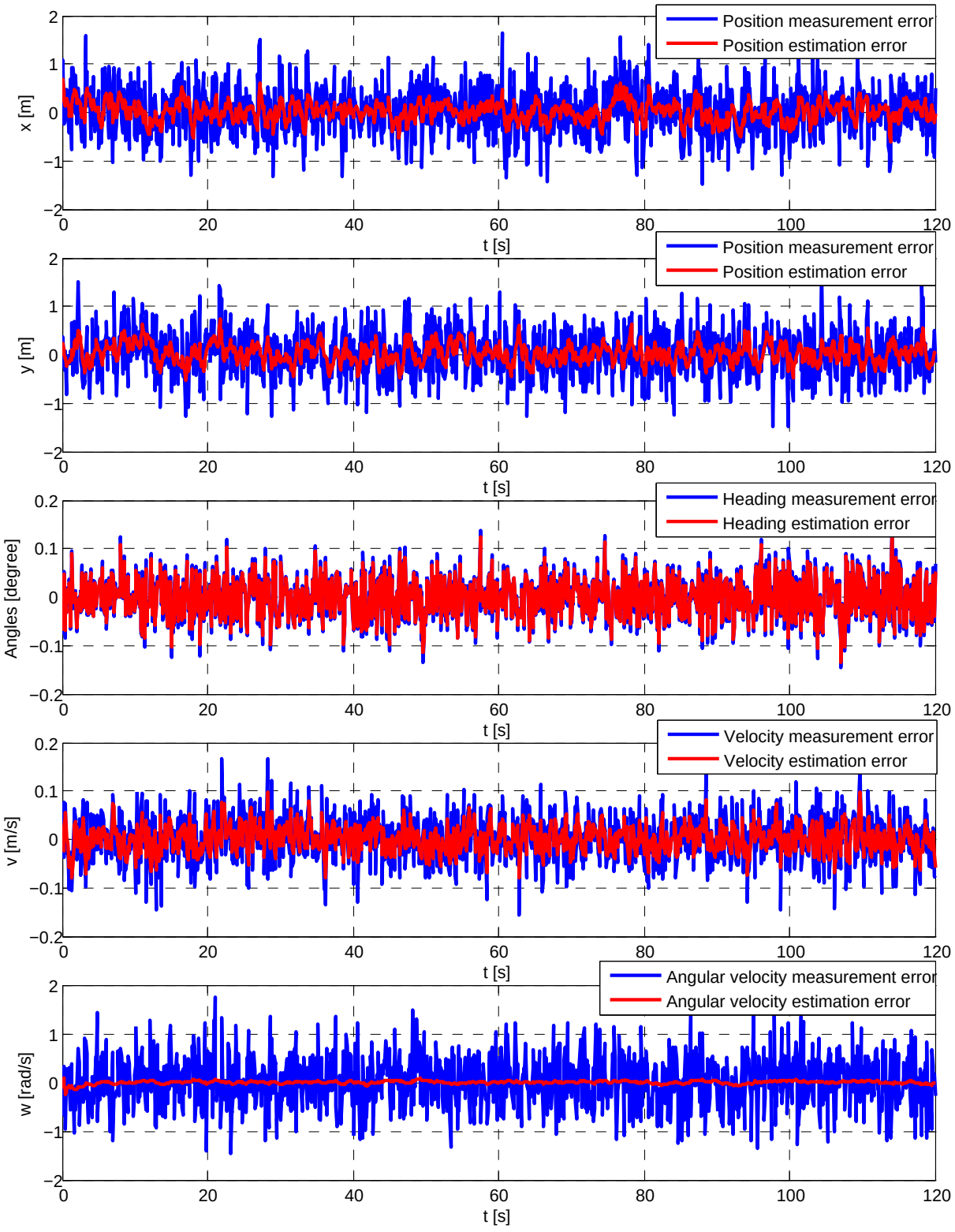


Figure 18 – Estimation errors on the default case with DGPS, measured vs estimated. Input and true wind variables are constant

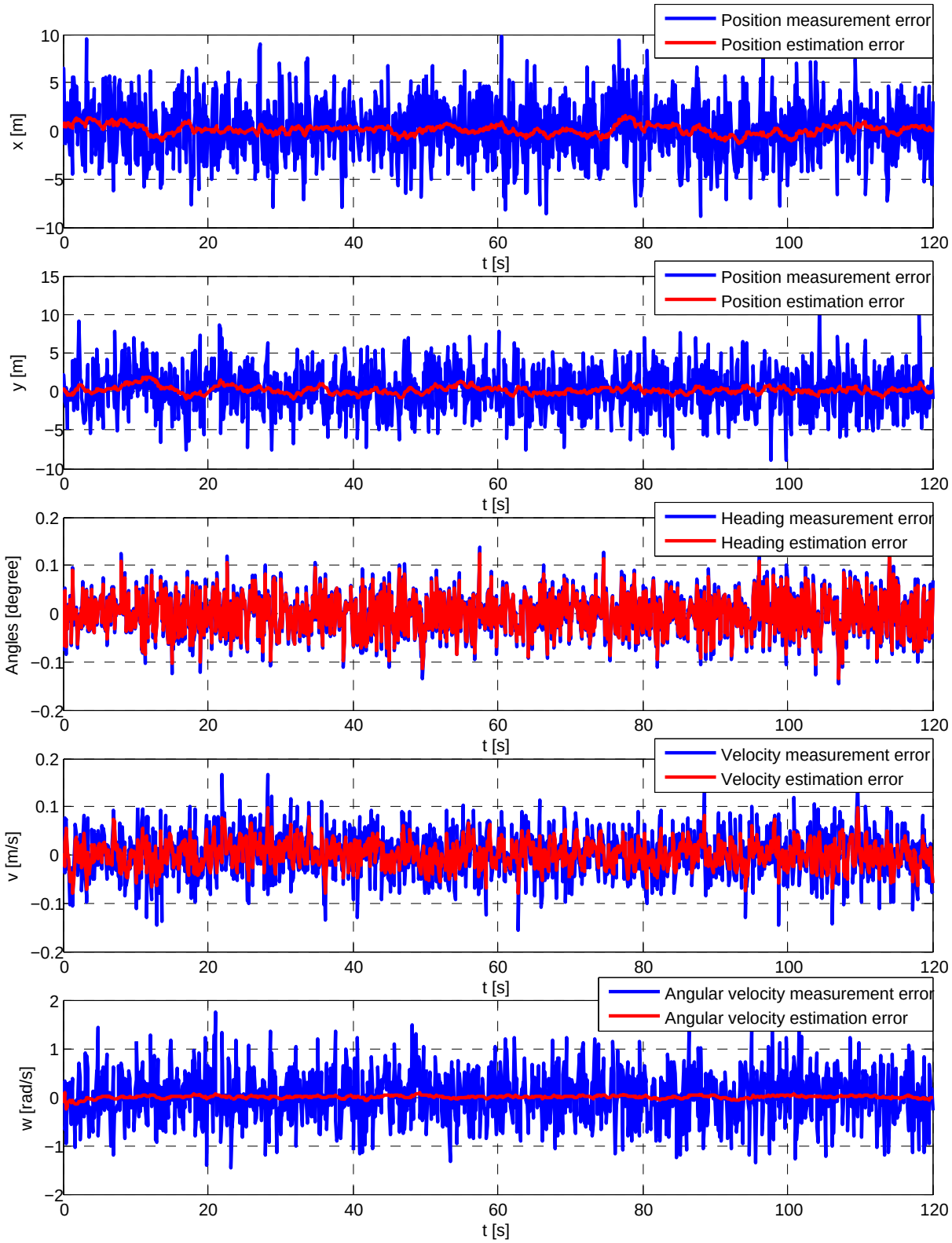


Figure 19 – Estimation errors on the default scenario without DGPS, measured vs estimated. Input and true wind variables are constant

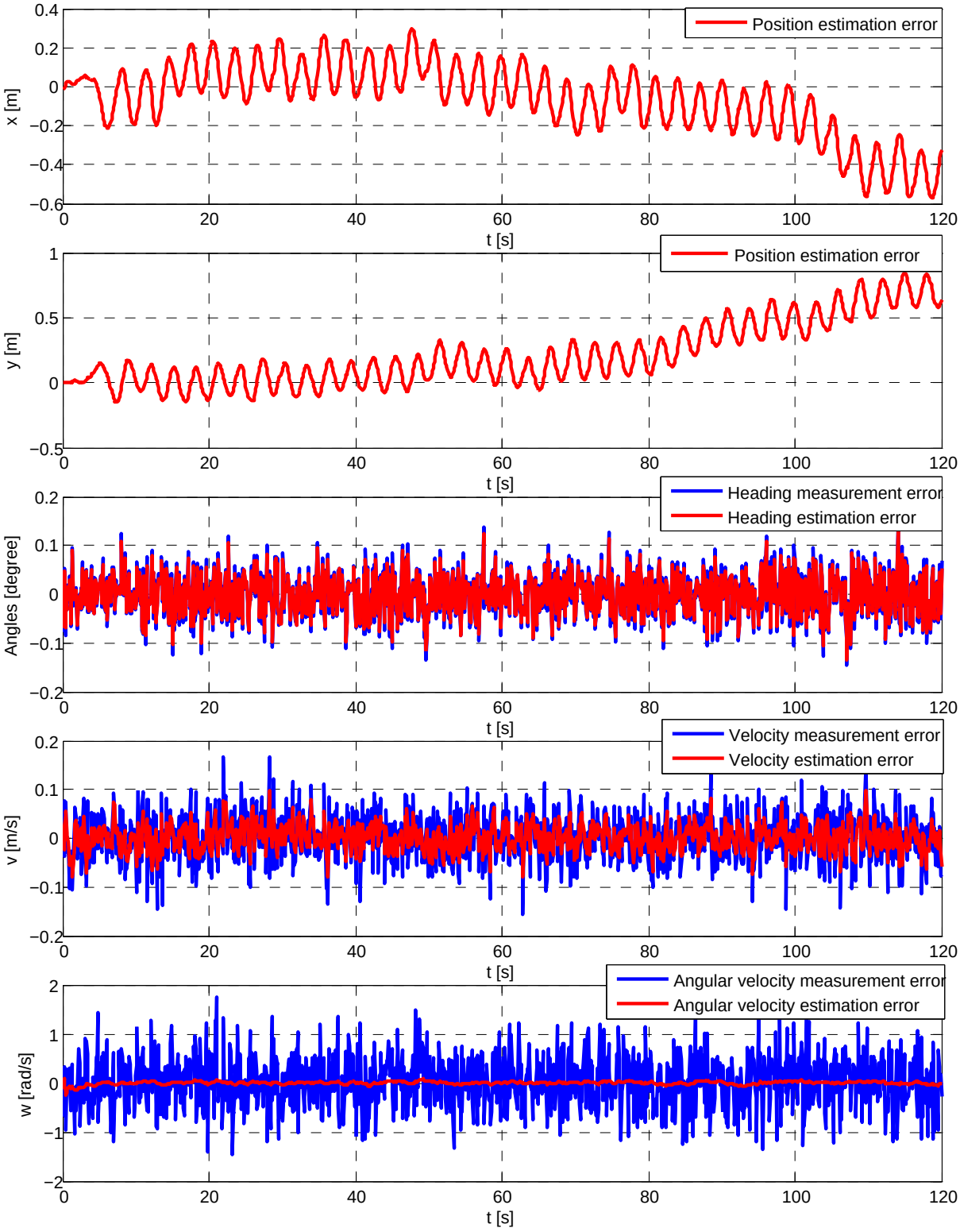


Figure 20 – Estimation errors on the second case, measured vs estimated. Positions measurement errors are not defined because their sensor functions are zero, $H_x = 0$ and $H_y = 0$. Input and true wind variables are constant

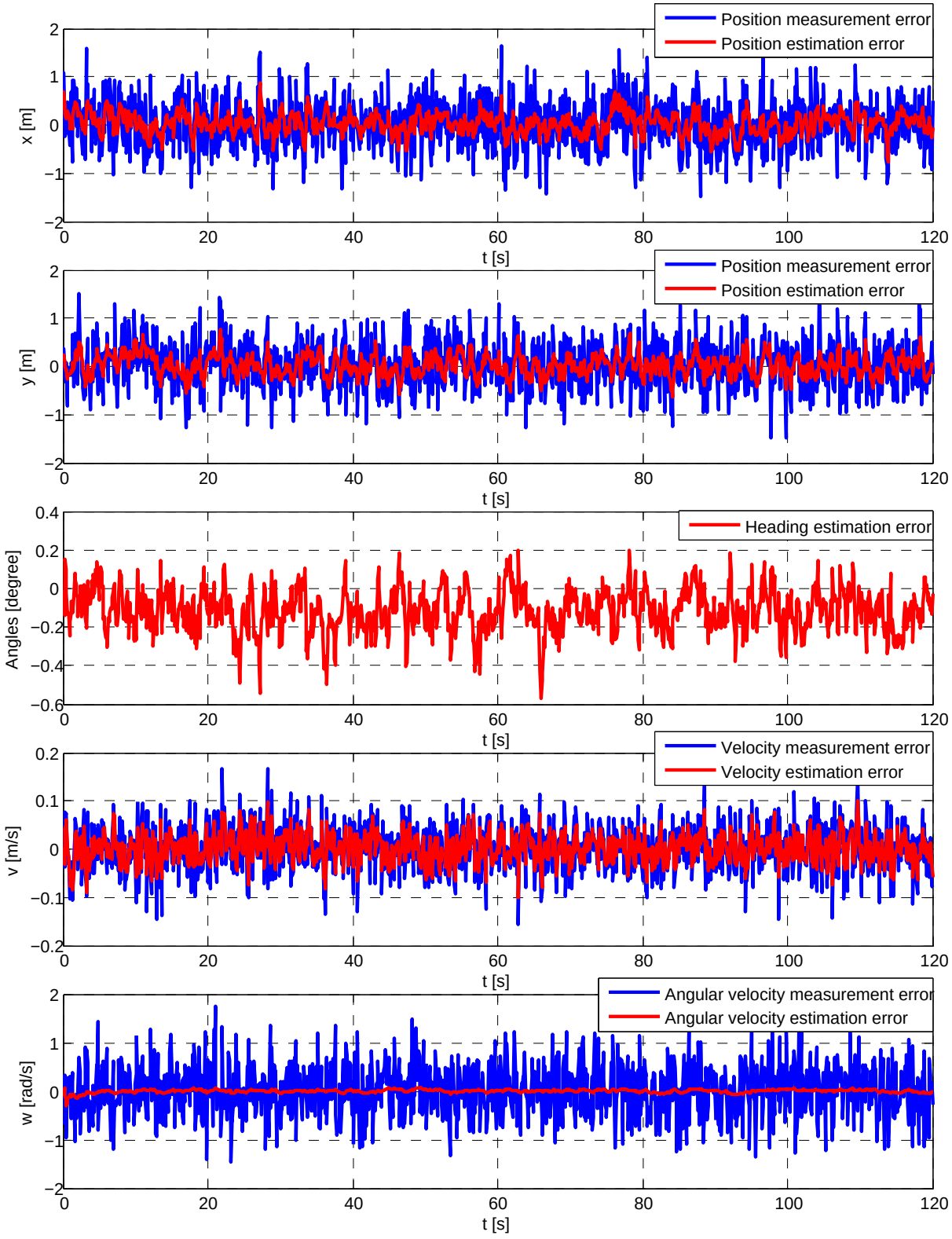


Figure 21 – Estimation errors on the third case, measured vs estimated. Heading measurement error is not defined because its sensor function is zero, $H_\theta = 0$. Input and true wind variables are constant

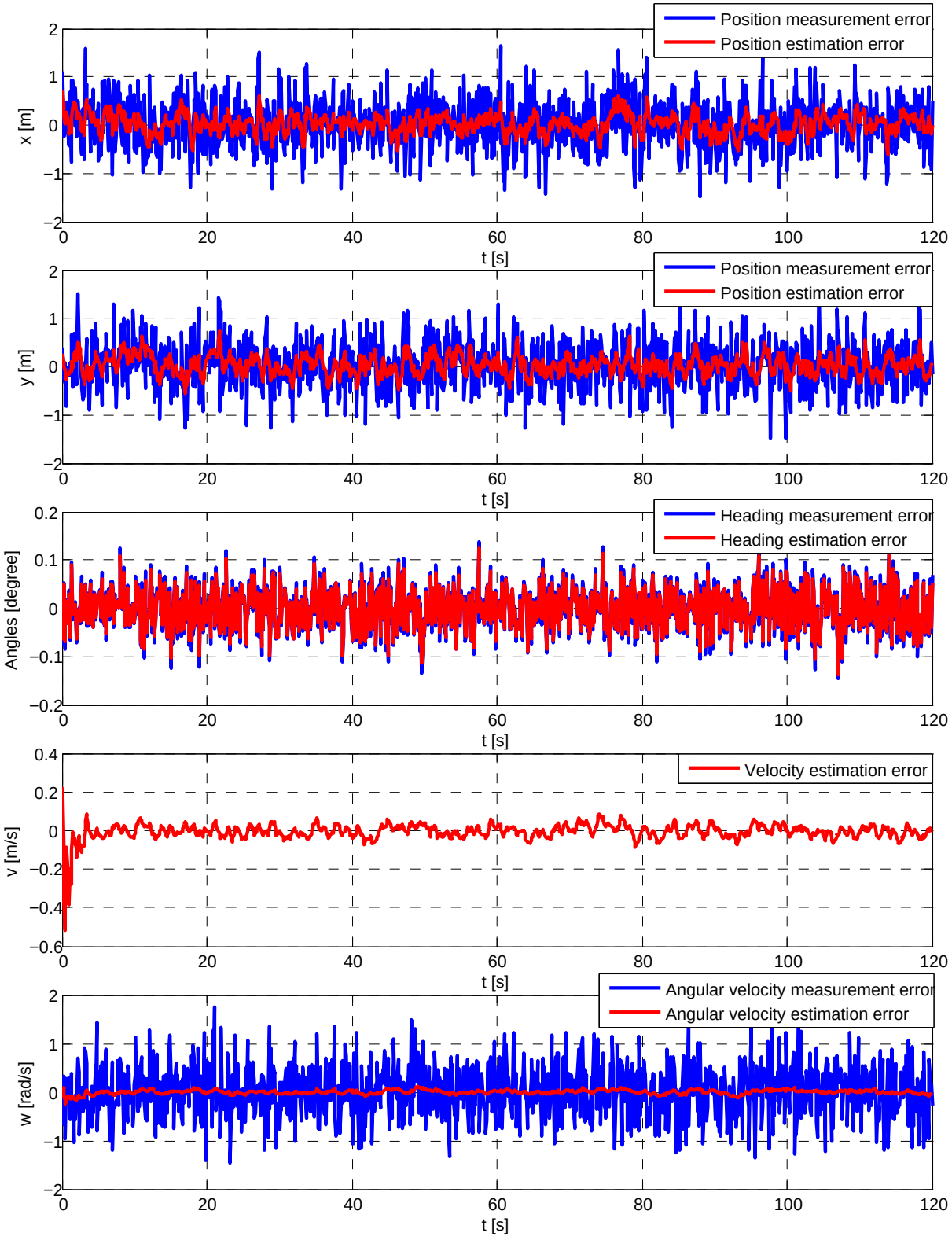


Figure 22 – Estimation errors on the fourth case, measured vs estimated. Velocity measurement error is not defined because its sensor function is zero, $H_v = 0$. Input and true wind variables are constant

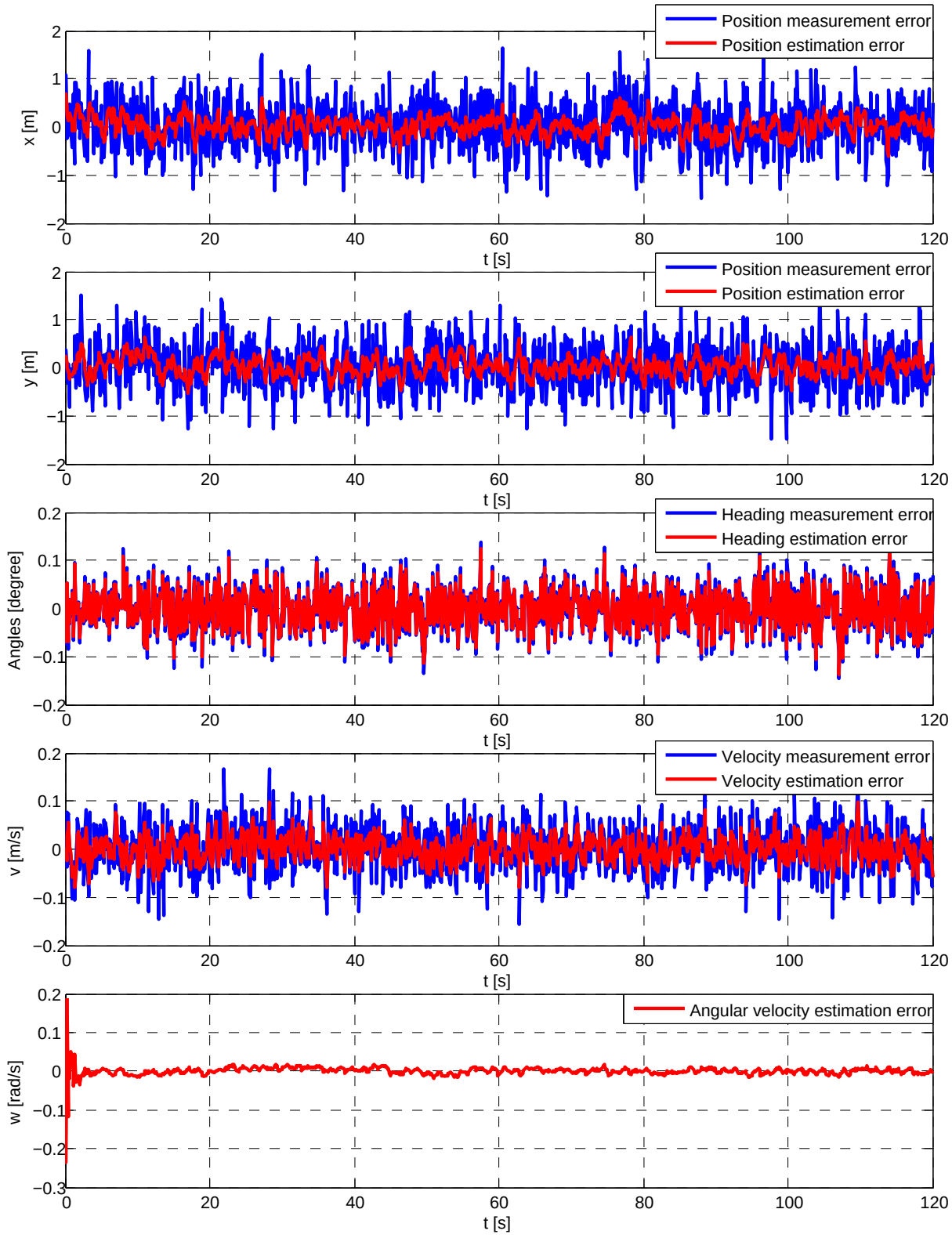


Figure 23 – Estimation errors on the fifth case, measured vs estimated. Angular velocity measurement error is not defined because its sensor function is zero, $H_w = 0$. Input and true wind variables are constant

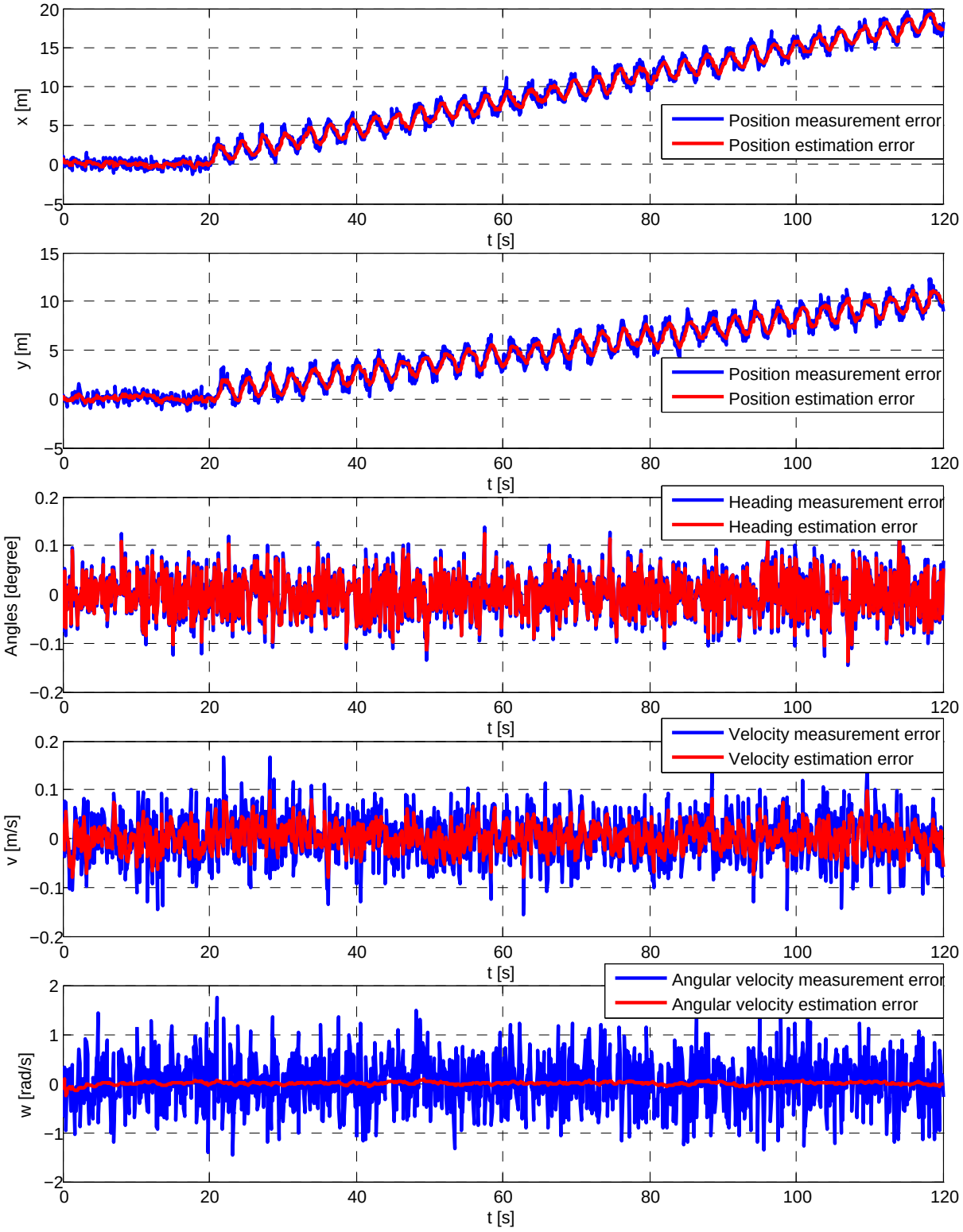


Figure 24 – Estimation errors on the case of permanent GPS stuck fault, measured vs estimated. Input and true wind variables are constant

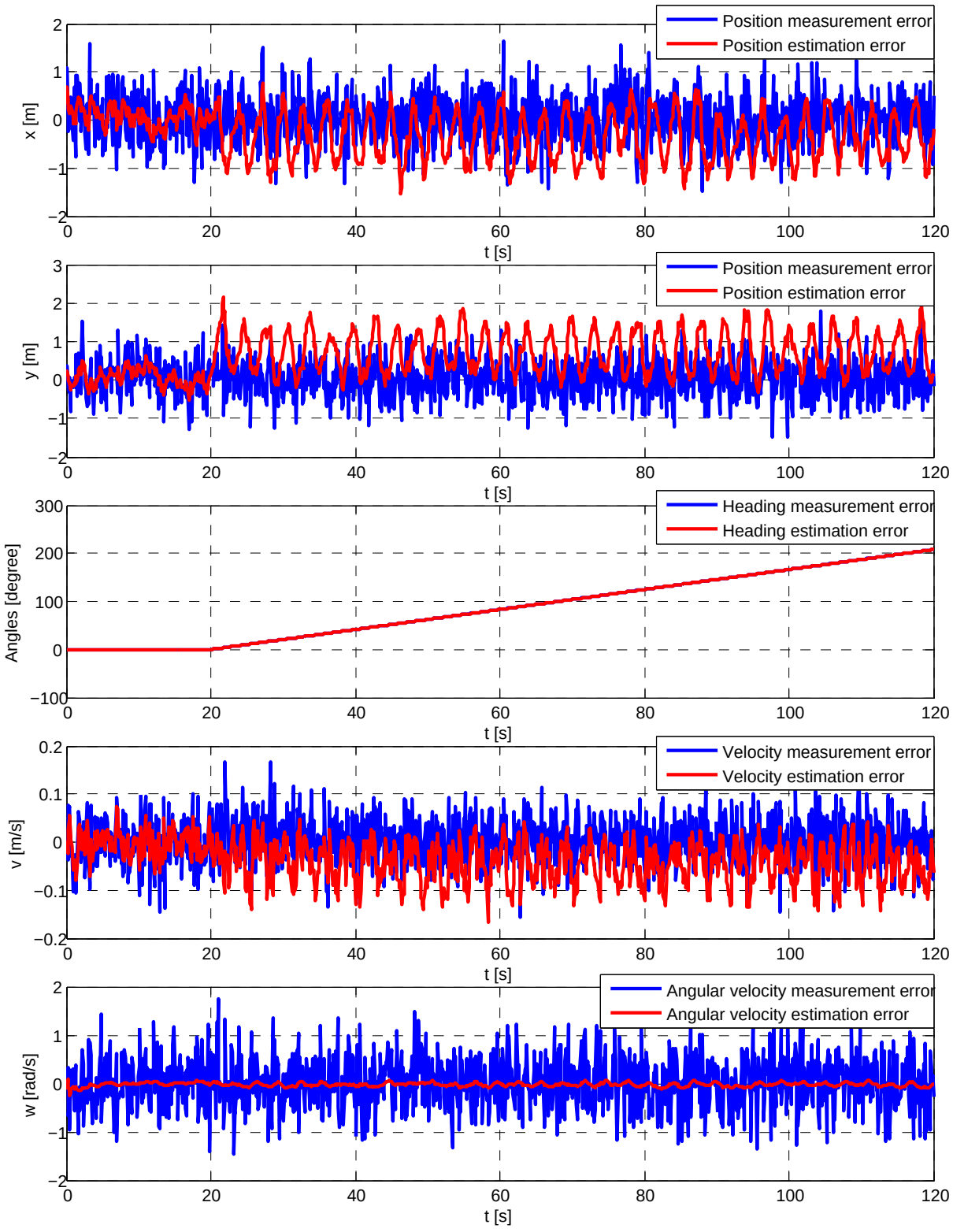


Figure 25 – Estimation errors on the case of permanent orientation stuck fault, measured vs estimated. Input and true wind variables are constant

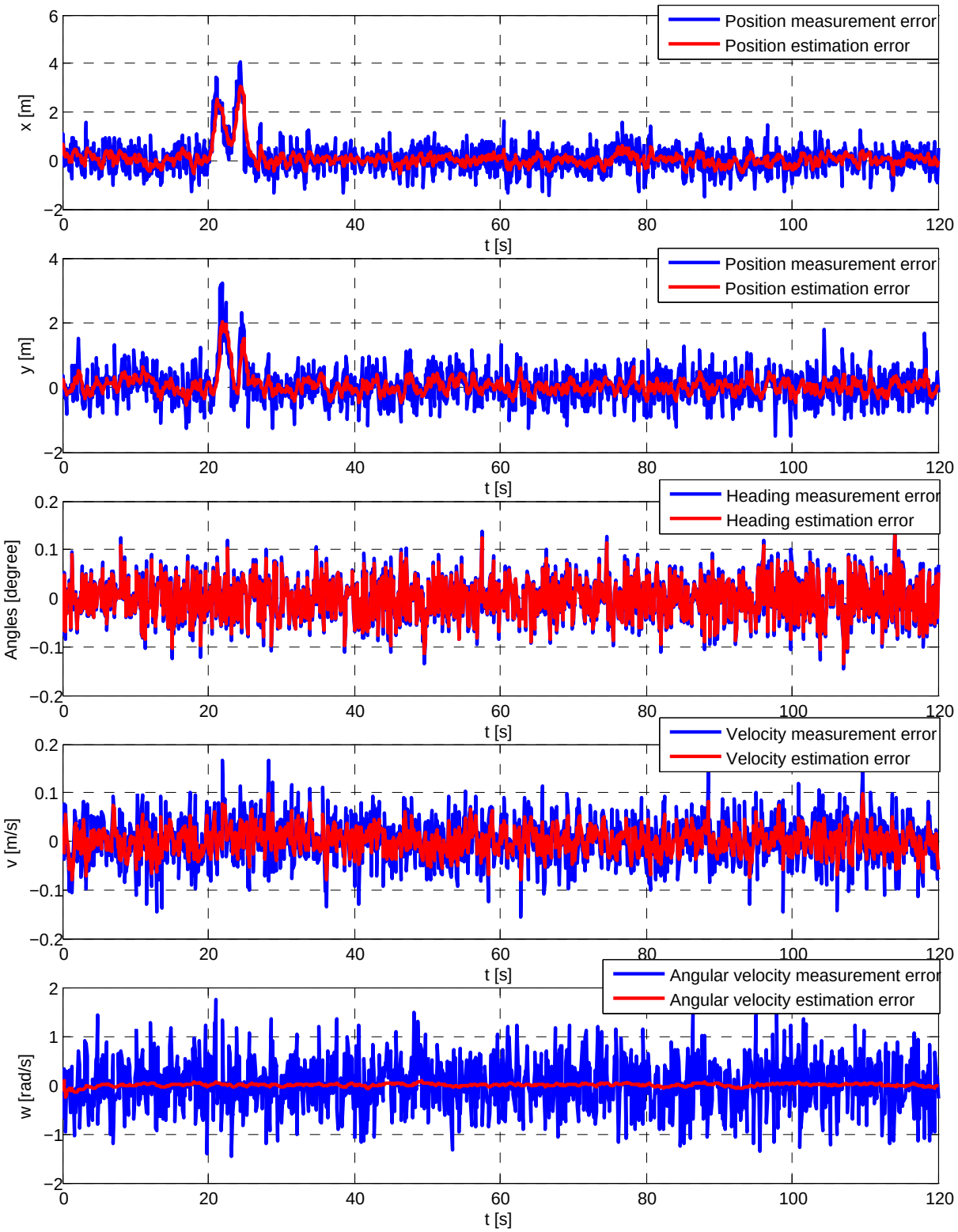


Figure 26 – Estimation errors on the case of intermittent GPS stuck fault, measured vs estimated. Input and true wind variables are constant

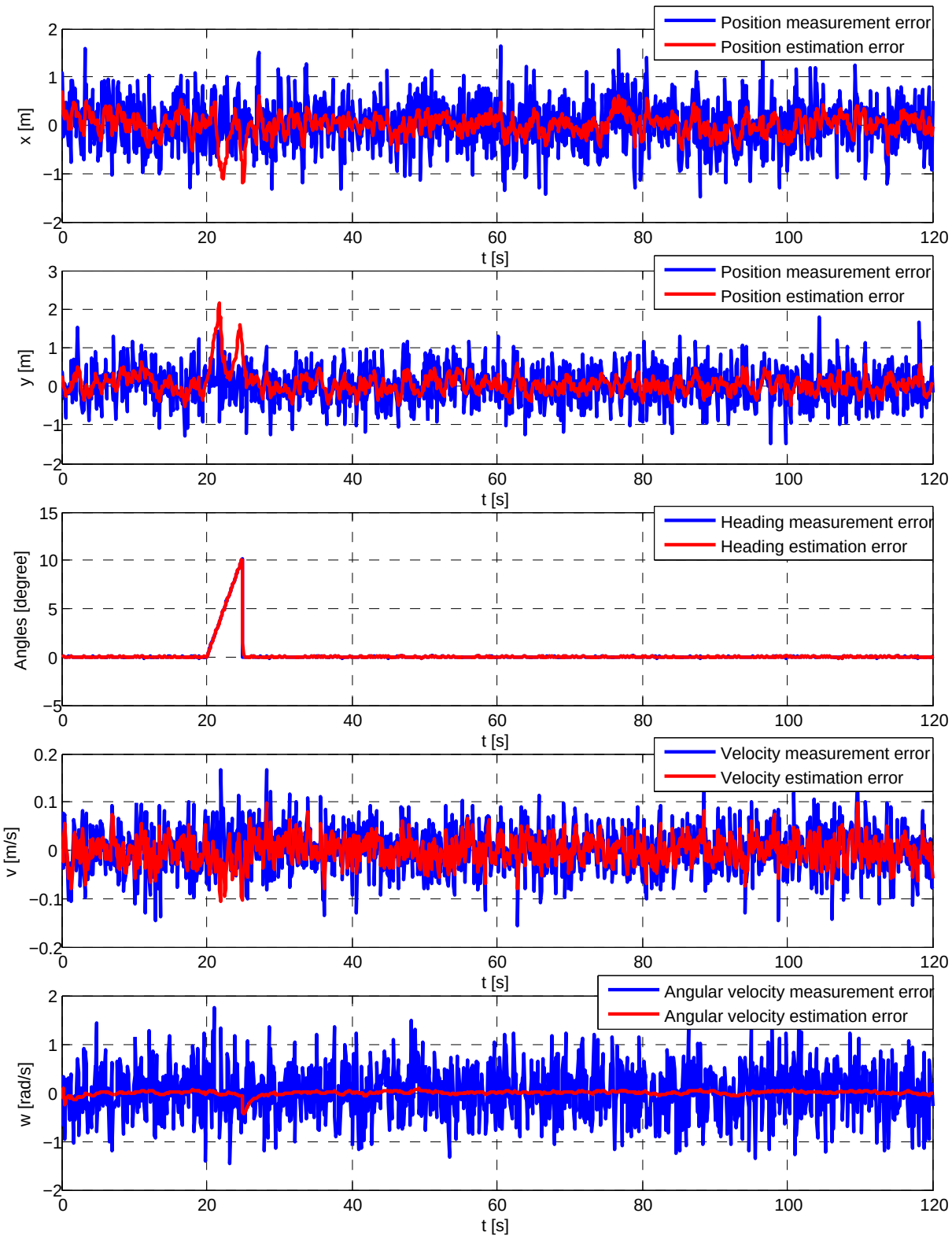


Figure 27 – Estimation errors on the case of intermittent orientation stuck fault, measured vs estimated. Input and true wind variables are constant

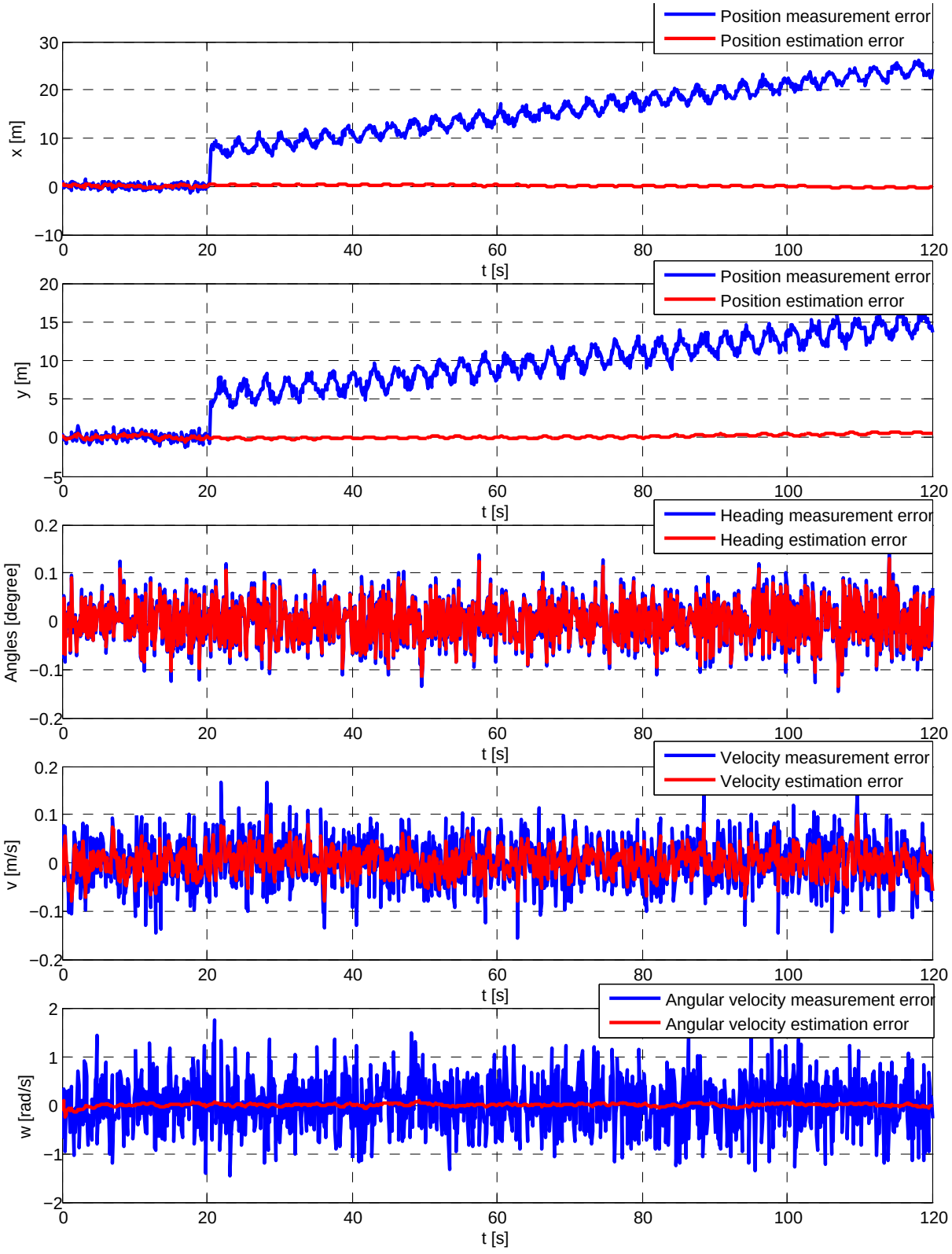


Figure 28 – Estimation errors on the case of permanent GPS stuck fault with correction, measured vs estimated. Fault signal detection delay of 5 seconds. Input and true wind variables are constant

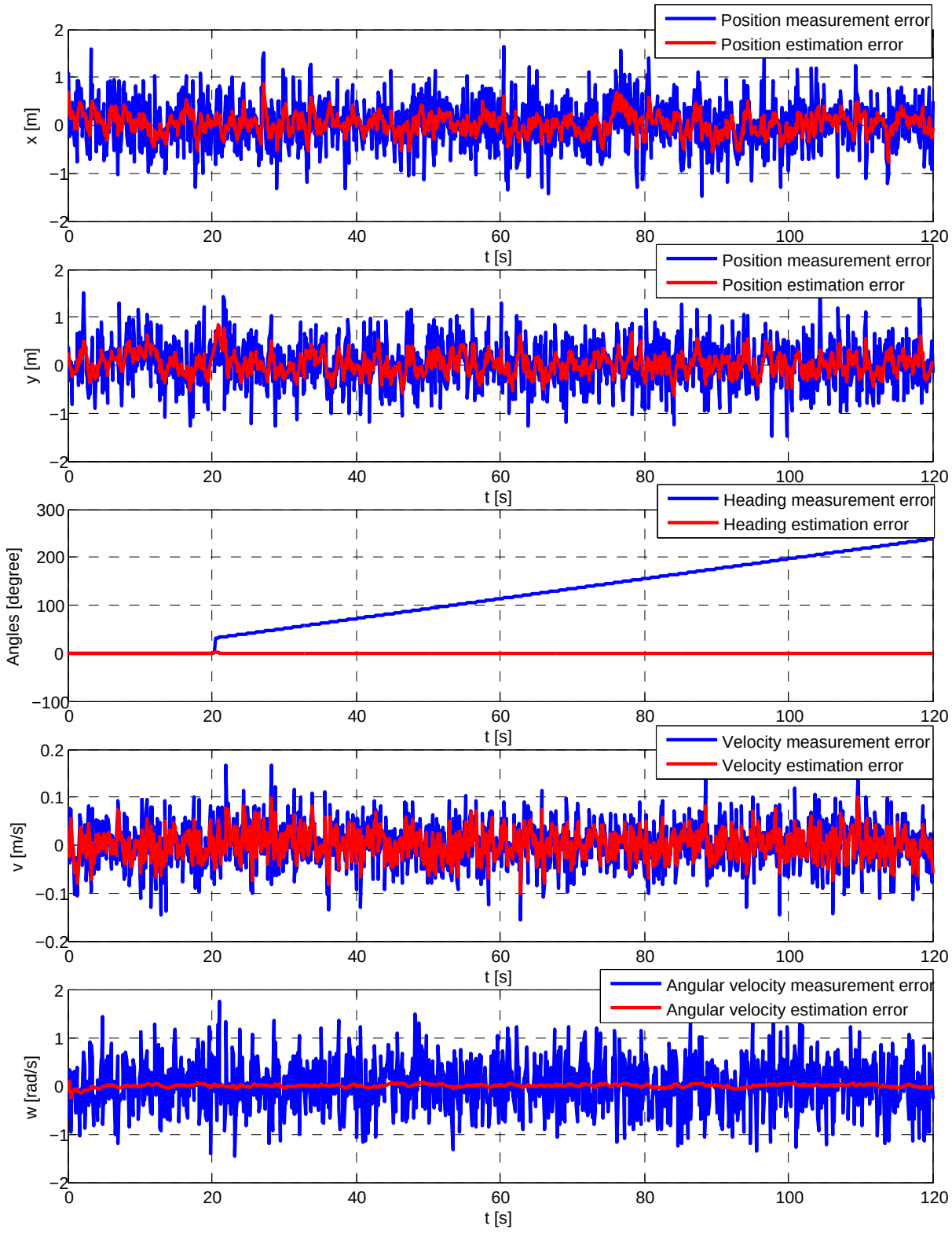


Figure 29 – Estimation errors on the case of permanent orientation stuck fault with correction, measured vs estimated. Fault signal detection delay of 5 seconds. Input and true wind variables are constant

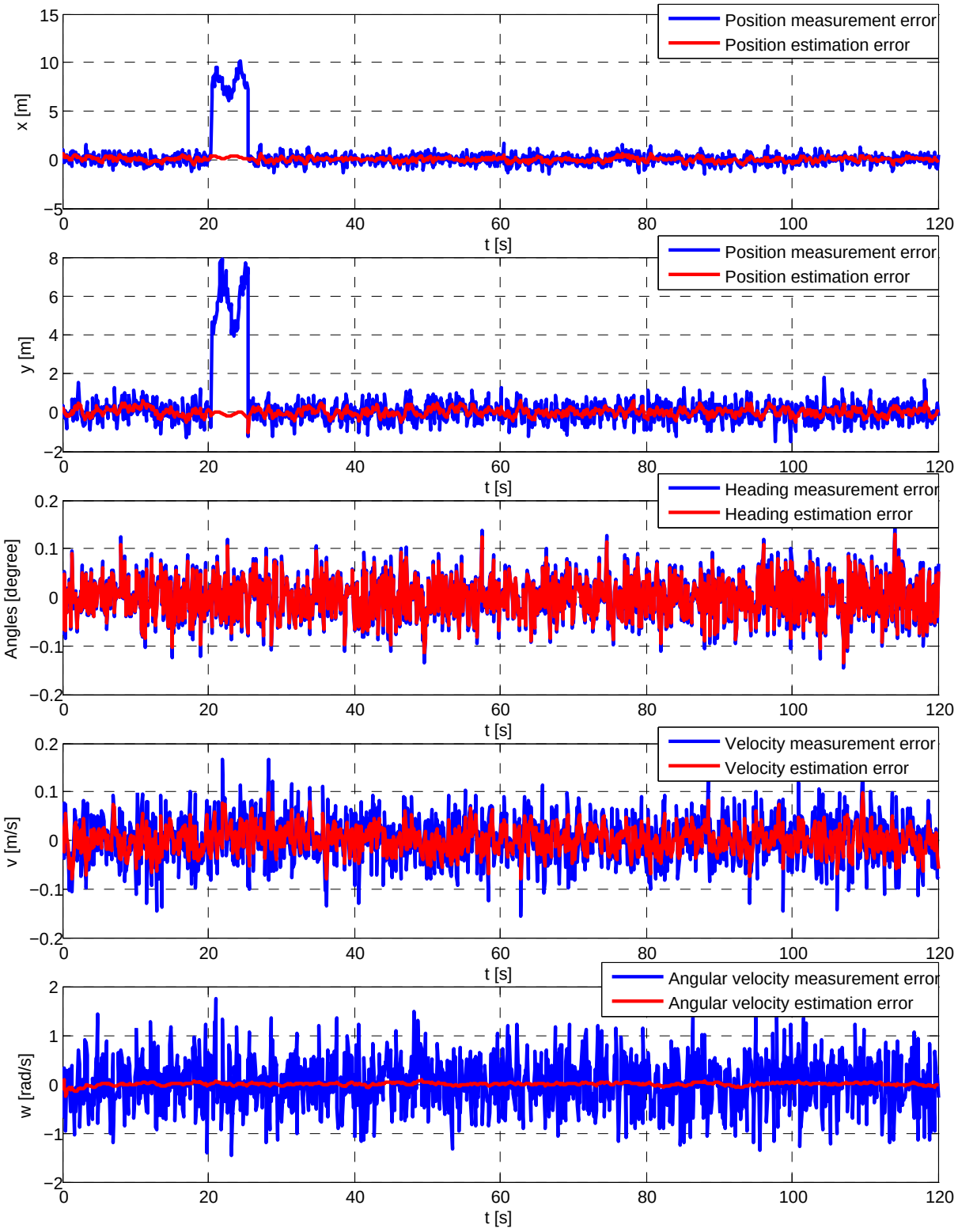


Figure 30 – Estimation errors on the case of intermittent GPS stuck fault with correction, measured vs estimated. Fault signal detection delay of 5 seconds. Input and true wind variables are constant

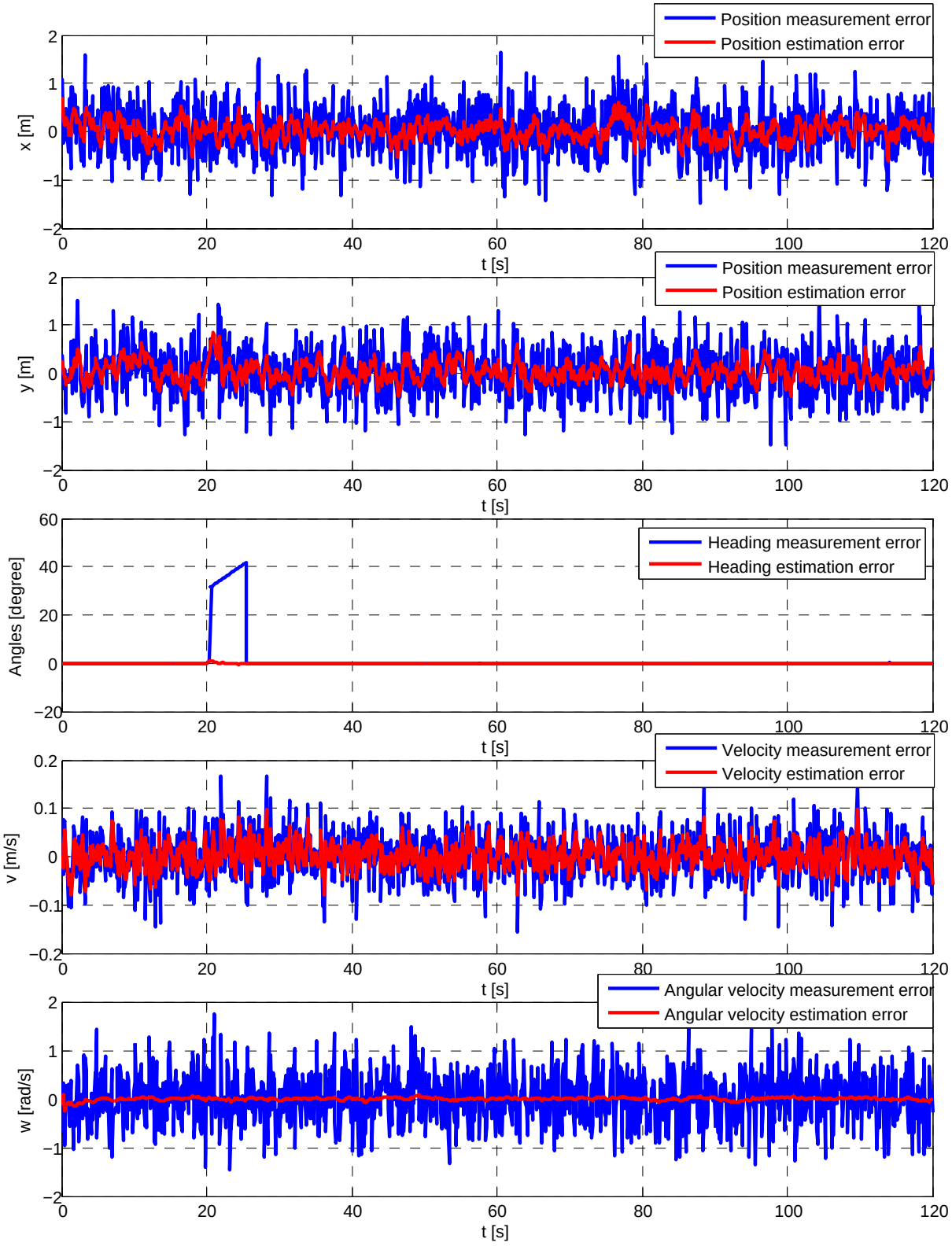


Figure 31 – Estimation errors on the case of intermittent orientation stuck fault with correction, measured vs estimated. Fault signal detection delay of 5 seconds. Input and true wind variables are constant

5 Conclusion

In this work a Extended Kalman Filter was designed and simulated in the context of a sailboat operation. The proposal was to generate good estimations based on measurement noises and process noises represented by gaussian white noises, and for that, an EKF was chosen based on its long history of optimal results.

The proposal behind the utilization of the EKF lays on it's easiness of implementation compared to its alternatives, while still presenting great results for quasilinear systems like the sailboat models in the literature.

What is found is that while the EKF indeed works, and can present great results for the scenarios displayed, its performance is still invariably proportional to the quasilinearity of the system. Meaning that parameters that increase the nonlinearities of the system, will highly influence the quality of the EKF as a estimator for the system. For example, parameters like mass p_9, that influences the changes in the velocity, and the moment of inertia p_{10}, that affects the angular acceleration, mean that the heavier the boat, the worse the EKF will perform.

The simulations show that p_{10} in special, might be responsible for the worse performance of the boat studied in the thesis, compared to the one in [14]. The reason for that is the angular velocity reflect directly the difference in the orientation, and it adds a great deal of nonlinearity to the positions state equations, though the trigonometrical functions.

In the end, even with limits of the EKF are clearly show by the simulations, the good results based on very noisy measurements are undeniable. Showing great results even for cases with really high measurement noise covariances. And though the implementation of the sensor failure correction method, a great solution to the sensor stuck failure was developed. This project allows for the further development of the boat's control system and energy saving solutions, contributing massively to eventual physical implementation of the sailboat for the Microtransat Challenge.

5.1 Further works

As can be concluded, given the extension of this project and the time available to the author, the possibility to further improvements exists. Between them, some insights for future work based on this thesis are presented.

- Since it was concluded that for bigger boats the influence of the nonlinear terms

on the state-transition equations is increased, the research of solutions like the Unscented Kalman Filter or other sampling methods might be appealing. Particle filters, for example, already have great implementations regarding wind sensors [36].

- Utilization of the properties of the Kalman filter like having more measurements to improve the estimates, like addition of a Compass to improvement to heading measurements for example [37], can to be explored.
- Other ideas like steady state EKF [38], might present good results for systems in smaller scales, with less nonlinearities.
- Further research in stochastic equations [39], is necessary, to better understand the implications of process noise into the simulations of the nonlinear system utilized.

Bibliography

1 PARTS of a Sailboat. [S.l.]. Disponível em: <http://victoria.tc.ca/Community/Firstgarryoak/>. Citado 2 vezes nas páginas 13 and 29.

2 AMERICAS Cup AC72 True Vs Apparent Wind – Upwind. [S.l.]. Disponível em: <http://www.nauticed.org/sailing-blog/americas-cup-apparent-wind/>. Citado 2 vezes nas páginas 13 and 30.

3 APPENDIX: Sketch of the orbits of Ceres and Pallas, by Gauss. [S.l.]. Disponível em: <http://sites.math.rutgers.edu/~cherlin/History/Papers1999/weiss.html>. Citado 2 vezes nas páginas 13 and 35.

4 DIFFERENTIAL GPS. [S.l.]. Disponível em: <http://www.marineengineering.org.uk/page90.html>. Citado 2 vezes nas páginas 13 and 50.

5 GENTRY, A. A review of modern sail theory. *IN: ANCIENT INTERFACE XI PROC. ELEVENTH SYMP. ON THE AERO/HYDRONAUTICS OF SAILING, (SEATTLE, U.S.A.: SEP.12, 1981)*, v. 27, 01 1981. Citado na página 23.

6 GENTRY, A. How sails really work - "the airflow diagrams in the sailing books are wrong!". *SAIL Magazine, April 1973*, 1973. Citado na página 23.

7 RYAN M. WILSON. JILA and Department of Physics, University of Colorado. *The Physics of Sailing*. Boulder, Colorado 80309-0440, USA, 2010. Citado na página 23.

8 DAHL ANTON BENGS´EN, M. W. K. Robotic sailing 2014 - power management strategies for an autonomous robotic sailboat. In: _____. [S.l.]: Springer International Publishing, 2015. p. 47–55. Citado na página 24.

9 FOSSEN, T. *Guidance and control of ocean vehicles*. Wiley, 1994. ISBN 9780471941132. Disponível em: <https://books.google.de/books?id=cwJUAAAAMAAJ>. Citado na página 31.

10 GUILLOU, G. *Architecture multi-agents pour le pilotage automatique des voiliers de competition et extensions algébriques des réseauxseaux de petri*. Tese (PhD Dissertation) — Universite de Bretagne, 2011. Citado na página 31.

11 XIAO, J. J. L. Modeling and nonlinear heading control of sailing yachts. *IEEE Journal of Oceanic Engineering*, v. 39, n. 2, p. 256–268, 2014. ISSN 0364-9059. Citado na página 31.

12 JAULIN, L. Modélisation et commande d'un bateau à voile. In: *CIFA'2004 (Conférence Internationale Francophone d'Automatique)*. [S.l.: s.n.], 2004. Citado na página 31.

13 BUEHLER, M.; HEINZ, C.; KOHAUT, S. *Dynamic Simulation Model for an Autonomous Sailboat*. Sailing Team Darmstadt e.V. - TU Darmstadt, 2018. Citado 3 vezes nas páginas 31, 35, and 36.

14 JON, M. *Modeling, control and state-estimation for an autonomous sailboat*. Tese (Doutorado) — Uppsala University, 2015. Disponível em: <urn:nbn:se:uu:diva-261553>. Citado 5 vezes nas páginas 31, 48, 59, 60, and 85.

15 JAULIN, F. L. B. L. Robotic sailing 2013 - sailboat as a windmill. In: _____. [S.l.]: Springer International Publishing, 2014. p. 81–92. ISBN 10 978-3-319-02276-5. Citado na página 31.

16 JON DAHL KJELL, W. M. M. Robotic sailing 2015 - modeling and control for an autonomous sailboat: A case study. In: _____. [S.l.]: Springer International Publishing, 2016. p. 137–149. ISBN 10 978-3-319-23335-2. Citado na página 35.

17 GREWAL, M.; ANDREWS, A. Applications of kalman filtering in aerospace 1960 to the present [historical perspectives. *Control Systems, IEEE*, v. 30, p. 69 – 78, 07 2010. Citado na página 37.

18 GREWAL, M.; ANDREWS, A. Kalman filtering: theory and practice using matlab. *New York: John Wiley and Sons*, v. 14, 2001. Citado 5 vezes nas páginas 37, 42, 43, 47, and 60.

19 JULIER, J. K. U. S. J. *New extension of the Kalman filter to nonlinear systems*. 1997. Disponível em: <https://doi.org/10.1117/12.280797>. Citado na página 45.

20 WAN, R. V. D. M. E. A. The unscented kalman filter for nonlinear estimation. In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*. [S.l.: s.n.], 2000. p. 153–158. Citado na página 45.

21 CAM-M8 u-blox M8 Concurrent GNSS Antenna Modules Data Sheet. [S.l.], 2016. Disponível em: <https://www.u-blox.com/sites/default/files/CAM-M8-FW3_DataSheet_%28UBX-15031574%29.pdf>. Citado na página 49.

22 DIGGELEN, F. van. Gps accuracy: Lies, damn lies, and statistics. *GPS World*, 01 1998. Citado na página 49.

23 DATA sheet LSM9DS1. [S.l.], 2015. Disponível em: <https://www.st.com/resource/en/datasheet/DM00103319.pdf>. Citado na página 51.

24 RICCI FABRIZIO TAFFONI, D. F. L. On the orientation error of imu: Investigating static and dynamic accuracy targeting human motion. *PLOS ONE*, Public Library of Science, v. 11, n. 9, p. 1–15, 2016. Disponível em: <https://doi.org/10.1371/journal.pone.0161940>. Citado na página 51.

25 GODWIN MICHAEL AGNEW, J. S. A. Accuracy of inertial motion sensors in static, quasistatic, and complex dynamic motion. In: . [S.l.: s.n.], 2009. v. 131, n. 11. Citado na página 51.

26 JR, R. R. L. *Kalman and Bayesian Filters in Python*. [S.l.], 2018. Disponível em: <http://robotics.itee.uq.edu.au/~elec3004/2015/tutes/Kalman_and_Bayesian_Filters_in_Python.pdf>. Citado na página 52.

27 GREWAL, M.; ANDREWS, A. Kalman filtering: Theory and practice using matlab. In: _____. Wiley, 2011. (Wiley - IEEE), cap. 7, p. 297. ISBN 9781118210468. Disponível em: <https://books.google.de/books?id=sZbxLK-NKb0C>. Citado na página 52.

28 MATLAB. *version 9.0.0.341360 (R2016a)*. Natick, Massachusetts, USA: The MathWorks Inc., 2016. Citado na página 54.

29 SIMULINK. *version 8.7 (R2016a)*. Natick, Massachusetts, USA: The MathWorks Inc., 2016. Citado na página 54.

30 D'ERRICO, J. *Numerical differentiation*. 1.0. ed. woodchips@rochester.rr.com. Citado na página 54.

31 JOAQUIM KROO ILAN, A. J. M. An automated method for sensitivity analysis using complex variables. *38th Aerospace Sciences Meeting and Exhibit*, American Institute of Aeronautics and Astronautics, n. 689, 2000. Disponível em: <https://doi.org/10.2514/6.2000-689>. Citado na página 54.

32 J. STURDZA PETER, A. J. M. The connection between the complex-step derivative approximation and algorithmic differentiation. In: . American Institute of Aeronautics and Astronautics, 2001. Disponível em: <https://doi.org/10.2514/6.2001-921>. Citado na página 55.

33 ATAN2 and complex arguments. [S.l.]. Disponível em: <http://mathforum.org/kb/message.jspa?messageID=967546>. Citado na página 55.

34 WU, L.; HO, D. W. Fuzzy filter design for itô stochastic systems with application to sensor fault detection. *IEEE Transactions on Fuzzy Systems*, IEEE, v. 17, n. 1, p. 233–242, 2009. Citado na página 57.

35 †, T. V. R. M. S. S. Simulation of sensor failure accommodation in flight control system of transport aircraft: a modular approach. *World Journal of Modelling and Simulation*, v. 11, n. 1, p. 55–68, 2015. ISSN 1 746-7233. Citado na página 59.

36 CABRERA-GÁMEZ, J. et al. A virtual wind sensor based on a particle filter. In: ALVES, J. C.; CRUZ, N. A. (Ed.). *Robotic Sailing 2016*. Cham: Springer International Publishing, 2017. p. 69–78. ISBN 978-3-319-45453-5. Citado na página 86.

37 BASTERRETXEA-IRIBAR, I.; SOTéS, I.; URIARTE, J. I. Towards an improvement of magnetic compass accuracy and adjustment. *Journal of Navigation*, Cambridge University Press, v. 69, n. 6, p. 1325–1340, 2016. Citado na página 86.

38 SIMON, D. Kalman filtering. *Embedded systems programming*, v. 14, n. 6, p. 72–79, 2001. Citado na página 86.

39 LEWIS LIHUA XIE, D. P. F. L. (Ed.). *Optimal and Robust Estimation With an Introduction to Stochastic Control Theory*. 2. edition. ed. [S.l.]: Taylor & Francis Group, LLC, 2008. ISBN 13 978-1-4200-0829-6. Citado na página 86.