NTNU
Norwegian University of
Science and Technology

Prototype of HIL Test Platform for Autonomous USV

Simulation of Vessel Dynamics

Even Ødegaard

Master of Science in Cybernetics and Robotics
Submission date: June 2017
Supervisor: Morten Breivik, ITK
Co-supervisor: Rein Anders Apeland, Kongsberg Maritime

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Preface

“Essentially, all models are wrong, but some are useful.”

– George E. P. Box

This report presents the results of a project conducted as part of my Master’s Thesis in Engineering Cybernetics during the spring of 2017, and concludes my years at the Norwegian University of Science and Technology.

The project was carried out for Kongsberg Maritime AS, which is a well established supplier of systems for dynamic positioning and navigation, marine automation, subsea survey, marine simulation and training, and satellite positioning. This thesis is conducted as the modeling of vessel dynamics and a prototype implementation of a Hardware-In-the-Loop (HIL) simulator for two unmanned surface vehicles (USVs), building upon the results accomplished during the specialization project the fall of 2016. This thesis work is performed in the hope that it can be of use in an ongoing USV project which Kongsberg Maritime is conducting in collaboration with the Norwegian Defence Research Establishment (FFI). The thesis has been carried out as part of a larger project in cooperation with Kjetil Svae Børs-Lind, who has considered the prototype implementation of simulated dynamic sorroundings for the HIL simulator.

I would like to thank the following people for valuable feedback, guidance and discussions throughout the project:

Morten Breivik	Thesis supervisor
Rein Anders Apeland	Technical Lead, USV Solutions, Kongsberg Maritime
Kjetil Svae Børs-Lind	Fellow student

Trondheim, June 5, 2017

Even Ødegaard

Summary

This thesis considers the modeling of vessel dynamics and implementation of a prototype Hardware-In-The-Loop (HIL) simulator for two autonomous unmanned surface vehicles (USVs) of different dimensions in 6 degrees of freedom, including vessel, actuator, sensor and environmental models. The simulator is part of a larger HIL test platform, which also includes simulated dynamic sorroundings (Børs-Lind (2017b)). Through a versatile design, the prototype HIL test platform is implemented to fit a wide range of vessels and scenarios.

The USVs considered, *Odin* and *Jolner*, are modeled as displacement vessels in 6 degrees of freedom using basic knowledge about the vessel dimensions in combination with well known empirical formulas, conversion of parameters from similar vessels, and parameter fitting. Comparisons of results from sea trials and simulations show a satisfactory simulation model performance up to a speed where the semi-planing dynamics occur, which are not included in the vessel models. Simulation models of realistic conditions for wind, waves and current are included in the simulator, and interaction with the vessel is implemented through the use of estimated vessel parameters. Additionally, noise-free models of GPS, MRU, speed sensor and wind sensor are included in the simulator, providing realistic sensor feedback to the USV control system.

By including the real control system on Odin in the loop, testing of the HIL functionality has been performed. Vessel performance during a waypoint guidance mission is presented for both sea trials and HIL simulations, and results suggest that the interface between the simulator prototype and the vessel control system is suitable for HIL simulations.

Sammendrag

Denne masteroppgaven tar for seg modellering av fartøysdynamikk og implementasjon av en prototype Hardware-In-The-Loop (HIL) simulator for to autonome ubemannede overflatefartøy (USV) av forskjellige dimensjoner i 6 frihetsgrader, inkludert modeller av fartøy, aktuatorer, sensorer og vær. Simulatoren er del av en større HIL-testplattform, som også inkluderer et simulert dynamisk miljø (Børs-Lind (2017a)). Med et anvendelig design er prototype HIL-testplattformen implementert for å kunne brukes for en rekke forskjellige fartøy og scenarioer.

USVene denne oppgaven tar for seg, *Odin* og *Jolner*, er modellert som deplasementfartøyer i 6 frihetsgrader ved bruk av grunnleggende kunnskap om fartøyenes dimensjoner i kombinasjon med empiriske formler, konvertering av parametre fra fartøyer med lignende skrogtype, og kurvetilpasning. Sammenligninger av resultater fra sjøtester og simuleringer viser tilfredsstillende ytelse for simuleringsmodellen opp til en hastighet der den semiplanende fartøysdynamikken dominerer, som ikke er inkludert i modellene av fartøyene. Simuleringsmodeller av realistiske forhold for vind, strøm og bølger er inkludert i simulatoren, og interaksjon med fartøyet er implementert ved bruk av estimerte fartøysparametre. I tillegg til dette er støyfrie modeller av GPS, MRU, hastighetssensor og vindsensor inkludert i simulatoren, slik at sensordata gis i tilbakekobling til USVens kontrollsystem.

Testing av HIL-funksjonaliteten har blitt gjennomført ved å inkludere reguleringssystemet til Odin i sløyfen. Fartøyets oppførsel under et oppdrag der det følger flere veipunkter presenteres for både sjøtester og HIL-simuleringer, og resultatene tyder på at grensesnittet mellom prototypesimulatoren og fartøyets reguleringssystem egner seg for HIL-simuleringer.

Contents

List of Tables

List of Figures

Acronyms

BSD	Berkeley Software Distribution
DOF	Degree of Freedom
ECEF	Earth-Centered, Earth-Fixed
FFI	Norwegian Defence Research Establishment
GNSS	Global Navigation Satellite Systems
GPS	Global Positioning System
GUI	Graphical User Interface
HIL	Hardware In The Loop
IMU	Inertial Measurement Unit
ISSC	International Ship and Offshore Structures Congress
ITTC	International Towing Tank Conference
LOS	Line-of-Sight
MRU	Motion Reference Unit
NED	North-East-Down
NMEA	National Marine Electronics Association
ODE	Ordinary Differential Equation
PID	Proportional-Integral-Derivative
RAO	Response Amplitude Operator
ROS	Robot Operating System
RPM	Revolutions Per Minute
SNAME	Society of Naval Architects and Marine Engineers
USV	Unmanned Surface Vehicle

Chapter 1

Introduction

1.1 Motivation

Automatic and autonomous systems are rapidly advancing, performing numerous tasks in our daily lives. Some of these systems relieve humans of their labour, often outperforming the human capabilities, and operating in places or conditions that are unsuitable for humans. The advances of these systems has, among others, facilitated for the development of unmanned surface vehicles (USVs), which are vehicles that operate on the ocean surface without a crew. These systems are intended to aid and enhance marine operations for both military and civilian purposes, in areas related to seabed mapping, rescue operations, surveillance and intelligence among others. A large amount of experimental USVs exist today, and research is done with the vision of completely autonomous vessels, capable of performing comprehensive and complex tasks based on simple human orders. More thorough discussions of USV development can be found in Bertram (2008) and Liu et al. (2016), which presents progress reviews and discussions concerning several USV development projects.

For a USV to be able to autonomously perform actions without human intervention, the vessel needs a wide array of sensor information, in addition to a decision making system. For maneuvering at sea the vessel needs to be aware of its own surroundings, and able to follow the rules and regulations applying. An overview of a typical USV architecture is presented in Figure 1.1.

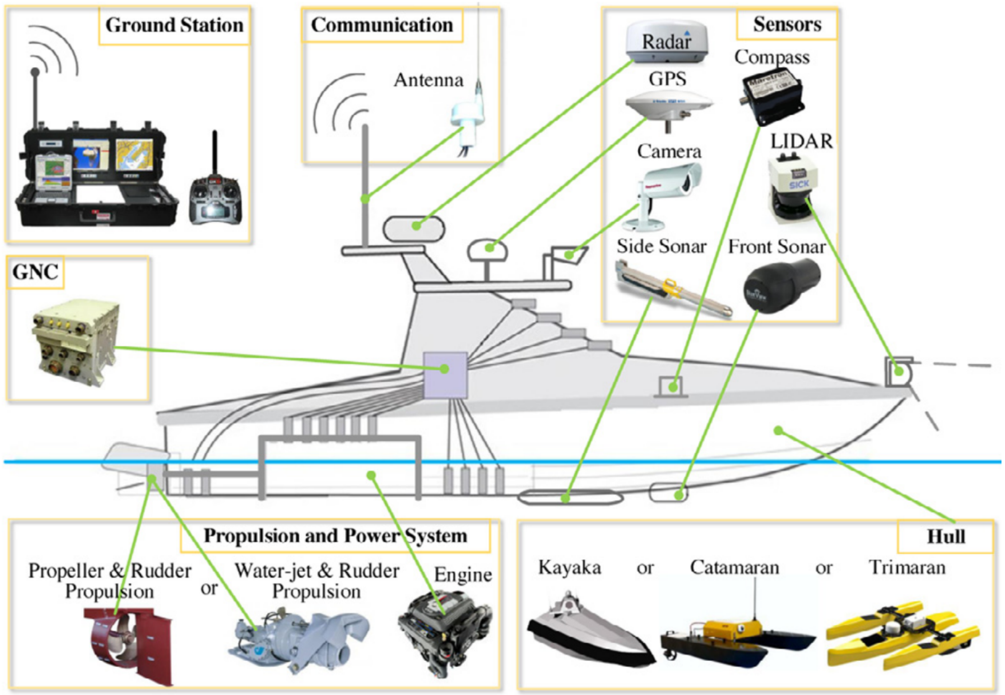


Figure 1.1: *Fundamental architecture of a typical USV. Courtesy of (Liu et al. (2016)).*

This thesis considers the prototype development of a Hardware-In-the-Loop (HIL) simulator, intended for testing of functionality necessary for the USV system to perform these operations. Development of a vessel control system is a large process, and the performance verification of such systems tends to be time consuming and expensive when performed on an actual vessel. To save resources, it is common to use simulators actively throughout the development process, to test and verify the performance of vital parts of the control system.

1.2 Background

1.2.1 Hardware-in-the-loop (HIL) Simulation

Implementation of a vessel control system is a fairly complex process, including everything from hardware installation to verification of performance. To reduce the time and cost related to the implementation of the vessel control system, it is quite common to perform several of these tasks in a simulator. The concept of HIL-simulation is to reconstruct the vessel plant at a degree where the control system faces the same interface as it would otherwise do onboard the vessel. This can be done in a variety of ways, it is possible to include all the hardware in the loop, to simulate all hardware, and anything in between. Examples of applications for HIL-simulations can be found in Baracos et al. (2001), Pugi et al. (2006) and Faruque and Dinavahi (2010).

An HIL simulator employs a real-time simulator as a replacement for the actual vessel, and simulates all signals from the vessel to the control system. As the control system outputs commands, the HIL simulator receives and applies these commands in order to simulate the resulting motion of the vessel. As both hardware and software making up the control system is included in the simulation, a broad range of both realistic operational and failure scenarios can be simulated in a controlled environment and systematic manner. Figure 1.2 illustrates the concept behind HIL simulators.

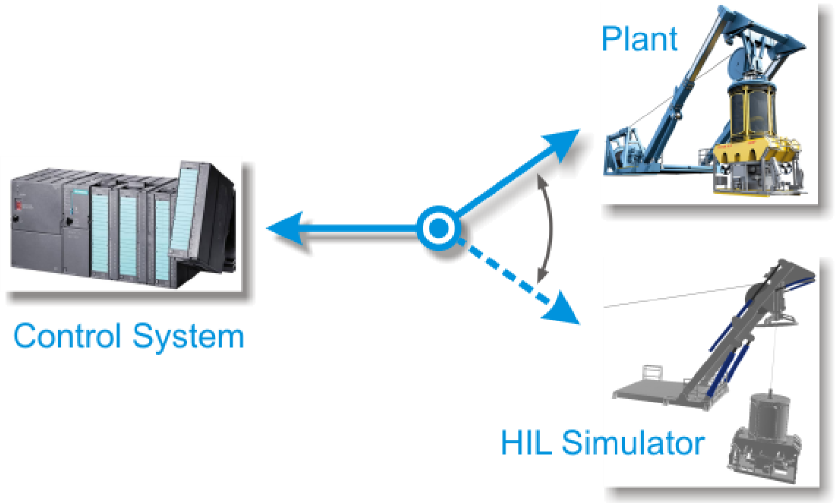


Figure 1.2: *Illustration of the concept behind HIL simulators. Courtesy of http://www.hil-simulation.com/*

While HIL simulations can not replace full-scale trials completely, the need for these trials are reduced, as extensive testing can be done early in the development process, and potential implementation-related problems can be tracked down and resolved prior to vessel installation. In addition to this, HIL-testing also gives the possibility to perform failure checks that would otherwise be challenging to complete on the actual vessel. An example illustration of a general HIL simulator setup is presented in Figure 1.3.

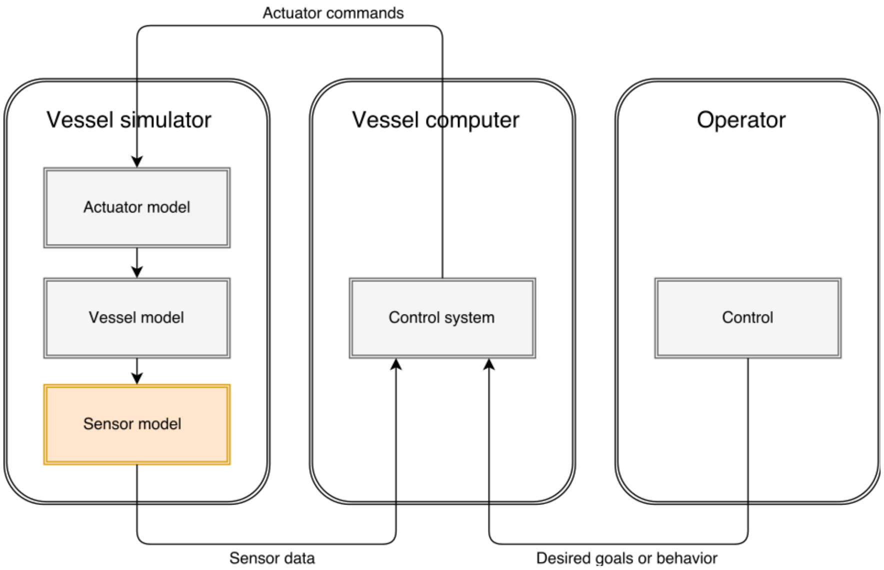


Figure 1.3: *Illustration of a general HIL simulator setup.*

1.3 Problem Formulation and Contribution

The master thesis work is divided into two parts and aims to result in a functional prototype of a HIL simulator platform for two different USVs operating in a simulated ocean environment including realistic weather conditions, obstacles and active agents. Specifically, this thesis will cover:

- Development of a simulator for the USV dynamics at low speed surveys modeled as displacement vessels, including realistic weather conditions and a system for logging of simulation data.

- Investigation of visualization in 6 DOFs using RViz, and if feasible also full 6 DOF simulation.

- Specification of the parametrization of vessel models.

- Implementation of simulated sensor data for measurement of position, attitude, velocity, acceleration and wind.

- Specification of interfaces for sensor data and actuator inputs.

- Verification of the simulated vessel dynamics vs. the real vessel dynamics.
- Testing and verification of HIL-functionality.

1.4 Outline of the Report

The rest of the report is organized into 5 chapters, described shortly below:

- Chapter 2 gives an introduction to the theoretical background for modeling of marine vessels, as well as relevant methods for estimating mathematical models of the vessels considered and the interaction with realistic weather conditions.
- Chapter 3 presents the vessels considered, the estimated mathematical models of the vessels, and the models of environmental forces and moments used in the simulator.
- Chapter 4 gives an overview of the Robot Operating System (ROS), which serves as a middleware platform in the simulator. Additionally, the prototype simulator implemented in this project is presented, with solutions for interfacing the USV control system, sensor simulation, visualization of the vessel state and logging of relevant data.
- Chapter 5 presents the results from several HIL simulations performed using the control system on Odin in the loop during a waypoint mission. Comparisons with results from the actual vessel during the same mission are presented.
- Chapter 6 presents the conclusions drawn from the obtained results, and provides recommendations for further work.

Chapter 2

Modeling of Marine Vessels

This chapter gives the theoretical background for modeling of marine vessels, thrusters and environmental forces. The models are utilized and elaborated through the rest of the chapter.

2.1 Reference Frames

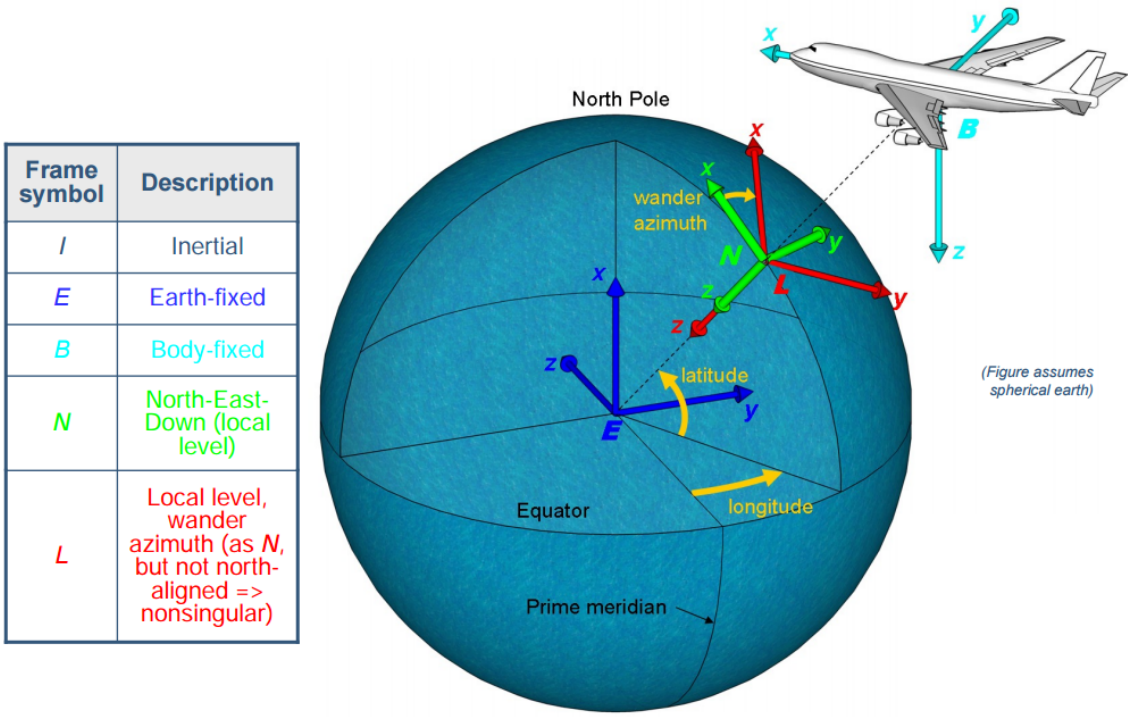
A reference frame is a predefined coordinate system of either two or three dimensions that defines the motion of this coordinate system relative to another coordinate system. In navigation, we often fix the coordinate system to the objects considered, and to the world the object navigates in (Vik (2009)).

2.1.1 ECEF, NED and BODY

When analyzing the motion of a surface vessel in 6 DOF, we usually define three frames of reference in order to navigate by means of a global navigation satellite system (GNSS) and/or compass:

ECEF

The Earth-centered Earth-fixed (ECEF) reference frame, denoted as $\{e\} = (x_e, y_e, z_e)$, has its origin o_e fixed at the Earth center, with its axes spinning along with the Earth rotation. The z-axis points north, the x-axis points to the intersection of the prime meridian and equator, and the



Frame symbol	Description
I	Inertial
E	Earth-fixed
B	Body-fixed
N	North-East-Down (local level)
L	Local level, wander azimuth (as *N*, but not north-aligned => nonsingular)

Figure 2.1: *Visualisation of the coordinate frames considered. Courtesy of Gade (2009).*

y-axis stands perpendicular to the xz-plane, completing the right hand rule. This coordinate system is usually used for global navigation and control. For marine crafts moving at low speeds, the Earth rotation can be neglected, and thus $\{e\}$ can be considered as an inertial frame.

NED

The North-East-Down (NED) reference frame, denoted as $\{n\} = (x_n, y_n, z_n)$, is defined relative to the Earth's reference ellipsoid. The z-axis points downwards, perpendicularly to the tangent plane of the ellipsoid, the x-axis points towards true north, and the y-axis points towards east, to complete the right hand rule. The reference frame used for GPS navigation is the WGS-84 (National Imagery and Mapping Agency (2000)).

BODY

The body-fixed reference frame, denoted as $\{b\} = (x_b, y_b, z_b)$, is a moving coordinate frame that is fixed to the vessel, with the origin o_b of the frame chosen to be located at the center of the vessel body, in the water line. The x-axis goes from aft to fore, the y-axis points to starboard, and the z-axis points downward. As a result of this, when the vessel points towards true north, the BODY and NED frames are parallel to each other.

2.2 Degrees of Freedom

The dynamics of a marine craft is commonly represented by a set of ordinary differential equations (ODEs), describing the craft's motions in six degrees of freedom (DOF): *surge, sway, heave, roll, pitch* and *yaw*. The equations of motion are usually represented using generalized position, velocity and forces (Fossen (2011)), defined by the state vectors:

$$\boldsymbol{\eta} := [x, y, z, \phi, \theta, \psi]^T \tag{2.1}$$

$$\boldsymbol{\nu} := [u, v, w, p, q, r]^T \tag{2.2}$$

$$\boldsymbol{\tau} := [X, Y, Z, K, M, N]^T \tag{2.3}$$

where the generalized position $\boldsymbol{\eta}$ is expressed in $\{n\}$, and the generalized velocity vector $\boldsymbol{\nu}$ and force vector $\boldsymbol{\tau}$ are both expressed in $\{b\}$. The 6-DOF states are defined according to The Society of Naval Architects and Marine Engineers (1950):

		Forces and	Linear and	Position and
DOF		moments	angular velocities	Euler angles
1	Motions in the x direction (surge)	X	u	x
2	Motions in the y direction (sway)	Y	v	y
3	Motions in the z direction (heave)	Z	w	z
4	Rotation about the x axis (roll)	K	p	ϕ
5	Rotation about the y axis (pitch)	M	q	θ
6	Rotation about the z axis (yaw)	N	r	ψ

Table 2.1: *The notation of SNAME for marine vessels*

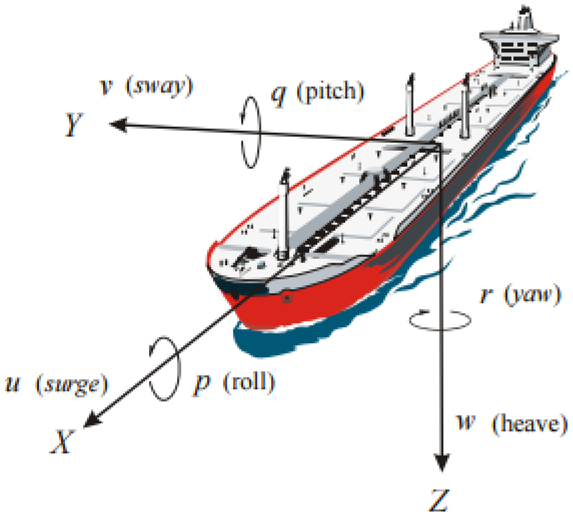


Figure 2.2: *Motion in 6 Degrees of Freedom (DOF). Courtesy of Fossen (2011).*

2.3 Froude Number

In hydrodynamics, we usually distinguish between *Seakeeping theory* and *Maneuvering theory* ((Fossen, 2011). In seakeeping theory, we typically represent the dynamics of ships at zero or constant speed in waves, like ocean structures and dynamically positioned vessels. In maneuvering theory, we represent the dynamics of a ship moving at a positive speed in restricted calm waters, using six ODEs to describe the kinematics.

From a hydrodynamic point of view, marine crafts can be classified according to their maximum operating speed, commonly described by the *Froude number*:

$$Fn := \frac{U_r}{\sqrt{gL_{pp}}} \tag{2.4}$$

where U_r is the craft speed relative to the water, L_{pp} is the overall submerged length of the craft, and g is the acceleration of gravity. In (Faltinsen (2005)), the following classifications are presented:

$Fn < 0.4$	Displacement Vessels	The buoyancy force dominates relative to the hydrodynamic force effect
$0.4 - 0.5 < Fn < 1.0 - 1.2$	Semi-displacement Vessels	The buoyancy force is not dominant at the maximum operating speed for a high speed submerged hull type of craft
$Fn > 1.0 - 1.2$	Planing Vessels	The hydrodynamic force mainly carries the weight

Table 2.2: *Classification of Froude numbers*

In this project, the vessels *Odin* and *Jolner* will be considered. As the object here is simulation of low speed surveys for localization and mapping, these vessels can be considered as displacement vessels under operation. Maneuvering theory will then be used to represent the dynamics.

2.4 Kinematics

The generalized velocities $\dot{\boldsymbol{\eta}}_b$ and $\boldsymbol{\nu}_n$ are related through the following kinematic transformation (Fossen (2011)).

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{2.5}$$

$$\boldsymbol{J} := \begin{bmatrix} \boldsymbol{R}(\boldsymbol{\Theta}) & \boldsymbol{0}_{3x3} \\ \boldsymbol{0}_{3x3} & \boldsymbol{T}(\boldsymbol{\Theta}) \end{bmatrix} \tag{2.6}$$

Where $\boldsymbol{\Theta} = [\phi, \theta, \psi]$ is the Euler angles, and

$$\boldsymbol{R}(\boldsymbol{\Theta}) = \begin{bmatrix} c(\psi)c(\theta) & -s(\psi)c(\phi) + c(\psi)s(\theta)s(\phi) & s(\psi)s(\phi) + c(\psi)c(\theta) \\ s(\psi)c(\theta) & c(\psi)c(\phi) + s(\phi)s(\theta)s(\psi) & -c(\psi)s(\phi) + s(\theta)s(\psi)c(\phi) \\ -s(\theta) & c(\theta)s(\phi) & c(\theta)c(\phi) \end{bmatrix} \tag{2.7}$$

$$\boldsymbol{T}(\boldsymbol{\Theta}) = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi)/c(\theta) & c(\phi)/c(\theta) \end{bmatrix}, \theta \neq \pm\frac{\pi}{2} \tag{2.8}$$

where $s(x) = \sin(x)$, $c(x) = \cos(x)$, $t(x) = \tan(x)$. For $\theta = \pm\frac{\pi}{2}$ a singularity occurs, which can be avoided using *unit quaternions*. As $\theta = \pm\frac{\pi}{2}$ is not a realistic situation for a surface vessel, the Euler angle representation will be used in this prototype.

2.5 Kinetics

Through use of the *Newton-Euler formulation*, the rigid-body kinetics can be derived. According to (Fossen (2011)), this gives:

$$\boldsymbol{M}_{RB}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}_{RB} \tag{2.9}$$

where $\boldsymbol{M}_{RB}$ is the rigid-body mass matrix, $\boldsymbol{C}_{RB}$ is the rigid-body Coriolis and centripetal matrix due to the rotation of $\{b\}$ about $\{n\}$, and $\boldsymbol{\tau}_{RB}$ represents the external forces and moments expressed in $\{b\}$:

$$\boldsymbol{\tau}_{RB} = -\boldsymbol{M}_A\dot{\boldsymbol{\nu}} - \boldsymbol{C}_A(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{G}\boldsymbol{\eta} - \boldsymbol{\mu} + \boldsymbol{\tau} \tag{2.10}$$

where $\boldsymbol{M}_A$ and $\boldsymbol{C}_A(\boldsymbol{\nu})$ represents hydrodynamic added mass, $\boldsymbol{D}(\boldsymbol{\nu})$ represents hydrodynamic damping, $\boldsymbol{G}$ the generalized restoring forces, $\boldsymbol{\mu}$ represents the fluid memory effects, and $\boldsymbol{\tau}$ represent the control inputs as generalized forces. By introducing *ocean currents*, we can augment the model further, and introduce the relative velocity vector:

$$\boldsymbol{\nu}_r = \boldsymbol{\nu} - \boldsymbol{\nu}_c \tag{2.11}$$

where $\boldsymbol{\nu}_c = [u_c^b, v_c^b, 0]^T$ is a vector of current velocities in $\{b\}$. Considering the ocean currents, we obtain the following equations:

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{C}_A(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{G}\boldsymbol{\eta} + \boldsymbol{\mu} = \boldsymbol{\tau} + \boldsymbol{\tau}_{wind} + \boldsymbol{\tau}_{wave} \tag{2.12}$$

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu}_r + \boldsymbol{\nu}_c^n \tag{2.13}$$

2.5.1 Inertia

The rigid-body mass matrix $\boldsymbol{M}_{RB}$ is the contribution of the rigid physical structure of the vessel, and the added mass matrix $\boldsymbol{M}_A$ represents the contribution from the hydrodynamically added mass, which can be seen as a virtual mass added to the system due to an acceleration or deceleration of the vessel. This imposes an acceleration or deceleration of the volume of surrounding fluid, which adds a virtual mass to the vessel. The rigid-body mass matrix $\boldsymbol{M}_{RB}$ is composed as:

$$\boldsymbol{M}_{RB} = \begin{bmatrix} m\boldsymbol{I}_{3x3} & -m\boldsymbol{S}(\boldsymbol{r}_g^b) \\ m\boldsymbol{S}(\boldsymbol{r}_g^b) & \boldsymbol{I}_b \end{bmatrix} \tag{2.14}$$

$$= \begin{bmatrix} m & 0 & 0 & 0 & mz_g & -my_g \\ 0 & m & 0 & -mz_g & 0 & mx_g \\ 0 & 0 & m & my_g & -mx_g & 0 \\ 0 & -mz_g & my_g & I_x & -I_{xy} & -I_{xz} \\ mz_g & 0 & -mx_g & -I_{yx} & I_y & -I_{yz} \\ -my_g & mx_g & 0 & -I_{zx} & -I_{zy} & I_z \end{bmatrix} \tag{2.15}$$

where m is the vessel mass, $\boldsymbol{I}_b$ is the inertia matrix about the vessel's center of buoyancy, and $\boldsymbol{S}(\boldsymbol{r}_g^b)$ is a skew symmetric matrix representing the distance between the buoyancy and gravity centers of the vessel. The hydrodynamic added mass will in general depend on the frequency of motion due to water surface effects, but Fossen (2011) states that the added mass can be approximated to a constant frequency independent matrix, based on the assumption that the surge motion is decoupled and that the vessel is starboard-port symmetric. This gives the following added mass matrix:

$$\boldsymbol{M}_A = - \begin{bmatrix} X_{\dot{u}} & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{\dot{v}} & 0 & Y_{\dot{p}} & 0 & Y_{\dot{r}} \\ 0 & 0 & Z_{\dot{w}} & 0 & Z_{\dot{q}} & 0 \\ 0 & K_{\dot{v}} & 0 & K_{\dot{p}} & 0 & K_{\dot{r}} \\ 0 & 0 & M_{\dot{w}} & 0 & K_{\dot{q}} & 0 \\ 0 & N_{\dot{v}} & 0 & N_{\dot{p}} & 0 & N_{\dot{r}} \end{bmatrix} \tag{2.16}$$

2.5.2 Coriolis and Centripetal Forces

As the dynamics of the vessel are stated in the non-inertial body frame, the Coriolis and centripetal effects introduces nonlinear terms in the Coriolis-centripetal matrix. The Coriolis and centripetal terms are stated as

$$\boldsymbol{C}(\boldsymbol{\nu},\boldsymbol{\nu}_r) = \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{C}_A(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r \tag{2.17}$$

where $\boldsymbol{C}_{RB}(\boldsymbol{\nu})$ relates to the contribution from the rigid body, and $\boldsymbol{C}_A(\boldsymbol{\nu}_r)$ to the added mass. Fossen (2011) uses the following rigid-body and added mass Coriolis and centripetal matrices:

$$\boldsymbol{C}_{RB}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & 0 & m(y_g q + z_g r) & -m(x_g q - w) & -m(x_g r + v) \\ 0 & 0 & 0 & -m(y_g p + w) & m(z_g r + x_g p) & -m(y_g r - u) \\ 0 & 0 & 0 & -m(z_g p - v) & -m(z_g q + u) & m(x_g p + y_g q) \\ -m(y_g q + z_g r) & m(y_g p + w) & m(z_g p - v) & 0 & -I_{yz} q - I_{xz} p + I_z r & I_{yz} r + I_{xy} p - I_y q \\ m(x_g q - w) & -m(z_g r + x_g p) & m(z_g q + u) & I_{yz} q + I_{xz} p - I_z r & 0 & -I_{xz} r - I_{xy} q + I_x p \\ m(x_g r + v) & m(y_g r - u) & -m(x_g p + y_g q) & -I_{yz} r - I_{xy} p + I_y q & I_{xz} r + I_{xy} q - I_x p & 0 \end{bmatrix} \tag{2.18}$$

$$\boldsymbol{C}_A(\boldsymbol{\nu}_r) = \begin{bmatrix} 0 & 0 & 0 & 0 & -a_3 & a_2 \\ 0 & 0 & 0 & a_3 & 0 & -a_1 \\ 0 & 0 & 0 & -a_2 & a_1 & 0 \\ 0 & -a_3 & a_2 & 0 & -b_3 & b_2 \\ a_3 & 0 & -a_1 & b_3 & 0 & -b_1 \\ -a_2 & a_1 & 0 & -b_2 & b_1 & 0 \end{bmatrix} \tag{2.19}$$

where

$$a_1 = X_{\dot{u}} u_r + X_{\dot{v}} v_r + X_{\dot{w}} w_r + X_{\dot{p}} p + X_{\dot{q}} q + X_{\dot{r}} r \tag{2.20}$$

$$a_2 = Y_{\dot{u}} u_r + Y_{\dot{v}} v_r + Y_{\dot{w}} w_r + Y_{\dot{p}} p + Y_{\dot{q}} q + Y_{\dot{r}} r \tag{2.21}$$

$$a_3 = Z_{\dot{u}} u_r + Z_{\dot{v}} v_r + Z_{\dot{w}} w_r + Z_{\dot{p}} p + Z_{\dot{q}} q + Z_{\dot{r}} r \tag{2.22}$$

$$b_1 = K_{\dot{u}} u_r + K_{\dot{v}} v_r + K_{\dot{w}} w_r + K_{\dot{p}} p + K_{\dot{q}} q + K_{\dot{r}} r \tag{2.23}$$

$$b_2 = M_{\dot{u}} u_r + M_{\dot{v}} v_r + M_{\dot{w}} w_r + M_{\dot{p}} p + M_{\dot{q}} q + M_{\dot{r}} r \tag{2.24}$$

$$b_3 = N_{\dot{u}} u_r + N_{\dot{v}} v_r + N_{\dot{w}} w_r + N_{\dot{p}} p + N_{\dot{q}} q + N_{\dot{r}} r \tag{2.25}$$

2.5.3 Damping Forces

All marine vessels that displaces water are affected by hydrodynamic damping, which is mainly caused by potential damping, skin friction, wave drift damping, lifting forces and vortex shedding (Fossen (2011)). Due to symmetry around the xz-plane it is common to assume decoupled surge dynamics, which gives the following linear damping matrix $\boldsymbol{D}_L$:

$$\boldsymbol{D}_L = -\begin{bmatrix} X_u & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_v & 0 & Y_p & 0 & Y_r \\ 0 & 0 & Z_w & 0 & Z_q & 0 \\ 0 & K_v & 0 & K_p & 0 & K_r \\ 0 & 0 & M_w & 0 & M_q & 0 \\ 0 & N_v & 0 & N_p & 0 & N_r \end{bmatrix} \tag{2.26}$$

At low velocities, this linear contribution represents the dominating forces, while at higher velocities, the nonlinear contribution $\boldsymbol{D}_{NL}(\boldsymbol{v}_r)$ increases. A simplified representation of the nonlinear contribution can be obtained by considering the nonlinear surge damping and the crossflow drag.

2.5.3.1 Nonlinear Surge Damping

In Fossen (2011) the following surge damping coefficient is suggested for low-speed maneuvering:

$$X_{|u|u} = -\frac{1}{2}\rho A_x C_x \tag{2.27}$$

where A_x is the frontal project area and C_x is the current coefficient, given by

$$C_x = -\frac{S}{A_x} C_f(u_r) \tag{2.28}$$

$$C_f(u_r) = \frac{0.075}{(log_{10} R_n - 2)^2} + C_R \tag{2.29}$$

$$R_n(u_r) = \frac{u_r L_{pp}}{1 \times 10^{-6} m/s^2} \tag{2.30}$$

where C_R represents residual friction due to hull roughness. The current coefficients are usually estimated from experiments in up to 1 m/s currents.

2.5.3.2 Cross-flow Drag Principle

For relative current angles $|\beta_c - \psi| >> 0$ the cross-flow drag principle may be applied. This will provide an estimate for the non-linear damping force in sway and the yaw moment:

$$Y = -\frac{1}{2}\rho \int_{\frac{-L_{pp}}{2}}^{\frac{L_{pp}}{2}} T(x) C_d^{2D}(x) |v_r + xr|(v_r + xr) dx \tag{2.31}$$

$$N = -\frac{1}{2}\rho \int_{\frac{-L_{pp}}{2}}^{\frac{L_{pp}}{2}} T(x) C_d^{2D}(x) x |v_r + xr|(v_r + xr) dx \tag{2.32}$$

where $T(x) = T$ is the draft, assumed to be constant, $v_r = v - v_c$ is the relative sway velocity, and C_d^{2D} is the 2-D drag coefficient, estimated using *Hoerner's curve*, see Figure 2.3. Having numerical values for $Y_{|v|v}$ and $N_{|v|v}$ from (2.31) and (2.32), the values can be compared to the scaled restoring maneuvering coefficients from similar vessel types (Halvorsen (2008)), where these coefficients are known. The scaling could be performed using the *Bis system*, presented in Table 2.3. The Bis system can be used for zero-speed as well as high-speed applications, since division of speed u is avoided, and thus it could be beneficial to use if Dynamic Positioning is to be simulated.

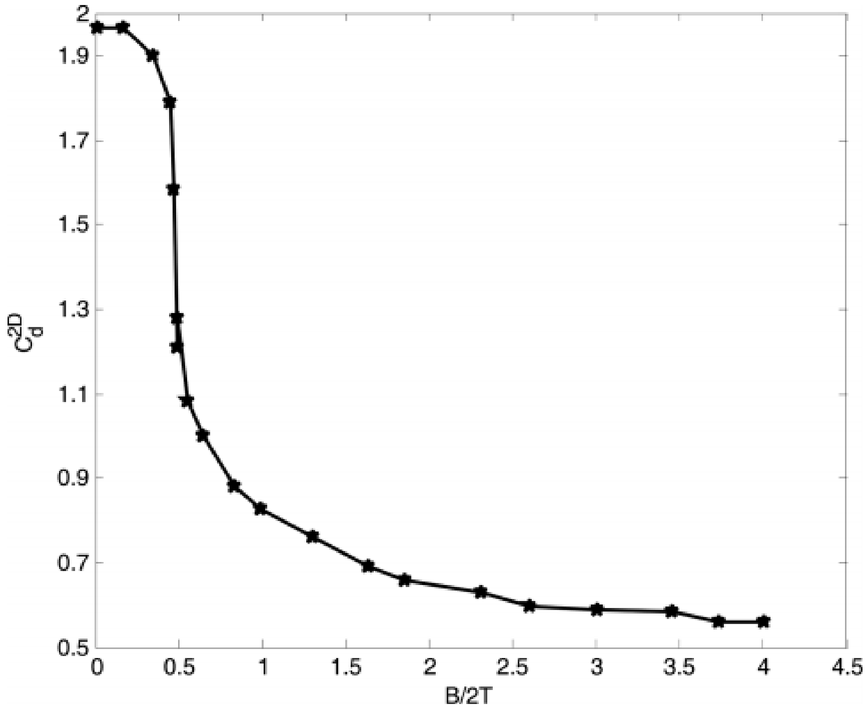


Figure 2.3: *C_d^{2D} as a function of $B/2T$. Courtesy of Hoerner (1965).*

Unit	Prime System	Bis System
Length	L	L
Mass	$\frac{1}{2}\rho L^3$	$\mu\rho\nabla$
Inertia moment	$\frac{1}{2}\rho L^5$	$\mu\rho\nabla L^2$
Time	$\frac{L}{U}$	$\sqrt{L/g}$

Table 2.3: *Relevant normalization variables used for the Prime and Bis systems. Courtesy of Fossen (2011).*

2.5.4 Restoring Forces

For surface vessels, restoring forces are usually referred to as metacentrically stability, where a metacentric stable vessel will resist inclinations away from its steady state or equilibrium points in heave, roll and pitch. The restoring forces will be dependent on the vessel's metacentric height, the location of the center of gravity (CG) and center of buoyancy (CB), and the shape and size of the water plane. As the vessel rolls, the CB moves due to the changed distribution of submerged volume, whereas the CG is always fixed at the same point. This results in moment in roll which will try to restore the equilibrium. The concept is illustrated in Figure 2.4. For surface vessels, the restoring forces are usually represented by a linear approximation. By assuming asymmetry about the yz-plane, Fossen (2011) states that the restoring force matrix $\boldsymbol{G}$ is defined as:

$$\boldsymbol{G} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -Z_z & 0 & -Z_\theta & 0 \\ 0 & 0 & 0 & -K_\phi & 0 & 0 \\ 0 & 0 & -M_z & 0 & -M_\theta & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} > 0 \tag{2.33}$$

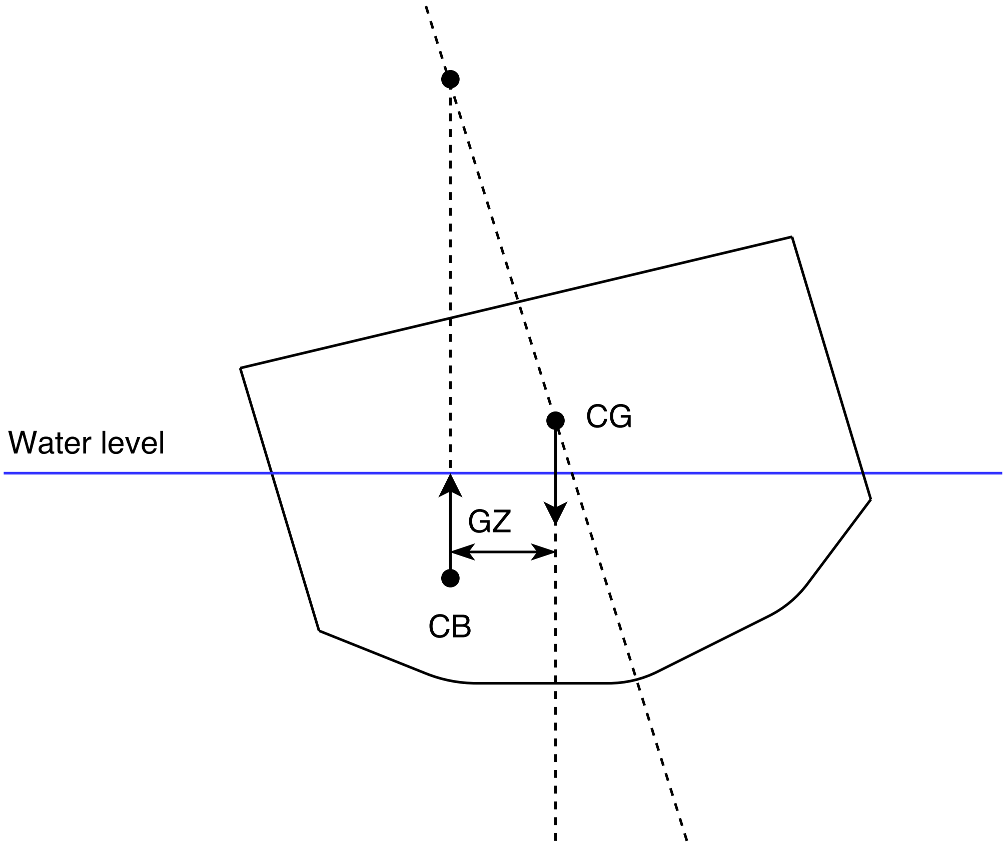


Figure 2.4: *Illustration of the longitudinal restoring forces acting on a surface vessel.*

2.5.5 Fluid Memory Effects

The radiation forces at a given time depends on the history of the vessel's velocity up to the present time, which means that once the vessel changes the momentum of the fluid, this will affect the forces in the future. This phenomenon is known as the fluid memory effect $\boldsymbol{\mu}$, which acts as a low-pass filtered damper. According to Fossen (2011) it has the following dynamics:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}_r \boldsymbol{x} + \boldsymbol{B}_r \delta \boldsymbol{\nu} \tag{2.34}$$

$$\boldsymbol{\mu} = \boldsymbol{C}_r \boldsymbol{x} \tag{2.35}$$

$$\delta \boldsymbol{\nu} = \boldsymbol{\nu} - \begin{bmatrix} u \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad \boldsymbol{\mu} = \begin{bmatrix} \mu_u \\ \mu_v \\ \mu_w \\ \mu_p \\ \mu_q \\ \mu_r \end{bmatrix} \tag{2.36}$$

where the dimension of $\boldsymbol{x}$ and the matrices $\boldsymbol{A}_r$, $\boldsymbol{B}_r$ and $\boldsymbol{C}_r$ depend on the order of the identified transfer functions.

2.6 Actuators

2.6.1 Thruster Dynamics

The dynamics for both electric and combustion engines are non-linear. According to Fossen (2011), these dynamics can be approximated with a linear first order model for modeling and simulation:

$$\dot{u}_i = -\frac{1}{T_i}(u_i - u_{i,r}) \tag{2.37}$$

where u_i is the actuator state, $u_{i,r}$ is the actuator setpoint, and T_i is the actuator time constant. Considering the possibility of *azimuth* thrusters, we introduce the thruster angle α_i, and a first order model for the angular dynamics:

$$\dot{\alpha}_i = -\frac{1}{T_{\alpha,i}}(\alpha_i - \alpha_{i,r}) \tag{2.38}$$

The control forces and moments for all thrusters $\boldsymbol{f}$ are expressed as $\boldsymbol{f} = \boldsymbol{K}\boldsymbol{u}$, where $\boldsymbol{u}$ is a vector containing the state of each actuator u_i, and $\boldsymbol{K}$ is a diagonal force coefficient matrix of dimensions $i \times i$. Through the use of a thrust configuration matrix $\boldsymbol{T}(\boldsymbol{\alpha})$, we can express the actuator forces and moments as:

$$\boldsymbol{\tau} = \boldsymbol{T}(\boldsymbol{\alpha})\boldsymbol{K}\boldsymbol{u} \tag{2.39}$$

where $\boldsymbol{T}(\boldsymbol{\alpha})$ relates to the position and angle of each actuator relative to CG. From Fossen (2011), we have that the thrust F from a variable-speed, fixed-pitch propeller can be modelled as:

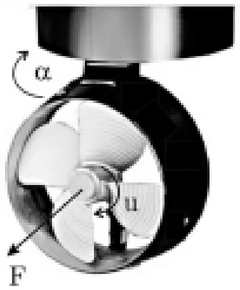


Figure 2.5: *Illustration of an azimuth thruster. Courtesy of Fossen (2011).*

$$F(n) = Kn|n| \tag{2.40}$$

where n is the propeller revolutions per minute. This reduces (2.39) to:

$$\boldsymbol{\tau} = \boldsymbol{T}(\boldsymbol{\alpha})\boldsymbol{F}(\boldsymbol{n}) \tag{2.41}$$

which for a vessel with two rear mounted azimuth thrusters gives the following:

$$\boldsymbol{\tau} = \begin{bmatrix} X \\ Y \\ N \end{bmatrix} = \begin{bmatrix} cos(\alpha_1) & cos(\alpha_2) \\ sin(\alpha_1) & sin(\alpha_2) \\ l_{x_1} sin(\alpha_1) - l_{y_1} cos(\alpha_1) & l_{x_2} sin(\alpha_2) - l_{y_2} cos(\alpha_2) \end{bmatrix} \begin{bmatrix} K_1 n_1|n_1| & 0 \\ 0 & K_2 n_2|n_2| \end{bmatrix} \tag{2.42}$$

where l_{x_1}, l_{x_2}, l_{y_1} and l_{y_2} are the moment arms in yaw.

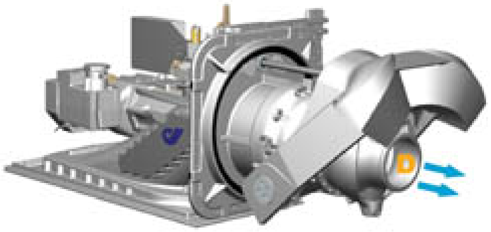


Figure 2.6: *Illustration of a waterjet propulsion system. Courtesy of HamiltonJet*

In the case of water jet thrusters, the response on the thrust magnitude will be somewhat similar to the response described in (2.37), usually with a slightly lower time constant. However, through use of an *astern deflector*, the water jet system is able to change the angle of the thrust vector in a very short time, in any direction. This enables water jet driven vessels to be extremely responsive, and in the case of two or more independent water jet engines, the vessel will also be fully actuated in the xy-plane. Water jet powered vessels suffer from the *Coandă effect*, which is the tendency of a fluid to follow an adjacent flat or curved surface. But since this effect gives a relatively small contribution to the vessel dynamics, it will be excluded from the simulations during prototyping.

2.6.2 Rudder Dynamics

Rudders are the primary steering device for most conventional marine vessels. They are usually located aft of the craft, providing a rudder force F_y, which will be a function of the rudder deflection. This force in *sway* will produce a yaw moment, which is used for control of the vessel. As a result of this, it is necessary to obtain a model of the rudder force, as well as knowing the exact location of the rudder. After obtaining this, the control force due to a rudder can be modelled using (2.37), by defining a sufficient time constant. The vessels considered do not use rudders for control, and thus these forces will not be considered while prototyping.

2.7 Modeling and Parametrization of Environmental Forces and Moments

Modeling the environmental forces and moments acting on the vessel is a task consisting of both modeling the disturbance in itself and modeling of the interaction between the vessel and the disturbance, which in turn gives the forces and moments acting on the vessel. It is common in most marine control applications to assume the principle of superposition, which usually gives good approximations. "In general, the environmental forces will be highly nonlinear and both additive and multiplicative to the dynamic equations of motion" (Fossen (2011)). As the models used will be rough approximations of the real vessels, so will the environmental forces and moments.

There exist several different systems for description of the environmental conditions at sea. One widely used system is the Beaufort wind scale, which describes wind and wave magnitude at open seas. The system is inaccurate for shore conditions, but nevertheless, it provides a good measure of the conditions at sea.

Beaufort number	Description	Wind speed	Wave height	Sea condition
0	Calm	< 0.3 m/s	0 m	Calm, mirror-like
1	Light air	0.3-1.5 m/s	0-0.2 m	Rippled
2	Light breeze	1.6-3.3 m/s	0.2-0.5 m	Small wavelets
3	Gentle breeze	3.4-5.5 m/s	0.5-1 m	Large wavelets
4	Moderate breeze	5.5-7.9 m/s	1-2 m	Small waves
5	Fresh breeze	8-10.7 m/s	2-3 m	Moderate waves
6	Strong breeze	10.8-13.8 m/s	3-4 m	Long waves, white foam
7	High wind	13.9-17.1 m/s	4-5.5 m	Breaking waves
8	Gale	17.2-20.7 m/s	5.5-7.5 m	Moderately high waves
9	Strong gale	20.8-24.4 m/s	7-10 m	High waves
10	Storm	24.5-28.4 m/s	9-12.5 m	Very high waves
11	Violent storm	28.5-32.6 m/s	11.5-16 m	Exceptionally high waves
12	Hurricane	≥ 32.7 m/s	≥ 14 m	Huge waves

Table 2.4: *The Beaufort wind force scale. Courtesy of* `https://www.britannica.com`*.*

2.7.1 Wind Forces and Moments

Mathematical models of wind forces and moments are used in both simulations and control systems, to improve performance and robustness of the system. The effects of wind are commonly divided into a mean, a slowly-varying, and a rapidly varying component. In reality, wind is a three-dimensional phenomenon, but the most common descriptions are restricted to velocities in the horizontal plane, parametrized by the velocity U and the direction ψ.

2.7.1.1 Modeling

According to Sørensen (2013), the mean velocity at z meters above the surface, $\bar{U}(z)$, can be calculated as:

$$\bar{U}(z) = \bar{U}_{10}\frac{5}{2}\sqrt{\kappa}ln\frac{z}{10e^{-\frac{2}{5*\sqrt{\kappa}}}} \tag{2.43}$$

where $\bar{U}_{10}$ is the wind speed at $10m$ elevation, and κ is the sea surface drag coefficient. Slow variations in the mean wind velocity can be implemented using a first order Gauss-Markov process:

$$\dot{\bar{U}} + \mu\bar{U} = \omega \tag{2.44}$$

where ω is Gaussian white noise and the magnitude of the velocity is restricted by saturation elements

$$0 \leq \bar{U}_{min} \leq \bar{U} \leq \bar{U}_{max} \tag{2.45}$$

A widely used formulation for the wind gust components is the Harris wind spectrum (Davenport (1977)):

$$S(f) = \frac{4\kappa L\bar{U}_{10}}{(2+\tilde{f}^2)^{5/6}} \tag{2.46}$$

$$\tilde{f} = \frac{Lf}{\bar{U}_{10}} \tag{2.47}$$

where L is a scaling length and f is the frequency in Hz. The wind gust over time with N gust components can then be calculated using the following equation:

$$U_g(t) = \sum_{i=1}^{N}\sqrt{2S(f_i)\Delta f_i}cos(2\pi f_t t + \phi_i) \tag{2.48}$$

where Δf_i is the frequency interval and ϕ_i is an evenly distributed phase angle. The total wind realization can then be written as

$$U(z,t) = \bar{U}(z) + U_g(t) \tag{2.49}$$

More recent wind spectrums gives alternative representations, among others the Norsok Standard (NORSOK).

2.7.1.2 Vessel Interaction

Letting v_w and γ_w denote the wind speed and angle of attack, the wind forces and moments acting on a marine craft can be computed using the following approach (Fossen (2015)):

$$X_{wind} = \frac{1}{2}\rho_a V_w^2 C_X(\gamma_w) A_{F_w} \tag{2.50}$$

$$Y_{wind} = \frac{1}{2}\rho_a V_w^2 C_Y(\gamma_w) A_{L_w} \tag{2.51}$$

$$Z_{wind} = \frac{1}{2}\rho_a V_w^2 C_Z(\gamma_w) A_{F_w} \tag{2.52}$$

$$K_{wind} = \frac{1}{2}\rho_a V_w^2 C_K(\gamma_w) A_{L_w} H_{L_w} \tag{2.53}$$

$$M_{wind} = \frac{1}{2}\rho_a V_w^2 C_M(\gamma_w) A_{F_w} H_{F_w} \tag{2.54}$$

$$N_{wind} = \frac{1}{2}\rho_a V_w^2 C_N(\gamma_w) A_{L_w} L_{oa} \tag{2.55}$$

where H_{Lw} and H_{Fw} are centroids above the water line of the frontal and lateral projected areas A_{F_w} and A_{L_w}, V_w represents the wind speed from (2.49), ρ_a is the air density, and

$$\gamma_w = \psi - \beta_w - \pi \tag{2.56}$$

where β is the wind direction in $\{n\}$. Blendermann (1994) gives the following expressions for the wind coefficients:

$$C_X(\gamma_w) = -CD_l \frac{A_{L_w}}{A_{F_w}} \frac{cos(\gamma_w)}{1 - \frac{\delta}{2}(1 - \frac{CD_l}{CD_t}) sin^2(2\gamma_w)} \tag{2.57}$$

$$C_Y(\gamma_w) = -CD_t \frac{sin(\gamma_w)}{1 - \frac{\delta}{2}(1 - \frac{CD_l}{CD_t}) sin^2(2\gamma_w)} \tag{2.58}$$

$$C_K(\gamma_w) = \kappa C_Y(\gamma_w) \tag{2.59}$$

$$C_N(\gamma_w) = \left(\frac{s_L}{L_{oa}} - 0.18(\gamma_w - \frac{\pi}{2})\right) C_Y(\gamma_w) \tag{2.60}$$

$$CD_l = CD_{l_{AF}}(\gamma_w) \frac{A_{F_w}}{A_{L_w}} \tag{2.61}$$

Where the coefficients $CD_{l_{AF}}$, CD_t, δ and γ are dependent on the vessel type. Assuming symmetricity with respect to the xz and yz planes, the wind coefficients for horizontal motions, C_X, C_Y and C_N, can be approximated by

$$C_X(\gamma_w) \approx -c_x cos(\gamma_w) \tag{2.62}$$

$$C_Y(\gamma_w) \approx c_y sin(\gamma_w) \tag{2.63}$$

$$C_N(\gamma_w) \approx c_n sin(2\gamma_w) \tag{2.64}$$

In Fossen (2011), the following approximations are presented:

$$c_x \in \{0.5, 0.9\} \tag{2.65}$$

$$c_y \in \{0.7, 0.95\} \tag{2.66}$$

$$c_n \in \{0.05, 0.2\} \tag{2.67}$$

Considering a vessel moving at a forward speed, we introduce the relative wind speed V_{rw} and angle of attack γ_{rw}:

$$V_{rw} = \sqrt{u_{rw}^2 + v_{rw}^2} \tag{2.68}$$

$$\gamma_{rw} = -atan2(v_{rw}, u_{rw}) \tag{2.69}$$

where the relative velocities are

$$u_{rw} = u - V_w cos(\beta_w - \psi) \tag{2.70}$$

$$v_{rw} = v - V_w sin(\beta_w - \psi) \tag{2.71}$$

By redefining (2.50)-(2.55), we can model the wind forces on the given vessels as

$$\boldsymbol{\tau}_{wind} = \frac{1}{2}\rho_a V_{rw}^2 \begin{bmatrix} C_X(\gamma_{rw})A_{Fw} \\ C_Y(\gamma_{rw})A_{Lw} \\ C_Z(\gamma_w)A_{Fw} \\ C_K(\gamma_w)A_{Lw}H_{Lw} \\ C_M(\gamma_w)A_{Fw}H_{Fw} \\ C_N(\gamma_{rw})A_{Lw}L_{oa} \end{bmatrix} \tag{2.72}$$

To use (2.72), we need to estimate the frontal and lateral projected areas of the vessels considered, in addition to performing experiments with the values in (2.65)-(2.67) during implementations. The air density can be modelled as a function of the temperature, but in this prototype, it will be represented by a constant value in order to simplify implementation.

2.7.2 Current Forces and Moments

2.7.2.1 Modeling

The forces and moments due to ocean currents can be implemented by replacing the generalized velocity vector with relative velocities as described in (2.11). Ocean currents are often assumed irrotational and constant in $\{n\}$. For a two-dimensional constant, irrotational current, a model can represented as:

$$\boldsymbol{v}_c^n = \begin{bmatrix} V_c\cos(\beta_c) \\ V_c\sin(\beta_c) \end{bmatrix} \tag{2.73}$$

which can be transformed to $\{b\}$ using the Euler angle rotation matrix:

$$\boldsymbol{v}_c^b = \boldsymbol{R}(\psi)\boldsymbol{v}_c^n = \begin{bmatrix} V_c\cos(\beta_c - \psi) \\ V_c\sin(\beta_c - \psi) \end{bmatrix} \tag{2.74}$$

where V_c is the current speed and β_c is the current *sideslip angle.* In computer simulations, the ocean current velocity and direction can be generated by the use of a first-order Gauss-Markov process:

$$\dot{V}_c + \mu_V V_c = w_V \tag{2.75}$$

$$\dot{\beta}_c + \mu_\beta \beta_c = w_\beta \tag{2.76}$$

$$V_{min} \leq V_c(t) \leq V_{max} \tag{2.77}$$

$$\beta_{min} \leq \beta_c(t) \leq \beta_{max} \tag{2.78}$$

where w_V and w_β are Gaussian white noises, and μ_V and μ_β are design constants. The relationship between the current speed V_c, current sideslip angle β_c and current angle of attack γ_c relative to the bow is visualized in Figure 2.7.

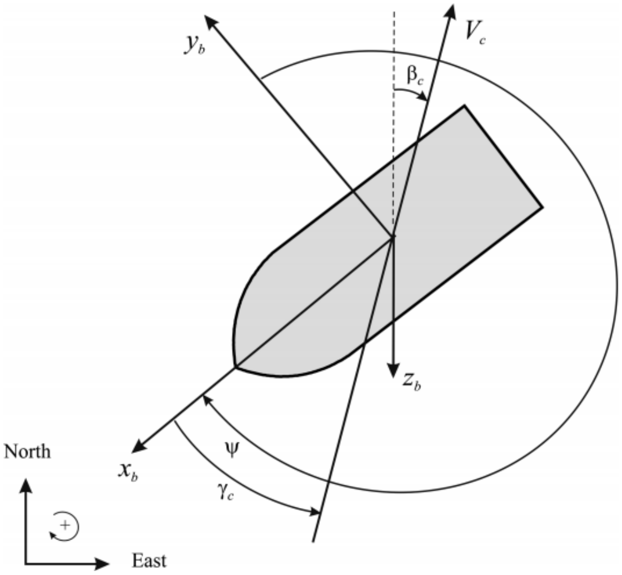


Figure 2.7: *Current speed V_c, current sideslip angle β_c and current angle of attack γ_c relative to the bow. Courtesy of Fossen (2011).*

The forces and moments due to ocean currents will in this thesis be implemented by the use of (2.12), which is obtained under the assumption of irrotational and constant ocean currents in $\{n\}$. As a result of this, the ocean currents will be implemented as irrotational, with slowly varying speed and constant direction.

2.7.3 Wave Forces and Moments

Ocean waves have an irregular, stochastic nature, and must be modeled accordingly, which motivates the use of energy spectra in the representation. The linear wave forces and moments are purely oscillatory, while higher order wave forces have magnitudes proportional to the square (or higher order) of the wave amplitudes. Second order wave effects include mean loads, slowly varying loads, and rapidly varying loads (Fossen (2011)).

2.7.3.1 Modeling

According to Faltinsen (2005), we can approximate the second-order wave force as a summation of second-order transfer functions of different frequency wave components:

$$\tau^{i}_{wave^2} = \sum_{j=1}^{N}\sum_{i=1}^{N} A_j A_k [T^{ic}_{jk} cos((\omega_k - \omega_j)t + (\epsilon_k - \epsilon_j)) + T^{is}_{jk} sin((\omega_k - \omega_j)t + (\epsilon_k - \epsilon_j))] \tag{2.79}$$

where ω_i represents the wave frequencies, ϵ_i is random phase angles, N is the number of wave components considered, $A_i = \sqrt{2S(\omega_i)\Delta\omega}$ is the wave amplitudes determined from the wave spectrum $S(\omega)$, and T^{ic}_{jk} and T^{is}_{jk} can be interpreted as second order transfer functions for the difference frequency loads (Faltinsen (2005)). A frequently used wave spectrum for open sea conditions, is the ITTC/ISSC-spectrum:

$$S(\omega) = \frac{A}{\omega^5} \cdot exp(-\frac{B}{\omega^4}) \tag{2.80}$$

$$A = 0.31 \cdot H_s^2 \cdot \omega_p^4 \tag{2.81}$$

$$B = 1.25 \cdot \omega_p^4 \tag{2.82}$$

where H_s denotes the significant wave height, and ω_p is the peak wave frequency.

2.7.3.2 Vessel Interaction

The wave-induced forces and moments τ_{wave} can be realized by defining the significant wave height H_s and the average zero-crossing wave period T_z for the simulation. This is then used as parameters in the chosen wave spectrum, $S(\omega)$, which is used in (2.79) to calculate the forces and moments. The use of response amplitude operators (RAOs) in (2.79) requires that the RAO tables are computed using a hydrodynamic program, as the wave forces depends on the geometry of the craft. According to Fossen (2011), an approximated model of the wave-induced forces and moments can be implemented by adding the wave-frequency motion to the vessel motion, which gives a total motion $\boldsymbol{y}$:

$$\boldsymbol{y} = \boldsymbol{\eta} + \boldsymbol{\eta}_{\omega} \tag{2.83}$$

From Pierson and Moskowitz (1963) we see that the average wave period T_{ω} within feasible weather conditions is roughly $1 < T_{\omega} < 6$ seconds. For a prototype implementation, the simulation of wave forces may then be implemented by using the wave spectrum to model the sea surface elevation over time, and representing the wave forces as a sinusoidal change in position, dependent on the sea surface elevation $\zeta(t)$ (Sørensen (2013)):

$$\zeta(t) = \sum_{n=1}^{N} \sqrt{2 \cdot S(\omega_n) \cdot \Delta(\omega)} \cdot sin(\omega_n \cdot t + \epsilon_n) \tag{2.84}$$

Chapter 3

Modeling of the Considered Vessels

This chapter aims to give an overview of the two vessels considered, including a description of the intended usage, as well as the actuators and sensors each vessel is equipped with. Furthermore, an overview of the methods used to obtain mathematical models of the vessels is presented, along with the resulting models.

3.1 Jolner

During the summer of 2016, Kongsberg Maritime and the Norwegian Defence Research Establishment (FFI) initiated a student project named Survey Explorer. The project team consisted of 6 students from the Norwegian University of Science and Technology and the Georgia Institute of Technology, and spanned over 8 weeks. The goal of the project was to develop and equip an autonomous Unmanned Surface Vehicle (USV) for seabed mapping, using a state-of-the-art multi-beam echo sounder from Kongsberg Maritime. The vessel used, 'Jolner', is a $1.65m$ long research boat delivered by DeepOcean, equipped with a moon pool for mounting underwater equipment, and a radio link for remote control and communication with land.

Possible applications for this product could be related to harbor surveillance systems and seabed mapping among others. Throughout the summer, the goals of the project were accomplished, and thus, more ambitious goals will follow in the years to come.

Jolner is equipped with two electric DC motor thrusters, mounted in the aft of the vessel, both producing a force F_x in the x-direction needed for transit. As the vessel does not make use of a rudder for steering, the yaw moment used for steering control is produced as a result of the difference in force produced from the two thrusters.

Figure 3.1: *The team working on Jolner in the Survey Explorer 2016 summer project of Kongsberg Maritime and FFI. From left: Jørgen Apeland, Even Ødegaard, Rune Nordmo, Mariusz Eivind Grøtte, Peder Aaby, Kjetil Børs-Lind. Courtesy of Kongsberg Gruppen.*

3.2 Odin

In 2015, the Norwegian Department of Defence initiated concept studies considering future marine systems for mine sweeping and clearance. One of the key properties of such a system would be to benefit from autonomous and modular systems, while still maintaining a low cost. As a result of this, FFI acquired the vessel 'Odin', to be used as a development platform for both mine sweeping and, in turn, other applications related to autonomy. The vessel is a 10.9m long water jet powered boat. In the years to come, Kongsberg Maritime and FFI will cooperate in the development of Odin as a USV fit for mine counter measures.

Odin uses two HamiltonJet water jets for propulsion, both located in the aft of the vessel. The water jet system is able to change the angle of the thrust vector by control of the nozzle direction and the *astern deflector*. The yaw moment used for steering control is also produced by the water jets.

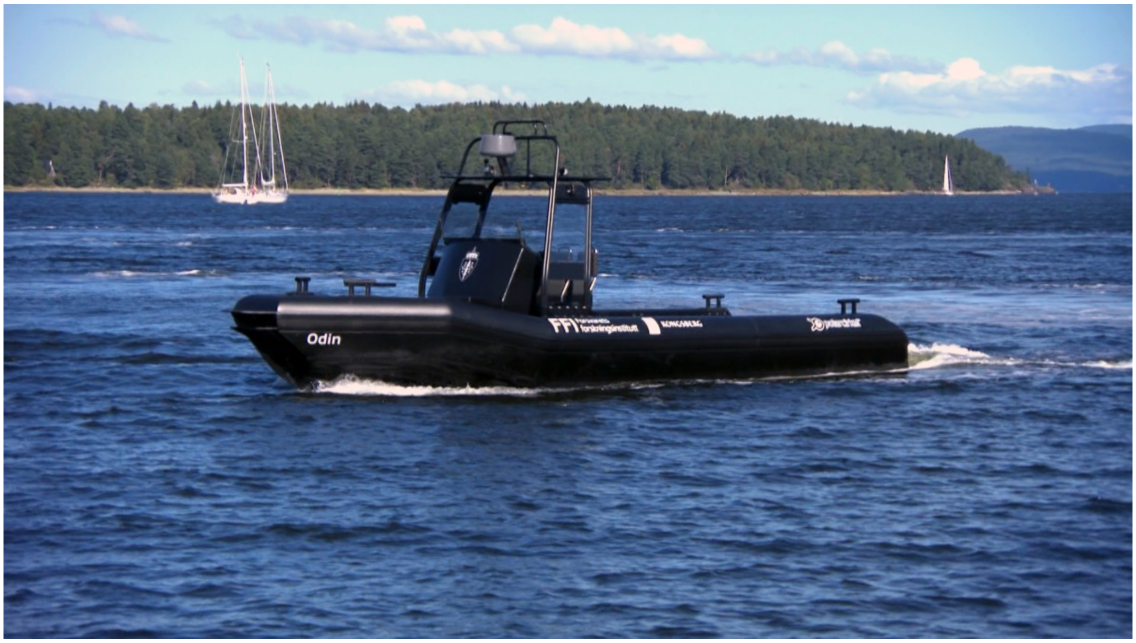


Figure 3.2: *The USV Odin. Courtesy of Philip Hofgaard, NRK.*

3.3 Sensors

In Figure 3.3, the overall layout of sensors and system architecture for Odin is visualized. The layout of sensors and system architecture for Jolner is still a work in progress, but it will likely be quite similar to the one in Figure 3.3. The control system on Odin utilizes Kongsberg's Navigation Processing Suite (NavP), included in *Navigation computer* in Figure 3.3. For this project, NavP will be considered as a sensor providing navigation data. The considered simulated sensors are then the wind sensor, speed sensor, GPS and NavP.

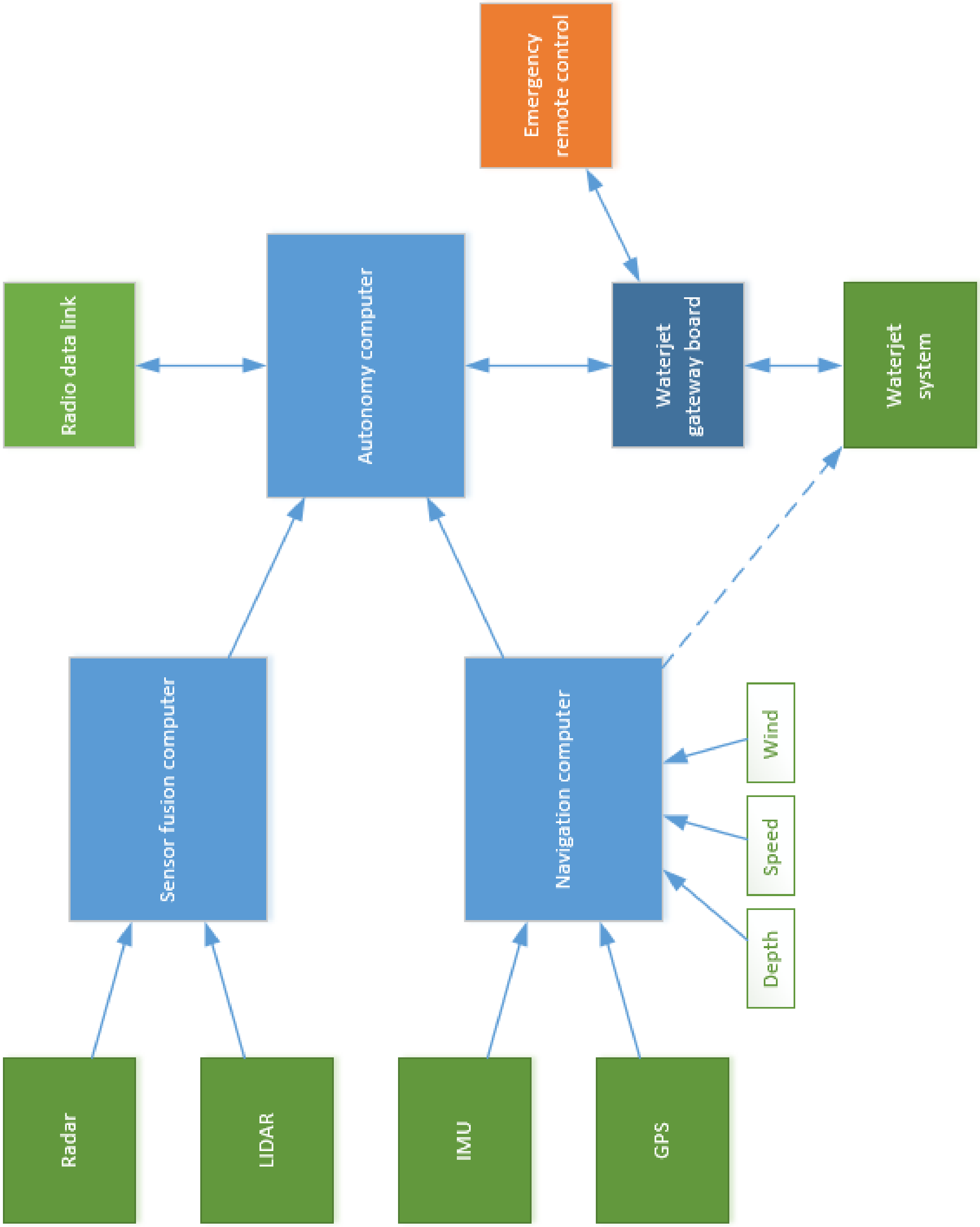


Figure 3.3: *Overview of system architecture and network layout on Odin, including sensors and processing units. Courtesy of Kongsberg Maritime.*

3.3.1 Wind Sensor

For measuring the relative wind speed, an *anemometer* is used, often including a vane, being able to measure the relative wind direction. Assuming the wind sensor is not connected to any other instruments, the wind sensor will output the relative wind speed and direction.

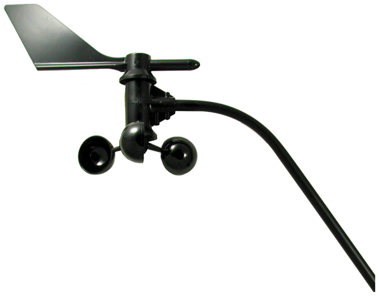


Figure 3.4: *Traditional anemometer with vane. Courtesy of http://www.davisnet.com.*

3.3.2 Speed Sensor

The speed sensor could work in a variety of ways, measuring either the relative speed (speed through water), the speed over ground, or both. As the GPS will provide the speed over ground, it seems reasonable to assume that the speed sensor considered in Figure 3.3 measures the speed through water, as the combination of the two measurements would provide estimates on the current speed and direction, given by the difference of the speed over ground and the speed through water. Sensors for measuring speed through water ranges from simple paddle wheels, measuring only the surge velocity, to highly advanced *Doppler Velocity Log*s (DVL), measuring velocity in both surge and sway.

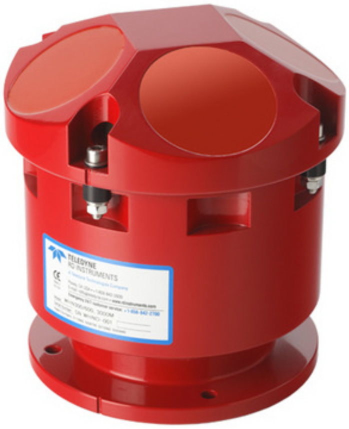


Figure 3.5: *Doppler Velocity Log. Courtesy of Teledyne.*

3.3.3 NavP

This software is used for inertial navigation. The software uses sensor data from the Inertial Measurement Unit (IMU) and GPS as input, processes the sensor data, and outputs filtered navigation data. The GPS is used to provide an initial position and heading, as well as aiding the filter in NavP, while the IMU provides intertial measurements to the filter.

3.3.3.1 IMU

Figure 3.6: *The HG-9848 IMU used on Odin. Courtesy of Honeywell.*

The Inertial Navigation System (INS) on Odin will estimate position, velocity and attitude using data from the IMU. The HG9848 IMU used on Odin consists of three digital ring laser gyroscopes and a 3-axis accelerometer, providing extremely accurate measurements of acceleration and angular velocity. As the IMU data is processed by NavP to obtain filtered navigation data, the IMU itself will not be simulated.

3.3.4 GPS

The GPS used on Odin combines GNSS signals and IMU data, providing high accuracy heading, position, heave, roll and pitch measurements. The resulting data is transmitted via standard NMEA 0183 protocol (Langley (1995)) for interpretation by the control system, describing among others latitude, longitude, height, heading, track and speed of the vessel. The GPS data is used to aid the filter in NavP, but could also be used as a standalone sensor, and thus the GPS will be simulated.

3.4 Available Parameters for Jolner and Odin

Although a full system identification is not performed for the vessels considered, we still have some parameters available:

	Odin:	**Jolner:**
Length between perpendiculars (L_{pp}):	10.5 m	1.65 m
Mass (m):	6000 kg	114 kg
Position of actuators (x_t, y_t, z_t):	$(-4.5, 0.57, 0.3)$,	$(-0.825, 0.1505, 0.2)$,
	$(-4.5, -0.57, 0.3)$	$(-0.825, -0.1505, 0.2)$
Draft height T:	0.7 m	0.2 m
Beam B:	3.5 m	0.695 m
Center of gravity (x_g, y_g, z_g):	$(-1.5, 0, -0.08)$	$(-0.2, 0, 0)$
Center of buoyancy (x_b, y_b, z_b):	N/A	$(-0.25, 0, 0.1)$

Table 3.1: *Available parameters for Odin and Jolner*

Using the information available in Table 3.1 along with methods described in Chapter 2, the mathematical models of the vessels considered are estimated as described in this section.

3.5 Inertia

Odin

In calculations performed by FFI, the mass m, moments of inertia I_{ij} and center og gravity (x_g, y_g, z_g) is estimated. Using this information, the rigid-body mass matrix $\boldsymbol{M}_{RB}$ can be calculated as described in (2.15), resulting in the following:

$$\boldsymbol{M}_{RB} = \begin{bmatrix} 6000 & 0 & 0 & 0 & -480 & 0 \\ 0 & 6000 & 0 & 480 & 0 & -9000 \\ 0 & 0 & 6000 & 0 & 9000 & 0 \\ 0 & 480 & 0 & 3066.5 & -21.13 & -519.34 \\ -480 & 0 & 9000 & -21.13 & 18992.13 & -1.21 \\ 0 & -9000 & 0 & -519.34 & -1.21 & 20927.7 \end{bmatrix} \tag{3.1}$$

As no effort has been done to identify the added mass parameters, a reasonable approach would be to use the parameters presented in Kjerstad (2010), which considers a vessel of similar dimensions utilizing the same hull type, and scale these according to the current vessel using the Bis system, shown in Table 2.3. This approach results in the following added mass matrix:

$$\boldsymbol{M}_A = \begin{bmatrix} 527.36 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2128.05 & 0 & 1850.18 & 0 & 700.68 \\ 0 & 0 & 16458.80 & 0 & 6537.05 & 0 \\ 0 & 1475.70 & 0 & 5435.66 & 0 & 506.83 \\ 0 & 0 & 13302.00 & 0 & 94768.52 & 0 \\ 0 & 632.29 & 0 & 272.42 & 0 & 13204.24 \end{bmatrix} \tag{3.2}$$

Jolner

Using the mass and center of gravity presented in Table 3.1, estimates are found for the matrices $m\boldsymbol{I}_{3x3}$, $m\boldsymbol{S}(\boldsymbol{r}_g^b)$ presented in (2.14). To estimate the matrix $\boldsymbol{I}_b$, the parameters from Kjerstad (2010) will be scaled, using the Bis system, resulting in the following mass matrix:

$$\boldsymbol{M}_{RB} = \begin{bmatrix} 114.1 & 0 & 0 & 0 & 11.4 & 0 \\ 0 & 114.1 & 0 & -11.4 & 0 & -22.8 \\ 0 & 0 & 114.1 & 0 & 22.8 & 0 \\ 0 & 11.4 & 0 & 5.1224 & 0 & 0 \\ 11.4 & 0 & 22.8 & 0 & 23.34 & 0 \\ 0 & -22.8 & 0 & 0 & 0 & 22.99 \end{bmatrix} \tag{3.3}$$

As no added mass parameters are identified, the same approach as for Odin is chosen for Jolner, using the added mass parameters presented in Kjerstad (2010), and scaling these according to the current vessel using the Bis-system, resulting in the following added mass matrix:

$$\boldsymbol{M}_A = \begin{bmatrix} 10.02 & 0 & 0 & 0 & 0 & 0 \\ 0 & 40.43 & 0 & 5.52 & 0 & 2.09 \\ 0 & 0 & 312.72 & 0 & 19.52 & 0 \\ 0 & 4.41 & 0 & 2.55 & 0 & 0.24 \\ 0 & 0 & 39.72 & 0 & 44.46 & 0 \\ 0 & 1.89 & 0 & 0.13 & 0 & 6.2 \end{bmatrix} \tag{3.4}$$

3.6 Coriolis and Centripetal Forces

Using the information available in (3.1), (3.3) and Table 3.1, the rigid body Coriolis and centripetal matrix C_{RB} for both vessels can be implemented using (2.18). Using the parameters estimated in (3.2) and (3.4), the added mass Coriolis and centripetal matrix C_A for both vessels can be implemented using (2.19).

3.7 Damping Forces

Odin

As there has not been performed a thorough system identification on Odin, the linear damping terms is scaled from the Viknes-vessel presented in Kjerstad (2010), using the Bis system, which should give a rough approximation, as the vessels are of similar size and type. This approach results in the following linear damping matrix:

$$\boldsymbol{D}_L = \begin{bmatrix} 266.26 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1075.90 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 178.06 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6728.50 \end{bmatrix} \tag{3.5}$$

Kjerstad (2010) presented two different values for the linear yaw damping parameter N_r, one computed by the hydrodynamical software ShipX, and one fitted to recorded data from sea trials. In this conversion, the data computed by the hydrodynamical software is used.

Jolner

Using the same approach for Jolner results in the following linear damping matrix:

$$\boldsymbol{D}_L = \begin{bmatrix} 12.76 & 0 & 0 & 0 & 0 & 0 \\ 0 & 51.57 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.21 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 13.19 \end{bmatrix} \tag{3.6}$$

while the linear 3 DOF damping matrix for Jolner estimated in Ødegaard (2016) is:

$$\boldsymbol{D}_L = \begin{bmatrix} 4.48 & 0 & 0 \\ 0 & 4.51 & 9.57 \\ 0 & -1.72 & 13.73 \end{bmatrix} \tag{3.7}$$

The surge and sway damping parameters presented in Ødegaard (2016) are scaled from the CyberShip Enterprize II parameters presented in Skjetne et al. (2004), which considers a displacement vessel. Since Jolner utilizes a planing hull, these parameters are likely not directly scaleable. As a result of this, the linear damping matrix in (3.6) will be used in the simulator. The linear damping matrices in (3.5) and (3.6) are simplifications of (2.26), containing only the most important linear damping terms.

3.7.1 Nonlinear Surge Damping

Odin

In the prototype simulator, the nonlinear surge damping parameter $X_{|u|u}$ is calculated using (2.27)-(2.30). In Fossen (2011), it is suggested that the form factor $k = 0.1$ is used under transit, and an estimate for the wetted surface S of the hull is found by using the Denny-Mumford equation (W. F. Durand and G. R. McDermott (1894)):

$$S = L(1.7H + C_B B) \tag{3.8}$$

By comparing results from sea trials and simulations, it is found that the residual friction coefficient $C_R = 0.014$, which results in the following nonlinear surge damping:

$$X_{|u|u} = 21487.33\left(\frac{0.075}{(log_{10} R_n - 2)^2} + 0.014\right) \tag{3.9}$$

Figure 3.7 features the simulated surge velocity using actuator data from an actual sea trial, compared with the surge velocity from said sea trial. The plot shows good vessel model performance up to a certain surge speed, where the semi-planing dynamics occur, which are not implemented in this prototype.

Jolner

Using the same approach for Jolner, the following nonlinear surge damping is estimated:

$$X_{|u|u} = 736.48\left(\frac{0.075}{(log_{10} R_n - 2)^2} + 0.014\right) \tag{3.10}$$

Figure 3.8 presents the nonlinear surge damping forces at different speeds for both Odin and Jolner.

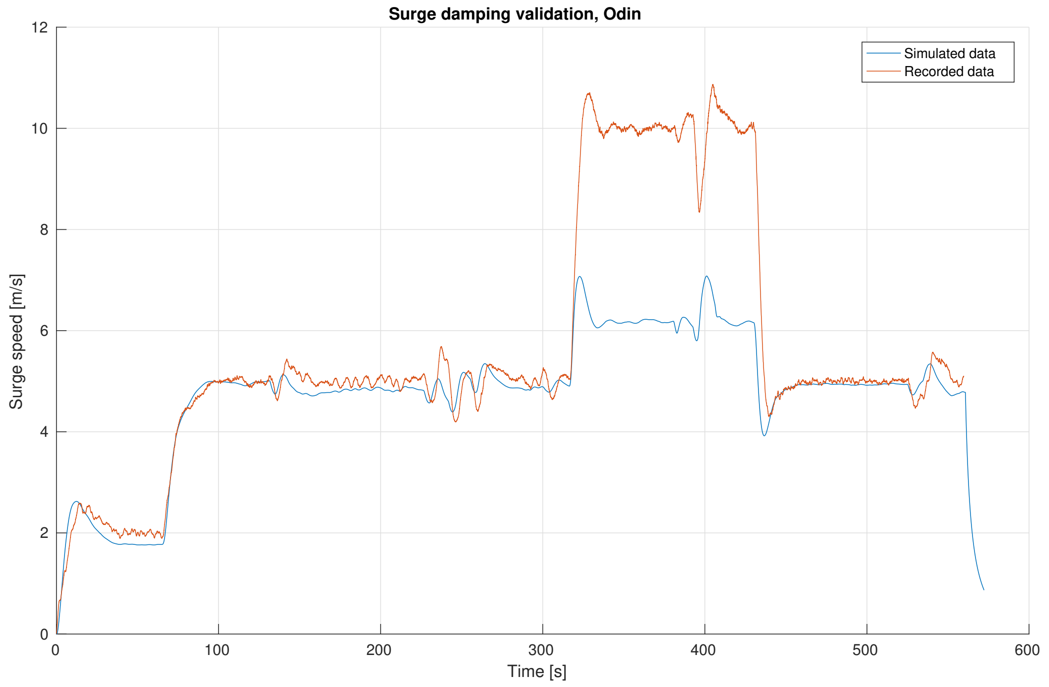


Figure 3.7: Validation of the surge speed of Odin using recorded vessel data.

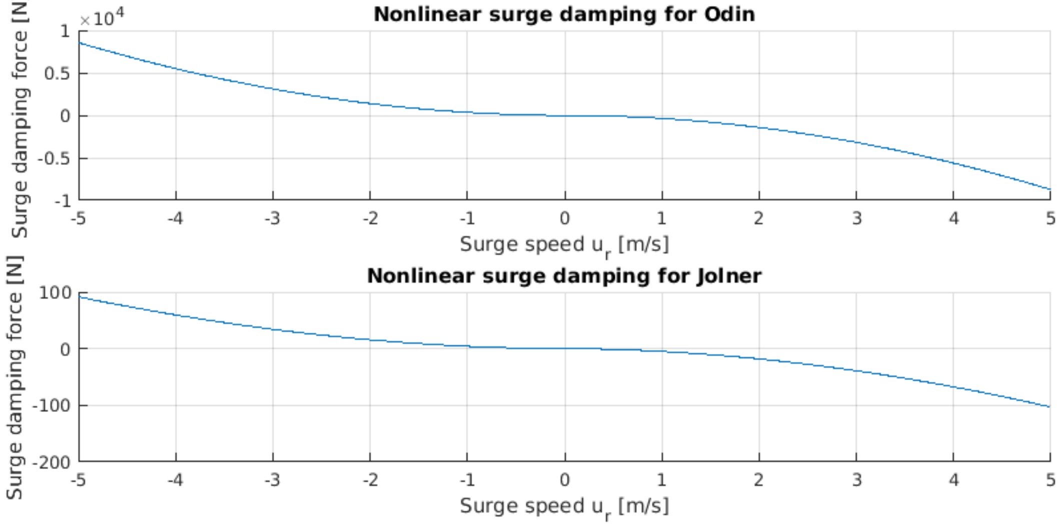


Figure 3.8: *Visualization of the nonlinear surge damping forces for different relative surge speeds u_r.*

3.7.2 Cross-flow Drag Principle

To calculate the nonlinear damping force in sway and the yaw moment, a constant 2-D current coefficient C_d^{2D} was estimated using Hoerner's curve, which is presented in Figure 2.3. Additionally, as no exact model of the draft $T(x)$ is available, it was assumed to be constant $T(x) = T$. Using this, the integrals presented in (2.31) and (2.32) are solved numerically in the simulator at each timestep. When fitting the model of Odin to the recorded data, the 2D current coefficient was adjusted to the following:

$$Y = -116.5 \int_{\frac{-L_{pp}}{2}}^{\frac{L_{pp}}{2}} |v_r + xr|(v_r + xr)dx \tag{3.11}$$

$$N = -233 \int_{\frac{-L_{pp}}{2}}^{\frac{L_{pp}}{2}} x|v_r + xr|(v_r + xr)dx \tag{3.12}$$

The resulting forces and moments are visualized in Figure 3.9.

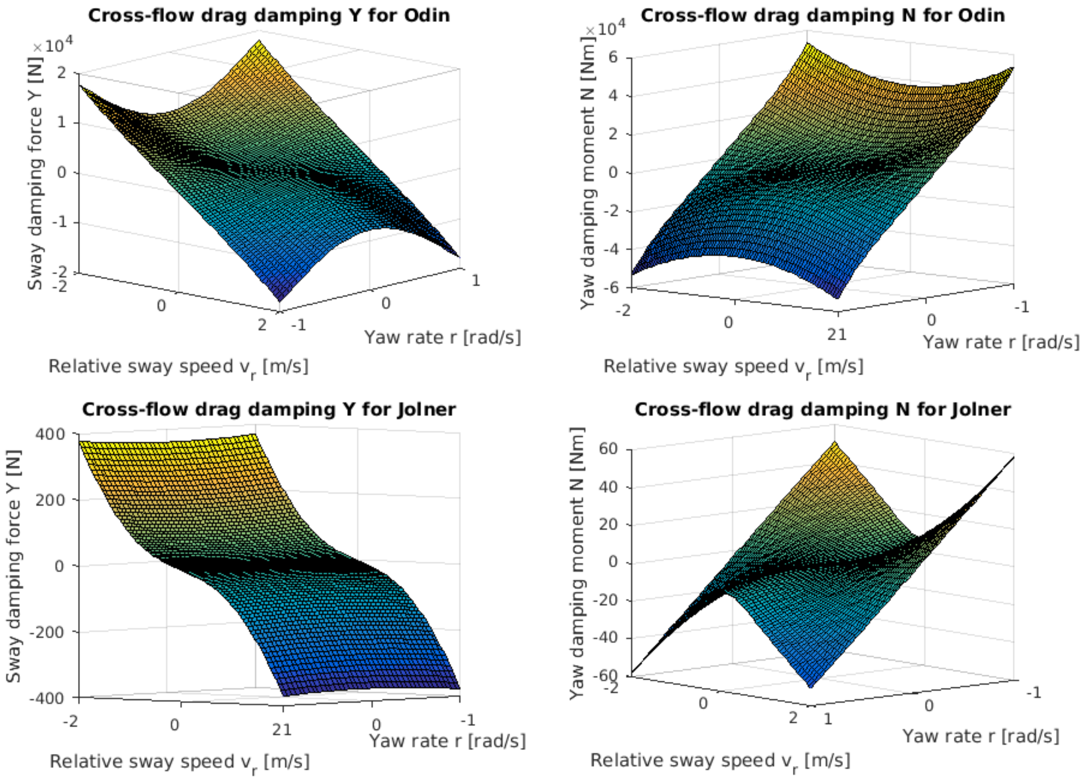


Figure 3.9: *Visualization of the cross-flow drag damping forces and moments for different relative sway speeds v_r and yaw rates r.*

Figures 3.10 and 3.11 features the simulated sway speed and yaw rate using actuator data from an actual sea trial, compared with the sway speed and yaw rate from said sea trial. The plots shows good vessel model performance, the model follows the recorded data closely, with some occasional deviations. The environmental conditions during sea trials are not known, and some deviations are expected as a result of this.

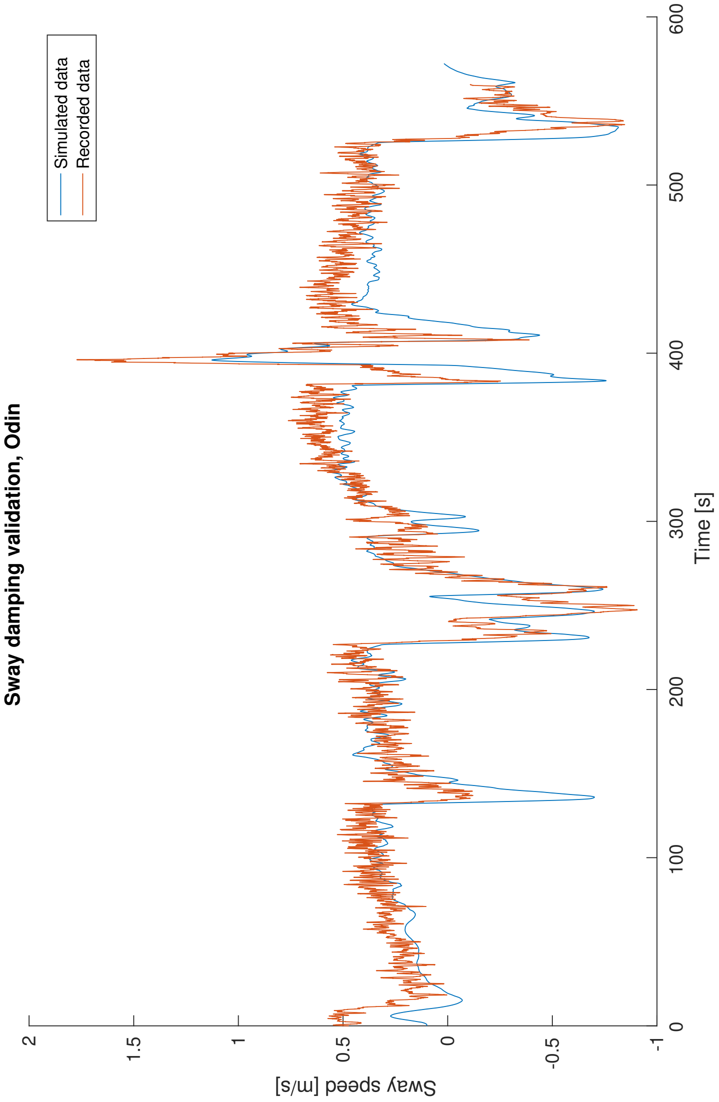


Figure 3.10: *Validation of the sway speed of Odin using recorded vessel data.*

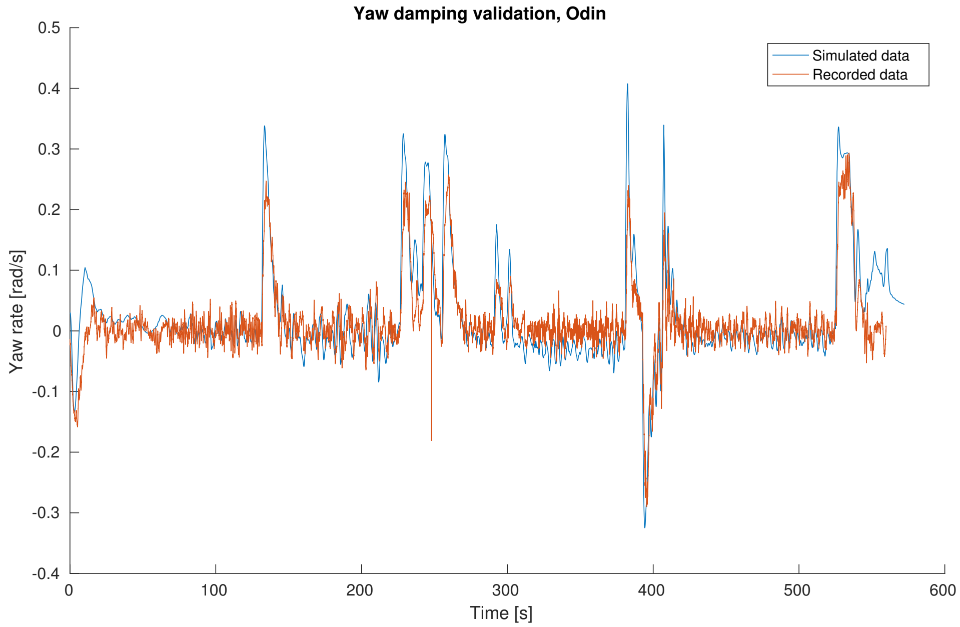


Figure 3.11: *Validation of the yaw rate of Odin using recorded vessel data.*

3.8 Restoring Forces

Odin

The restoring forces were scaled from the values presented in Kjerstad (2010) using the Bis system, resulting in the following restoring force matrix:

$$\boldsymbol{G} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 120237.27 & 0 & -10797.81 & 0 \\ 0 & 0 & 0 & 83341.50 & 0 & 0 \\ 0 & 0 & -10797.81 & 0 & 740195.25 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.13}$$

By using this approach, the restoring forces for the Viknes 830 vessel considered in Kjerstad (2010) are scaled to fit Odins dimensions. This does not necessarily represent the actual restoring forces for Odin, but acts as a rough approximation.

Jolner

Using the same approach for Jolner results in the following restoring force matrix:

$$\boldsymbol{G} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 14537.77 & 0 & -205.15 & 0 \\ 0 & 0 & 0 & 248.83 & 0 & 0 \\ 0 & 0 & -205.15 & 0 & 2210.01 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.14}$$

3.9 Fluid Memory Effects

Each of the elements in

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_u \\ \mu_v \\ \mu_w \\ \mu_p \\ \mu_q \\ \mu_r \end{bmatrix} \tag{3.15}$$

consists of the sum of the output of three independent linear systems, which results in a total of 18 independent linear systems representing the fluid memory effects. In the simulator, the fluid memory effects computed by the hydrodynamical software in Kjerstad (2010) are implemented, and the resulting output is scaled to internally to fit the simulated vessel at each timestep.

3.10 Actuators

The two vessels considered are equipped with quite different actuators, which are modeled through the use of different time constants and states, implemented in a separate module. By subscribing to different ROS-topics for each model, the module becomes versatile enough to handle both types of actuator models, as long as the control system for the simulated vessel publishes actuator messages to the given topic.

3.10.1 Jolner

From experiments performed with Jolner during the summer 2016, it is known that a propeller speed of $n = 0.35 \cdot n_{max}$ on both thrusters is sufficient to maintain a surge velocity of $1.2m/s$, and a propeller speed $n = n_{max}$ on both thrusters is sufficient to maintain a surge velocity of $3m/s$. These experiments were performed in calm weather, and are assumed to be unaffected by environmental forces. From information provided by *DeepOcean*, the supplier of the USV, we also know that $n_{max} \approx 3000RPM$. Using this information, an approximation of the propeller thrust as a function of RPM can be fitted to the simulation model.

By implementing a simplified speed controller in the simulation model, using the parameters presented in this chapter, results show that to maintain a steady speed of $1.2m/s$, a total force in surge of $25N$ is required, and to maintain a steady speed of $3.0m/s$, a total force in surge of $250N$ is required. Using (2.39), K can then be found, which gives the relationship between force and RPM shown in Figure 3.12. By estimating this through use of the simulator, a proper response in simulations is achieved, even though the results may deviate from the actual thruster dynamics. The thrusters used on Jolner are brushless DC motors, and will have a relatively fast dynamic response, and thus low time constants T_i (Yedamale (2003)). Through this approach, the following actuator model is found:

$$\dot{n}_i = -\frac{1}{T_{n_i}}(n_i - n_{i,r}) \tag{3.16}$$

$$\begin{bmatrix} X \\ Y \\ N \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ -l_{y_1} & -l_{y_2} \end{bmatrix} \begin{bmatrix} K_1 & 0 \\ 0 & K_2 \end{bmatrix} \begin{bmatrix} |n_1|n_1 \\ |n_2|n_2 \end{bmatrix} \tag{3.17}$$

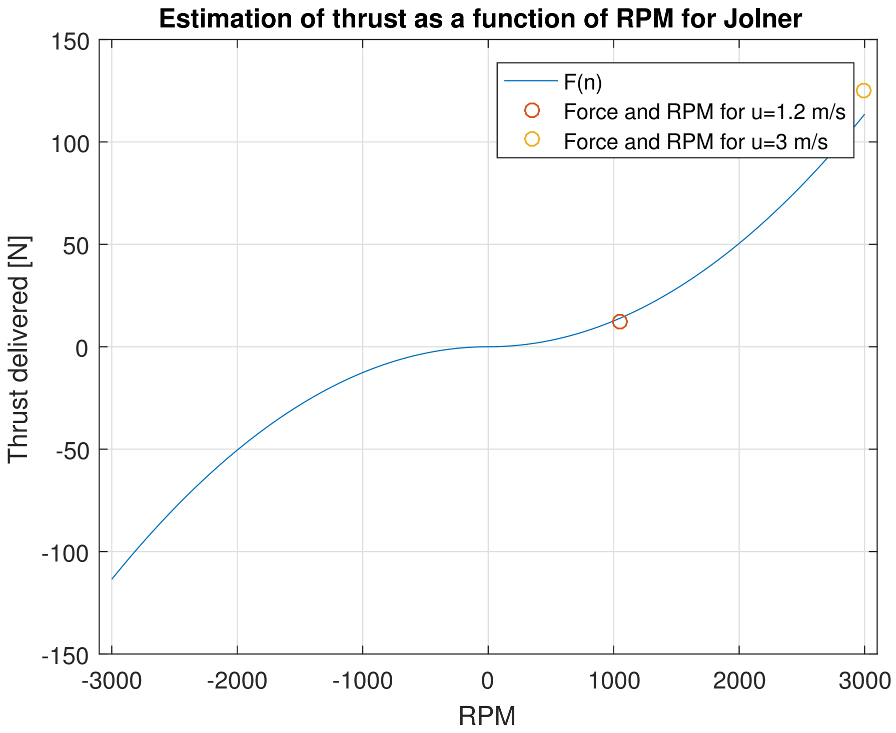


Figure 3.12: *The estimated relationship $F(n)$ between the RPM and force delivered by one thruster for Jolner.*

3.10.2 Odin

Odin is equipped with two water jet thrusters for propulsion, both located in the aft of the vessel. The resulting forces and moments are given by the RPM, nozzle direction and astern deflector for each thruster. The control system on Odin will output desired values for all these components, where the output range $[-100, 100]$ corresponds with the ranges defined below:

- **Engine RPM:** $[400, 2000]$ RPM.
- **Nozzle:** $[-27, 27]$ degrees.
- **Astern deflector:** [Full reverse position, Full forward position].

By control of the engine RPM, the output force from the thruster is scaled, as shown in Figure 3.13. The vessel speed affects the resulting forces, which in general lowers as the vessel speed increases. As the vessel models derived in this thesis are valid only for low speeds, the output force from the water jet thrusters are assumed independent of the vessel speed.

By assuming that the output forces are independent of the relative speed through water for a Froude number < 0.4, the relationship between force and RPM is estimated as shown in Figure 3.14, using the resulting actuator forces for $u_r = 2m/s$.

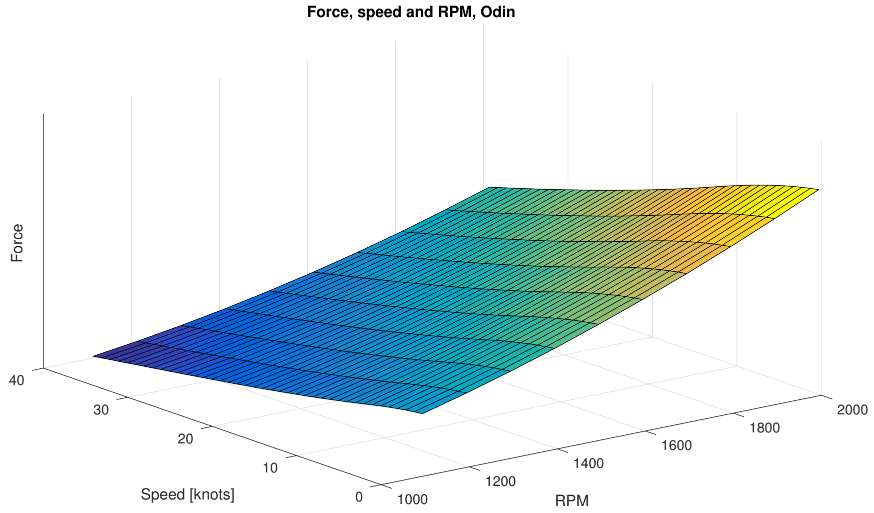


Figure 3.13: *Estimated relationship between RPM, speed and force for Odin. The resulting force is plotted without numerical data, on request by FFI, which is responsible for the estimates.*

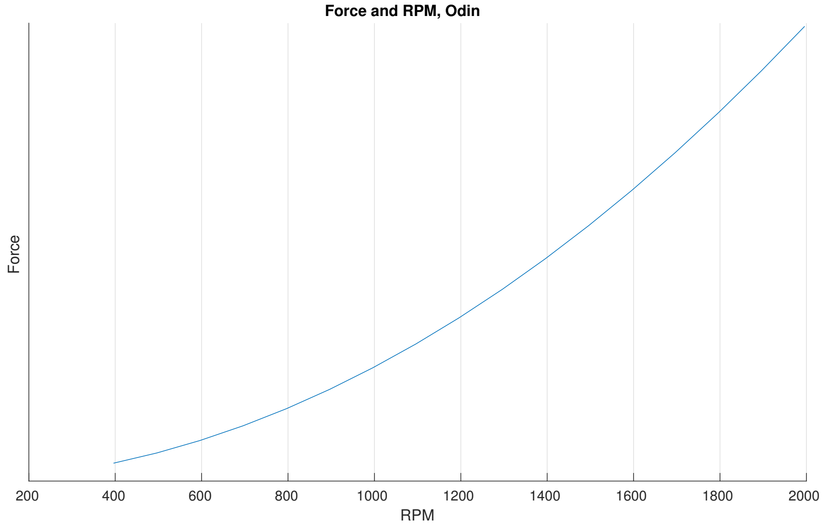


Figure 3.14: *Estimated relationship between RPM and force for Odin. The resulting force is plotted without numerical data, on request by FFI, which is responsible for the estimates.*

When the value for the astern deflector is set to 100, the deflector is open, and the water outflow is directed in the aft direction, resulting in a positive surge force. When the value reaches 0, the water outflow is divided into three components, one in the aft direction, and two in the fore direction, negating each other, and resulting in zero force. When the value reaches -100, the water outflow is directed in the fore direction, resulting in a negative surge force.

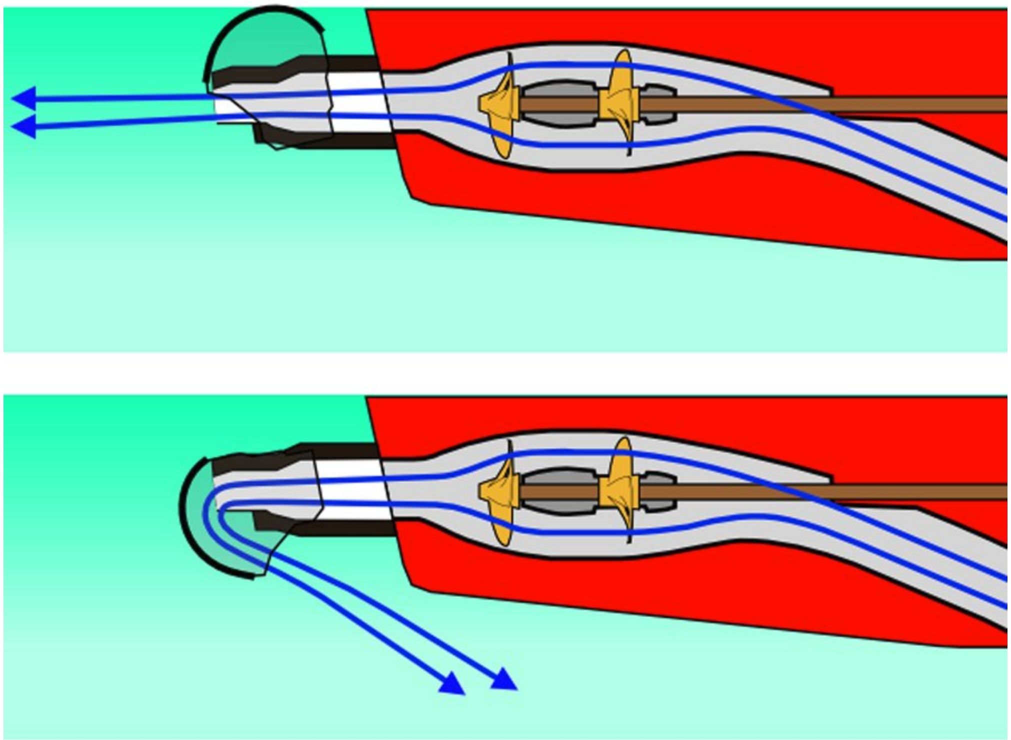


Figure 3.15: *Illustration of different astern deflector positions. Courtesy of* `http://www.hivets.com`.

By controlling the nozzle direction, the angle of the resulting force vector can be controlled, resulting in both forces in surge and sway, and moment in yaw. When the astern deflector is set to 0, the nozzle direction can still be used to control the yaw moment of the vessel, while the surge force will remain zero.

As no exact model of the relationship between force and astern deflector position exists, the control system on Odin mainly uses only three values for now, forward, zero force and reverse. In this prototype simulator, the relationship between the astern deflector and the output force is linearized around these three states, to provide an estimate of the astern deflector dynamics. Through experiments with the water jet thrusters performed by FFI, time constants for the engine RPM, astern deflectors and the jet nozzles are found, and by using these time constants in (2.38), the dynamics of the engine RPM, astern deflectors and jet nozzles are implemented, resulting in the step-responses shown in Figure 3.16. Using this approach, the following actuator model is found:

$$\dot{\alpha}_i = -\frac{1}{T_{\alpha_i}}(\alpha_i - \alpha_{i,r}) \tag{3.18}$$

$$\dot{\beta}_i = -\frac{1}{T_{\beta_i}}(\beta_i - \beta_{i,r}) \tag{3.19}$$

$$\dot{n}_i = -\frac{1}{T_{n_i}}(n_i - n_{i,r}) \tag{3.20}$$

$$\begin{bmatrix} X \\ Y \\ N \end{bmatrix} = \begin{bmatrix} \beta_1 cos(\alpha_1) & \beta_2 cos(\alpha_2) \\ sin(\alpha_1) & sin(\alpha_2) \\ l_{x_1} sin(\alpha_1) - l_{y_1} cos(\alpha_1) & l_{x_2} sin(\alpha_2) - l_{y_2} cos(\alpha_2) \end{bmatrix} \begin{bmatrix} K_1 & 0 \\ 0 & K_2 \end{bmatrix} \begin{bmatrix} |n_1|n_1 \\ |n_2|n_2 \end{bmatrix} \tag{3.21}$$

Where α represents the nozzle direction, β represents the astern deflector state, and n is the engine RPM.

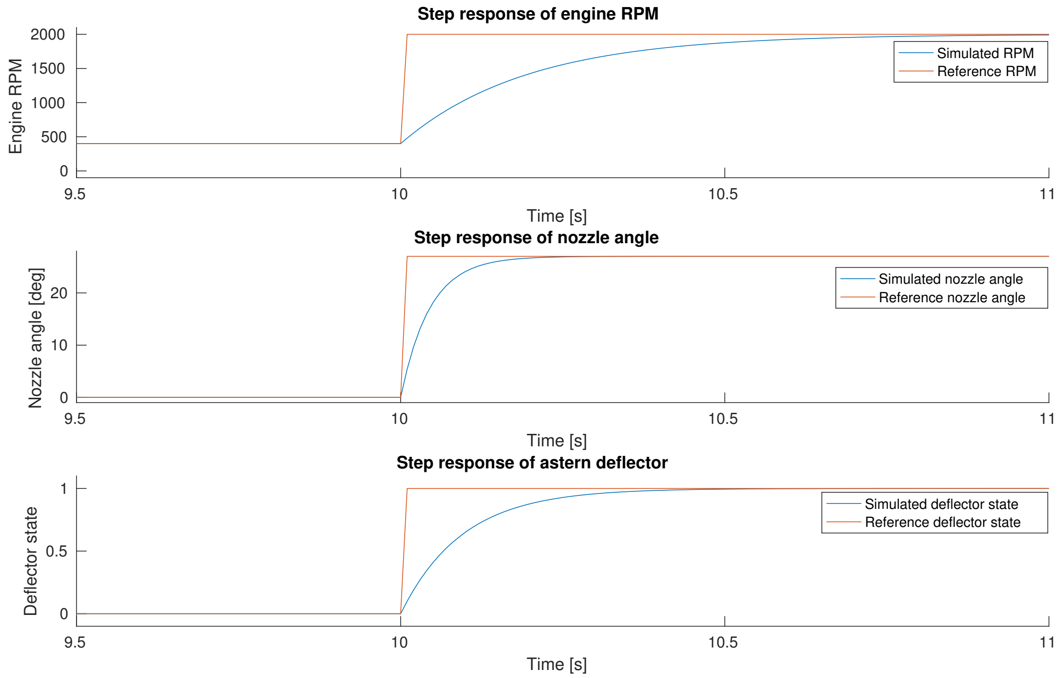


Figure 3.16: *Simulated step responses for engine RPM, astern deflector and jet nozzle angle on Odin.*

3.11 Modeling and Parametrization of Environmental Forces and Moments

This section presents the implementation and results of the environmental models presented in Chapter 2. The modeling of the disturbance itself is considered separately, followed by interaction with each of the vessels considered. All environmental models are implemented in C++, and data for both environmental states and corresponding forces is published to separate ROS-topics to facilitate for logging and visualization.

3.11.1 Wind Forces and Moments

3.11.1.1 Modeling

The mean and slowly varying wind velocity are modeled using a first order Gauss-Markov process, with saturation elements restricting the maximum and minimum velocity. The wind gust component is modeled using the Harris wind spectrum presented in (2.46), which gives the total wind realization presented in (2.49). To find the velocity at z metres above the surface, the total wind velocity is used in (2.43), using the vertical distance to the centroid of A_{LW}, s_L, as the height z. The wind direction is modeled with a slowly varying element around a mean direction, using a first-order Gauss-Markov process. Results from simulations of wind velocity and direction are shown in Figures 3.17 and 3.18.

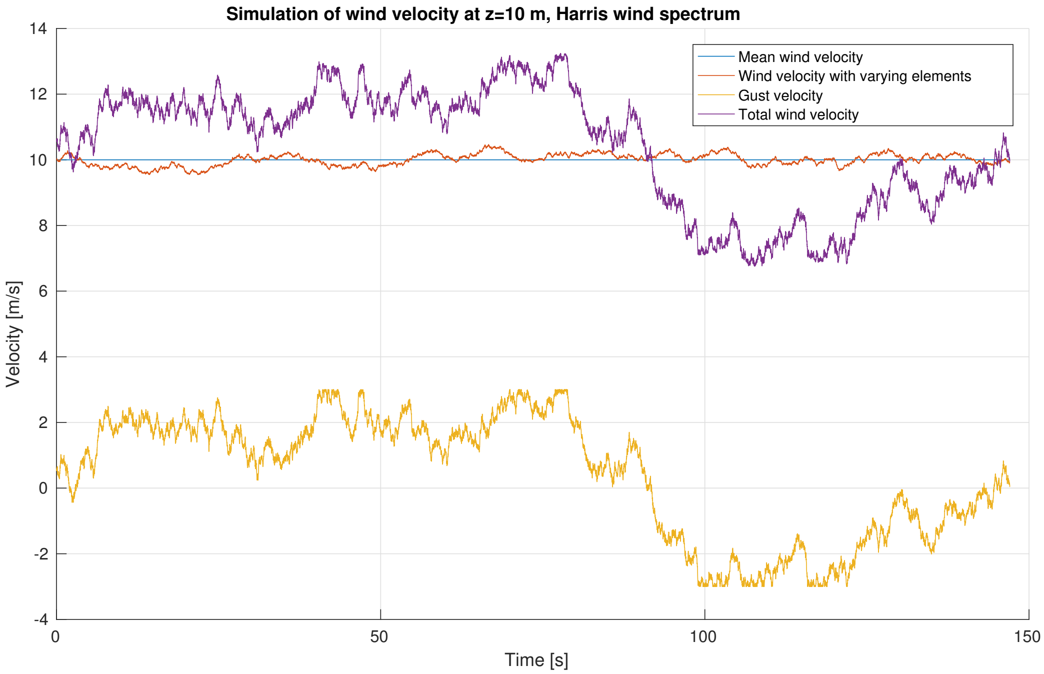


Figure 3.17: *Simulation of wind velocity over 150 seconds, using the Harris wind spectrum.*

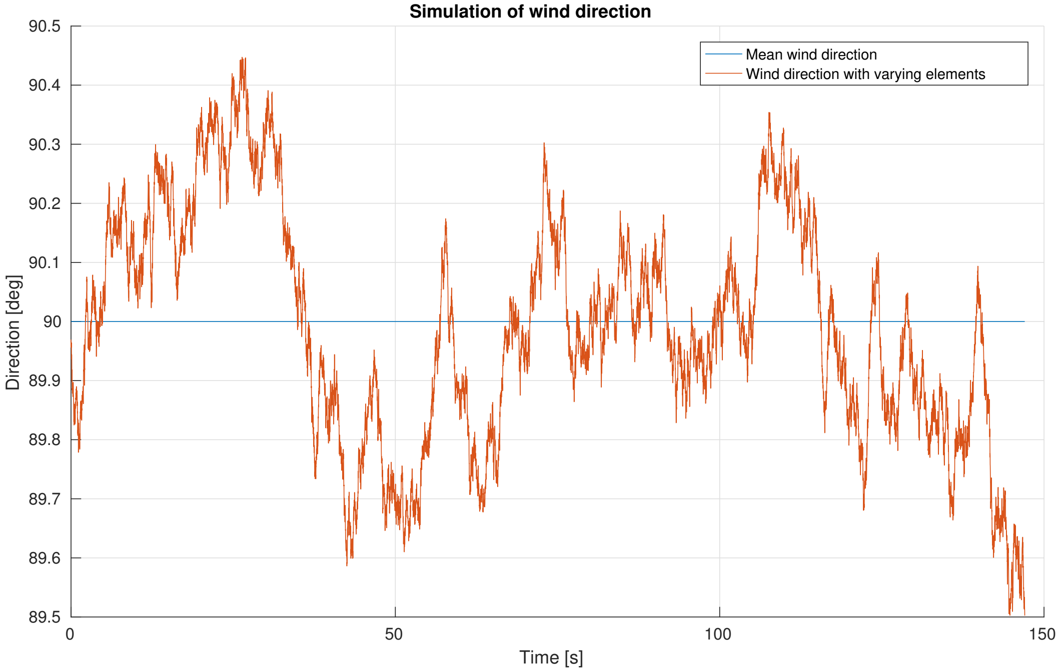


Figure 3.18: *Simulation of wind direction over 150 seconds.*

3.11.1.2 Vessel Interaction

In the simulator, a wind disturbance is implemented in three degrees of freedom, surge, sway and yaw, using the equations presented in Section 2.7.1. To calculate the approximate wind forces and moments acting on the vessel, the vessels' frontal and lateral projected areas and overall length are estimated. Figure 3.19 shows two images of Odin obtained from a 3D drawing of the vessel, which are then cut at the waterline and binarized.

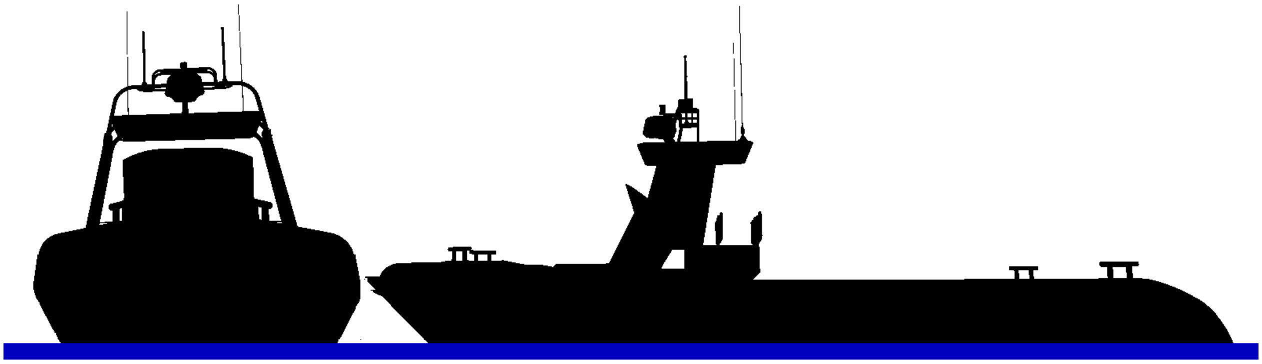


Figure 3.19: *Contours of Odin.*

Knowing the overall dimensions of the vessel, the pixel count can be used to estimate the frontal and lateral projected areas. Using this method, the following parameters are obtained:

$$A_{F_w} = 6.4m^2 \tag{3.22}$$

$$A_{L_w} = 11.31m^2 \tag{3.23}$$

$$s_H = 0.4m \tag{3.24}$$

$$s_L = 1.2m \tag{3.25}$$

As the wheelhouse is placed in the fore, the vessel is not symmetrical with respect to the yz plane, and the wind coefficients presented in (2.57)-(2.61) should be used, giving a nonsymmetrical wind profile. In Fossen (2011), wind characteristics for different vessel types are defined, giving values for the coefficients $CD_{l_{AF}}$, CD_t, δ and γ. The vessel Odin is assumed to have characteristics similar to a speed boat, which results in the wind coefficients presented in Figure 3.20.

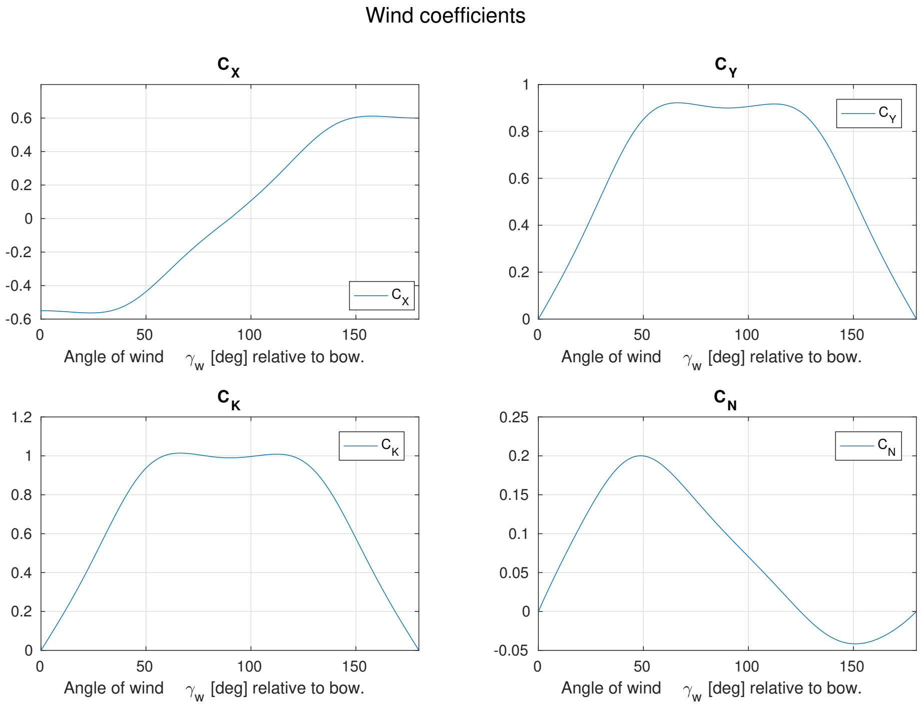


Figure 3.20: *Wind coefficients for Odin, based on generalized vessel type characteristics presented in Fossen (2011).*

Using these coefficients, the following wind vector is used in the simulator:

$$\boldsymbol{\tau}_{wind} = \frac{1}{2}\rho_a V_{rw}^2 \begin{bmatrix} C_X(\gamma_{rw})A_{F_w} \\ C_Y(\gamma_{rw})A_{L_w} \\ 0 \\ C_K(\gamma_{rw})A_{L_w}H_{L_w} \\ 0 \\ C_N(\gamma_{rw})A_{L_w}L_{oa} \end{bmatrix} \tag{3.26}$$

As the vessel Jolner is quite heavy loaded, the hull section above the waterline is in average ≈ 5 cm high, in addition to the GPS antenna. For this implementation, the wind forces and moments influencing Jolner will be assumed to be negligible, as the prevailing forces and moments will be given by the actuators and hydrodynamics of the vessel.

3.11.2 Current Forces and Moments

3.11.2.1 Modeling

The current velocity and direction are modeled using a first-order Gauss-Markov process, as described in (2.75)-(2.78). This gives a slowly varying current direction and velocity, with saturation elements restricting the minimum and maximum values. Results from simulations of current velocity and direction are shown in Figures 3.21 and 3.22.

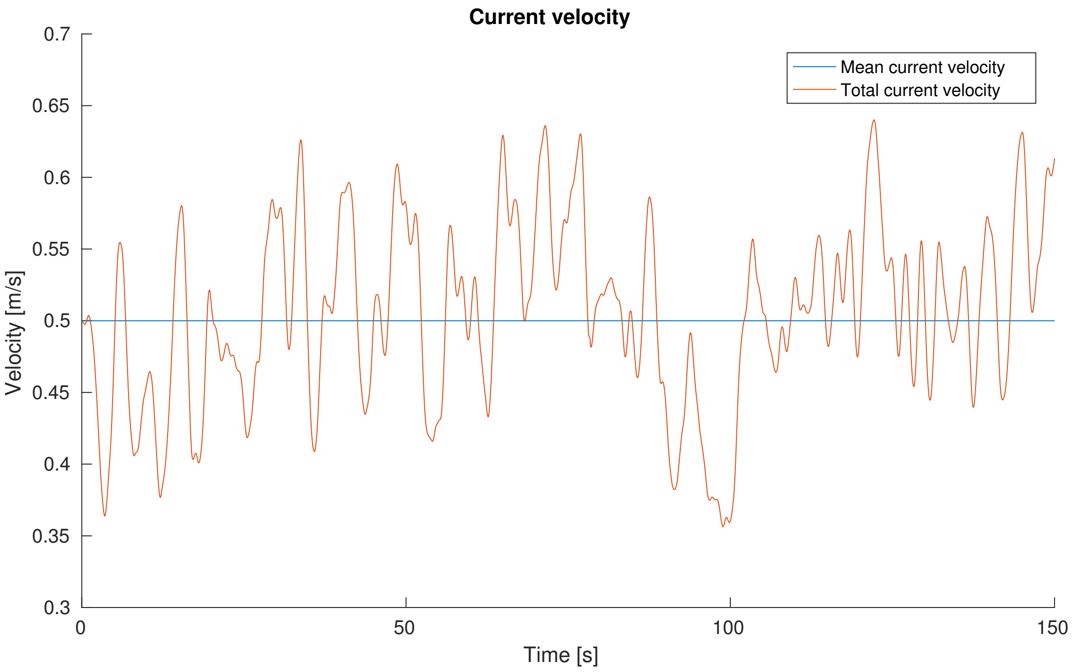


Figure 3.21: *Simulation of current velocity over 150 seconds.*

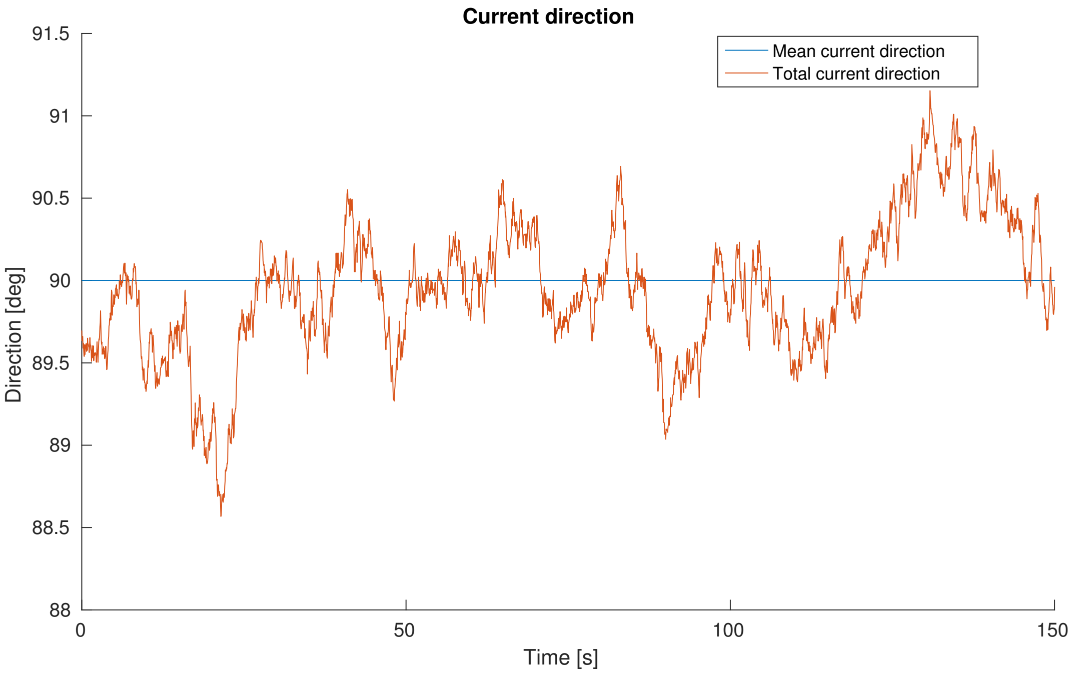


Figure 3.22: *Simulation of current direction over 150 seconds.*

3.11.2.2 Vessel Interaction

The forces and moments due to ocean currents are implemented by introducing relative velocities, as described in Section 2.7.2.1. As the simulated ocean current is irrotational and slowly varying in $\{n\}$, this is a good approximation of the current forces and moments, according to Fossen (2011).

3.11.3 Wave Forces and Moments

3.11.3.1 Modeling

As the RAO tables for the vessels considered are not computed, the approach presented in Section 2.7.3.1 is not directly applicable in this case. Fossen (2011) presents a simplified approach to the modeling of wave induced forces and moments, directly interacting with the vessel, as described in the next section. Using this approach, the heave position of the vessel is assumed to correspond with the sea surface elevation. An example of the sea surface elevation using a significant wave height $H_s = 2.5m$ is shown in Figure 3.23. The simulation results can then be controlled by use of the following relationship:

$$H_s = 4\sqrt{var(\zeta)} \tag{3.27}$$

Using the results presented in Figure 3.23, this gives a calculated significant wave height $H_s = 2.556m$.

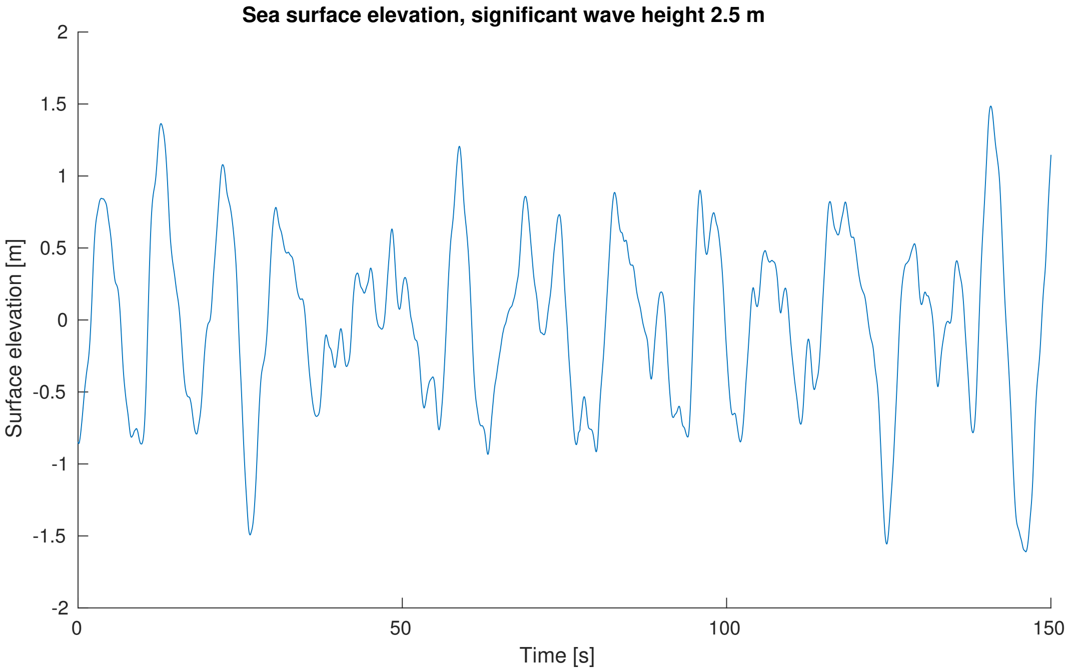


Figure 3.23: *Simulation of the sea surface elevation over 150 seconds, with significant wave height* $H_s = 2.5m$.

3.11.3.2 Vessel Interaction

Fossen (2011) suggests the following approach for modelling the wave forces and moments in 6 DOF:

$$X_{wave} = \frac{K_w^{\{1\}} s}{s^2 + 2\lambda\omega_e^{\{1\}} + (\omega_e^{\{1\}})^2} w_1 \tag{3.28}$$

$$Y_{wave} = \frac{K_w^{\{2\}} s}{s^2 + 2\lambda\omega_e^{\{2\}} + (\omega_e^{\{2\}})^2} w_2 \tag{3.29}$$

$$\vdots$$

$$N_{wave} = \frac{K_w^{\{6\}} s}{s^2 + 2\lambda\omega_e^{\{6\}} + (\omega_e^{\{6\}})^2} w_6 \tag{3.30}$$

where $\omega_e^{\{i\}}$ denotes the frequency of encounter, ω_0 denotes the wave peak frequency, and λ is a damping coefficient. The frequency of encounter should be used for a ship moving at a forward speed $U > 0$, using the wave peak frequency to compute ω_e. The sea state description in Table 2.4 can be used to find appropriate values for ω_0, resulting in the approximate values shown in

Table 3.2.

Beaufort number	Wave height	Sea condition	ω_0
0	0 m	Calm, mirror-like	n/a
1	0-0.2 m	Rippled	1.9858
2	0.2-0.5 m	Small wavelets	1.2559
3	0.5-1 m	Large wavelets	0.8881
4	1-2 m	Small waves	0.6280
5	2-3 m	Moderate waves	0.5127
6	3-4 m	Long waves, white foam	0.4440
7	4-5.5 m	Breaking waves	0.3787
8	5.5-7.5 m	Moderately high waves	0.3243
9	7-10 m	High waves	0.2808
10	9-12.5 m	Very high waves	0.2512
11	11.5-16 m	Exceptionally high waves	0.2293
12	≥ 14 m	Huge waves	0.2220

Table 3.2: *Wave peak frequencies related to the Beaufort scale.*

By transforming (3.28)-(3.30) to the time domain, defining $\dot{x}_{w1}^{\{i\}} = x_{w2}^{\{i\}}$ and $x_{w2}^{\{i\}} = y_{w}^{\{i\}}$ as state variables, linear state-space models on the following form can be obtained for each element $^{\{i\}}$ in $\boldsymbol{\tau}_{wave}$:

$$\dot{\boldsymbol{x}}^{\{i\}} = \begin{bmatrix} \dot{x}_{w1}^{\{i\}} \\ \dot{x}_{w2}^{\{i\}} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -(\omega_e^{\{i\}})^2 & -2\lambda\omega_e^{\{i\}} \end{bmatrix} \begin{bmatrix} x_{w1}^{\{i\}} \\ x_{w2}^{\{i\}} \end{bmatrix} + \begin{bmatrix} 0 \\ K_w^{\{i\}} \end{bmatrix} w \tag{3.31}$$

$$y_w^{\{i\}} = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_{w1}^{\{i\}} \\ x_{w2}^{\{i\}} \end{bmatrix} \tag{3.32}$$

The amplitudes of the forces and moments are then adjusted by choosing the constants $K_w^{\{i\}}$. For a marine craft moving at a speed $|U| > 0$, the wave peak frequency will be shifted according to

$$\omega_e(U, \omega_0, \beta) = \left| \omega_0 - \frac{\omega_0^2}{g} U cos(\beta) \right| \tag{3.33}$$

where the definition of the encounter angle β is shown in Figure 3.24.

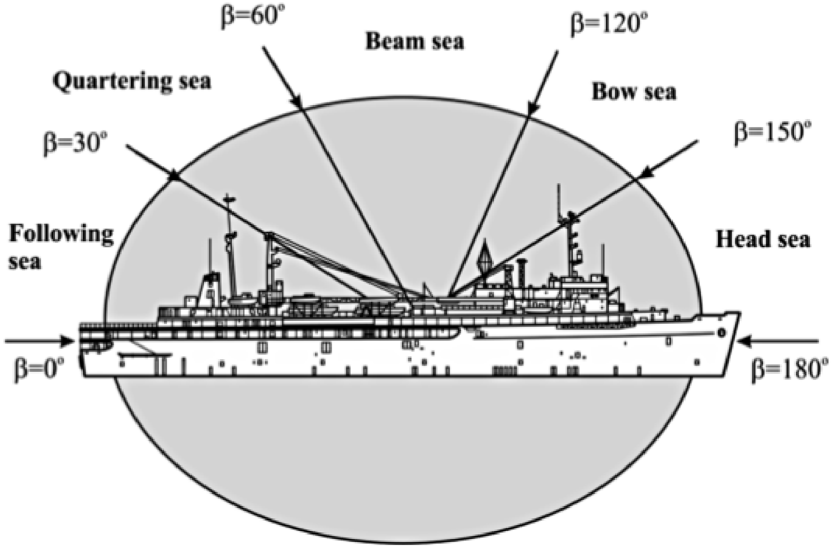


Figure 3.24: *Definition of wave encounter angle β. Courtesy of Fossen (2011).*

3.12 Model Discussion

In the vessel models presented, most parameters are results of semi-empirical formulas and scaled parameters from similar vessel types, with the exception of the nonlinear surge, sway and yaw damping for Odin, which was adjusted to fit recorded vessel data. The data were collected during tests performed by FFI, in order to validate controller performance and identify vessel parameters, and the tests consisted of zigzag maneuvers, turning circles and straight lines at different velocity and heading/heading rate setpoints. The recorded data consists of the following:

- Position and attitude relative to start position $(x, y, z, \phi, \theta, \psi)$
- Linear and angular velocities (u, v, w, p, q, r)
- GPS data $(latitude, longitude, height, speed, track)$
- Reference velocity, heading and heading rate $(u_{ref}, \psi_{ref}, r_{ref})$
- Desired actuator states $(RPM, deflector, nozzleangle)$

As the recorded data does not include information about wind, current or waves, there are uncertainties tied to the fitted parameters. Additionally, as hydrodynamic damping is highly dependent on details in the geometry of the vessel hull, the scaled parameters does not necessarily represent the vessel dynamics optimally. Despite of these uncertainties, the models described in this chapter still gives good performance for low speed applications, and the difference between actual vessel data and the simulation data is small up to a surge speed of $\approx 5m/s$, as shown in Figures 3.7, 3.10 and 3.11. Using (2.4), the Froude number for Odin at $5m/s$ is calculated as $Fn = 0.49$, which according to Table 2.3 approximately represents the transition to a semi-displacement vessel. When the semi-planing dynamics occur, there is a large deviation between the speeds of the simulated and actual vessel, as these dynamics are not included in the simulation model.

The actuator model for Odin presented in Section 3.10.2 is obtained by FFI, using state feedback from the actuators to identify time constants for RPM, nozzle angle and deflector position. The relationship between force and RPM is estimated at different surge speeds during sea trials, and as such, the actuator model for Odin is quite accurate. As this thesis considers modeling and simulation of low speed surveys for localization and mapping, the semi-planing vessel dynamics is neglected. As the impact of the surge speed u in the relationship between force and RPM is negligible at low speeds, the simulator uses the force/RPM relationship described in (2.40), which is independent of u.

As there is no actuator state feedback available on Jolner, the actuator model presented in Section 3.10.1 is obtained using characteristics for similar actuators, which results in a rough estimate of the actuator dynamics. For more accurate actuator dynamics, a relationship between force and RPM and time constants should be identified during trials.

Chapter 4

Simulator Implementation

The simulator is implemented in C++ using the Robot Operating System (ROS) framework, and it is an implementation of the equations of motion in 6DOF described in Chapter 2, using the vessel models presented in Chapter 3. Considering that the simulator should be versatile enough to be used for several different surface vessels, the simulator utilizes the ROS parameter server to load vessel parameters at runtime. The simulator also includes modules for simulation of environmental forces and realistic sensor data, in addition to a numerical solver.

4.1 Robot Operating System (ROS)

This section aims to give a brief introduction to the main concepts and conventions regarding ROS.

4.1.1 Introduction

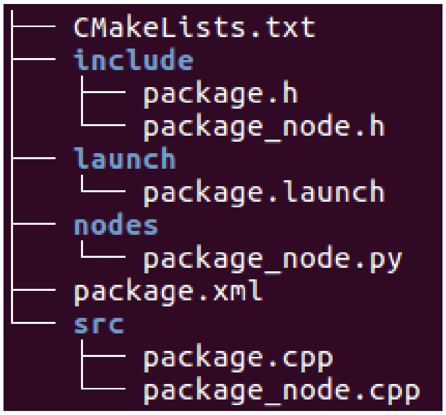
The Robot Operating System is a flexible framework for writing robot software. From its start in 2007, it was developed at multiple institutions, with significant resources provided by Willow Garage (ROS (2014)). The software has gradually become a widely used platform in robotics communities, partly thanks to the time and expertise contributed by countless researchers. The software itself is a collection of tools, libraries and conventions aiming to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms (ROS (2014)). ROS is also language independent, supporting among others C++, Python and

Lisp, in addition to experimental libraries for Java and Lua. It currently only runs on Unix-like platforms, with Ubuntu being the primary supported operating system.

The community supporting ROS has developed a large amount of software packages, simplifying the process of creating new solutions, and ensuring the possibility to build upon earlier work. As the core parts of ROS is licensed under the standard three-clause BSD license, it allows for reuse in commercial and closed source products. The community packages are using several different licenses, both Apache 2.0, GPL, MIT and proprietary licenses. Each package available is required to specify a license, so it is easy to decide if an interesting package meets the given licensing needs.

4.1.2 Concepts in ROS

ROS packages are the most common way of organizing code in ROS. A package can be compared to a computer program, containing one or more *nodes* and configuration files, in addition to a *package manifest* and a *makefile*. The *nodes* are processes that performs computations, which can be combined together into a graph, and communicate with one another. Each *node* may publish or subscribe to a *topic*, and there may be multiple concurrent publishers and subscribers on a single topic, which are generally unaware of each others existence. The *package manifest* provides metadata about a given package, like name, version, license and description of the package. ROS uses *CMake* to configure compilation of ROS packages, and provides the possibility to define and compile custom *message types*, which defines the data structures for messages sent.



```
├── CMakeLists.txt
├── include
│   ├── package.h
│   └── package_node.h
├── launch
│   └── package.launch
├── nodes
│   └── package_node.py
├── package.xml
└── src
    ├── package.cpp
    └── package_node.cpp
```

Figure 4.1: *Example of ROS package file structure.*

The *Parameter Server* stores data by key in the ROS *Master*, which provides name registration and lookup to the rest of the computation graph. This allows nodes to fetch parameters both at launch time and/or run-time. ROS also provides *bags* as a format for saving and playing back ROS message data on a given *topic*.

4.1.3 Reference Frames

The ROS standard units of measureand coordinate conventions are described in Foote and Purvis (2010). The coordinate frames are right handed, and the body coordinate frame standard is defined as:

- x forward,
- y left,
- z up.

The global reference coordinate system used is the East, North, Up (ENU) convention. In ROS, the position and orientation of a robot is refered to as *pose*, where the position is given as $\boldsymbol{p} = (x, y, z)^T$, and the orientation as a quaternion $\boldsymbol{q} = (\epsilon_1, \epsilon_2, \epsilon_3, \eta)^T$. Velocity is split into linear velocity $\boldsymbol{v} = (u, v, w)^T$ and angular velocity $\boldsymbol{\omega} = (p, q, r)^T$.

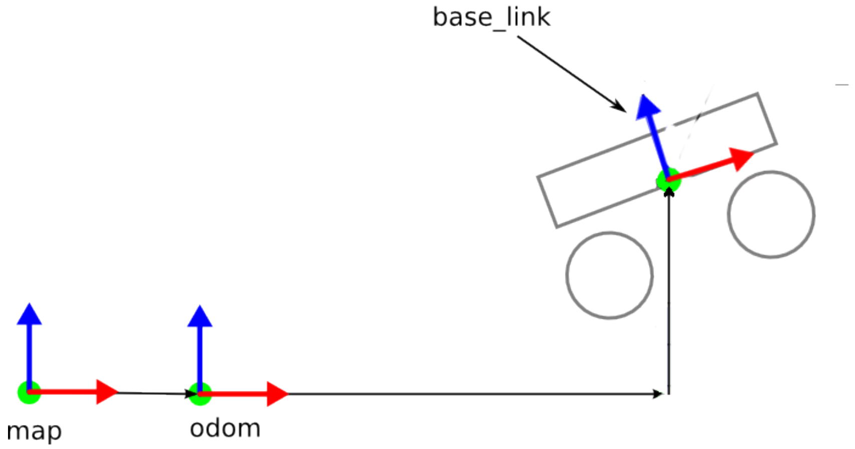


Figure 4.2: *ROS coordinate frames. The red, green and blue axes represents the x, y and z axes.*

For mobile platforms ROS commonly uses three main frames, which are described in Meeussen (2010): *map*, *odom* and *base__link*. The *map* frame is a global world fixed frame, with it's z-axis pointing upwards. The *odom* frame is also a world fixed frame, but this frame can drift over time, without any bounds, which makes it less useful as a global reference. This frame is typically used for local sensing and acting, and often computed based on information from sensors. The *base__link* frame is rigidly attached to the mobile robot base, acting as the *body* frame.

4.2 Interface Between Simulator and Control System

In this section, the interface between the prototype HIL simulator and the USV control system will be described. As both Jolner's and Odin's control system are a work in progress, details regarding protocol, physical interface and similar details is omitted. Instead, a more general interface is specified, describing the information relevant to exchange between the simulator and control system. After completion of the control systems, this general interface should be augmented with smaller modules, able to transmit and receive the information described in this section.

The HIL simulator is running on a single laptop, with Ubuntu or Debian OS to utilize ROS functionality. The development of the simulator is divided into two master theses; one for the simulation of the USV dynamics (this thesis) and one for the simulation of the surroundings of the USV (Børs-Lind (2017a)). By dividing the development into two tasks, it becomes natural to split the simulation software into two executive modules, *Dynamics* and *Surroundings*. The weather conditions are decided in the *Surroundings*-module, and sent to the *Dynamics*-module for detailed simulations, including how the wind, current and waves influences the motion of the USV, as well as the varying components of the weather conditions. In addition to this, actuator inputs are sent from the USV control system to the *Dynamics*-module, enabling calculation of the vessels attitude and position at any given time, as well as the linear and angular velocities. The position of the USV is also sent to the *Surroundings*-module, to visualize the vessel in the simulated sorroundings. Knowing the detailed values of the weather conditions in addition to the actuator inputs of the vessel at any given time, the *Dynamics*-module is then able to generate appropriate sensor data for the USV's *wind sensor*, *MRU*, *GPS* and *speed sensor*, and sending it to the USV Control System. An overview of the HIL software layout is presented in Figure 4.3, while the structure of the Dynamics module is presented in Figure 4.4.

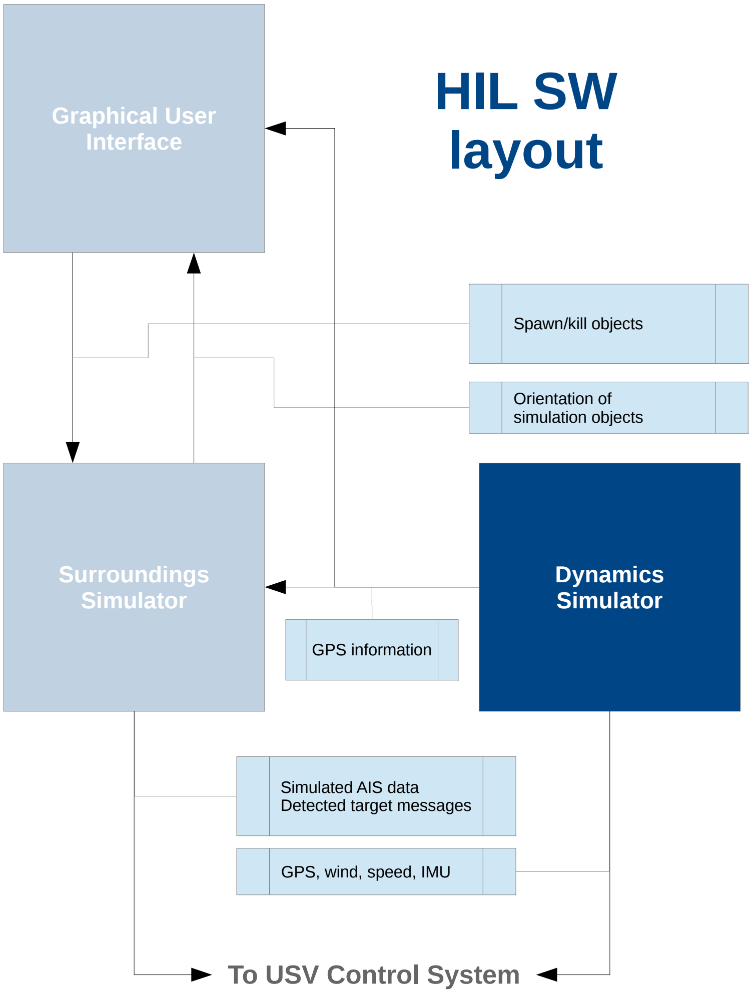


Figure 4.3: *Overview of the design of HIL simulation setup. The solid blue module, "Dynamics Simulator", is considered in this thesis, while the other, transparent modules are considered in Børs-Lind (2017a).*

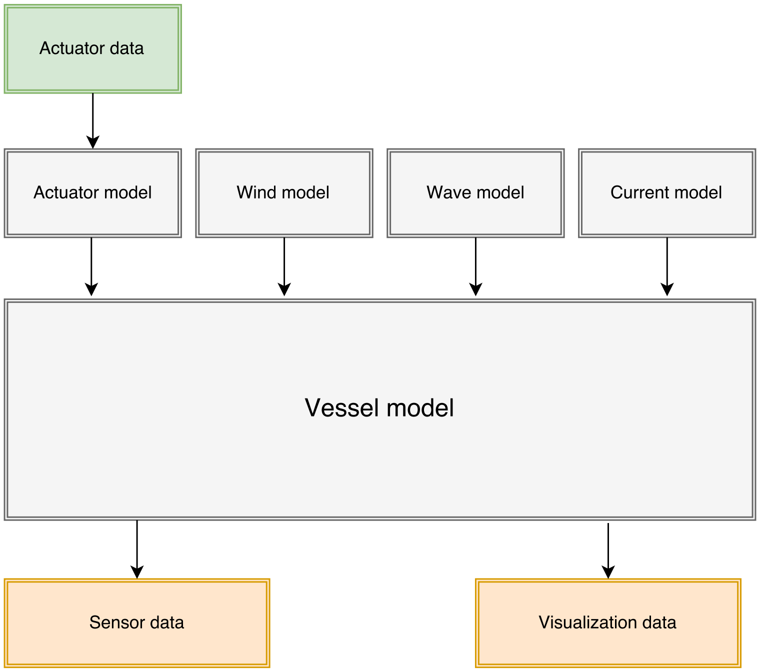


Figure 4.4: *Structure of the Dynamics Simulator.*

4.3 Numerical Solver

Numerical simulations are usually a trade-off between simulation time and precision. The mathematical models presented in this report are approximations, and we know for certain that the models are not 100% accurate. As a result of this, the simulations performed will be approximations of the vessels behavior, and the need for solver precision decreases. However, assuming that system identification of the vessels considered is performed in the future, it would be convenient if the solver(s) used in the simulations are relatively accurate. Considering this, the *Dormand-Prince* method would be a reasonable choice of solver in the following simulations. This is a variable-step explicit Runge-Kutta method, which provides the possibility to specify the desired accuracy of the computed solution, and then have automatic selection of the step-size that ensures the required accuracy (Egeland and Gravdahl (2003)). On the other hand, as the interface with the USV control system introduces some timing demands, a fixed-step solver could simplify the simulation of sensor data as well as receiving the thrust inputs. A good solution would then be to use the Dormand-Prince method with a fixed step-size, rendering it a fifth order Runge-Kutta method. Similar to the vessel parameters, the step-size is also loaded from the ROS parameter server at runtime. In Figure 4.5, the resulting positions from simulations with five different step sizes are shown. The simulations are performed for the vessel Jolner, with no external forces, using constant surge force $X = 100N$ and yaw moment $N = 5Nm$ for all five runs. It is obvious from Figure 4.5 that the choice of step size is of great importance for the precision, and we see that the increased precision converges to zero as the step size gets sufficiently small. As the simulations in this prototype will run in real-time, there is also a lower limit on the step size, related to the computational power available at the computer running the simulations.

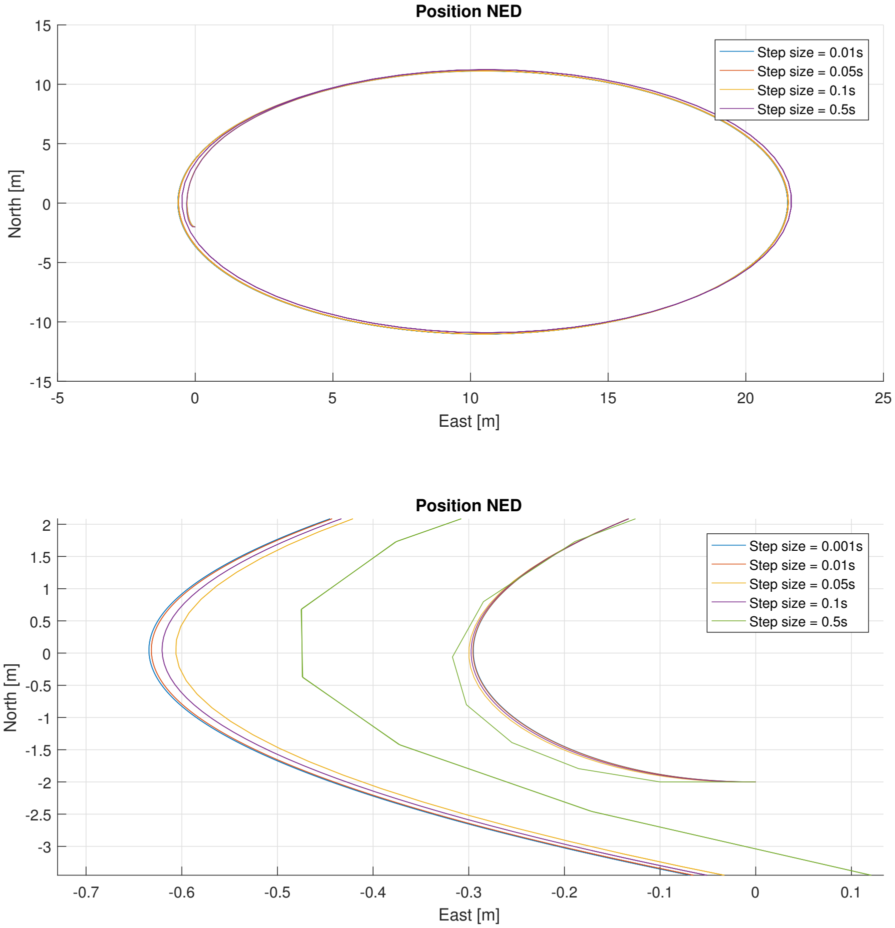


Figure 4.5: *Comparison of the simulated positions of Jolner using different step sizes, without the influence of environmental forces. The simulations are performed for 100 seconds, with a constant surge force $X = 100N$ and yaw moment $Y = 5Nm$. The lower figure presents a zoomed partition of the position plots presented in the upper figure.*

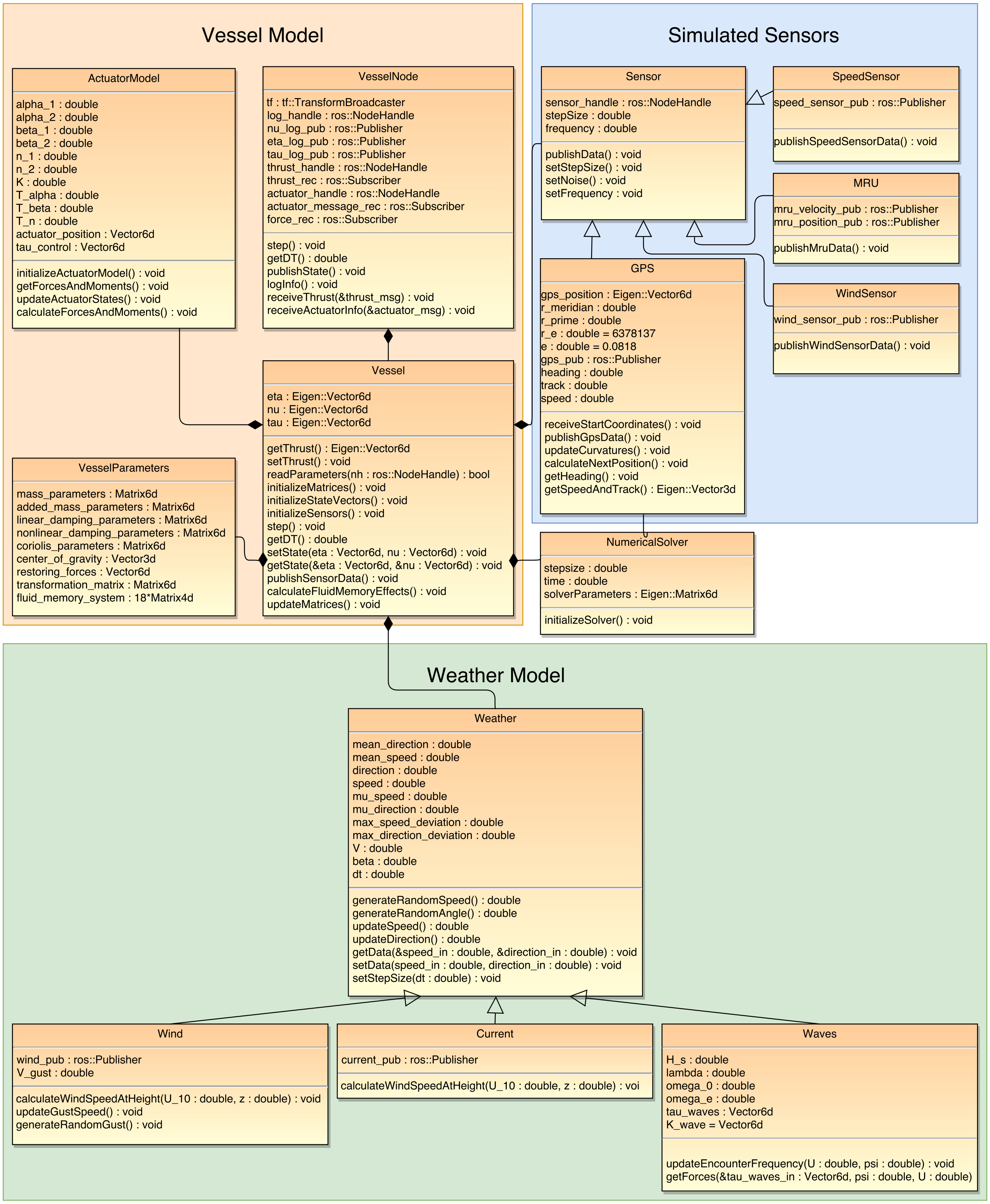

Vessel Model
ActuatorModel
alpha_1 : double
alpha_2 : double
beta_1 : double
beta_2 : double
n_1 : double
n_2 : double
K : double
T_alpha : double
T_beta : double
T_n : double
actuator_position : Vector6d
tau_control : Vector6d
initializeActuatorModel() : void
getForcesAndMoments() : void
updateActuatorStates() : void
calculateForcesAndMoments() : void
VesselNode
tf : tf::TransformBroadcaster
log_handle : ros::NodeHandle
nu_log_pub : ros::Publisher
eta_log_pub : ros::Publisher
tau_log_pub : ros::Publisher
thrust_handle : ros::NodeHandle
thrust_rec : ros::Subscriber
actuator_handle : ros::NodeHandle
actuator_message_rec : ros::Subscriber
force_rec : ros::Subscriber
step() : void
getDT() : double
publishState() : void
logInfo() : void
receiveThrust(&thrust_msg) : void
receiveActuatorInfo(&actuator_msg) : void
Vessel
eta : Eigen::Vector6d
nu : Eigen::Vector6d
tau : Eigen::Vector6d
getThrust() : Eigen::Vector6d
setThrust() : void
readParameters(nh : ros::NodeHandle) : bool
initializeMatrices() : void
initializeStateVectors() : void
initializeSensors() : void
step() : void
getDT() : double
setState(eta : Vector6d, nu : Vector6d) : void
getState(&eta : Vector6d, &nu : Vector6d) : void
publishSensorData() : void
calculateFluidMemoryEffects() : void
updateMatrices() : void
VesselParameters
mass_parameters : Matrix6d
added_mass_parameters : Matrix6d
linear_damping_parameters : Matrix6d
nonlinear_damping_parameters : Matrix6d
coriolis_parameters : Matrix6d
center_of_gravity : Vector3d
restoring_forces : Vector6d
transformation_matrix : Matrix6d
fluid_memory_system : 18*Matrix4d
Simulated Sensors
Sensor
sensor_handle : ros::NodeHandle
stepSize : double
frequency : double
publishData() : void
setStepSize() : void
setNoise() : void
setFrequency : void
SpeedSensor
speed_sensor_pub : ros::Publisher
publishSpeedSensorData() : void
MRU
mru_velocity_pub : ros::Publisher
mru_position_pub : ros::Publisher
publishMruData() : void
GPS
gps_position : Eigen::Vector6d
r_meridian : double
r_prime : double
r_e : double = 6378137
e : double = 0.0818
gps_pub : ros::Publisher
heading : double
track : double
speed : double
receiveStartCoordinates() : void
publishGpsData() : void
updateCurvatures() : void
calculateNextPosition() : void
getHeading() : void
getSpeedAndTrack() : Eigen::Vector3d
WindSensor
wind_sensor_pub : ros::Publisher
publishWindSensorData() : void
NumericalSolver
stepsize : double
time : double
solverParameters : Eigen::Matrix6d
initializeSolver() : void
Weather Model
Weather
mean_direction : double
mean_speed : double
direction : double
speed : double
mu_speed : double
mu_direction : double
max_speed_deviation : double
max_direction_deviation : double
V : double
beta : double
dt : double
generateRandomSpeed() : double
generateRandomAngle() : double
updateSpeed() : double
updateDirection() : double
getData(&speed_in : double, &direction_in : double) : void
setData(speed_in : double, direction_in : double) : void
setStepSize(dt : double) : void
Wind
wind_pub : ros::Publisher
V_gust : double
calculateWindSpeedAtHeight(U_10 : double, z : double) : void
updateGustSpeed() : void
generateRandomGust() : void
Current
current_pub : ros::Publisher
calculateWindSpeedAtHeight(U_10 : double, z : double) : voi
Waves
H_s : double
lambda : double
omega_0 : double
omega_e : double
tau_waves : Vector6d
K_wave = Vector6d
updateEncounterFrequency(U : double, psi : double) : void
getForces(&tau_waves_in : Vector6d, psi : double, U : double)

Figure 4.6: *Class diagram describing the structure of the Dynamics simulation system.*

4.4 Sensor Simulation

For this project, the considered sensor data is speed, wind speed, wind direction, acceleration and angular rate, in addition to GPS data. As no sensors are perfect, the true sensor signals will have a certain error, noise and drift, which can also be modeled for the simulated sensors.

As shown in Figure 4.6, this prototype uses a base class *Sensor* for the sensors, providing functionality for modeling of noise, setting sensor output frequency and publishing sensor data, while each specific sensor class is derived from this base class, with specific features for the given sensor. Detailed models of sensor noise, error and drift can specified by loading sensor parameters from the ROS parameter server for each specific sensor.

4.4.1 Wind Sensor

The wind sensor in the simulator is assumed to be disconnected from any other instruments, and as such, it will output the appearing wind speed and direction. It is assumed that the wind sensor onboard both vessels is able to measure both relative wind speed and direction. Using the true wind speed V_w and direction β_w, the appearing wind speed V_{rw} and direction γ_{rw} is obtained by using (2.68)-(2.71). The simulated wind sensor then publishes a ROS-message containing the relative wind speed V_{rw} and direction γ_{rw}.

4.4.2 Speed Sensor

To facilitate for simulation of the different speed sensors described in Section 3.3.2, the simulated speed sensor will output relative speed in surge, and alternately in sway, in addition to a time-stamp. The speed through water will be assumed to be error-free, and the speed sensor data can then be simulated by sending the speed through water using the desired interface.

4.4.3 NavP

NavP uses sensor data from the IMU and the GPS to obtain filtered navigation data, with estimates of position, velocity and attitude. By simulating the NavP output data directly, the data frequency demanded by the navigation system lowers, as the estimations are assumed to be performed within NavP. The simulated NavP module will then output a message consisting of measured acceleration and angular velocity, estimates of position, velocity and attitude, and a time-stamp. The simulated data will in this thesis be considered to be error-free, to simplify implementation.

4.4.4 GPS

As the position given from (2.13) is defined in a three-dimensional Cartesian frame, this has to be transformed to the NED frame using the WGS-84 ellipsoid, defined in Table 4.1, which is currently used for GPS.

Parameter	Explanation
$r_e = 6378137m$	Equatorial radius of ellipsoid (semimajor axis).
$r_p = 6356752m$	Polar axis radius of ellipsoid (semiminor axis).
$\omega_e = 7292115 \cdot 10^{-11} rad/s$	Angular velocity of Earth.
$\mu_g = 3986005 \cdot 10^8 m^3/s^2$	Gravitational constant of Earth.
$e = 0.0818$	Eccentricity of Earth.

Table 4.1: *Overview of the WGS-84 parameters.*

By defining the origin of the Cartesian frame as the vessel's origin in the NED frame with latitude $\mu(0)$ and longitude $l(0)$, the position $[\mu(t), l(t)]$ at any time t can be described as an offset from the NED frame origin. A suggested approach for this is presented in Vik (2009):

$$\begin{bmatrix} \dot{\mu} \\ \dot{l} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} \frac{1}{r_{mer}+h} & 0 & 0 \\ 0 & \frac{1}{(r_{prime}+h)cos\mu} & 0 \\ 0 & 0 & -1 \end{bmatrix} \boldsymbol{v}^n_{\{1,2,3\}} \tag{4.1}$$

where r_{mer} and r_{prime} are the prime and meridian curvatures of Earth, defined by:

$$r_{mer} = \frac{r_e(1-e^2)}{(1-e^2 sin^2\mu)^{1.5}} \tag{4.2}$$

$$r_{prime} = \frac{r_e}{(1-e^2 sin^2\mu)^{0.5}} \tag{4.3}$$

The simulated GPS will then publish a ROS message describing latitude, longitude, height, heading, track, speed, roll and pitch measurements, in addition to a time-stamp. The simulated GPS will be considered to be error-free during implementation.

4.5 Visualization of Simulation Data

Using the approaches presented in Chapter 2 and Chapter 3, the simulated vessels' position and attitude in 6 DOF are obtained. To visualize this, the simulator publishes the vector $\boldsymbol{\eta}$ along with a timestamp in a `tf::TransformBroadcaster` on the topic `/tf`. By uploading a description of the vessel 3D model to the ROS Parameter Server, the vessel position and attitude can be visualized in the ROS 3D visualization tool RViz by fetching the vessel description from the ROS Parameter Server and subscribing to the `/tf` topic.

To visualize the actuator states during runtime, RViz Markers are used, in the shape of two arrows. The arrow length and direction corresponds to the actuator RPM and angle, and as such, the user gets an overview of the state of each actuator. An example of the vessel and actuator state visualization is shown in Figure 4.7.

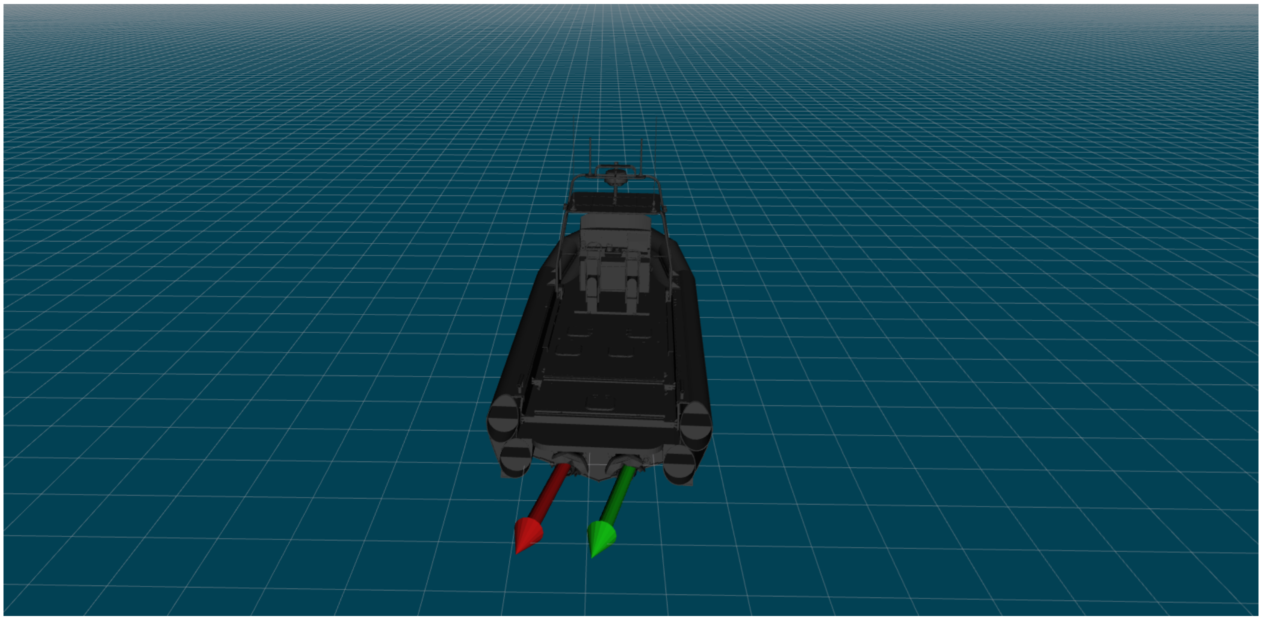


Figure 4.7: *Visualization of the vessel Odin in RViz during simulation.*

4.6 Logging of Simulation Data

The results from simulations should be saved and archived, in order for the developers to be able to compare the current performance with previous ones, without having to run the simulations all over again. This is especially important when considering the use of automatic performance tests, as these tests are usually performed without interaction from the developers.

In this prototype, this functionality is implemented through the use of a ROS package called *rosbag*, which is a set of tools for recording from and playing back to given ROS topics (Field et al. (2015)). Using this package, all messages published on the ROS topics defined for logging of simulation data are stored in a *.bag* file with timestamps, allowing the developers to play back data and regenerate the chain of events, in addition to providing the possibility of importing simulation data to among others *Matlab*, in order to inspect the results more closely.

In this prototype simulator, several ROS nodes and topics are used for logging of the simulation data, including position and attitude, linear and angular velocities, current speed and direction, wind speed and direction, actuator forces and moments and sensor data. By using *rosbag*, the developers can then subscribe to log-data from only the nodes they find relevant, and leave the rest of the topics unused. An example of the information flow is given in Figure 4.8, and a sequence diagram for the Dynamics simulator is presented in Figure 4.9.

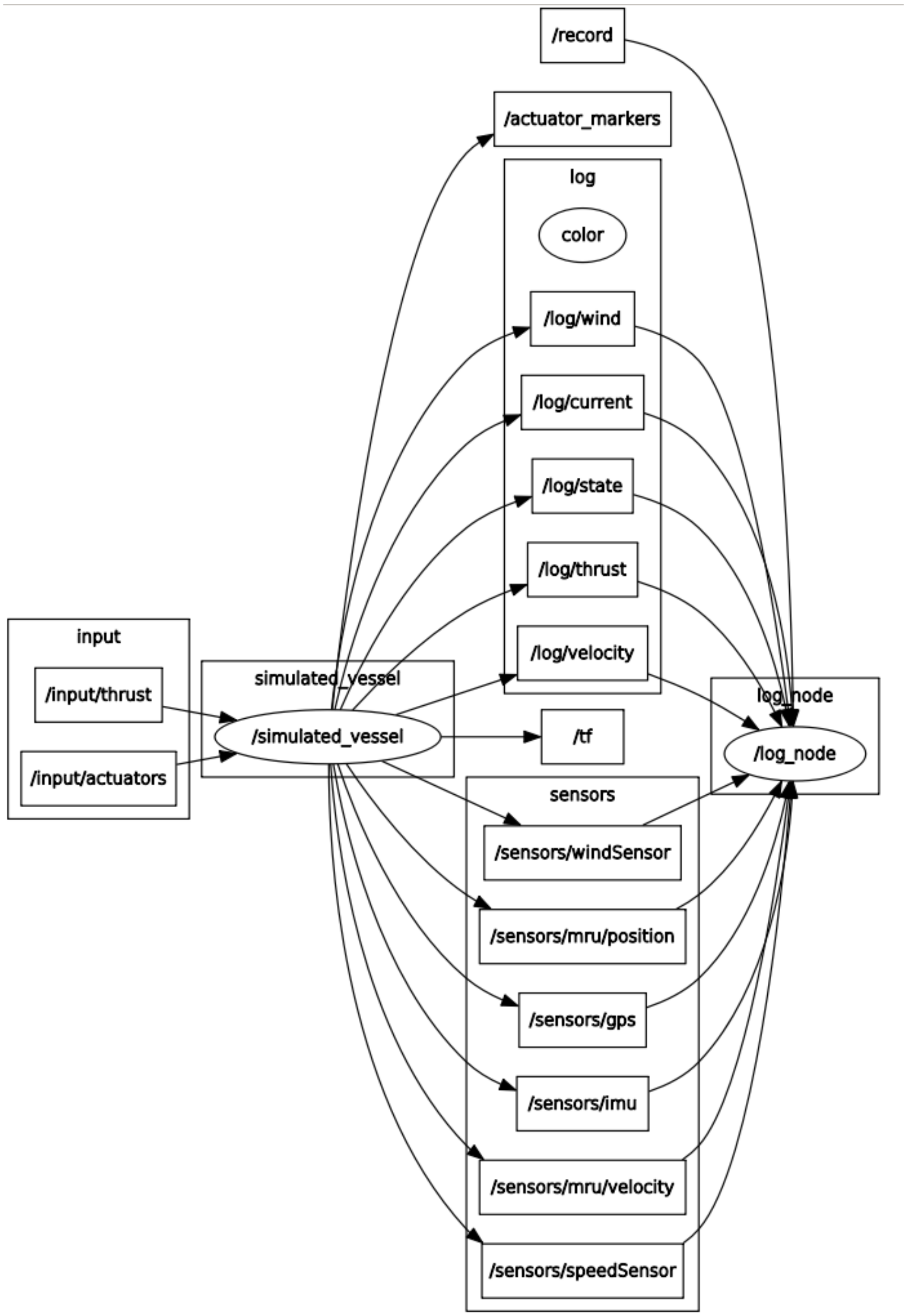


Figure 4.8: *An overview of the ROS Nodes and Topics used in the simulator, generated by the package `rqt_ graph`.*

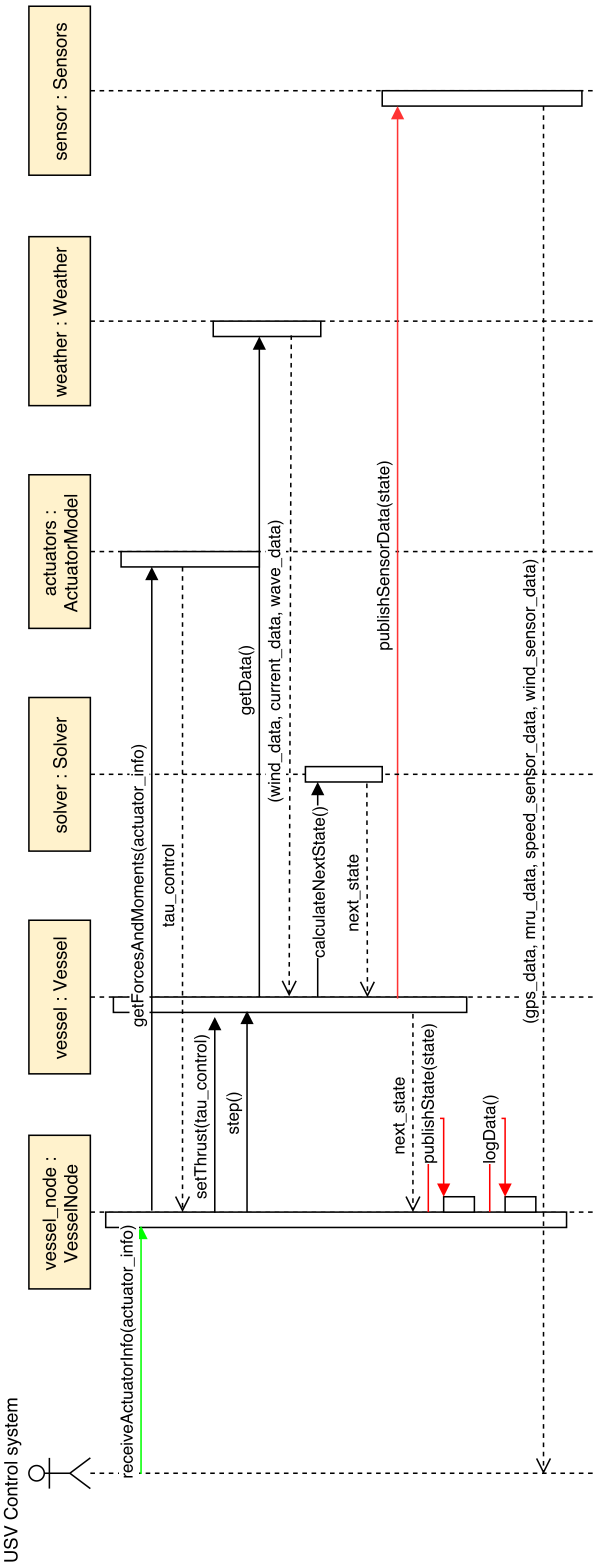


Figure 4.9: *Sequence diagram for the dynamics simulator. To improve readability, wind, current and waves are represented as weather, and GPS, NavP, wind sensor and speed sensor are represented as sensors. Red lines denotes methods utilizing ROS Publishers, and green lines denotes methods utilizing ROS Subscribers.*

Chapter 5

HIL Simulation Results

This chapter presents comparisons of results from simulations performed with the control system on Odin included in the loop. The control system is performing a waypoint guidance mission in the HIL simulator in three separate simulations, using three different sets of environmental conditions. The simulation results are presented and compared with results from one sea trial using the same mission.

HIL testing is normally conducted in several phases of the vessel development, and the first phase is usually an extensive software test conducted at a factory, enabling thorough testing through virtual sea trials before the vessel is built. Using this approach enables the possibility of testing and verification of the computer software and vessel performance. In order to achieve the test objectives, the simulator should be able to simulate a wide selection of realistic scenarios regarding weather and sorroundings, ensuring the desired functionality and vessel performance during both normal and extreme conditions.

As this thesis considers the development of an HIL simulator for two existing USVs, the vessel control systems are already developed and tuned for the actual vessels through sea trials. The vessel models estimated in this thesis are based on approximations, and some deviations are expected. As a result of this, the vessel controllers are not necessarily properly tuned for the simulation models.

By performing these simulations the interface between the control system on Odin and the Dynamics simulator is tested, and the control system performance can be compared for the simulated and actual vessel. Additionally, the use of different environmental conditions serves to verify the robustness of the control system. The mission consists of a set of four different waypoints, which are repeated several times. The control system used on Odin in these scenarios consists of a waypoint guidance controller, using a Line-Of-Sight (LOS) guidance scheme, which outputs a desired heading to a PID heading controller. The waypoint guidance controller generates a path using straight lines and circular arcs connecting the waypoints, and the turning point of the vessel is dependent on the angle between the waypoints, as shown in Figure 5.1. The reference speed is set to $5 m/s$, which is controlled by a PI-controller independent of the heading and waypoint controller.

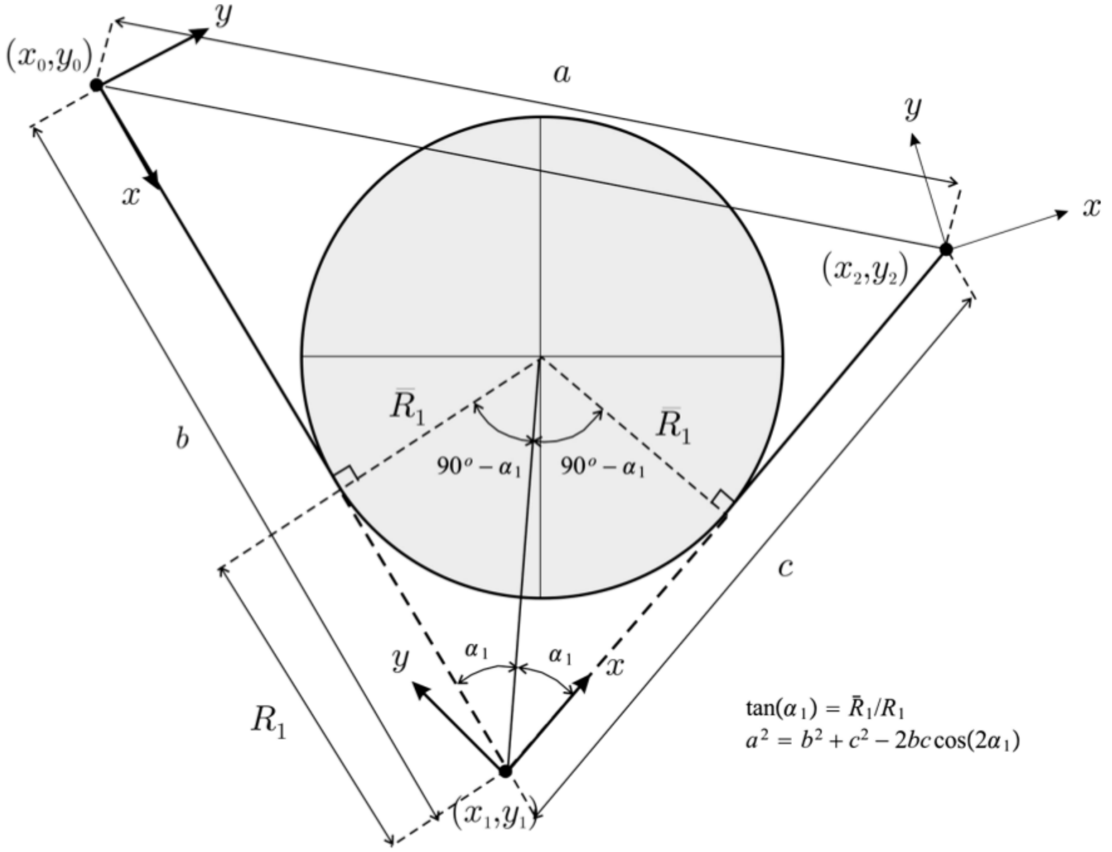


Figure 5.1: *Circle with radius $\bar{R}_1$ inscribed between the waypoints (x_0, y_0), (x_1, y_1) and (x_2, y_2). The point where the circle arc intersects the straight line represents the turning point of the ship. Courtesy of Fossen (2011)*

The sea trial used for comparison took place in Horten, Norway, at May 8, 2017. Details regarding the environmental conditions during the sea trial are not known, and as such they can not be reconstructed during simulations. This is expected to lead to some deviations between the simulated and actual vessel, as seen in the simulation results. The following scenarios are used in the HIL simulations:

1. **Calm weather:** Simulation performed without influence from the environmental conditions.

2. **Gale, calm sea:** Simulation performed using mean wind speed $V_w = 20m/s$, mean wind direction $\beta_w = 0deg$, mean current speed $V_c = 0.5m/s$, mean current direction $\beta_c = 0deg$, and significant wave height $H_s = 0m$.

3. **Gale, significant wave height 1m:** Simulation performed using mean wind speed $V_w = 20m/s$, mean wind direction $\beta_w = 0deg$, mean current speed $V_c = 0.5m/s$, mean current direction $\beta_c = 0deg$, and significant wave height $H_s = 1m$.

5.1 Calm Weather

Scenario 1 considers a simulated mission performed without influence from the environmental conditions, which gives the following results:

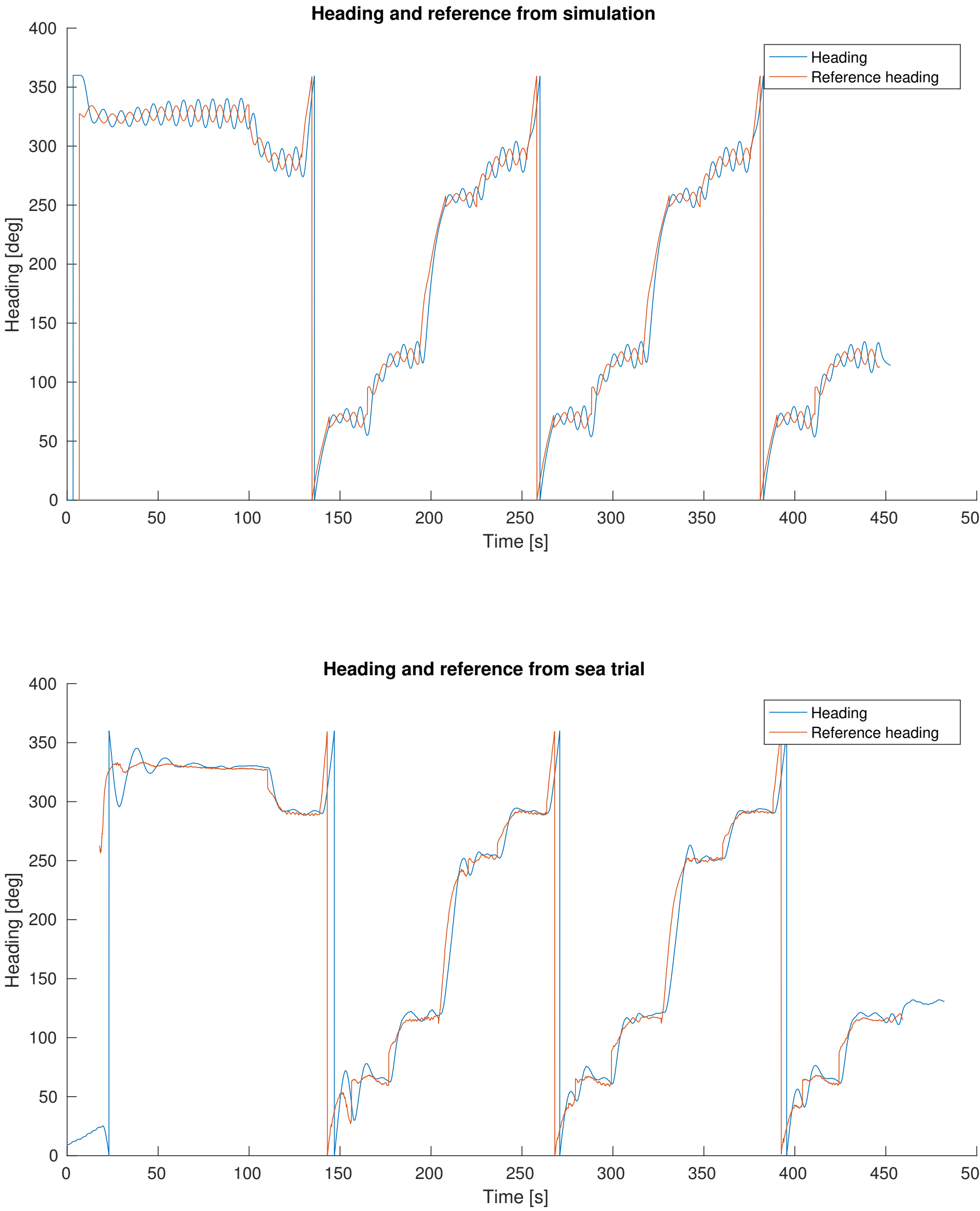


Figure 5.2: *Comparison of heading and reference heading from sea trials and simulation, scenario 1.*

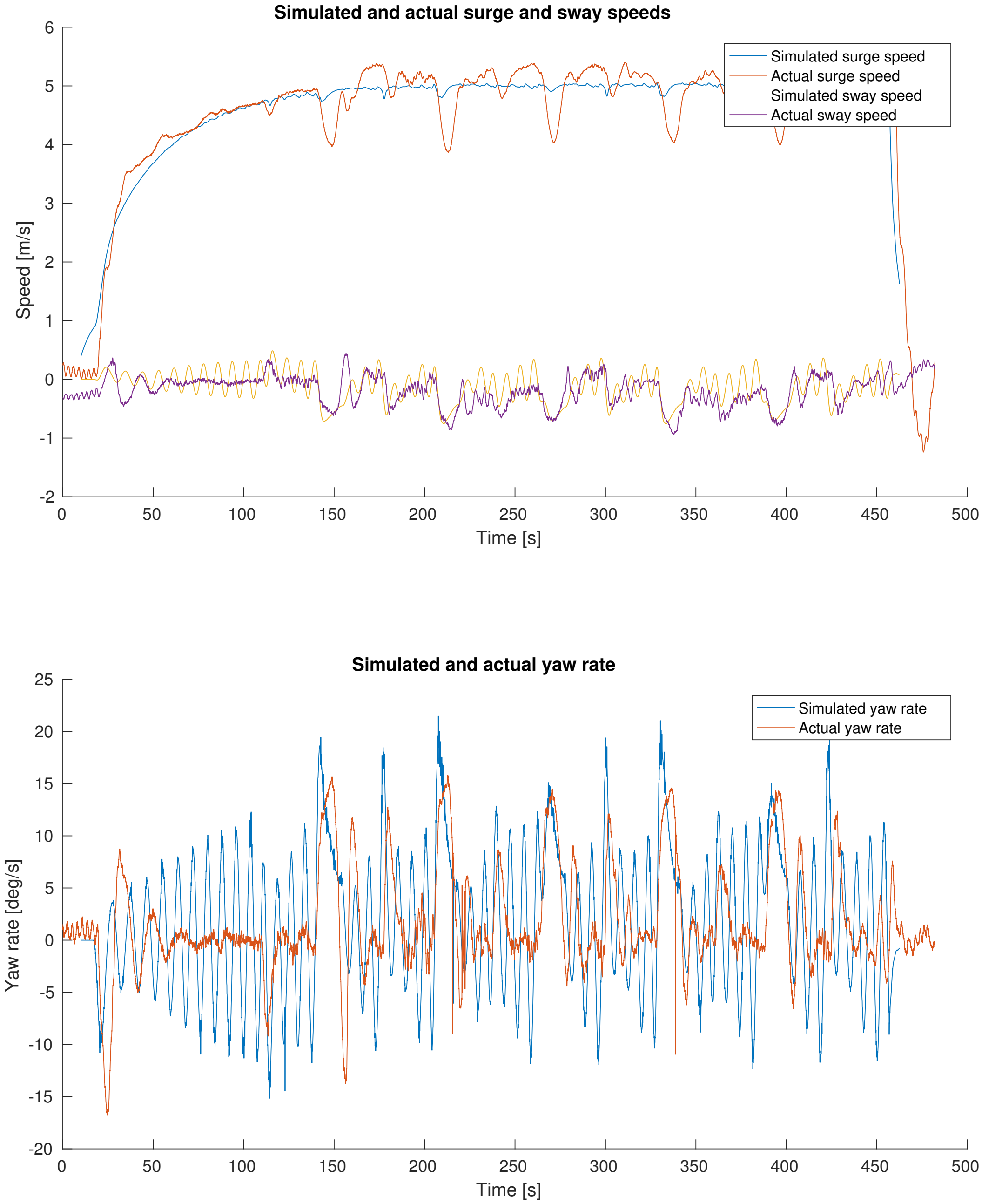


Figure 5.3: *Comparison of linear and angular velocities from sea trials and simulation, scenario 1.*

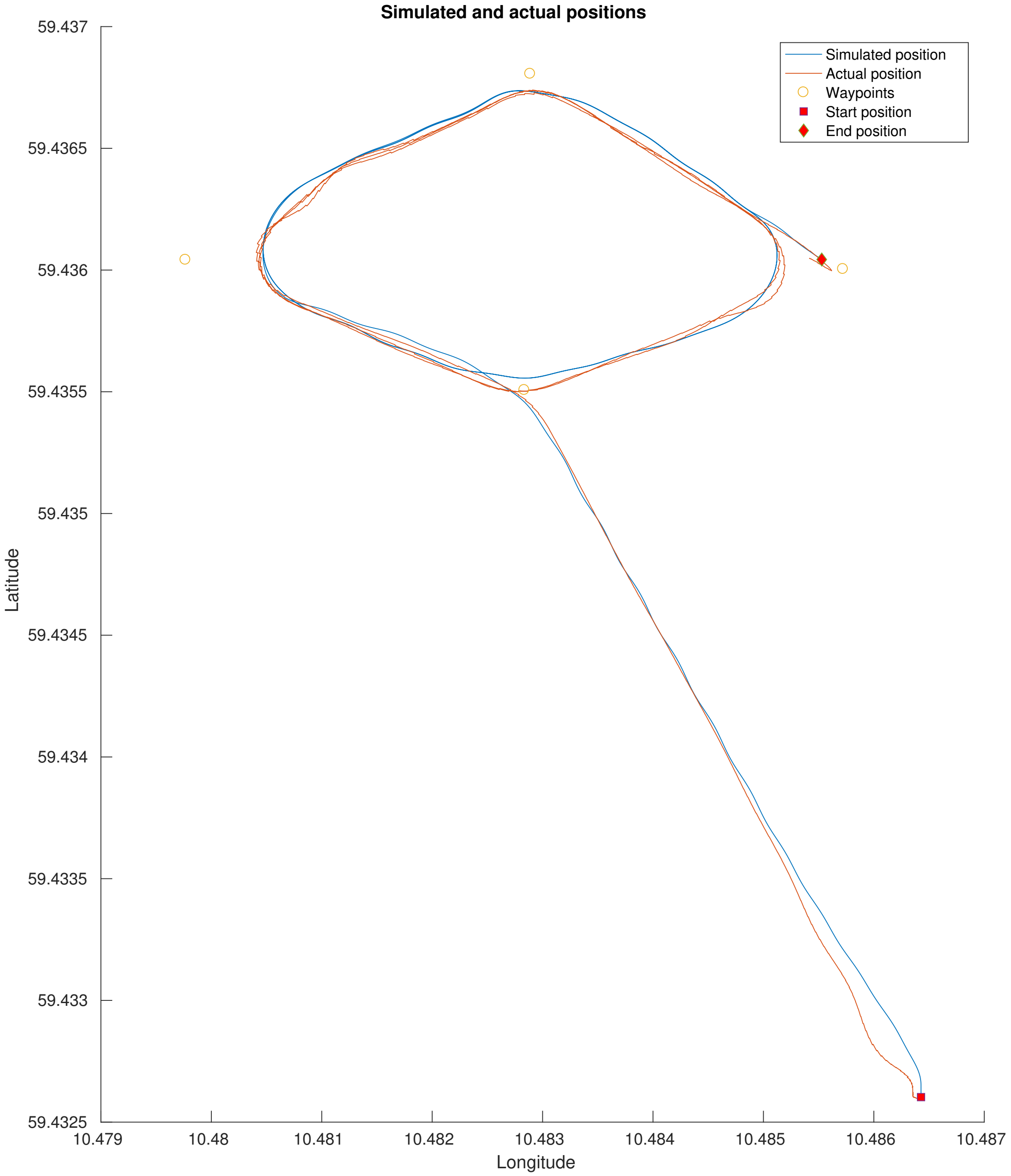


Figure 5.4: *Comparison of position and waypoints from sea trials and simulation, scenario 1.*

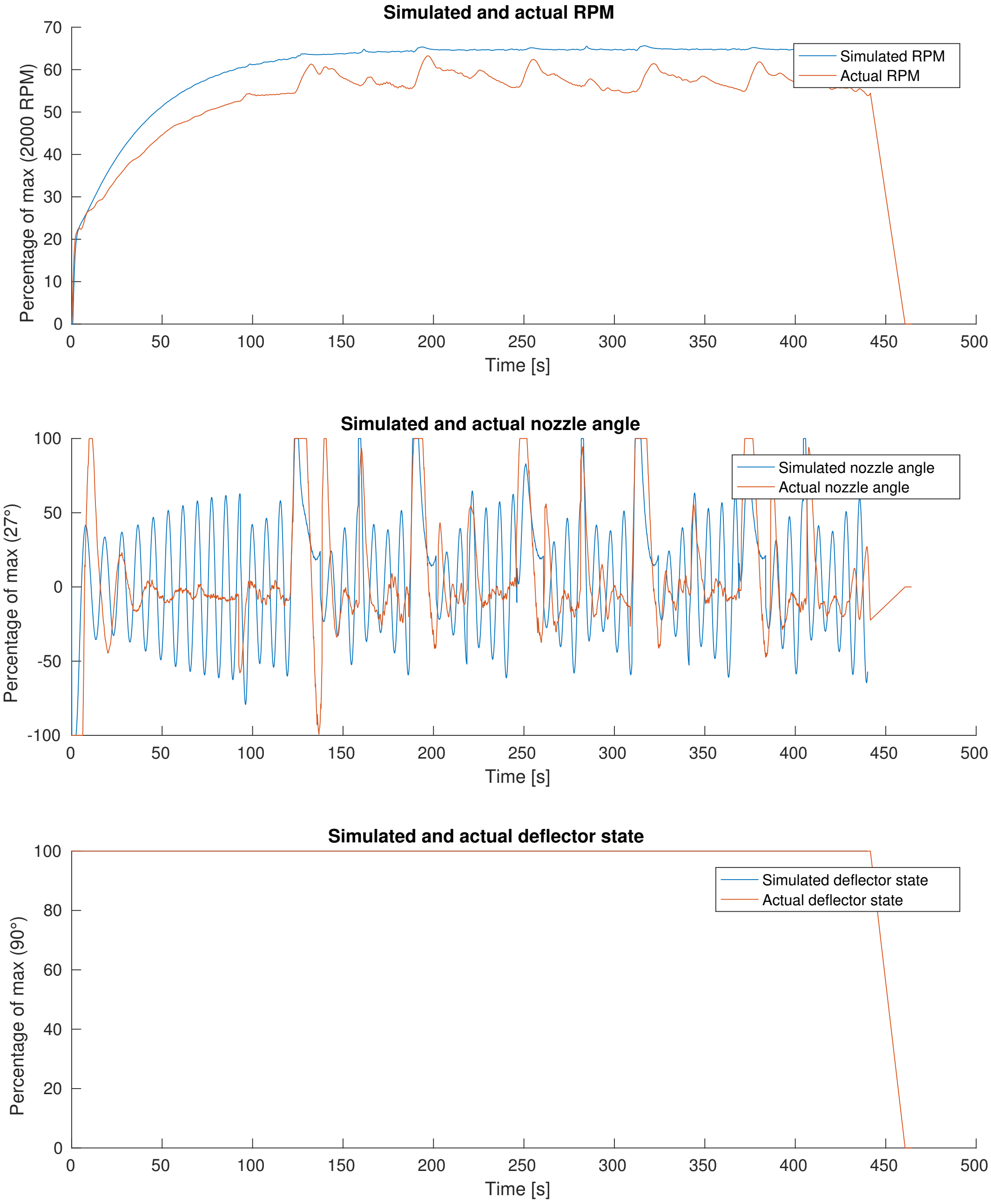


Figure 5.5: *Comparison of desired actuator states from sea trials and simulation, scenario 1.*

The results from this scenario shows differences in the heading controller performance for the simulated and actual vessel. In Figure 5.2, it can be seen that the actual vessel has a properly tuned controller, following the reference closely. The heading controller for the simulated vessel is slightly underdamped, leading to oscillations in the heading. As the LOS guidance generates a reference heading based on the vessel's position relative to the LOS vector, this behaviour leads to oscillations in the reference heading, which in turn affects the yaw rate, sway speed and actuator states, as shown in Figures 5.3 and 5.5. Figure 5.3 shows that the speed controller has similar performance for the simulated and actual vessel. The yaw rate and sway speed has a more significant effect on the surge speed of the actual vessel. A possible reason for this is that the damping matrix used in the simulator assumes decoupled surge dynamics under the assumption of symmetry about the xz-plane. Furthermore, it is observed that the actual surge speed in Figure 5.3 drops periodically. Weather data acquired from `http://www.yr.no` shows an average wind speed of $12m/s$ and direction north-northeast in Horten at the time of sea trials, which could influence both the surge and sway speeds differently at different headings. From Figure 5.4 it is obvious that the waypoint guidance controller used in the control system is able to perform well for both the simulated and actual vessel.

5.2 Gale, Calm Sea

Scenario 2 considers a simulated mission performed with a mean wind speed of $20 m/s$, and a mean current speed of $0.5 m/s$, with a significant wave height $H_s = 0m$, which gives the following results:

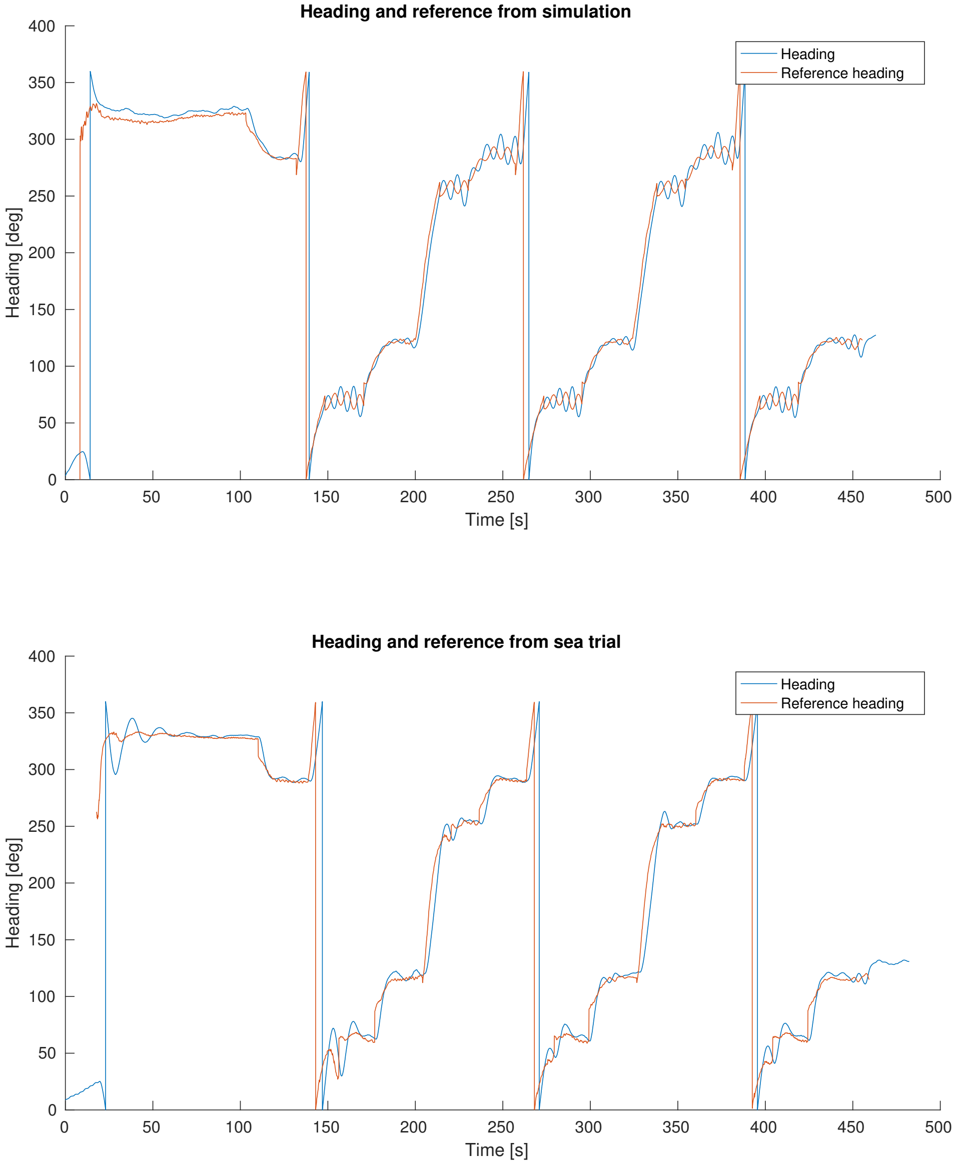


Figure 5.6: *Comparison of heading and reference heading from sea trials and simulation, scenario 2.*

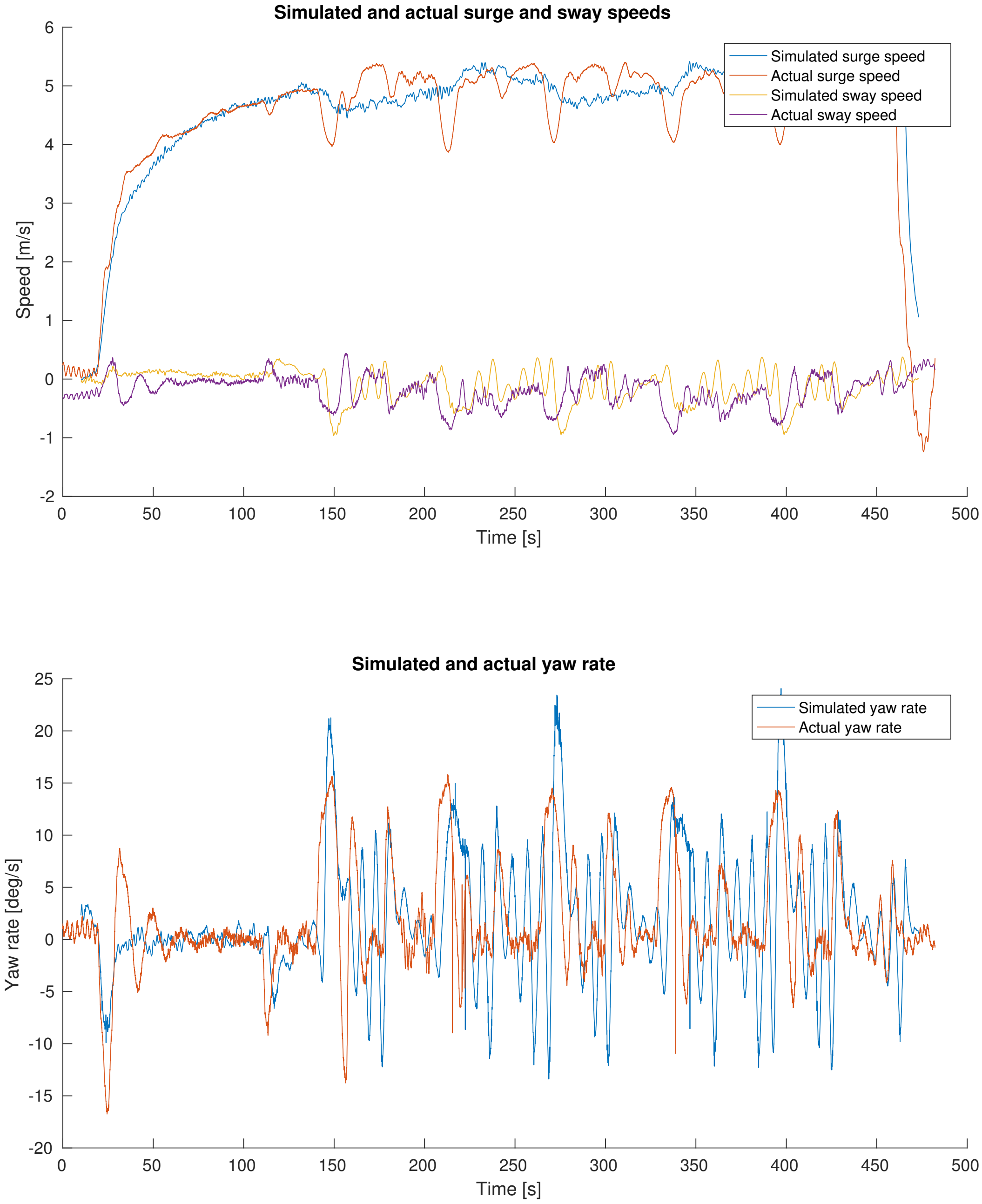


Figure 5.7: *Comparison of linear and angular velocities from sea trials and simulation, scenario 2.*

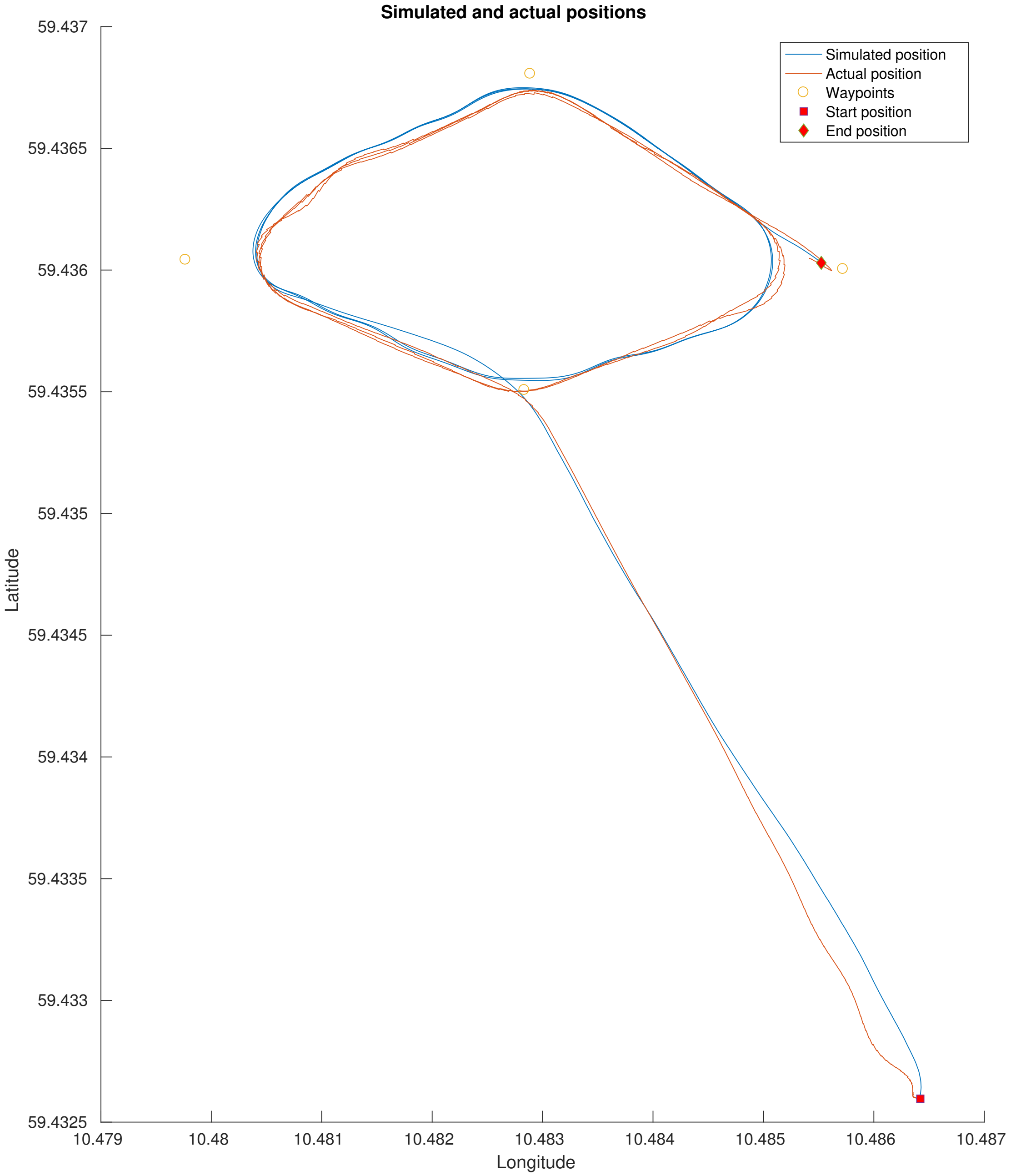


Figure 5.8: *Comparison of position and waypoints from sea trials and simulation, scenario 2.*

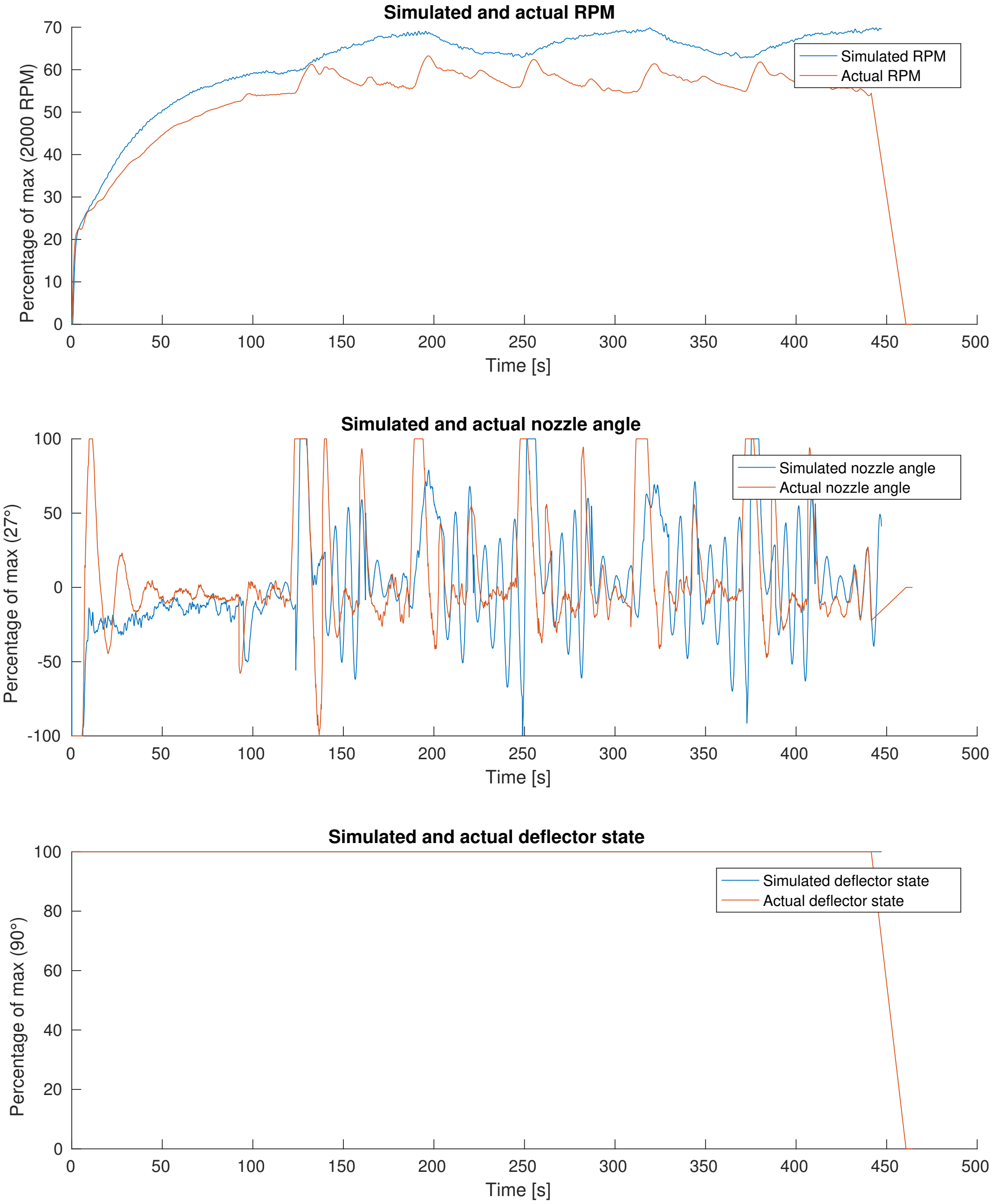


Figure 5.9: *Comparison of desired actuator states from sea trials and simulation, scenario 2.*

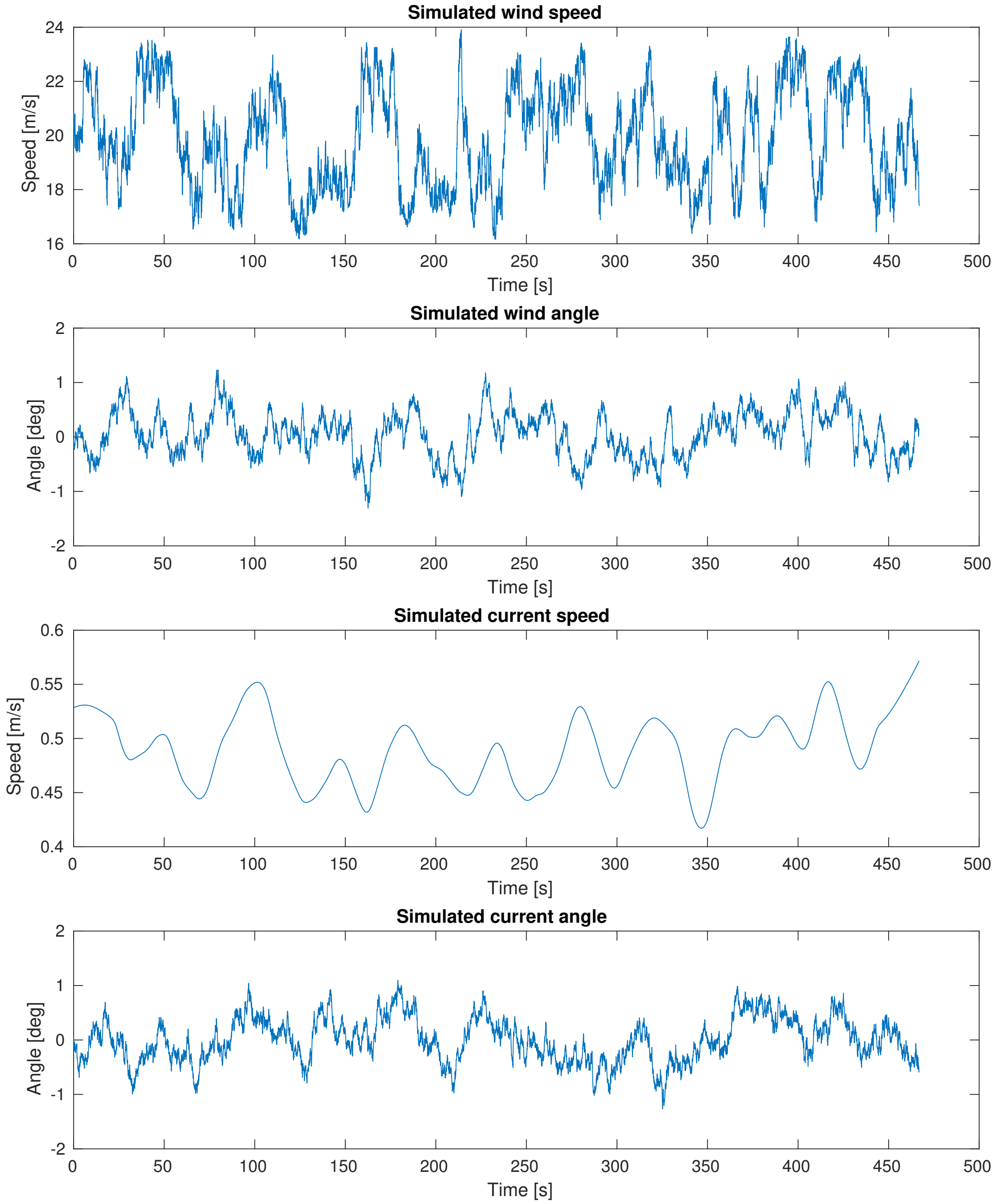


Figure 5.10: *Speed and angle for wind and current from simulation, scenario 2.*

Results from this scenario shows results similar to the *Calm Sea* scenario. The heading controller is not properly tuned for the simulation model, leading to oscillations in both heading and heading reference, while the speed controller performs quite well. It is obvious from Figure 5.7 that the wind and current speed affects the vessel speed, and that the integral term is able to compensate for this, reaching the reference speed after a certain time. As a result of this, the position plot in Figure 5.8 shows good waypoint guidance controller performance for both the simulated and the actual vessel.

5.3 Gale, Significant Wave Height 1m

Scenario 3 considers a simulated mission performed with a mean wind speed of $20m/s$, and a mean current speed of $0.5m/s$, with a significant wave height $H_s = 1m$, which gives the following results:

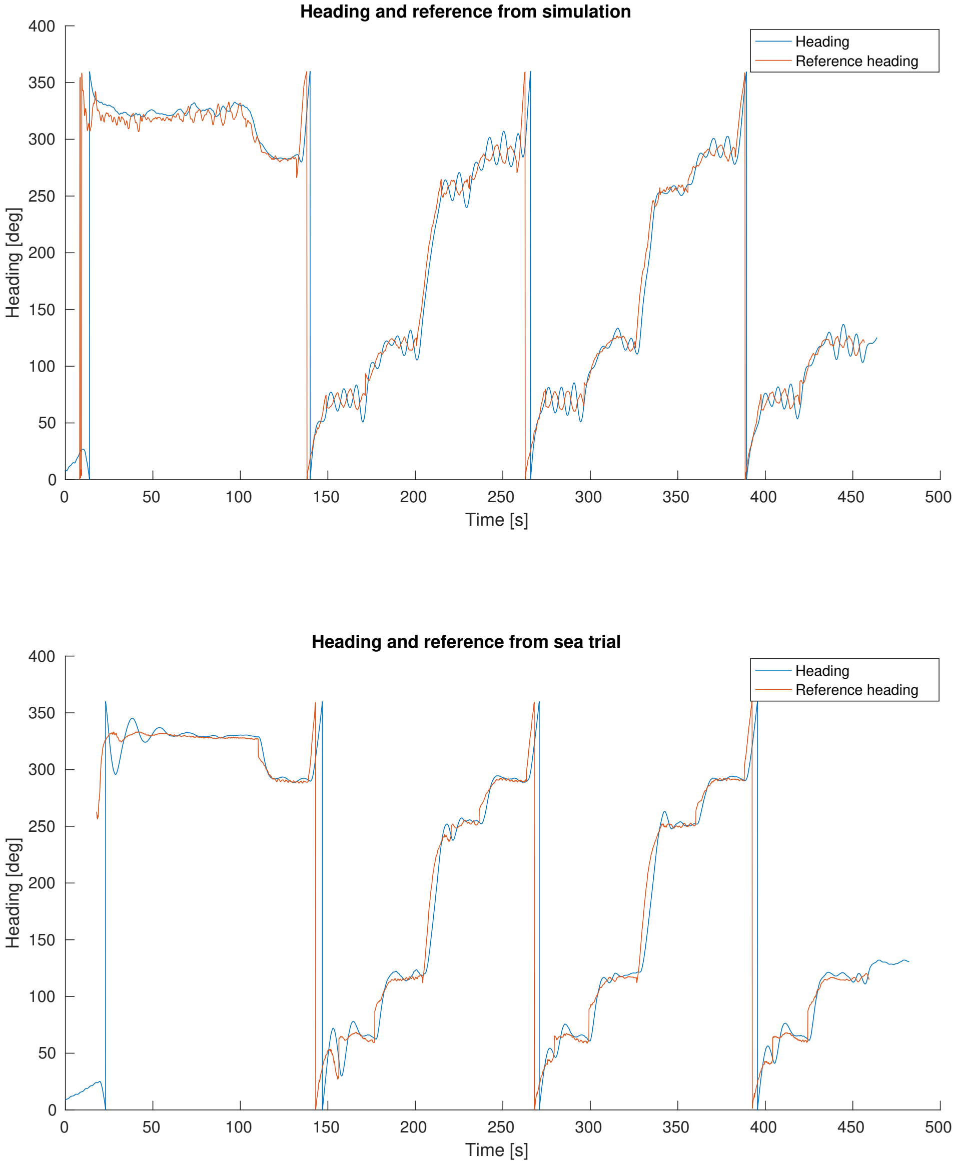


Figure 5.11: *Comparison of heading and reference heading from sea trials and simulation, scenario 3.*

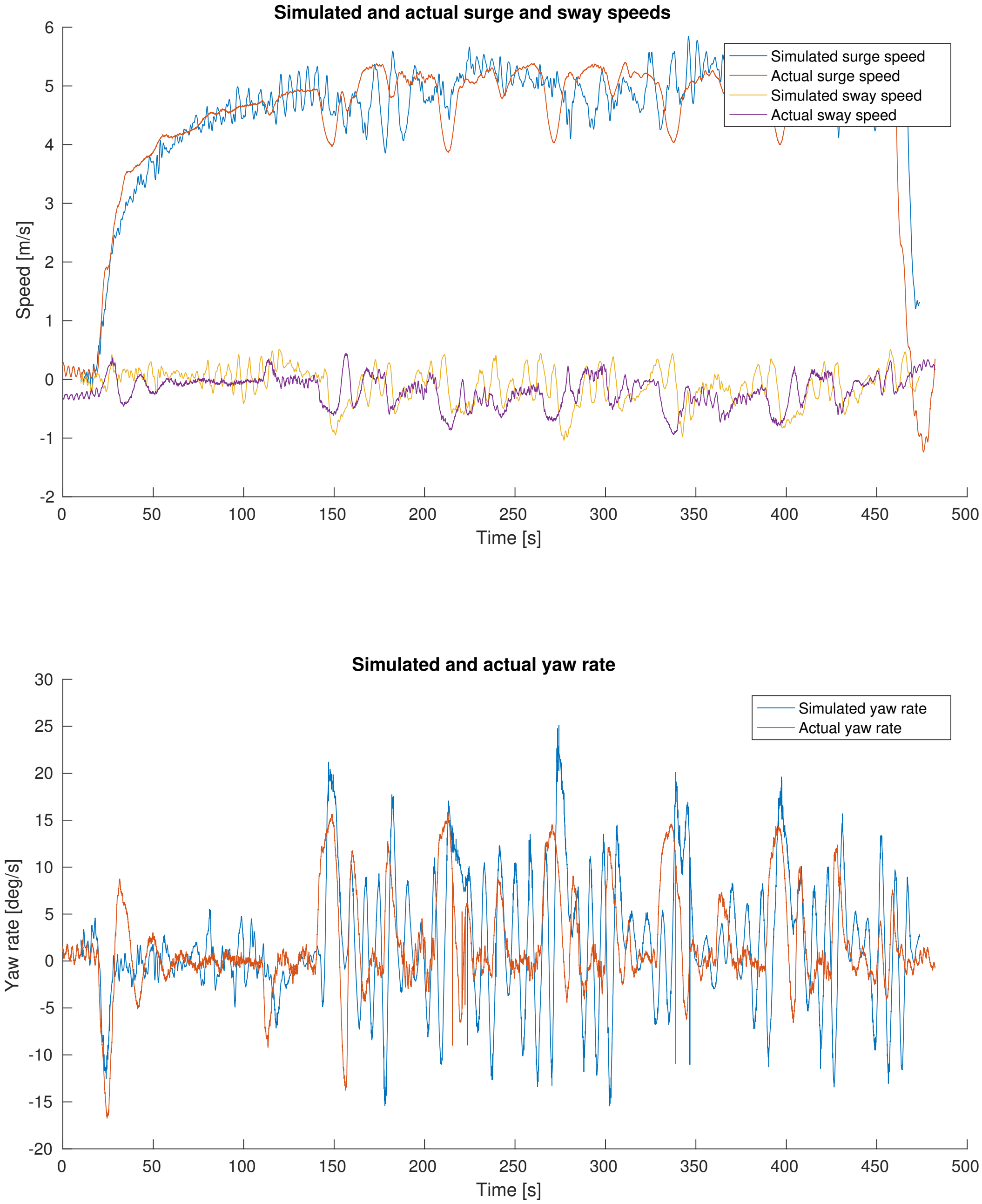


Figure 5.12: *Comparison of linear and angular velocities from sea trials and simulation, scenario 3.*

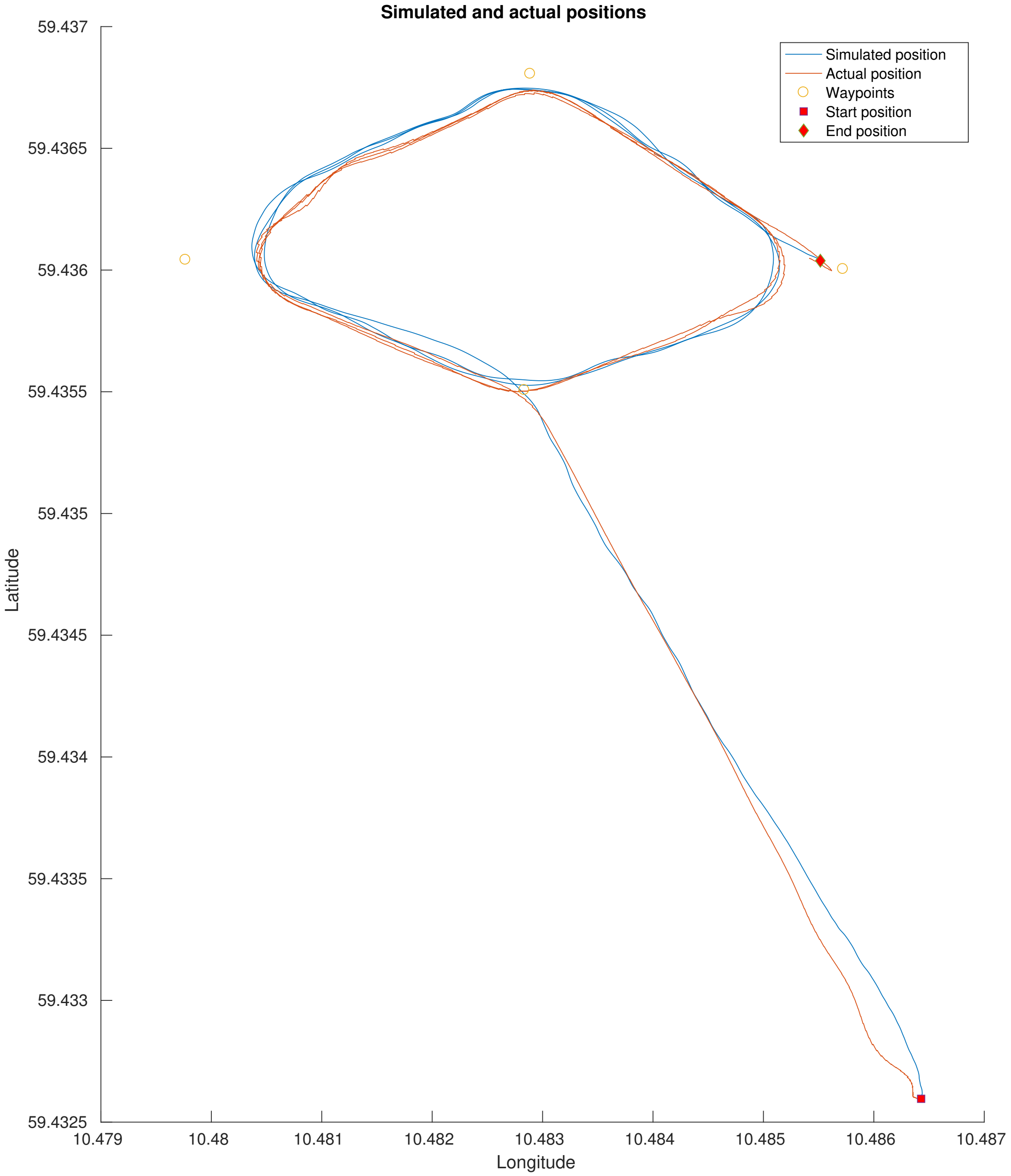


Figure 5.13: *Comparison of position and waypoints from sea trials and simulation, scenario 3.*

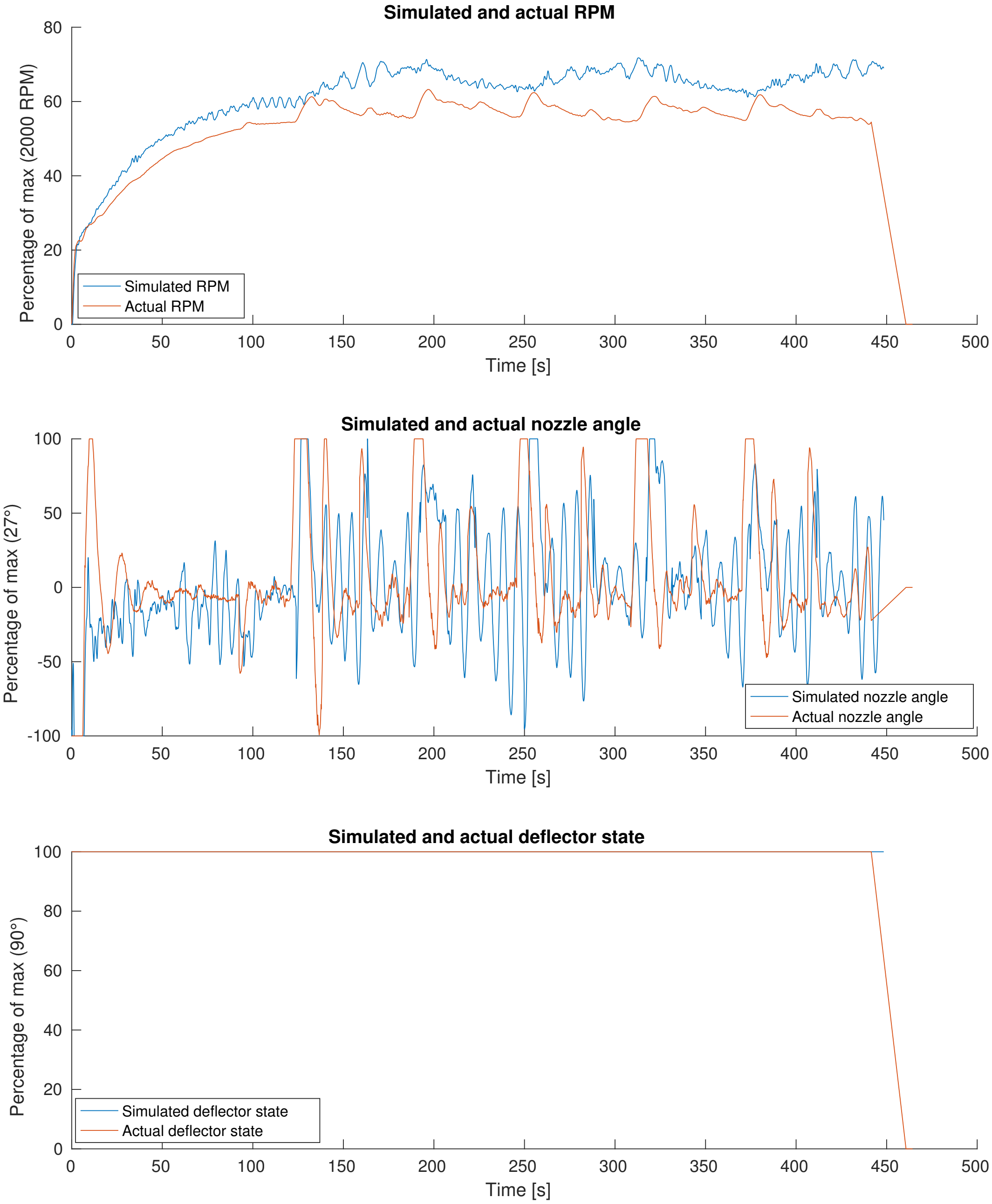


Figure 5.14: *Comparison of desired actuator states from sea trials and simulation, scenario 3.*

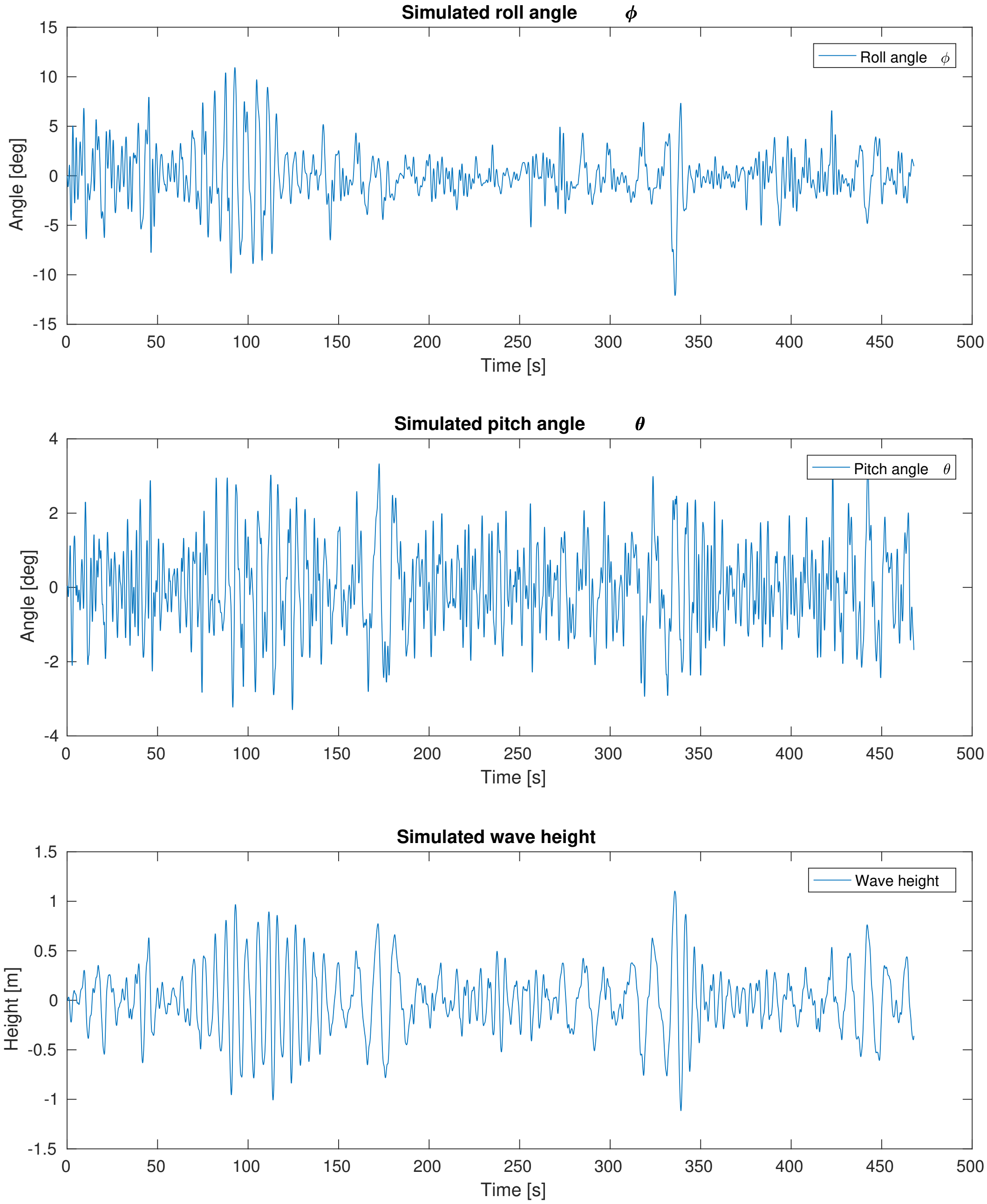


Figure 5.15: *Roll angle, pitch angle and wave height from simulation, scenario 3.*

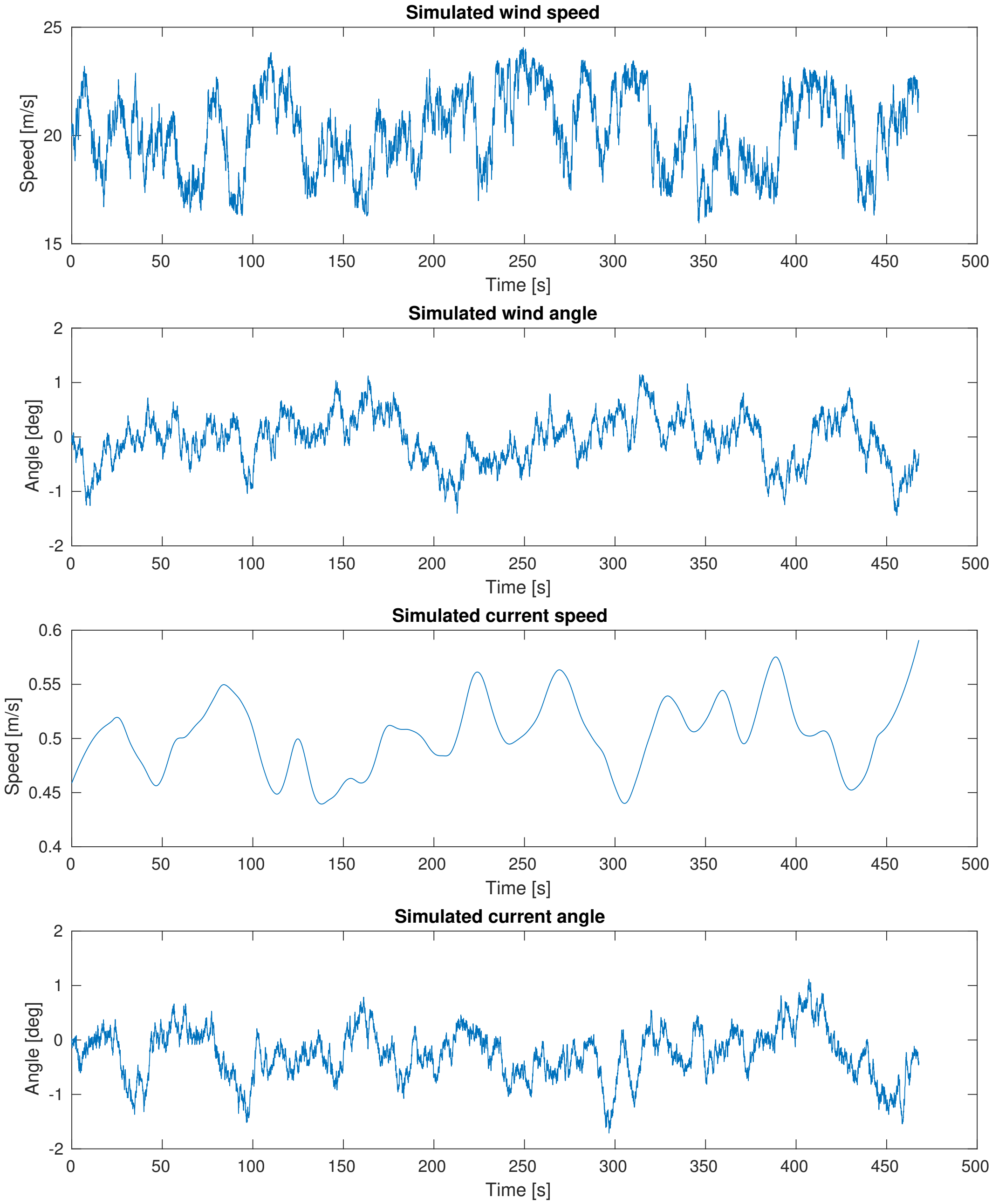


Figure 5.16: *Speed and angle for wind and current from simulation, scenario 3.*

This scenario introduces wave-induced forces and moments in the simulations. The simulator features an experimental module for simulation of wave effects, as RAO tables for Odin is not available, and the wave height over time is presented in Figure 5.15. In Figure 5.11, it can be seen that the wave forces and moments affect the reference heading, as the vessel position is influenced by the wave effects, giving a noisy reference heading for the simulated vessel. Nevertheless, the heading controller performs quite well, with the same oscillations as seen in the previous scenarios. From Figure 5.12 it is obvious that the waves affect the surge and sway speeds of the simulated vessel, and in Figure 5.14 it can be seen that the speed controller compensates for this by altering the engine RPM.

Chapter 6

Conclusion and Further Work

6.1 Conclusion

6 degrees of freedom (DOF) mathematical models of the vessel dynamics for the vessels Jolner and Odin have been developed, including actuator dynamics, hydrodynamic added mass and damping, restoring forces, fluid memory effects and interaction with wind, current and waves. The simulated vessel dynamics of Odin was compared with the actual vessel dynamics, and results shows a good model performance. For performance testing using a Hardware-In-The-Loop (HIL) simulator, the simulated surge and yaw dynamics are the most important, both of which can be seen to follow the actual vessel dynamics closely.

A prototype simulator for simulation of a general unmanned surface vehicle (USV) in 6 DOFs has been implemented in C++ using the Robot Operating System (ROS) as a framework. The prototype simulator utilizes the mathematical model of the USV, and includes modules for simulation of wind, current and waves. Additionally, to obtain the necessary HIL functionality, the prototype simulator includes a module for simulated sensors, including GPS, wind sensor, speed sensor and NavP, and a module for receiving actuator commands from the control system.

Different simulation scenarios have been conducted, using the control system on Odin in the loop during a waypoint guidance mission. These tests show that the interface between the simulator and the control system is working properly, the control system is able to receive sensor data from the simulator, and the simulator is able to receive the calculated actuator commands from the control system. Comparisons of the simulated and actual vessel performance during the same mission are presented, showing similar behaviour. The heading controller used in the control system is developed for the actual vessel, and as such it is not properly tuned for the simulation model. This leads to oscillations in the simulated vessel heading, a problem that could be solved by either tuning the heading controller or enhancing the simulation model accuracy. The speed controller used in the control system performs well for both the simulated and actual vessel, and simulation results shows that the water jets on Odin run at approximately the same revolutions per minute for the same reference speed. The simulated scenarios includes several different environmental conditions, ranging from no wind, current or waves to $20m/s$ wind, $0.5m/s$ current and significant wave height $H_s = 1m$. Simulation results indicates that the waypoint guidance controller used in Odin's control system performs satisfactory in the simulation model.

6.2 Further Work

In further work, the following aspects should be considered:

- **Verification of the simulated vessel dynamics for Jolner:** As logs from sea trials for Jolner is not available at the time being, verification of the mathematical model of Jolner has not been performed.

- **Implementation of realistic sensor noise models:** The simulated sensors described in this thesis are considered to be error-free.

- **Implementation of modules for simulation of possible fault situations:** One of the key properties of an autonomous USV is the ability to handle certain situations without interaction from a user. These situations could include a potential collision, engine malfunctions and loss of sensor signal.

- **Simulation of IMU data:** The HIL simulator developed in this thesis simulates filtered navigation data directly, and does not consider the IMU data. For testing and verification of the filter performance, the IMU data should be simulated.

- **Computation/estimation of Response Amplitude Operators (RAOs):** The simulated wave forces and moments acting on the vessels uses a simplified approach, giving a rough approximation. To obtain more accurate results, RAO tables for both vessels should estimated.

- **Computation of the fluid memory effects for both vessels** The simulation models uses the fluid memory effects of the Viknes 830 vessel, presented in Kjerstad (2010), and scales these according to the simulated vessel's dimensions. For more accurate results, the fluid memory effects of both vessels considered should be computed.

- **Improving the simulation model of Odin:** Comparisons of HIL simulation results and sea trials shows that the heading controller is properly tuned for the actual vessel, and slightly underdamped for the simulated vessel. For the heading controller used in Odin's control system to work properly, the simulation model of Odin should be improved with more accurate dynamics. Alternately, a more robust controller able to handle different vessel dynamics could be implemented, or the heading controller could simply be tuned for the simulation model.

Appendix A

Setting up the Simulator

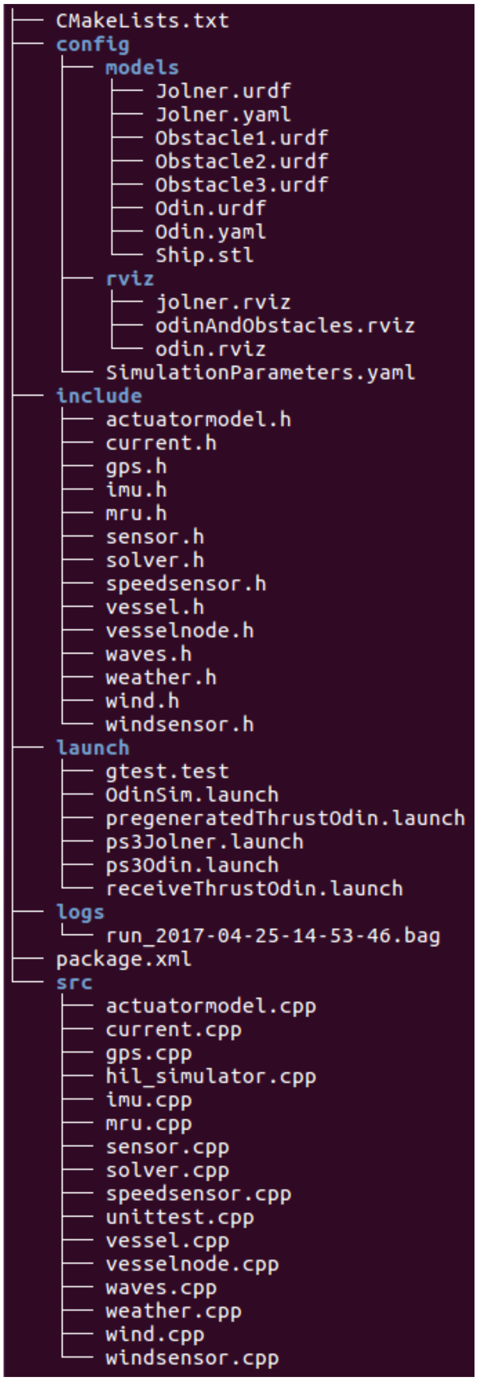
The implementation of the simulator system can be found at GitHub: `https://github.com/kjetilbl/hilsim`. The repository features packages for simulation of both the *Dynamics* and the *Surroundings* modules, in addition to several smaller packages made to facilitate for testing and development of the simulator. This appendix aims to give an overview of the packages utilized in the *Dynamics* module, and how to set these up for simulation. For details regarding the *Surroundings* module, see Børs-Lind (2017b).

A.1 Installation Requirements

In order to use the packages described above, there are several requirements. The user should run a working distribution of ROS on Ubuntu (see `http://wiki.ros.org`). Additionally, the 3D models of the vessels considered in this thesis are not available for distribution, and thus the user needs to obtain these models directly from FFI or Kongsberg Maritime, or alternatively use different 3D models from elsewhere.

A.2 Detailed Description of the Simulator Package

The simulator_prototype package implements a general 6 DOF surface vessel simulator. The equations of motion for a general surface vessel in 6 DOFs, as described in Chapter 2, is implemented in vessel.cpp, and the ROS subscribers and publishers for vessel and actuator states is implemented in vesselnode.cpp. To facilitate for simulation of several different vessels, the parameters estimated in Chapter 3 are uploaded to the ROS Parameter Server at launch, which is called by the simulator to obtain the parameters for a specific vessel. Mathematical models of the two vessels considered in this thesis are found in config/models/<VESSEL_NAME>.yaml, along with XML specifications of the 3D models used in the simulator, named <VESSEL_NAME>.urdf. Using the same approach, it is possible to specify environmental conditions, sensor frequencies, start position and step size used by the simulator in config/SimulationParameters.yaml. The 3D models used in this project are not included in this thesis, as they are not intended for public use.



```
├── CMakeLists.txt
├── config
│   ├── models
│   │   ├── Jolner.urdf
│   │   ├── Jolner.yaml
│   │   ├── Obstacle1.urdf
│   │   ├── Obstacle2.urdf
│   │   ├── Obstacle3.urdf
│   │   ├── Odin.urdf
│   │   ├── Odin.yaml
│   │   └── Ship.stl
│   ├── rviz
│   │   ├── jolner.rviz
│   │   ├── odinAndObstacles.rviz
│   │   └── odin.rviz
│   └── SimulationParameters.yaml
├── include
│   ├── actuatormodel.h
│   ├── current.h
│   ├── gps.h
│   ├── imu.h
│   ├── mru.h
│   ├── sensor.h
│   ├── solver.h
│   ├── speedsensor.h
│   ├── vessel.h
│   ├── vesselnode.h
│   ├── waves.h
│   ├── weather.h
│   ├── wind.h
│   └── windsensor.h
├── launch
│   ├── gtest.test
│   ├── OdinSim.launch
│   ├── pregeneratedThrustOdin.launch
│   ├── ps3Jolner.launch
│   ├── ps3Odin.launch
│   └── receiveThrustOdin.launch
├── logs
│   └── run_2017-04-25-14-53-46.bag
├── package.xml
└── src
    ├── actuatormodel.cpp
    ├── current.cpp
    ├── gps.cpp
    ├── hil_simulator.cpp
    ├── imu.cpp
    ├── mru.cpp
    ├── sensor.cpp
    ├── solver.cpp
    ├── speedsensor.cpp
    ├── unittest.cpp
    ├── vessel.cpp
    ├── vesselnode.cpp
    ├── waves.cpp
    ├── weather.cpp
    ├── wind.cpp
    └── windsensor.cpp
```

Figure A.1: *File structure of the simulator_prototype package.*

To facilitate for simulations of different vessels under different scenarios, several launch files are included in the package. These launch files start up several different ROS Nodes and a ROS Master, as well as loading the vessel parameters and 3D model descriptors to the ROS Parameter Server. When initiated, the simulator publishes and subscribes to messages using the following ROS Topics and message types:

Topic	Message type	Action
`/log/wind`	`geometry_msgs::Twist`	Publish
`/log/current`	`geometry_msgs::Twist`	Publish
`/log/state`	`geometry_msgs::Twist`	Publish
`/log/thrust`	`geometry_msgs::Twist`	Publish
`/log/velocity`	`geometry_msgs::Twist`	Publish
`/tf`	`tf::TransformBroadcaster`	Publish
`/sensors/mru/position`	`geometry_msgs::Twist`	Publish
`/sensors/mru/velocity`	`geometry_msgs::Twist`	Publish
`/sensors/windSensor`	`geometry_msgs::Twist`	Publish
`/sensors/speedSensor`	`geometry_msgs::Twist`	Publish
`/sensors/gps`	`simulator_messages::Gps`	Publish
`/input/thrust`	`geometry_msgs::Twist`	Subscribe
`/input/actuators`	`simulator_messages::ActuatorMessage`	Subscribe
`/input/environment`	`simulator_messages::ActuatorMessage`	Subscribe

Table A.1: *Overview of the ROS Topics used by the simulator.*

A.3 Package Overview

The following packages are developed during the work on this thesis:

- `ps3_joy_thrusters`: This package is used to generate actuator messages for Jolner. It subscribes to the topic `joy` to receive input from the PlayStation3-controller connected, and uses the controller message to generate an *ActuatorMessage*, where the thruster RPM corresponds to the vertical joystick positions. This is then published on the topic `input/actuators`.

- `ps3_thrust_input`: This package is used to generate control forces and moments for Odin. It subscribes to the topic `joy` to receive input from the PlayStation3-controller connected, and uses the controller message to generate a *Twist*-message, containing values for surge and sway force, and yaw moment. This is then published on the topic `input/thrust`.
- `simulator_messages`: This package contains no source code, it's sole purpuse is to generates definitions of all the custom messages used in this project. This includes messages for actuator states, setting environmental state, GPS data and obstacle command.
- `simulator_prototype`: This package contains the simulator described in this thesis.
- `txt_to_actuator_info`: This package reads a *.txt*-file with actuator input logs from actual sea trials, and publishes the inputs on the topic `input/actuators` at a given rate. This gives the possibility of comparing results from sea trials and simulations using the same actuator inputs.

A.4 Running the Simulator

The following steps should be performed in order to get the simulator running for the first time:

1. Install Ubuntu on the computer intended for simulation. (`https://www.ubuntu.com/download/desktop/install-ubuntu-desktop`).
2. Install the latest distribution of ROS for Ubuntu (`http://wiki.ros.org/kinetic/Installation/Ubuntu`).
3. Open the terminal and create a ROS workspace:

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/src
```

4. Clone the HIL-simulator repository into the `src`-folder:

   ```
   $ git clone https://github.com/kjetilbl/hilsim
   ```

5. Build the `simulator_messages` package first, as several simulator packages depends on this package:

   ```
   $ cd .. && catkin_make --pkg simulator_messages
   ```

6. Build the workspace:

   ```
   $ catkin_make
   ```

7. Source your new `setup.*sh` file:

   ```
   $ source devel/setup.bash
   ```

8. Obtain 3D models of Odin and/or Jolner, copy these into `/home` in the computer used for simulation.

9. Set the desired environmental conditions for the simulation in `config/SimulationParameters.yaml`.

10. The simulator should now be ready for use. Test it by launching a simulation of Odin using pregenerated actuator-inputs:

    ```
    $ roslaunch simulator_prototype pregeneratedThrustOdin.launch
    ```

Bibliography

Baracos, P., Murere, G., Rabbath, C. A., and Jin, W. (2001). Enabling PC-based HIL simulation for automotive applications. pages 721–729.

Bertram, V. (2008). Unmanned surface vehicles-a survey. *Skibsteknisk Selskab, Copenhagen, Denmark*, 1:1–14.

Blendermann, W. (1994). Parameter identification of wind loads on ships. *Wind engineering and industrial aerodynamics*, 51:339–351.

Børs-Lind, K. S. (2017a). Prototype of HIL test platform for autonomous USV, simulation and visualization of vessel surroundings. Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway.

Børs-Lind, K. S. (2017b). *Specification of Simulated Environment and User Interface for HIL Testing of USV.* Specialization Project Report, Norwegian University of Science and Technology, Trondheim, Norway.

Davenport, A. G. (1977). The prediction of risk under wind loading. In *Proceedings of the 2nd International Conference on Structural Safety and Reliability*, pages 511–538, Munich, Germany.

Egeland, O. and Gravdahl, T. (2003). *Modeling and Simulation for Automatic Control.* Marine Cybernetics, 2nd edition.

Faltinsen, O. M. (2005). *Hydrodynamic of High-Speed Marine Vehicles.* Cambridge University Press.

Faruque, M. O. O. and Dinavahi, V. (2010). Hardware-in-the-loop simulation of power electronic systems using adaptive discretization. *IEEE Transactions on Industrial Electronics*, 57(4):1146–1158.

Field, T., Leibs, J., and Bowman, J. (2015). Rosbag. *http://wiki.ros.org/rosbag*, Accessed: 18.05.17.

Foote, T. and Purvis, M. (2010). Standard units of measure and coordinate conventions. *Open Source Robotics Foundation*, Accessed: 09.03.2017, at http://www.ros.org/reps/rep-0103.html.

Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, Ltd, 1st edition.

Fossen, T. I. (2015). *Mathematical Models of Ships and Underwater Vehicles*, pages 701–706. Springer London, London.

Gade, K. (2009). Introduction to inertial navigation and Kalman filtering. In *Tutorial for IAIN World Congress*, Stockholm, Sweden.

Halvorsen, H. (2008). Dynamic positioning for unmanned surface vehicles. Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway.

Hoerner, S. F. (1965). *Fluid Dynamic Drag*. Hartford House.

Kjerstad, Ø. K. (2010). Weather-optimal positioning control for underactuated USVs. Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway.

Langley, R. (1995). NMEA 0183: A GPS receiver. *GPS world*.

Liu, Z., Zhang, Y., Yu, X., and Yuan, C. (2016). Unmanned surface vehicles: An overview of developments and challenges. *Annual Reviews in Control*, 41:71 – 93.

Meeussen, W. (2010). Coordinate frames for mobile platforms. *Open Source Robotics Foundation*, Accessed: 09.03.2017, at http://www.ros.org/reps/rep-0105.html.

National Imagery and Mapping Agency (2000). Department of Defense, World Geodetic System, 1984: Its Definition and Relationships with Local Geodetic Systems. Technical report, National Geospatial-Intelligence Agency, USA.

NORSOK. Norsok standards. *https://www.standard.no/en/sectors/energi-og-klima/petroleum/norsok-standards*, Accessed: 20.12.2016.

Pierson, W. J. and Moskowitz, L. (1963). A proposed spectral form for fully developed wind seas based on the similarity theory of s. a. Kitaigorodskii. Technical report, New York University, School of Engineering and Science.

Pugi, L., Malvezzi, M., Tarasconi, A., Palazzolo, A., Cocci, G., and Violani, M. (2006). HIL simulation of WSP systems on MI-6 test rig. *Vehicle System Dynamics*, 44(sup1):843–852.

ROS (2014). The robot operating system. *http://www.ros.org/about-ros/*, Accessed: 18.05.17.

Skjetne, R., Smogeli, Ø. N., and Fossen, T. I. (2004). A nonlinear ship manoeuvering model: Identification and adaptive control with experiments for a model ship. *Modeling, Identification and Control*, 25(1):3–27.

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

The Society of Naval Architects and Marine Engineers (1950). Nomenclature for treating the motion of a submerged body through a fluid. *Technical and Research Bulletin*, (1–5).

Vik, B. (2009). Integrated satellite and inertial navigation systems. *Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway.*

W. F. Durand and G. R. McDermott (1894). An approximate formula for the wetted surface of ships. In *General Meeting of The Society of Naval Architects and Marine Engineers*, New York, USA.

Yedamale, P. (2003). Brushless DC (BLDC) motor fundamentals. Technical report, Microchip Technology Incorporated, USA.

Ødegaard, E. (2016). *Specification and First Prototype for Simulation of Autonomous USV*. Specialization Project Report, Norwegian University of Science and Technology, Trondheim, Norway.