NTNU

Norwegian University of
Science and Technology

Development of a Dynamic Positioning System for the ReVolt Model Ship

Henrik Alfheim

Kjetil Muggerud

Master of Science in Cybernetics and Robotics

Submission date: June 2017

Supervisor: Morten Breivik, ITK

Co-supervisor: Edmund Førland Brekke, ITK
Egil Eide, IET
Øystein Engelhardtsen, DNV GL

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Preface

This thesis is the concluding part of the authors' 2-year Masters degree in Cybernetics and Robotics, graduating from the Norwegian University of Science and Technology (NTNU) in June 2017.

Autonomous Surface Vehicles is chosen as a topic based on personal interests and its increasing relevance for use in transportation, surveillance and so forth. We find our work highly relevant and hope that our contribution on creating a platform on which autonomous systems can be implemented and tested, will help the industry move forward.

We would like to thank our academic supervisors, Morten Breivik, Egil Eide and Edmund Førland Brekke, for all the help and support throughout the thesis. The knowledgeable discussions throughout the process have motivated continuous learning and given valuable perspectives to our research. Our gratitude also goes to Øystein Engelhardtsen and Hans Anton Tvete at DNV GL for the opportunity to work on ReVolt in this thesis.

Furthermore, we want to thank Andreas Lindahl Flåten and Erik Wilthil for knowledgeable inputs on programming and Kalman Filter. We would also like to thank Stefano Bertelli at NTNU, who assisted on practical matters of the research and in making the experimental tests of ReVolt possible.

A special thanks goes out to both our partners and family, for their patience and valuable inputs. This has been a great thesis to work on, where we have learned a lot and withstood the long days required to complete the experimental tests. We are ready and motivated for whatever lies ahead, and hope that the future users of ReVolt find working with ReVolt as inspiring as we have.

Trondheim, June 2017

Henrik Lemcke Alfheim
Kjetil Muggerud

Abstract

Autonomous surface vehicles are still in the early stages of development, and show great potential. These vessels can be used in different scientific and commercial operations, outperforming manned vessels in safety, endurance and cost efficiency.

A Dynamic Positioning (DP) control system has been developed for, and implemented on, a scale model of DNV GLs concept ship ReVolt. This model-scale ship, called ReVolt, is used as a test platform for an autonomous ferry, as well as an early test platform for sensors and control systems for autonomous vessels. As a result, this thesis primarily focus' on the functionality and implementation of the components and control system required to achieve DP capabilities on ReVolt.

A DP control system has been developed, consisting of a 3-Degree of Freedom reference filter and a Proportional-Integral-Derivative (PID) controller with a model-based feedforward from the reference. DNV GL's thrust allocation has been implemented and an Error-state Kalman Filter is developed. The DP control system is implemented on ReVolt's onboard computer, which runs the Robot Operating System (ROS) on top of a Linux shell. The control system has furthermore been simulated in MATLAB to develop and test its functionality.

Extensive field tests has been completed, where the main objective was to achieve station keeping and low-speed maneuvering capabilities on ReVolt. Different setups in the thrust allocation and controller have been assessed to determine the best overall setup for ReVolt. In addition, the control system has been applied to a docking scenario to asses its applicability in docking.

Sammendrag

Autonome overflatefartøy er fremdeles på et tidlig utviklingsstadium, men har et stort potensiale. Disse kjøretøyene kan bli brukt i ulike forskning og kommersielle operasjoner, som vil utkonkurerer bemannede kjøretøy p sikkerhet, utholdenhet og kostnads effektivitet.

Et dynamisk posisjons (DP) kontrollsystem har blitt utviklet for, og implementert på, en skalamodell av DNV GLs konseptskip ReVolt. Dette skalamodellskipet, fra nå av kalt ReVolt, er brukt som en testplattform for en autonom ferge og er i tillegg en tidlig testplattform for sensorer og kontrollsystemer tiltenkt autonome fartøy. Som et resultat av dette fokuserer oppgaven hovedsakelig på funksjonalitet og implementering av komponenter og algoritmer nødvendig for å oppnå DP funksjonaliteter på ReVolt.

Et DP kontrollsystem har blitt utviklet, bestående av et referansefilter i tre frihetsgrader og en Proporsjonal-Integrasjon-Derivasjon (PID) kontroller med en modellbasert foroverkobling fra referansen. DNV GLs trust allokering har blitt implementert og et Kalman Filter er utviklet. DP kontrollsystemet er implementert på ReVolts kjørecomputer, som kjører Robot Operating System (ROS) oppå operativsystemet Linux. Kontrollsystemet har videre blitt simulert i MATLAB for å bevise dets funksjonalitet.

Omfattende felttester har blitt utført, hvor hovedformålet var å oppnå stasjonær posisjonering og lavhastighets manøvreringsevner på ReVolt. Ulik konfigurering av trust allokeringen og kontrolleren er utprvd for å fastslå konfigurasjonen som gir best ytelse for ReVolt. Kontrollsystemet har videre blitt anvendt på et dokkingscenario for å vurdere dets anvendelighet for dokking.

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AHRS	Attitude Heading Reference System
ARPA	Automatic Radar Plotting Aid
BODY	Body-fixed
CG	Center of Gravity
CO	Center of Origin
CSV	Comma-Separated Values
DOF	Degree of Freedom
DP	Dynamic Positioning
ECEF	Earth-Centered-Earth-Fixed
EGNOS	European Geostationary Navigation Overlay Service
EOL	End-of-Life
ESC	Electronic Speed Controller
ESKF	Error-State Kalman Filter
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GUI	Graphical User Interface
HIL	Hardware In the Loop
I/O	Input/Output
IADC	Integral of Absolute Differentiated Control

IAE	Integrated Absolute Error
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IP	Internet Protocol
LIDAR	Light Detection and Ranging
LLA	Longitude-Latitude-Altitude
LOS	Line of Sight
LPP	Length Between Perpendiculars
LQR	Linear Quadratic Regulator
NED	North-East-Down
NMEA	National Marine Electronics Association
NTNU	Norwegian University of Science and Technology
OBC	Onboard Computer
PID	Proportional-Integral-Derivative
PWM	Pulse Width Modulation
ROS	Robot Operating System
RTK	Real-Time Kinematic
SBAS	Satellite-Based Augmentation System
STT	Stadt Towing Tank
TA	Thrust Allocation
TCP	Transmission Control Protocol
USB	Universal Serial Bus
WADAM	Wave Analysis by Diffraction and Morison

Nomenclature

{B}	Denotation of the value being a bias signal
{N}	Denotation of the value being a noise signal
{b}	Denotation of the Body-fixed frame
{d}	Denotation of the value being a desired signal from the reference filter
{e}	Denotation of the Earth-Centered-Earth-Fixed frame
{g}	Denotation of the Geodetic Longitude-Latitude-Altitude frame
{n}	Denotation of the North-East-Down frame
{ref}	Denotation of the value being a reference signal from the operator
$\boldsymbol{\eta}$	Position and orientation vector
$\boldsymbol{\nu}$	Linear and angular velocity vector
$\boldsymbol{\tau}$	Forces and moments vector
$\mathbf{p}$	Position vector
$\boldsymbol{\Theta}$	Vector of Euler angles
$\mathbf{v}$	Linear velocity vector
$\boldsymbol{\omega}$	Angular velocity vector
$\mathbf{f}$	Force vector
$\mathbf{m}$	Moment vector
N, E, D	North, East, Down position in the North-East-Down frame
x, y, z	Body-fixed position in the x, y, z direction
ϕ, θ, ψ	Euler angle about the x, y and z axis

u, v, w	Body-fixed linear velocity in the x, y and z direction
p, q, r	Body-fixed angular velocity about the x, y and z axis
X, Y, Z	Body-fixed force in the x, y and z direction
K, M, N	Body-fixed moment about the x, y and z axis
l, μ	Longitude and Latitude
$\mathbf{M}$	Inertia Matrix
$\mathbf{M_{RB}}$	Rigid-body Inertia Matrix
$\mathbf{M_A}$	Added Mass Matrix
$\mathbf{C}$	Coriolis-Centripetal Matrix
$\mathbf{C_{RB}}$	Rigid-body Coriolis-Centripetal Matrix
$\mathbf{C_A}$	Added Mass Coriolis-Centripetal Matrix
$\mathbf{D}$	Damping Matrix
$\mathbf{D_L}$	Linear Damping Matrix
$\mathbf{D_n}$	Non-linear Damping Matrix
$\mathbf{\Delta}$	Relative damping ratio matrix
$\mathbf{\Omega}$	Natural frequency matrix
K_P,K_D,K_D	Controller gains for the Proportional-Integral-Derivative controller
$\mathbf{F}$	Thrust force vector
F_i	Thrust force for thruster i
$\boldsymbol{\alpha}$	Thruster angle vector
α_i	Thruster angle for thruster i
l_{x_i},l_{y_i}	Distance from the Center of Gravity to thruster i in the x and y direction
$\boldsymbol{q}$	Quaternion vector, where $\boldsymbol{q} = [q_w, \boldsymbol{q}_v]^{\mathrm{T}} = [q_w, q_x, q_y, q_z]^{\mathrm{T}}$
Δt	Variable time step
$\mathbf{I}$	Identity matrix
$\mathbf{R}$	Rotation matrix
$\mathbf{S}(\boldsymbol{x})$	Skew matrix of a vector $\boldsymbol{x}$
h	Fixed time step
n	Degrees of Freedom
s(x), c(x)	Sine and cosine of x

Chapter 1

Introduction

This chapter provides background and motivation for the work on this thesis, a problem description and the thesis' relevance to the local community. The authors contributions are described as well as an outline for this report.

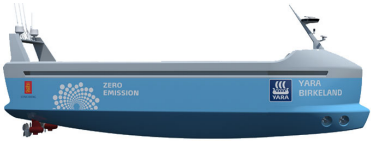
1.1 Background

During the past century automation has revolutionized numerous industries by replacing manual labor with automated machinery, and in recent years automation has entered our homes in the form of lawn mowers, vacuum cleaners or even the house itself. A shift from automated systems, which specialize on one task, to autonomous systems, which is capable of self governing complex tasks without human interaction, has emerged. Examples of autonomous systems are Google's self driving cars and the Mars rover Curiosity.

The automation of ships began back in the early 1900's when the American entrepreneur Elmer Sperry's gyrocompass invention made it possible to get a reliable measurement of the ships' heading (Breivik, 2010). A few years later, in 1922, the Russian-born electrical engineer Nicolas Minorsky wrote a paper with a theoretical analysis of the Proportional-Integral-Derivative (PID) controller for autopilots in ships. Simultaneously Sperry conducted field trials with his so-called gyropilot, popularly termed "Metal Mike" which proved to be a success. Although unaware of each other, Sperry and Minorsky laid the foundation for the successful application of ship autopilots in the following years.

The next major advances in control-related ship technology came in the 1960's and 70's with computer based bridge control, power management systems, Automatic Radar Plotting Aid (ARPA) and Dynamic Positioning (DP). Norway having the fourth largest trading fleet and a booming offshore industry at the time, played a key role in many of these advances. This has resulted in an advanced Norwegian ship automation industry, with

(a) Stella. Courtesy of Rolls-Royce.



(b) Yara Birkeland. Courtesy of Yara.

Figure 1.1: Rolls-Royce and Kongsberg Maritime's flagship projects.

companies such as Rolls-Royce and Kongsberg Maritime being key technology suppliers within the ship market.

Inspired by Google's self driving cars, the industry now focus on taking the next leap. Currently, both Rolls-Royce and Kongsberg Maritime, through respectively Stella and Yara Birkeland, are working towards being the first company to develop an autonomous ship for industrial use. Stella is a double ended ferry that will have additional equipment installed for autonomous testing purposes (Rolls-Royce, 2016), and Yara Birkeland proclaimed to be the first fully electric and autonomous container ship (Yara, 2017). Both ships are shown in Figure 1.1.

1.2 Motivation: Autonomous Ferry

There are many cities built around rivers which have small islands in the city center. To ease the access for pedestrians, a small autonomous passenger ferry can be advantageous compared to bridges, both due to costs and navigability for other vessels. There are currently several cable driven ferries in Norway, a few shown in Figure 1.2, which potentially can be replaced by autonomous versions.

(a) Cable ferry Nesøya - Brønnøya, Norway.

(b) Cable ferry Espevær - Bømlo, Norway.

Figure 1.2: Cable ferries operational in Norway.

In Trondheim, the possibility of building a footbridge from Ravnkloa to Fosenkaia is considered. However this would create an obstacle for the boat traffic in the river and impair the maritime and cultural values the river represents for Trondheim. As a result of this, the idea of an autonomous ferry to cross the river was brought to life. This ferry will travel between Ravnkloa and Venstre Kanalkai, as shown in Figure 1.3 (Stensvold, 2016).

Figure 1.3: Map of the planned route for the autonomous ferry. Courtesy of Egil Eide.

The idea is that the ferry will be operational for most of the day and be a ”ferry on demand”. The stretch across the river is about 95 meters and the ferry will only sail across the river when there are passengers on board or it has been ”demanded” by a passenger on the other side. The ferry is planned to have up to 12 passengers, including bikes, and be designed to accommodate wheelchairs and prams, see Figure 1.4. The ferry is planned to be operational by 2019, having gone through three phases of design and testing. The first of these three phases includes the development and testing of a control system and sensors on the experimental platform ReVolt.

1.2.1 Extension of Previous Work

During the summer 2016 the authors worked on interfacing an Onboard Computer (OBC) with actuators and sensors on the model ship ReVolt. Manual control and a rudimentary heading controller, using a Inertial Navigation System (INS), was achieved. The work continued into a joint project during the fall 2016, which resulted in the acquirement of an additional Global Navigation Satellite System (GNSS) and implementation of a simple DP control system (Alfheim and Muggerud, 2016). Furthermore, during field tests ReVolt achieved station keeping abilities. This could however be improved and extended to include maneuvering which is one of the objectives of this thesis.

Figure 1.4: Conceptual image of the autonomous ferry in Trondheim, seen from Vestre Kanalkai. Courtesy of Svein Aanond Aanondsen, NTNU.

1.3 Problem Description

This thesis considers the problem ”Development of a Dynamic Positioning System for the ReVolt Model Ship”. This work includes:

- Improve the current DP system for the model ship ReVolt, including reference filter, thruster allocation, motion controller and observer.
- A simulation of the DP system, using CyberShip II as a ship model, for proof of concept.
- Improve the system architecture of the system on ReVolt to allow for a easier user experience.
- Perform experimental sea trials to obtain results and test the DP system of ReVolt.

1.4 Contributions

The thesis has generated a lot of media attention around the autonomous ferry project and unmanned vessels in general. A summary of all the media coverage created by this thesis is found in Appendix A.

The main contributions are listed below:

- Improvement of the reference filter to include a saturating element.
- Creation of a 3-Degree of Freedom (DOF) mathematical model of ReVolt to improve the motion controller with a model-based reference feedforward term.
- Improvement of the thrust allocation to include the dynamics of thrusters.

- Development of a simulation using the ship model of CyberShip II in MATLAB to verify that each part of the DP system works as expected.
- Implementation of the improved DP system on the OBC in ReVolt. This included translating MATLAB code into Python and adding the necessary code to make the algorithms work together in another environment.
- Assessment of weaknesses and strengths of the navigational sensors used in the observer onboard ReVolt.
- Develop the software structure and a Graphical User Interface (GUI) to make it easier to operate ReVolt.
- Execution of numerous experiments in Trondheim harbour to assess the performance of the improved control system of ReVolt, where station keeping and low-speed maneuvering was achieved.
- Participation in a live demonstration of unmanned surface and sub-surface vehicles at the Norwegian University of Science and Technology (NTNU) Ocean Week on the 3rd May 2017.
- Suggestion of future work which can be done with ReVolt on its path to becoming an autonomous vessel.

1.5 Outline

The report is organized in the following manner:

- Chapter 2 is an introduction of the background theory which is used throughout the report.
- Chapter 3 goes through the theory for the guidance system which is used in the DP system.
- Chapter 4 goes through the theory for the control system, more specifically the motion controller and the thrust allocation.
- Chapter 5 introduces the navigation system, which includes a Kalman Filter, wild point filter and lever arm compensation.
- Chapter 6 describes the sensors and actuators on the experimental platform ReVolt.
- Chapter 7 describes how the components and system structure is implemented.
- Chapter 8 describes the creation of the mathematical model of ReVolt.
- Chapter 9 describes the result from the simulation of the DP system.
- Chapter 10 describes the performance of the observer and the experimental results of sea-trials with ReVolt.
- Chapter 11 and 12 are a discussion and conclusion of the thesis as a whole.

Chapter 2

Theoretical Background

The theoretical background needed in this report is covered in this chapter.

2.1 Reference Frames

The kinematic relationship for a vessel can be described in various reference frames. The most common frames used for terrestrial navigation include a global system, a local geographic system and lastly a system fixed to the vessel itself.

The frames, as seen Figure 2.1, are:

- **ECEF** Earth-Centered-Earth-Fixed (ECEF) frame is a Cartesian coordinate system which rotates around the Earth's spin axis. A fixed point on the Earth's surface can be represented by a fixed set of coordinates denoted $\{e\}, \mathbf{P_e} = (x_e, y_e, z_e)^\mathsf{T}$. ECEF can also be represented as a geodetic coordinate frame Longitude-Latitude-Altitude (LLA), which is widely used in GNSS-based navigation. It characterizes a coordinate point near the Earth's surface with longitude, latitude and altitude denoted by $\{g\}, \mathbf{P_g} = (l, \mu, h)^\mathsf{T}$
- **NED** The local North-East-Down (NED) frame is defined relative to the Earth's reference ellipsoid. The z-axis points downward perpendicular to the tangent plane, and the x-axis points towards the true north. This makes it intuitive for local navigation and is denoted by $\{n\}, \mathbf{P_n} = (x_n, y_n, z_n)^\mathsf{T}$
- **BODY** The Body-fixed (BODY) frame is moving and rotating with the vehicle, where usually the x-axis points in the forward direction, the y-axis to the right and the z-axis downward. Denoted by $\{b\}, \mathbf{P_b} = (x_b, y_b, z_b)^\mathsf{T}$

The following notation will be used to describe position, linear velocities and angular velocities (Fossen, 2011):

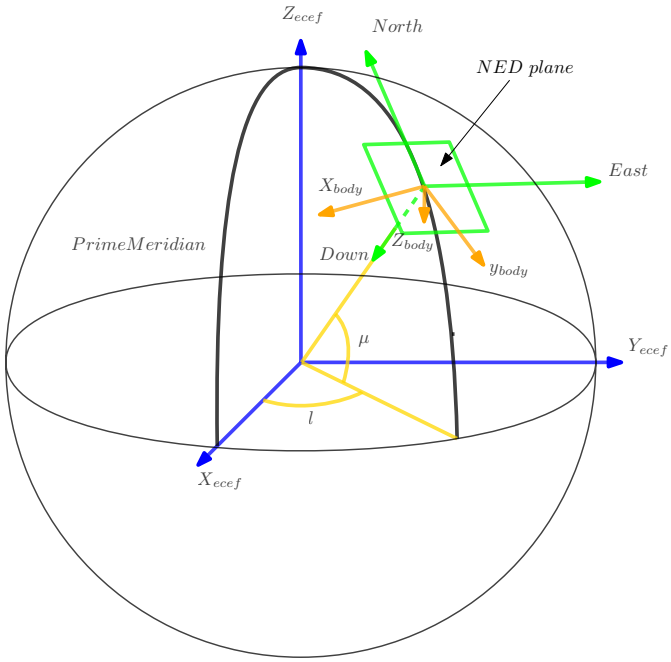


Figure 2.1: The different reference frames. Blue is ECEF, green is NED, orange is BODY and yellow is latitude and longitude.

$$
\begin{aligned}
\mathbf{v}^e_{b/n} &= \text{linear velocity of the point } o_b \text{ with respect to } \{\text{n}\} \text{ expressed in } \{\text{e}\} \\
\boldsymbol{\omega}^b_{n/e} &= \text{angular velocity of } \{\text{n}\} \text{ with respect to } \{\text{e}\} \text{ expressed in } \{\text{b}\} \\
\mathbf{f}^n_b &= \text{force with line of action through the point } o_b \text{ expressed in } \{\text{n}\} \\
\mathbf{m}^n_b &= \text{moment about the point } o_b \text{ expressed in } \{\text{n}\} \\
\mathbf{p}^n_{n/b} &= \text{distance from } \{\text{n}\} \text{ to } \{\text{b}\} \text{ expressed in } \{\text{n}\} \\
\boldsymbol{\Theta}_{nb} &= \text{Euler angles between } \{\text{n}\} \text{ and } \{\text{b}\}
\end{aligned}
$$

For a marine craft moving in 6-DOF, six independent coordinates are needed to describe position and orientation. The notation of forces and motion in Table 2.1 are adopted from SNAME (1950). The elements in Table 2.1 can be written in vectors according to:

$$
\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{p}^n_{b/n} \\ \boldsymbol{\Theta}_{nb} \end{bmatrix} \quad , \quad \boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{v}^b_{b/n} \\ \boldsymbol{\omega}^b_{b/n} \end{bmatrix} \quad , \quad \boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{f}^b_b \\ \boldsymbol{m}^b_b \end{bmatrix} \tag{2.1}
$$

where

DOF		Forces and moments	Linear and angular velocities	Positions and Euler angles
1	motion in the x direction (surge)	X	u	x
2	motion in the y direction (sway)	Y	v	y
3	motion 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 for marine vessels adopted from SNAME (1950).

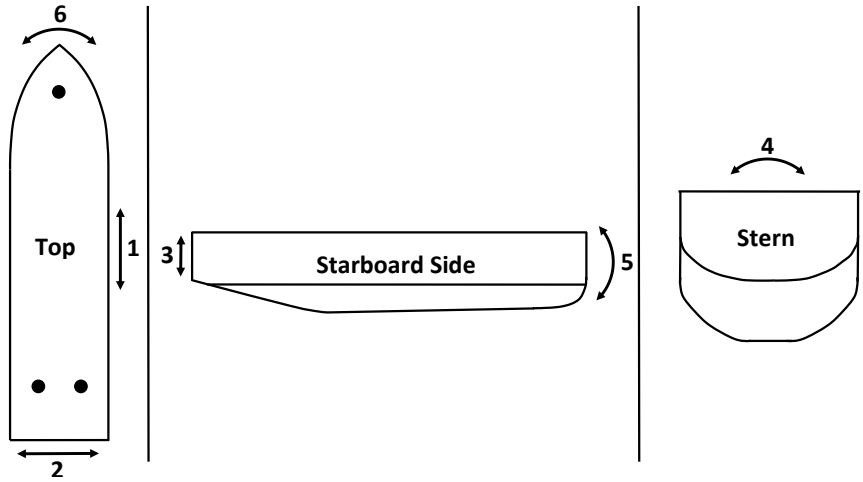


Figure 2.2: Degrees of Freedom of a vessel. The numbers correspond to those in Table 2.1.

NED position $\boldsymbol{p}^n_{b/n} = \begin{bmatrix} N \\ E \\ D \end{bmatrix} \in \mathbb{R}^3$ Attitude $\boldsymbol{\Theta}_{nb} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \in S^3$

Body-fixed linear velocity $\boldsymbol{v}^b_{b/n} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \in \mathbb{R}^3$ Body-fixed angular velocities $\boldsymbol{\omega}^b_{b/n} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \in \mathbb{R}^3$

Body-fixed force $\boldsymbol{f}^b_b = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \in \mathbb{R}^3$ Body-fixed moment $\boldsymbol{m}^b_b = \begin{bmatrix} K \\ M \\ N \end{bmatrix} \in \mathbb{R}^3$

2.1.1 Transformation Between ECEF and NED Frames

The rotation from ECEF to NED is stated in Vik (2014) and is is represented by the following matrix:

$$\mathbf{R}_n^e(\Theta_{en}) = \begin{bmatrix} -c(l)s(\mu) & -s(l) & -c(l)c(\mu) \\ -s(l)s(\mu) & c(l) & -s(l)c(\mu) \\ c(\mu) & 0 & -s(\mu) \end{bmatrix} \tag{2.2}$$

where $s = \sin$, $c = \cos$, l is longitude and μ is latitude.

2.1.2 Transformation Between BODY and NED Frames

The rotation from BODY to NED is stated in Vik (2014) and is is represented by the following matrix:

$$\mathbf{R}_b^n(\Theta) = \begin{bmatrix} c(\psi)c(\theta) & -s(\psi)c(\phi) + c(\psi)s(\theta)s(\phi) & s(\psi)s(\phi) + c(\psi)s(\theta)c(\phi) \\ s(\psi)c(\theta) & c(\psi)c(\phi) + s(\psi)s(\theta)s(\phi) & -c(\psi)s(\phi) + s(\psi)s(\theta)c(\phi) \\ -s(\theta) & c(\theta)s(\phi) & c(\theta)c(\phi) \end{bmatrix} \tag{2.3}$$

where $s = \sin$ and $c = \cos$.

The transformation from NED to BODY is given by

$$\mathbf{R}_b^n(\Theta)^\mathsf{T} = \mathbf{R}_n^b(\Theta) \tag{2.4}$$

2.2 The 3-DOF Ship Maneuvering Model

A ship's full maneuvering model contains 6-DOF differential equations of motion. These are referred to as surge, sway and heave for the ship's position, and roll, pitch and yaw for the orientation of the ship, see Table 2.1. Since the scope of this thesis is to maneuver a vessel in the horizontal plane, the model can be simplified. For a ship one can assume small amplitudes in roll and pitch, $\phi = \theta \approx 0$, and since the ship is floating, heave is also discarded $z \approx 0$.

From Fossen (2011), the 3-DOF marine craft equations of motion is written as:

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

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}_{thruster} + \boldsymbol{\tau}_{environmental} \tag{2.6}$$

where rotation is performed about the z-axis with the matrix

$$\boldsymbol{R}(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

In the rest of the thesis, the simplified notation of $\mathbf{R}$ is used to represent the rotation matrix from BODY to NED.

$\boldsymbol{M}$ in (2.6) is the inertia matrix, which is the sum of rigid-body inertia matrix $\boldsymbol{M}_{RB}$ and hydrodynamic added mass $\boldsymbol{M}_A$.

$$\boldsymbol{M} = \boldsymbol{M}_{RB} + \boldsymbol{M}_A = \begin{bmatrix} m & 0 & 0 \\ 0 & m & mx_g \\ 0 & mx_g & I_z \end{bmatrix} + \begin{bmatrix} -X_{\dot{u}} & 0 & 0 \\ 0 & -Y_{\dot{v}} & -Y_{\dot{r}} \\ 0 & -Y_{\dot{r}} & -N_{\dot{v}} \end{bmatrix} \tag{2.8}$$

where m is rigid-body mass, x_g is the distance from Center of Origin (CO) to Center of Gravity (CG) and I_z is the moment of inertia about the Z_b axis.

Further, $\boldsymbol{C}(\boldsymbol{\nu})$ in (2.6) consists of Coriolis and Centripetal terms which occurs due to rotation of the vessel about the NED frame:

$$\boldsymbol{C}(\boldsymbol{\nu}) = \boldsymbol{C}_{RB}(\boldsymbol{\nu}) + \boldsymbol{C}_A(\boldsymbol{\nu}) \tag{2.9}$$

where

$$\boldsymbol{C}_{RB}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & -m(x_g r + v) \\ 0 & 0 & mu \\ m(x_g r + v) & -mu & 0 \end{bmatrix} \tag{2.10a}$$

$$\boldsymbol{C}_A(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & Y_{\dot{v}} v + Y_{\dot{r}} r \\ 0 & 0 & -X_{\dot{u}} u \\ -Y_{\dot{v}} v - Y_{\dot{r}} r & X_{\dot{u}} u & 0 \end{bmatrix} \tag{2.10b}$$

$\boldsymbol{D}(\boldsymbol{\nu})$ consists of the damping forces which usually are modelled as a sum of constants and some velocity dependent terms (Fossen, 2011):

$$\begin{aligned} \boldsymbol{D}(\boldsymbol{\nu}) = \boldsymbol{D}_L + \boldsymbol{D}_n(\boldsymbol{\nu}) = & \begin{bmatrix} -X_u & 0 & 0 \\ 0 & -Y_v & -Y_r \\ 0 & -N_v & -N_r \end{bmatrix} + \\ & \begin{bmatrix} X_{|u|u}|u| & 0 & 0 \\ 0 & -Y_{|v|v}|v| - Y_{|r|v}|r| & -Y_{|v|r}|v| - Y_{|r|r}|r| \\ 0 & -N_{|v|v}|v| - N_{|r|v}|r| & -N_{|v|r}|v| - N_{|r|r}|r| \end{bmatrix} \end{aligned} \tag{2.11}$$

For a vessel operating around zero speed, it is reasonable to assume that linear damping $\boldsymbol{D}_L$ will dominate over $\boldsymbol{D}_n(\boldsymbol{\nu})$.

2.3 Global Navigation Satellite System

A Global Navigation Satellite System (GNSS) is a positioning system based on satellites. By the use of four or more satellites it is possible to acquire the 3D position of the receiver in the ECEF frame and transform it into more understandable Latitude and Longitude coordinates.

There are different GNSS to choose from, the most known being the Global Positioning System (GPS). GPS was created by the United States of America in 1993 and originally developed for military purposes (Vik, 2014). In later years, GPS has been opened for civilian use and is now used in many applications.

A conventional GNSS receiver uses pseudo-random codes, transmitted from the satellites, to determine the range to the satellites. By knowing the position of the satellites and these ranges, the receiver is able to compute a position. The accuracy is usually in the area of a few meters, but this can be improved using GNSS augmentation systems, like Satellite-Based Augmentation System (SBAS) or Differential GNSS like Real-Time Kinematic (RTK).

SBAS is a network of geostationary satellites which transmits correction data. The SBAS system which primarily covers Europe is called European Geostationary Navigation Overlay Service (EGNOS). To get a higher accuracy, RTK GNSS can be used. The receiver uses the signal from a RTK GNSS base station, with a known location, to correct the position of the receiver. This can give an accuracy in the range of a few centimeters. For an in-depth explanation of how GNSS and augmentation systems works, see Vik (2014).

There are several error sources for GNSS receivers. A possible error source for marine surface vessels is multipath interference, which occurs when the device receives reflected signals in addition to the direct Line of Sight (LOS) signal. These interference signals are generally reflected from the ground/sea, buildings or trees. Therefore, by mounting the antennas further away from the sea, it will help mitigate the interference from multipath (Parkinson, 1996).

The National Marine Electronics Association (NMEA), has developed a standard message structure for communication between GNSS and other hardware. This standard makes it easy to mix and match software and hardware, and for software developers life is made easier as they can create software for many different GNSS receivers and not only one. A common NMEA standard is NMEA 0183, which has several NMEA sentences containing different information.

Chapter 3

Guidance System

This chapter describes the design of a guidance system which can be part of a DP system, see Figure 3.1. This involves a position and attitude reference filter, saturation of the desired velocity and discretization of the reference filter.

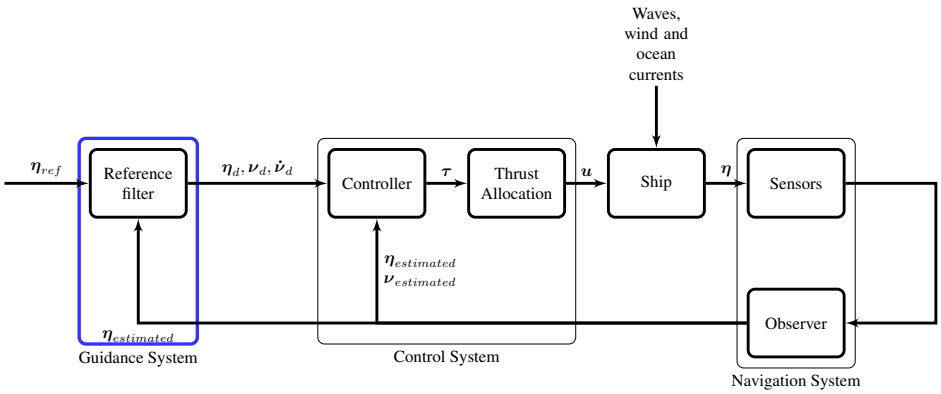


Figure 3.1: Block diagram of the DP system with the guidance system highlighted in blue.

3.1 Reference Filter

A reference filter is used to generate a suitable trajectory which the vessel can follow. For big changes in setpoint, the controller will act on a big error which in most cases will lead to a sudden huge demand in control effort. This may then generate unwanted behavior from the vessel. In order to generate a suitable trajectory for position and attitude, a third

order linear reference filter has been chosen. The reference filter is defined as stated in Fossen (2011):

$$\dddot{\boldsymbol{\eta}}_d + (2\boldsymbol{\Delta} + \mathbf{I})\boldsymbol{\Omega}\ddot{\boldsymbol{\eta}}_d + (2\boldsymbol{\Delta} + \mathbf{I})\boldsymbol{\Omega}^2\dot{\boldsymbol{\eta}}_d + \boldsymbol{\Omega}^3\boldsymbol{\eta}_d = \boldsymbol{\Omega}^3\boldsymbol{\eta}_{ref} \tag{3.1}$$

where $\boldsymbol{\eta}_d \in \mathbb{R}^3$ is the desired pose, given in the NED frame. The input to the filter is $\boldsymbol{\eta}_{ref} \in \mathbb{R}^3$ and is given in the NED frame. The output, $\boldsymbol{\eta}_d$ and the input, $\boldsymbol{\eta}_{ref}$ are given by:

$$\boldsymbol{\eta}_d = \begin{bmatrix} N_d \\ E_d \\ \psi_d \end{bmatrix} \qquad \boldsymbol{\eta}_{ref} = \begin{bmatrix} N_{ref} \\ E_{ref} \\ \psi_{ref} \end{bmatrix} \tag{3.2}$$

The diagonal matrix $\boldsymbol{\Delta} > 0$, contains the relative damping ratios, $\zeta_i, i = 1, 2, 3$, and $\boldsymbol{\Omega} > 0$ is a diagonal matrix with the the natural frequencies, $\omega_i, i = 1, 2, 3$.

The continuous equation for the reference filter in (3.1) can be written in the state-space form

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{3.3}$$

as

$$\begin{bmatrix} \dot{\boldsymbol{\eta}}_d \\ \ddot{\boldsymbol{\eta}}_d \\ \dddot{\boldsymbol{\eta}}_d \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{I} & 0 \\ 0 & 0 & \mathbf{I} \\ -\boldsymbol{\Omega}^3 & -(2\boldsymbol{\Delta} + \mathbf{I})\boldsymbol{\Omega}^2 & -(2\boldsymbol{\Delta} + \mathbf{I})\boldsymbol{\Omega} \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}_d \\ \dot{\boldsymbol{\eta}}_d \\ \ddot{\boldsymbol{\eta}}_d \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \boldsymbol{\Omega}^3 \end{bmatrix} \boldsymbol{\eta}^n_{\text{ref}} \tag{3.4}$$

The desired jerk, $\dddot{\boldsymbol{\eta}}$, acceleration, $\ddot{\boldsymbol{\eta}}$ and velocities,$\dot{\boldsymbol{\eta}}$ are all given in the NED frame. A block diagram of the reference filter is shown in Figure 3.2.

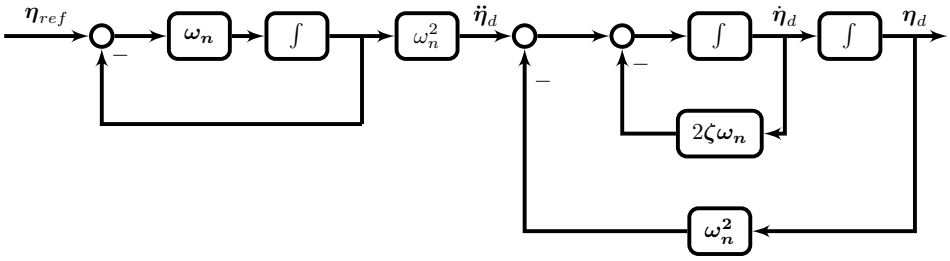


Figure 3.2: Block diagram of a 1-DOF reference filter.

3.1.1 Tuning the Reference Filter

The reference filter must be tuned not to exceed the velocity and acceleration values obtainable by the vessel as well as any limits set by the user.

An experimental study on how passenger comfort is related to a vehicles' longitudinal direction was conducted by The University of Texas at Austin for the Department of Transportation in Washington D.C. (Hoberock, 1976). The goal of the study was to determine which acceleration and jerk that would make the ride comfortable for passengers. Even though the study could not decisively conclude with any precise values for acceptable acceleration and jerk, it did indicate a range of values which are suitable. Acceleration should be in the range $0.11 - 0.15g$ or $1.08 - 1.47m/s^2$. The study also concluded that a jerk less than $0.30g/s$ would be acceptable for most passengers.

In the reference filter there are two parameters which can be used to tune the filter. These two values are the $\mathbf{\Delta} \in \mathbb{R}^{3\times1}$ and $\mathbf{\Omega} \in \mathbb{R}^{3\times1}$. $\mathbf{\Delta}$ is the damping ratio and determines how stiff the system is. An overshoot is not desirable when the reference filter will be used for movements close to obstacles like other ships or a dock. Since it is not desirable with an overshoot in any of the 3-DOF, the system has been tuned to be critically damped:

$$\mathbf{\Delta} = \begin{bmatrix} \zeta_N \\ \zeta_E \\ \zeta_\psi \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{3.5}$$

$\mathbf{\Omega}$ is the "natural frequency" of the filter or rather the convergence parameter. This determines how quickly the filter reacts and converges to the input.

3.1.2 Reference Saturation

A disadvantage of the linear reference filter is that it will only yield a suitable response for a fixed setpoint, while the response for a different setpoint will give a different behavior. This behavior may not be obtainable by the vessel. In order to assure that the filter always generates a suitable and obtainable trajectory, amplitude gain scheduling of the tuning parameters (Fossen, 2011) can be used or the input of the reference filter can be saturated.

A saturation of the input, $\boldsymbol{\eta}_{ref}$, has been created. If the input is larger than a set max value, the input will be saturated and moved along, ahead of the states of the filter, until it reaches the actual or final input value. This function can be seen as holding a carrot on a stick in front of a donkey, where the carrot is the reference point and the donkey is the filter state.

The maximum distance the input should be ahead of the ship, in the NED frame, is different depending on the desired heading of the ship. When the ship's desired heading is due north the reference, or carrot, can be place further away in the north direction, than when the ship is pointing due east. This is because the ship can achieve a higher surge speed than sway speed. The difference between the reference filter state and the final input is rotated into the BODY frame and saturated to the desired maximum distances, before being rotated back to the NED frame.

With this reference saturation, (3.1) is modified to include this feature.

$$\dddot{\boldsymbol{\eta}}_d + (2\mathbf{\Delta} + \mathbf{I})\mathbf{\Omega}\ddot{\boldsymbol{\eta}}_d + (2\mathbf{\Delta} + \mathbf{I})\mathbf{\Omega}^2\dot{\boldsymbol{\eta}}_d + \mathbf{\Omega}^3\boldsymbol{\eta}_d = \mathbf{\Omega}^3\boldsymbol{\eta}_{input} \tag{3.6}$$

where $\boldsymbol{\eta}_{input} = \boldsymbol{\eta}_d + \boldsymbol{R}(\psi_d)(sat(\boldsymbol{R}(\psi_d)^{\mathsf{T}}(\boldsymbol{\eta}_{ref} - \boldsymbol{\eta}_d)))$ and $sat()$ is the saturation function. A diagram of the reference filter including the input saturation can be seen in Figure 3.3.

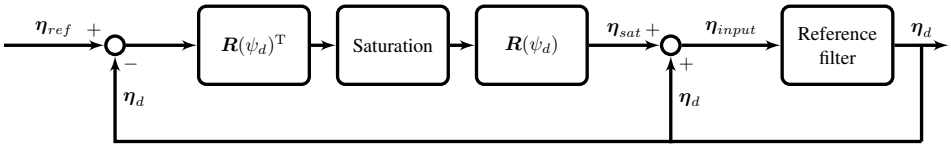


Figure 3.3: Block diagram of the reference filter with the "carrot function" implemented.

Figure 3.4 shows how the input to the filter is saturated. The saturated reference is given by

$$\boldsymbol{\eta}_{sat} = \boldsymbol{R}(\psi_d)(sat(\boldsymbol{R}(\psi_d)^{\mathsf{T}}(\boldsymbol{\eta}_{ref} - \boldsymbol{\eta}_d))) \tag{3.7}$$

In Figure 3.5, the desired velocity $\dot{\boldsymbol{\eta}}_d$ and acceleration $\ddot{\boldsymbol{\eta}}_d$ is shown. The velocity and acceleration limits to which the reference filter was tuned to in this example were:

$$\nu_u^b = 0.5m/s \quad -\nu_u^b = 0.2m/s \quad \pm\nu_v^b = 0.1m/s \quad \pm\nu_r^b = 3^\circ/s \tag{3.8}$$

$$\dot{\nu}_u^b = 0.25m/s^2 \quad -\dot{\nu}_u^b = 0.1m/s^2 \quad \pm\dot{\nu}_v^b = 0.05m/s^2 \quad \pm\dot{\nu}_r^b = 1.5^\circ/s^2 \tag{3.9}$$

Based on the velocity and acceleration limits given in (3.8) and (3.9) the following saturation limits were used:

$$x_{max}^b = 2.25 \quad -x_{max}^b = 1 \quad \pm y_{max}^b = 0.5 \quad \pm\psi_{max}^b = 15^\circ \tag{3.10}$$

which were set with the following tuning parameters:

$$\boldsymbol{\Delta} = \begin{bmatrix} \zeta_N \\ \zeta_E \\ \zeta_{psi} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad , \quad \boldsymbol{\Omega} = \begin{bmatrix} \omega_{n,N} \\ \omega_{n,E} \\ \omega_{n,\psi} \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.6 \\ 0.6 \end{bmatrix} \tag{3.11}$$

3.1.3 Discretization

By using the numerical integration scheme Euler's method as given in Egeland and Gravdahl (2002), the discrete equation of (3.6) becomes

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + h(\boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{\eta}_{input}) \tag{3.12}$$

which is shown to be stable if

$$|R(h\lambda)| = |1 + h\lambda| \leq 1 \tag{3.13}$$

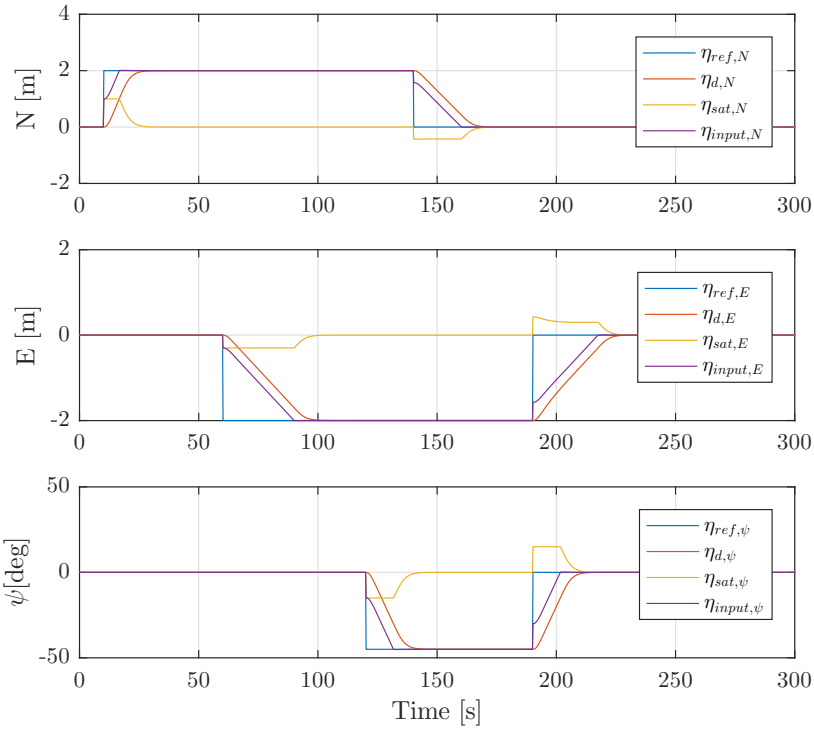


Figure 3.4: Relationship between reference, output, saturated input and input of the reference filter.

where $R(h\lambda)$ is the stability function, h the timestep and λ are the eigenvalues of the system. In other words, Euler's method is stable if $h\lambda$ is inside a circle with a radius of one around the point (-1,0).

Since the reference filter will only have real eigenvalues, see Section 3.1.1, this method will be stable if

$$h \leq -\frac{2}{\lambda} \tag{3.14}$$

3.1.4 Rotate Shortest Path

When computing a desired heading for the ship, the reference filter has to find the shortest way to the heading reference and not rotate more than necessary. The reference filter works in the area of $\psi = \pm 180°$ and when rotating from 179° to -179° the reference filter should rotate the ship across the 180° mark, not all the way across the 0° mark. The problem has been solved with the code given in Algorithm 3.1.

```
%Find remainder of difference between ref and des
psi_temp=mod((psi_ref-psi_d),360)

% Add 360 and find remainder
psi_shortest=mod(psi_temp+360,360);

%Find the shortest way to rotate
if (psi_shortest > 180)
    psi_shortest = psi_shortest - 360;
end
```

Algorithm 3.1: Algorithm to rotate the shortest way between two heading angles.

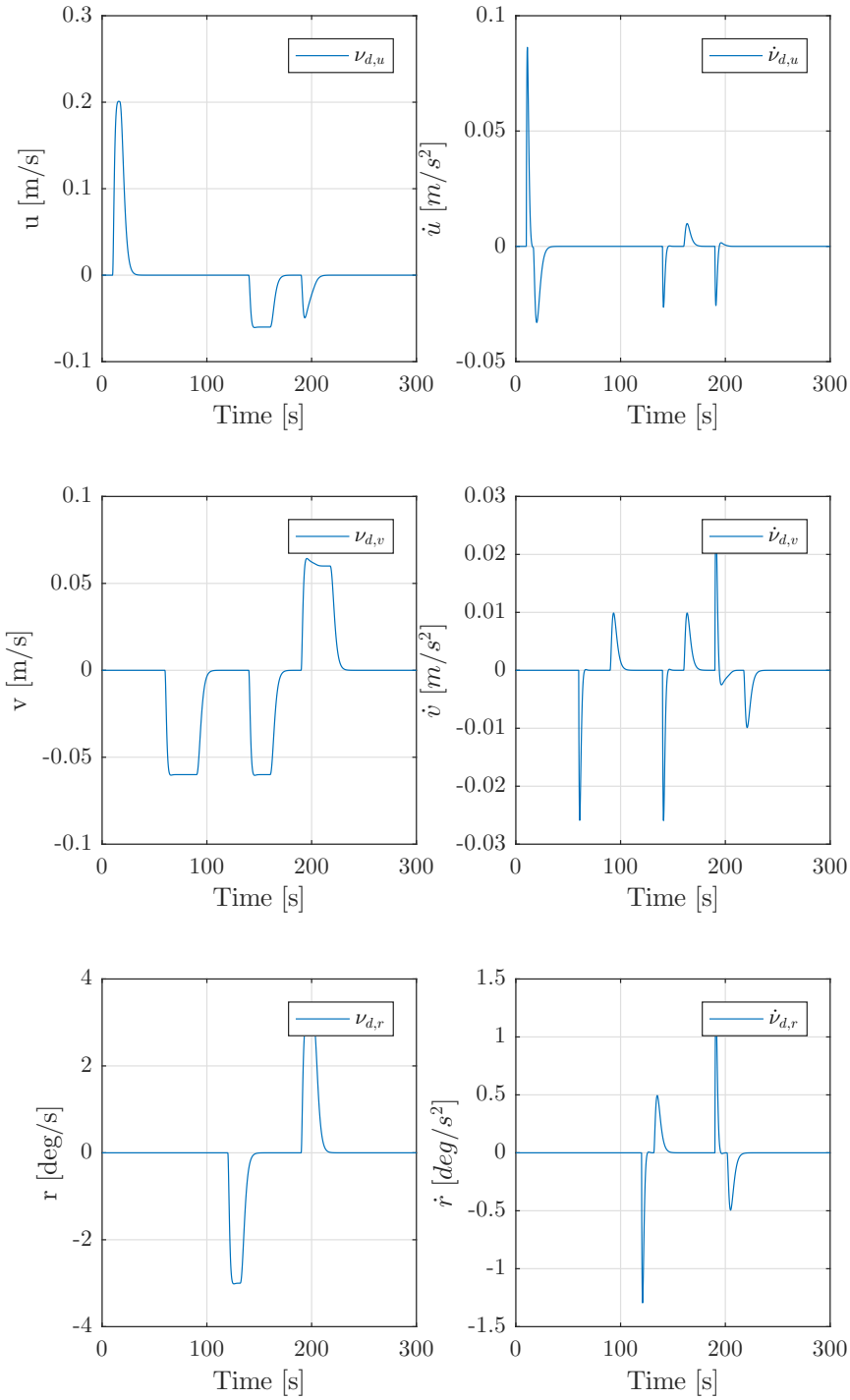


Figure 3.5: Desired velocity and acceleration from the reference filter.

Chapter 4

Control System

The control system forms the brain of the whole DP system. Here all the information comes together and the control signals are calculated before being passed on to the actuators. Figure 4.1 shows the control part of the DP system. Data of the vessel's pose and desired pose are fed into the controller, from which the controller calculates a desired control force. The thrust allocation then sets up the actuators in such a way that the desired control force is obtained. In this chapter, a motion controller with a feedforward from the reference and thrust allocation are presented.

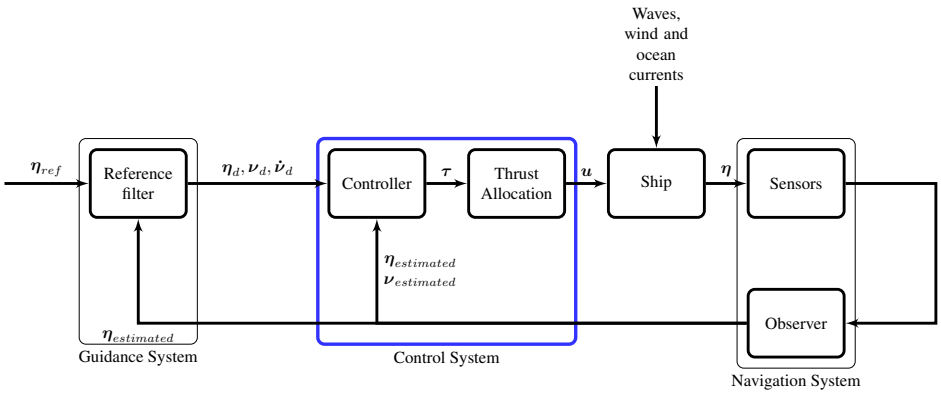


Figure 4.1: Block diagram of the DP system with the control system highlighted in blue.

4.1 Dynamic Positioning Controller

Dynamic positioning (DP) started out as an exotic control technology for geological sampling in deep waters during the 1960s, and was shortly after introduced to the offshore petroleum industry (Breivik et al., 2015). The technology became more advanced, especially with use of Kalman filter in DP, which was introduced in the late 1970s. Since then, DP has developed from a scientific technology to an industrial technology, and is today used in mass-marked applications such as the offshore petroleum industry. The DP systems are most commonly used to keep the vessel at a fixed heading and position, but can also be used to maneuver the vessel precisely over short or longer distances.

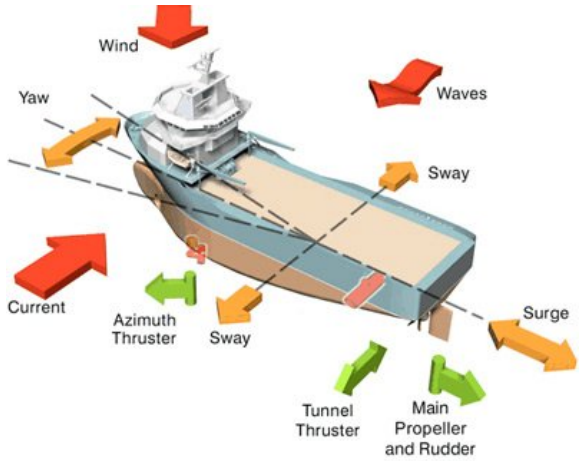


Figure 4.2: A vessel using a dynamic positioning control system must use its thrusters actively to counteract the environmental forces it is subjected too. Courtesy of Kongsberg Maritime.

There are several types of controllers that can be used for DP. These range from simple to advanced with a PID controller being a robust and industry standard used in many applications, the Linear Quadratic Regulator (LQR) which is based on a mathematical model of the vessel, to the more exotic controllers such as adaptive backstepping controller (Sørensen et al., 2016). As there is no verified existing mathematical model of the experimental vessel ReVolt, see Chapter 6, the PID controller is a solid choice. A feedforward term utilizing a vessel model obtained from simulations, should assist the controller with setpoint changes.

4.1.1 Control Algorithm

The overall control algorithm will consist of a PID feedback term and a reference feedforward term. This gives the following overall controller.

$$\boldsymbol{\tau} = \boldsymbol{\tau_{FF}} + \boldsymbol{\tau_{PID}} \tag{4.1}$$

Feedforward

In the controller, a model-based reference feedforward will be used, based on the model in (2.6), restated here in (4.2).

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}_{thruster} + \boldsymbol{\tau}_{environmental} \tag{4.2}$$

The model-based feedforward part of the control law is designed to feed the dynamics of the vessel forward, based on the model. This will give the vessel faster reactions according to the vessel dynamics. Therefore the feedforward term becomes:

$$\boldsymbol{\tau}_{FF} = \boldsymbol{M}\dot{\boldsymbol{\nu}}_{\boldsymbol{d}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu}_d + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu}_d \tag{4.3}$$

where $\boldsymbol{\nu}_d \in \mathbb{R}^n$ and $\dot{\boldsymbol{\nu}}_{\boldsymbol{d}} \in \mathbb{R}^n$ denotes the desired velocities and accelerations in the BODY frame, respectively.

When implementing the feedforward term in a practical system, $\boldsymbol{C}(\boldsymbol{\nu}_d)$, $\boldsymbol{D}(\boldsymbol{\nu}_d)$ and $\dot{\boldsymbol{\nu}}_d$ have been used instead of $\boldsymbol{C}(\boldsymbol{\nu})$, $\boldsymbol{D}(\boldsymbol{\nu})$ and $\dot{\boldsymbol{\nu}}$. This is because the desired setpoints are a much more stable signal than the system states signal. If the state signal becomes corrupted or is estimated incorrectly, the entire system can become unstable. The feedforward term then becomes as follows:

$$\boldsymbol{\tau}_{FF} = \boldsymbol{M}\dot{\boldsymbol{\nu}}_{\boldsymbol{d}} + \boldsymbol{C}(\boldsymbol{\nu}_d)\boldsymbol{\nu}_d + \boldsymbol{D}(\boldsymbol{\nu}_d)\boldsymbol{\nu}_d \tag{4.4}$$

PID

Since the vessel is prone to environmental disturbances and there are modelling errors in the mathematical model of the vessel, a feedback term is added to the controller. The PID controller tries to minimize the error between a system's desired setpoint, x_{d}, and the system's state x, to zero. This error is given by

$$e(t) = x(t) - x_{\mathrm{d}}(t) \tag{4.5}$$

The desired setpoint is determined by the user of the system, whilst the actual state of the system is measured by sensors or estimated By considering the error, the algorithm will compute a control command $\tau_{PID}(t)$ using a weighted sum

$$\tau_{PID}(t) = -K_P e(t) - K_I \int_0^t e(\tau)d\tau - K_D \frac{de(t)}{dt} \tag{4.6}$$

where $K_P > 0, K_I > 0$ and $K_D > 0$ are the proportional, integral and derivative parameters respectively. A PID controller is easy to implement as it only relies on the measured

states of the system and does not require any knowledge of the vessels dynamics. To implement the PID controller on a computer, the equation has been discretized using implicit Euler method, giving the following equation

$$\tau_{PID}(k) = -K_P e(k) - K_I \sum_{i=0}^{i} e(i)h - K_D \frac{1}{h}(e(k) - e(k-1)) \tag{4.7}$$

where h is the time between each execution of the controller algorithm, called timestep.

When using a PID controller in a DP control system, it is common that the desired set-point is in 3-DOF. This can be achieved by using matrices in the controller equation. By changing the state x to $\boldsymbol{\eta}^b \in \mathbb{R}^3$ in (4.7), where

$$\boldsymbol{\eta}^b = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix} \tag{4.8}$$

the control law becomes the following:

$$\boldsymbol{\tau}_{PID}(k) = -\mathbf{K}_P \boldsymbol{\eta}^b_{error}(k) - \mathbf{K}_I \sum_{i=0}^{i} \boldsymbol{\eta}^b_{error}(i)h - \mathbf{K}_D \frac{1}{h} \boldsymbol{\nu}_{error}(k) \tag{4.9}$$

where

$$\boldsymbol{\eta}^b_{error} = \boldsymbol{\eta}^b - \boldsymbol{\eta}^b_d \tag{4.10}$$
$$\boldsymbol{\nu}_{error} = \boldsymbol{\nu} - \boldsymbol{\nu}_d \tag{4.11}$$

Rotation to BODY

In the controller given in (4.1), the pose error, $\boldsymbol{\eta}_{error}$, desired linear and angular velocities and accelerations, are input in the NED frame and need to be expressed in the BODY frame. This is solved by using the rotation matrix $\boldsymbol{R}^\mathsf{T}$ given in (2.4).

$$\boldsymbol{\eta}^b_{error} = \boldsymbol{R}^\mathsf{T} \boldsymbol{\eta}^n_{error} \tag{4.12}$$
$$\boldsymbol{\nu}_d = \boldsymbol{R}^\mathsf{T} \dot{\boldsymbol{\eta}}^n_d \tag{4.13}$$
$$\dot{\boldsymbol{\nu}}_d = \boldsymbol{R}^\mathsf{T} \ddot{\boldsymbol{\eta}}^n_d + \boldsymbol{S}(\boldsymbol{\omega}) \boldsymbol{R}^\mathsf{T} \dot{\boldsymbol{\eta}}^n_d \tag{4.14}$$

where $\boldsymbol{\omega} = [0, 0, r]^\mathsf{T}$ since the controller is in 3-DOF.

4.1.2 Saturating Elements

The ship has a maximum amount of thrust which each propeller can produce, therefore the controller should not output a larger wanted force than the ship can handle. Saturating elements are placed in the controller to limit the output value if the controller computes a too large wanted force.

The maximum thrust matrix is given by:

$$\boldsymbol{\tau}_{\text{max}} = \begin{bmatrix} \tau_{X,\text{max}} \\ \tau_{Y,\text{max}} \\ \tau_{N,\text{max}} \end{bmatrix} \tag{4.15}$$

The saturating elements have been implemented according to

$$sat(x) = \begin{cases} sgn(x)x_{max} & if\ |x| \geq x_{max} \\ x & else \end{cases} \tag{4.16}$$

The saturating limit $\boldsymbol{\tau_i} \leq \boldsymbol{\tau}_{i,max}, i = 1, 2, 3$ is set to reflect the physical limitations of the ship. The saturating elements have been realized using the pseudocode given in Algorithm 4.1.

```
% Output Saturation
for i = 1:3
    if abs(tau(i))>tau_max(i)
        tau(i) = sign(tau(i))*tau_max(i);
    end
end
```

Algorithm 4.1: Saturating element.

4.1.3 Tuning the Controller

To tune a PID controller, techniques such as pole placement or manual tuning can be used. Pole placement is useful if a mathematical model of the system is available (Fossen, 2011), and manual tuning can be used if the physical system or a simulation is available.

When tuning the controller manually, the operator needs to think physically about what the gains mean for the system. With the error and velocity as defined in (4.11), the units become

$$\boldsymbol{\tau}_{PID} \begin{bmatrix} N \\ N \\ Nm \end{bmatrix}, \quad \boldsymbol{K}_P \begin{bmatrix} \frac{N}{m} \\ \frac{N}{m} \\ \frac{Nm}{deg} \end{bmatrix}, \quad \boldsymbol{K}_I \begin{bmatrix} \frac{N}{m^2} \\ \frac{N}{m^2} \\ \frac{Nm}{deg^2} \end{bmatrix}, \quad \boldsymbol{K}_D \begin{bmatrix} \frac{N}{m/s} \\ \frac{N}{m/s} \\ \frac{Nm}{deg/s} \end{bmatrix} \tag{4.17}$$

A frequently used method of tuning the PID is increasing the proportional gain until oscillations around the setpoint is achieved. Then adding derivative gain functioning as dampening to remove the oscillations. Lastly adding integral action removes the constant offset. As a rule of thumb, the controller gains should have the following relationships:

$$\mathbf{K}_D > \mathbf{K}_P \quad , \quad \mathbf{K}_I < \mathbf{K}_P \quad , \quad \mathbf{K}_I \approx \frac{\mathbf{K}_P}{10}$$

4.1.4 Yaw Wrapping

Since the rest of the DP system uses a yaw measurement between $\pm 180°$, the yaw angle in the controller has to stay between $\pm 180°$ if the vessel is to behave optimally. By keeping the measured heading between $\pm 180°$, the vessel will never need to rotate more than $180°$ and the controller will receive the shortest path to the desired heading. This yaw wrapping is performed on the measured heading angle of the ship before it is used in the controller. The yaw wrapping is implemented as described by the pseudocode in Algorithm 4.2.

```
%Wrap measured yaw within +/- pi(180 deg)
yaw = modulo(yaw,360)    %return the remainder of yaw/360
if yaw > 180:    % If yaw is larger than 180, subtract 360
    yaw = yaw - 360

elseif yaw < -180: % If yaw is smaller than 180, add 360
    yaw = yaw + 360
```

Algorithm 4.2: Pseudocode for wrapping of the yaw.

4.2 Thrust Allocation

Actuators can exert forces and moments on the ship and are used by the DP system to control the ship pose. How well the DP system's commanded forces are realized depends greatly on each actuator and their position on the vessel relative to CG as well as the Thrust Allocation (TA)'s ability to handle the thruster's dynamics.

4.2.1 Actuators

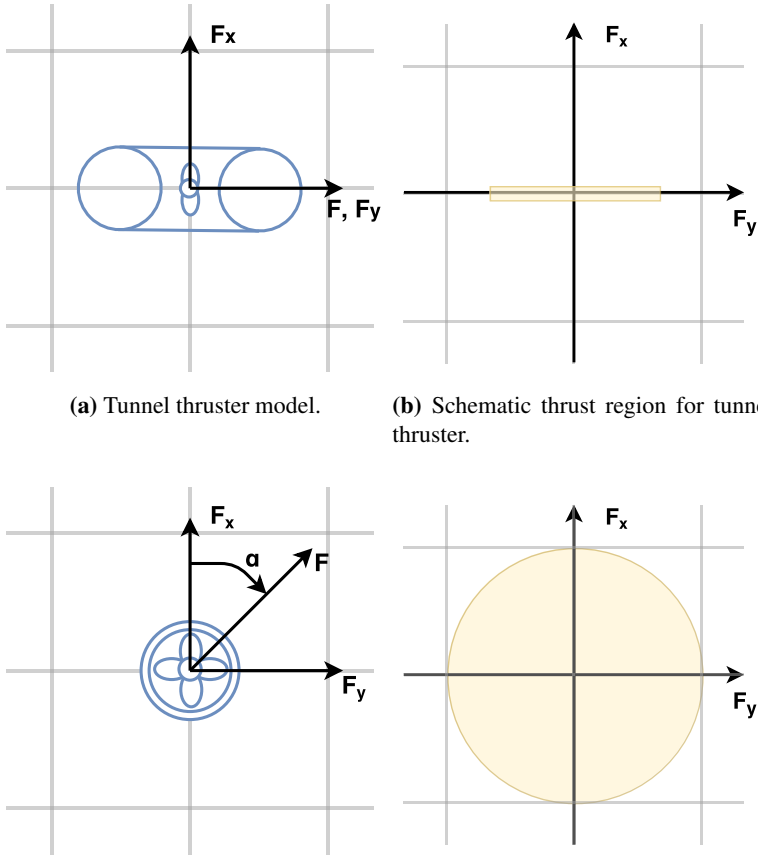
The most common actuators for a vessel using rotating propellers are:

Tunnel thrusters which can generate transverse thrust and are usually positioned at the bow and stern of the ship where they will produce the most moment. The angle of which they produce thrust is given as α and with its maximum obtainable thrust, T_{max} the thrust region becomes a line as in Figure 4.3b. Tunnel thrusters are mostly used for low-speed navigation.

Azimuth thrusters can be used for both steering and propulsion purposes. They can rotate 360° and can therefore generate thrust in any direction. The thrust region for an azimuth thruster is seen in Figure 4.3d.

Main propellers and rudders are designed to be very efficient in certain conditions. They are mostly used as the main propulsion for a ship, but are also commonly used in DP systems.

There are also other actuators for example fins which are used to stabilize roll motions, or water jet propulsion devices which can exert forces a lot like a main propeller with rudder.



(a) Tunnel thruster model.

(b) Schematic thrust region for tunnel thruster.

(c) Azimuth thruster model.

(d) Schematic thrust region for an azimuth thruster.

Figure 4.3: Thruster model and thrust region for a tunnel thruster and an azimuth thruster. The figure is adapted from Wit (2009).

4.2.2 Thruster Force

The relationship between the propeller thrust F and the shaft speed n is given by (Sørensen, 2013)

$$F = sign(n)K_T\rho D^4\mathsf{n}^2 \tag{4.18}$$

where D is the propeller diameter and ρ is the water density. The thrust coefficient $K_T > 0$ is usually found by performing a bollard pull test where the force applied to the vessel by the propeller is measured. In some cases, n is not the input to the thruster, thus a simplified model using the control input is written as a linear model (Fossen, 2011)

$$\boldsymbol{F} = \boldsymbol{K}\boldsymbol{u} \tag{4.19}$$

where $\boldsymbol{F} \in \mathbb{R}^n$ is the generated force by the thrusters, $\boldsymbol{K} \in \mathbb{R}^{r\times r}$ (r = number of thrusters) is the new force coefficient and $\boldsymbol{u} \in \mathbb{R}^r$ is the control input to the thruster. The force exerted from the thruster can then be measured at different control inputs, and by using linear regression, the force coefficients are found.

4.2.3 Thrust Configuration Matrix

To find the forces and moments the thrusters exert on the vessel $\boldsymbol{\tau} \in \mathbb{R}^n$, a thrust configuration matrix is set up. For this, the CG is assumed to be known and defined, as well as every thruster's position with respect to this point. The forces and moments applied to the vessel is now defined by

$$\boldsymbol{\tau} = \boldsymbol{T}(\boldsymbol{\alpha})\boldsymbol{F} \tag{4.20}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^r$ is a vector of azimuth angles and $\boldsymbol{T}(\boldsymbol{\alpha}) \in \mathbb{R}^{n\times r}$ is the thrust configuration matrix. By inserting (4.19), (4.20) becomes

$$\boldsymbol{\tau} = \boldsymbol{T}(\boldsymbol{\alpha})\boldsymbol{K}\boldsymbol{u} \tag{4.21}$$

For a 3-DOF system and with the thruster geometry presented in Figure 4.4, the dimensions become: $\tau \in \mathbb{R}^3$, $\boldsymbol{u} \in \mathbb{R}^3$ and $\boldsymbol{T}(\boldsymbol{\alpha}) \in \mathbb{R}^{3\times 3}$. The thrust configuration matrix for

each thruster is written as

$$\boldsymbol{T(\alpha_1)} = \begin{bmatrix} c(\alpha_1) \\ s(\alpha_2) \\ l_{x1}s(\alpha_1) \end{bmatrix} \tag{4.22a}$$

$$\boldsymbol{T(\alpha_2)} = \begin{bmatrix} c(\alpha_2) \\ s(\alpha_2) \\ l_{y2}c(\alpha_2 - l_{x2}s(\alpha_2) \end{bmatrix} \tag{4.22b}$$

$$\boldsymbol{T(\alpha_3)} = \begin{bmatrix} c(\alpha_3) \\ s(\alpha_3) \\ l_{y3}c(\alpha_3 - l_{x3}s(\alpha_3) \end{bmatrix} \tag{4.22c}$$

$$\tag{4.22d}$$

which combined gives

$$\boldsymbol{T}(\boldsymbol{\alpha}) = \begin{bmatrix} c(\alpha_1) & c(\alpha_2) & c(\alpha_3) \\ s(\alpha_1) & s(\alpha_2) & s(\alpha_3) \\ l_{x1}s(\alpha_1) & l_{y2}c(\alpha_2) - l_{x2}s(\alpha_2) & l_{y3}c(\alpha_3) - l_{x3}s(\alpha_3) \end{bmatrix} \tag{4.23}$$

The control input and azimuth angles are now

$$\boldsymbol{u} = \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix}^\mathsf{T} \qquad \boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \end{bmatrix}^\mathsf{T} \tag{4.24}$$

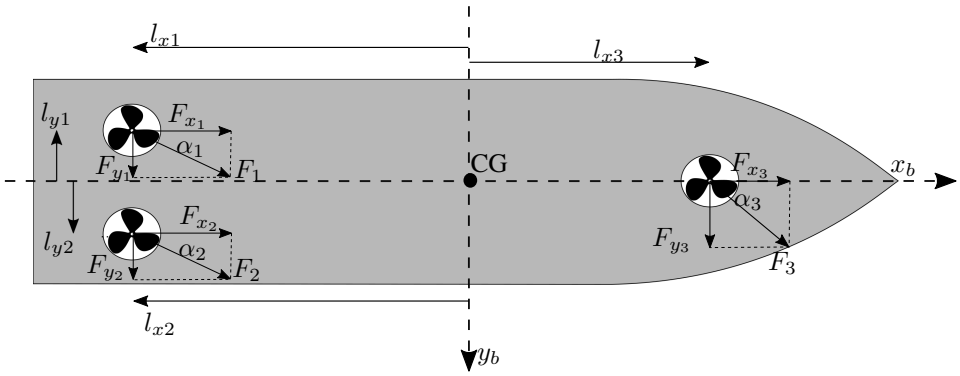


Figure 4.4: The thruster configuration for ReVolt introduced in Chapter 6, decomposed along the body x- and y-axis.

4.2.4 Unconstrained and Constrained Solution

Computation of the control input $\boldsymbol{u}$ and $\boldsymbol{\alpha}$ given a wanted $\boldsymbol{\tau}$, is a model-based optimization problem. In its most advanced form, it will handle constraints such as rotatable thrusters, rate saturations, pods wake influence and max thrust saturation.

Unconstrained

One solution to avoid this advanced mathematical problem is by setting the thrusters at a fixed angle. This results in the thruster configuration matrix $\boldsymbol{T}(\boldsymbol{\alpha}) = \boldsymbol{T}$. Now $\boldsymbol{u}$ is found using the pseudo inverse

$$\boldsymbol{u} = \boldsymbol{K}^{-1} T^{\mathsf{T}} (\boldsymbol{T}\boldsymbol{T}^{\mathsf{T}})^{-1} \tau \tag{4.25}$$

This method is unconstrained and will not consider saturations which will result in unobtainable commanded control inputs. Fossen (2011) suggest an extended thrust configuration matrix which will allow for rotatable thrusters. It will however still be unconstrained and not consider thruster dynamics, which will result in command inputs the thrusters can not achieve.

Constrained

To obtain a realistic thrust allocation which handles saturations and thruster dynamics, the optimization problem can be set up according to Fossen (2011) as

$$\begin{aligned} J &= min \left\{ \boldsymbol{s}^{\mathsf{T}} \boldsymbol{Q} \boldsymbol{s} + \boldsymbol{\Delta\alpha}^{\mathsf{T}} \boldsymbol{\Omega} \boldsymbol{\Delta\alpha} \right\} \\ subject\ to & \\ & \boldsymbol{T}(\boldsymbol{\alpha})\boldsymbol{F} - \boldsymbol{s} = \boldsymbol{\tau} \\ & \boldsymbol{F}_{min} \leq \boldsymbol{F} \leq \boldsymbol{F}_{max} \\ & \Delta\boldsymbol{F}_{min} \leq \Delta\boldsymbol{F} \leq \Delta\boldsymbol{F}_{max} \\ & \boldsymbol{\alpha}_{min} \leq \boldsymbol{\alpha} \leq \boldsymbol{\alpha}_{max} \\ & \boldsymbol{\Delta\alpha}_{min} \leq \boldsymbol{\Delta\alpha} \leq \boldsymbol{\Delta\alpha}_{max} \end{aligned} \tag{4.26}$$

shere $\boldsymbol{s}$ is a slack variable allowing the solution to lie within the thrust regions and not specifically on the boundary. Further, $\Delta\boldsymbol{F}$ is the maximum force the thrusters can apply and $\Delta\boldsymbol{\alpha}$ is the max angular rate for the thrusters.

As $\boldsymbol{T}(\boldsymbol{\alpha})$ is nonlinear, this optimization problem is nonconvex (Nocedal and Wright, 2006), and requires a significant amount of computations at each timestep. There are several ways to decrease the computation power needed, see Wit (2009).

For this thesis, DNV GL has supplied their thrust allocation which is described implemented in Section 7.2.4.

Chapter 5

Navigation System

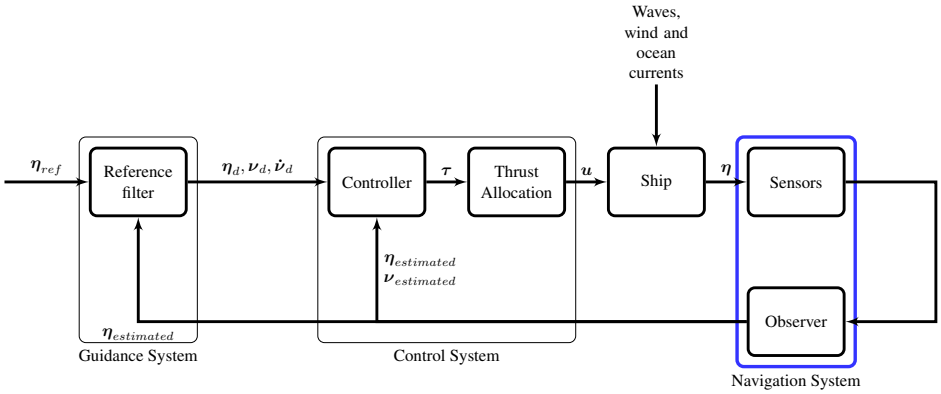


Figure 5.1: Block diagram of the DP system with the navigation system highlighted in blue.

Conventional DP systems are implemented with a navigation system, see Figure 5.1, which usually includes a state estimator (observer) for processing of sensor and navigation data. The quality of the data is checked as well as wild points removed. The data is then processed in a state estimator to filter noise and reconstruct unmeasured states. The most famous state estimator algorithm is the Kalman filter introduced in the 1960's (Brown and Hwang, 2012). Since then several observers based on passivity has been developed, see Grip et al. (2015). The chosen filter in this section is the Error-State Kalman Filter (ESKF).

This chapter provides an introduction to the Error-state Kalman Filter, a method to handle GNSS wild points and how to transfer GNSS measurements to a joint sensor frame.

5.1 Error-State Kalman Filter

The idea is to integrate accelerometer and gyrometer readings from an Inertial Measurement Unit (IMU) to obtain position, orientation and velocity, hereby called navigation state. Numerical integration of IMU readings leads to drift in the navigation state, which needs to be handled by fusing the information with absolute position readings from a GNSS. In the Kalman filter paradigm, there are two main strategies for approaching this. A total state filter calculates and corrects the navigational state from sensor measurements. The Error-State Kalman Filter (ESKF) uses the IMU as the reference navigation system and has aiding sensors such as a GNSS to estimate the error in the navigational state. These errors are then fed back into the navigational state. Arguments for choosing an ESKF:

- Since the error is estimated, the slower dynamical error model of the IMU is used as opposed to the faster dynamics of the vehicle model.
- Integration of the IMU data can be run at high frequencies independent of the Kalman filter.
- With a loss of GNSS measurements, the navigation state will continuously be integrated leading to what is called dead reckoning.
- Absolute position sensors can be added in a modular fashion.

The ESKF in this thesis is based on Sola's ESKF, (Solà, 2017) as Sola presents the material in a comprehensive way.

The idea behind the ESKF is to have a nominal state and an error state. The nominal state follows a nonlinear differential equation, while the error state is small, thus linearly integrable and suited for Gaussian filtering. The nominal state $\mathbf{x}$ will consist of integrated high-frequency IMU data $\mathbf{u}_m$, not taking into account noise terms $\mathbf{w}$ and other unmodeled imperfections. This will cause it to accumulate errors and the ESKF will estimate these errors in the error state. The noise and perturbations incorporated in the nominal state will be added into the error state. In parallel with integration of the nominal state, the ESKF propagates the error state's covariance until it receives a correction measurement from for example a GPS. The correction measurement makes the error state observable, although at a slower rate than the integration of the IMU data. After the correction measurement, the error state's mean is subtracted from the nominal-state, reset to zero, and covariance matrix updated. An overview of the states and equations used in the ESKF is presented in Table 5.1.

5.1.1 Quaternions and Euler Angles

In this chapter, the orientation of the vessel is given as a four parameter quaternion vector $\boldsymbol{q}$ which has the advantage of being singularity free (Fossen, 2011). The representation of quaternions is adapted from Solà (2017) and is stated as

Process noise	$\boldsymbol{Q}_k = \boldsymbol{Q}_k^\mathsf{T} > \mathbf{0}$
Measurement noise	$\boldsymbol{V}_k = \boldsymbol{V}_k^\mathsf{T} > \mathbf{0}$
True state	$\mathbf{x}_{t,k+1} = f_t(\mathbf{x}_{t,k}, \mathbf{u}_k, \mathbf{w}_k)$
Nominal state	$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$
Error state	$\delta\mathbf{x}_{k+1} = f_\delta(F\mathbf{x}_k, \delta\mathbf{x}_k, \mathbf{u}_{m,k}, \boldsymbol{i}) = \boldsymbol{F}_\mathbf{x}(\mathbf{x}_k, \mathbf{u}_{m,k}) \cdot \delta\mathbf{x}_k + \boldsymbol{F}_{\boldsymbol{i}} \cdot \boldsymbol{i}$
State observation	$\boldsymbol{y}_k = h(\mathbf{x}_{t,k}) + \boldsymbol{V}$
Initial state	$\mathbf{x}(0) = \mathbf{x}_0$
Initial error state	$\hat{\delta\mathbf{x}}(0) = \mathbf{0}$
Initial error state covariance	$\boldsymbol{P}(0) = \boldsymbol{P}_0$
Transition matrix	$\boldsymbol{F}_\mathbf{x} = \frac{\partial f}{\partial \delta\mathbf{x}}\big\vert_{\mathbf{x}_k, \mathbf{u}_{m,k}}$
Perturbation Jacobian	$\boldsymbol{F}_{\boldsymbol{i}} = \frac{\partial f}{\partial \boldsymbol{i}}\big\vert_{\mathbf{x}_k, \mathbf{u}_{m,k}}$
Observation Jacobian	$\boldsymbol{H}_k = \frac{\partial h}{\partial \delta\mathbf{x}}\big\vert_{\mathbf{x}_k} = \frac{\partial h}{\partial \mathbf{x}_t}\big\vert_{\mathbf{x}_k} \frac{\partial \mathbf{x}_t}{\partial \delta\mathbf{x}}\big\vert_{\mathbf{x}_k}$
Error covariance propagation	$\boldsymbol{P}_k = \boldsymbol{F}_\mathrm{x}\boldsymbol{P}_{k-1}\boldsymbol{F}_\mathrm{x}^\mathsf{T} + \boldsymbol{F}_i\boldsymbol{Q}\boldsymbol{F}_i^\mathsf{T}$
Kalman gain matrix	$\boldsymbol{K}_k = \boldsymbol{P}_k\boldsymbol{H}_k^\mathsf{T}(\boldsymbol{H}_k\boldsymbol{P}_k\boldsymbol{H}_k^\mathsf{T} + \boldsymbol{V})^{-1}$
Error state estimate update	$\hat{\boldsymbol{\delta\mathbf{x}_k}} = \boldsymbol{K}_k(\boldsymbol{y}_k - h(\mathbf{x}_t))$
Error covariance update	$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_k(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)^\mathsf{T} + \boldsymbol{K}_k\boldsymbol{V}\boldsymbol{K}_k^\mathsf{T}$
Injection of observed error	$\mathbf{x}_{k+1} = \mathbf{x}_k \oplus \hat{\delta\mathbf{x}}_k$
True error	$\delta\mathbf{x}_{k+1} = g(\delta\mathbf{x}_k) = \delta\mathbf{x}_k \ominus \hat{\delta\mathbf{x}}_k$
ESKF reset	$\hat{\delta\mathbf{x}}_{k+1} = 0$
Reset Jacobian	$\boldsymbol{G}_k = \frac{\partial g}{\partial \delta\mathbf{x}}\big\vert_{\hat{\delta\mathbf{x}}_k}$
Error covariance reset	$\boldsymbol{P}_k = \boldsymbol{G}_k\boldsymbol{P}_k\boldsymbol{G}_k$

Table 5.1: Error-State Kalman Filter overview, where $\oplus$ and $\ominus$ are the generic composition to respectively add and subtract. (Solà, 2017)

$$\boldsymbol{q} = \begin{bmatrix} q_w \\ \boldsymbol{q}_v \end{bmatrix} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} \tag{5.1}$$

The error dynamics are represented with Euler angles $\delta\theta$ as the error is small thus avoids the singularities.

5.1.2 True-State Kinematics

The kinematic equations for the true state $\mathbf{x}_t$ where $\{t\}$ denotes true are

$$\dot{\mathbf{p}}_t = \mathbf{v}_t \tag{5.2a}$$
$$\dot{\mathbf{q}}_t = \frac{1}{2}\mathbf{q}_t \otimes \boldsymbol{\omega}_t \tag{5.2b}$$
$$\dot{\mathbf{v}}_t = \mathbf{a}_t \tag{5.2c}$$
$$\dot{\mathbf{a}}_{Bt} = \mathbf{a}_w \tag{5.2d}$$
$$\dot{\boldsymbol{\omega}}_{Bt} = \boldsymbol{\omega}_w \tag{5.2e}$$

where $\otimes$ is the quaternion-product. The true acceleration $\mathbf{a}_t$ and angular rate $\boldsymbol{\omega}_t$ are measured by an IMU. These sensor measurements are in the BODY frame and are prone to bias and noise, denoted $\{B\}$ and $\{N\}$. The measurements, denoted $\{m\}$, is then stated as

$$\mathbf{a}_m = \boldsymbol{R}_t^{\mathsf{T}}(\mathbf{a}_t - \mathbf{g}_t) + \mathbf{a}_{Bt} + \mathbf{a}_N \tag{5.3}$$
$$\boldsymbol{\omega}_m = \boldsymbol{\omega}_t + \boldsymbol{\omega}_{Bt} + \boldsymbol{\omega}_N \tag{5.4}$$

where $\boldsymbol{R}_t^{\mathsf{T}} = \boldsymbol{R}(\boldsymbol{q}_t)^{\mathsf{T}}$ and $\boldsymbol{g} = \begin{bmatrix} 0 & 0 & g \end{bmatrix}^{\mathsf{T}}$ is the gravity vector. For more advanced gravity models, see Pavlis et al. (2012). Further, the kinematic system can be written to use the IMU measurements

$$\dot{\mathbf{p}}_t = \mathbf{v}_t \tag{5.5a}$$
$$\dot{\mathbf{q}}_t = \frac{1}{2}\mathbf{q}_t \otimes (\boldsymbol{\omega}_m - \boldsymbol{\omega}_{Bt} - \boldsymbol{\omega}_N) \tag{5.5b}$$
$$\dot{\mathbf{v}}_t = \boldsymbol{R}_t(\mathbf{a}_m - \mathbf{a}_{Bt} - \mathbf{a}_N) + \mathbf{g}_t \tag{5.5c}$$
$$\dot{\mathbf{a}}_{Bt} = \mathbf{a}_w \tag{5.5d}$$
$$\dot{\boldsymbol{\omega}}_{Bt} = \boldsymbol{\omega}_w \tag{5.5e}$$
$$\tag{5.5f}$$

The kinematic system can then be written on state space form using state vector $\mathbf{x}$, input vector $\mathbf{u}$, and perturbation vector $\mathbf{w}$ defined as

$$\mathbf{x}_t = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \mathbf{v} \\ \mathbf{a}_B \\ \boldsymbol{\omega}_B \end{bmatrix} \qquad \mathbf{u} = \begin{bmatrix} \mathbf{a}_m - \mathbf{a}_N \\ \boldsymbol{\omega}_m - \boldsymbol{\omega}_N \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} \mathbf{a}_w \\ \boldsymbol{\omega}_w \end{bmatrix} \tag{5.6}$$

5.1.3 Nominal and Error-State Kinematics

The nominal-state kinematics is modeled without noise as

$$\dot{\mathbf{p}} = \mathbf{v} \tag{5.7a}$$
$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes (\boldsymbol{\omega}_m - \boldsymbol{\omega}_B) \tag{5.7b}$$
$$\dot{\mathbf{v}} = \boldsymbol{R}(\mathbf{a}_m - \mathbf{a}_B) + \mathbf{g} \tag{5.7c}$$
$$\dot{\mathbf{a}}_B = 0 \tag{5.7d}$$
$$\dot{\boldsymbol{\omega}}_B = 0 \tag{5.7e}$$
$$\tag{5.7f}$$

in short $\dot{\mathbf{x}} = f_c(\mathbf{x}, \mathbf{u})$.

The linearized error-state kinematics are trivially obtained for position and both biases as these are derived from linear equations. The kinematics for quaternions and velocity are non-trivial, and detailed derivations can be found in Solà (2017). The error state $\delta\dot{\mathbf{x}} = f_{\delta c}(\delta\mathbf{x}, \mathbf{u}, \mathbf{w})$ is then written as

$$\delta\dot{\mathbf{p}} = \delta\mathbf{v} \tag{5.8a}$$
$$\delta\dot{\boldsymbol{\theta}} = -(\boldsymbol{\omega}_m - \omega_B) \times \delta\boldsymbol{\theta} - \delta\omega_B - \boldsymbol{\omega}_N \tag{5.8b}$$
$$\delta\dot{\mathbf{v}} = -\boldsymbol{R}(\mathbf{a}_m - \mathbf{a}_B) \times \delta\boldsymbol{\theta} - \boldsymbol{R}\delta\mathbf{a}_B - \boldsymbol{R}\mathbf{a}_N \tag{5.8c}$$
$$\delta\dot{\mathbf{a}}_B = \mathbf{a}_w \tag{5.8d}$$
$$\delta\dot{\boldsymbol{\omega}}_B = \boldsymbol{\omega}_w \tag{5.8e}$$
$$\tag{5.8f}$$

5.1.4 Discrete Nominal and Error State

The dynamics have to be discretized, which may be done using the forward Euler method. With this, the nominal state becomes

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \mathbf{v}_k \Delta t + \frac{1}{2}(\boldsymbol{R}(\mathbf{a}_m - \mathbf{a}_{B,k}) + \mathbf{g})\Delta t^2 \tag{5.9a}$$
$$\mathbf{q}_{k+1} = \mathbf{q}_k \otimes \mathbf{q}_k\left((\boldsymbol{\omega}_m - \boldsymbol{\omega}_{B,k})\Delta t\right) \tag{5.9b}$$
$$\mathbf{v}_{k+1} = \mathbf{v}_k + (\boldsymbol{R}(\mathbf{a}_m - \mathbf{a}_{B,k}) + \mathbf{g})\Delta t \tag{5.9c}$$
$$\mathbf{a}_{B,k+1} = \mathbf{a}_{B,k} \tag{5.9d}$$
$$\boldsymbol{\omega}_{B,k+1} = \boldsymbol{\omega}_{B,k} \tag{5.9e}$$
$$\tag{5.9f}$$

and the error state becomes

$$\delta\mathbf{p}_{k+1} = \delta\mathbf{p} + \delta\mathbf{v}\Delta t \tag{5.10a}$$
$$\delta\boldsymbol{\theta}_{k+1} = \boldsymbol{R}^{\mathsf{T}}\left((\boldsymbol{\omega}_m - \boldsymbol{\omega}_{B,k})\Delta t\right)\delta\boldsymbol{\theta} - \delta\boldsymbol{\omega}_{B,k}\Delta t + \boldsymbol{\theta_i} \tag{5.10b}$$
$$\delta\mathbf{v}_{k+1} = \delta\mathbf{v}_k + (-\boldsymbol{RS}\left(\mathbf{a}_m - \mathbf{a}_{B,k}\right)\delta\boldsymbol{\theta} - \boldsymbol{R}\delta\mathbf{a}_{B,k})\Delta t + \mathbf{v_i} \tag{5.10c}$$
$$\delta\mathbf{a}_{B,k+1} = \delta\mathbf{a}_{B,k} + \mathbf{a_i} \tag{5.10d}$$
$$\delta\boldsymbol{\omega}_{B,k+1} = \delta\boldsymbol{\omega}_{B,k} + \boldsymbol{\omega_i} \tag{5.10e}$$
$$\tag{5.10f}$$

The vectors $\boldsymbol{\theta_i}, \mathbf{v_i}, \mathbf{a_i}$ and $\boldsymbol{\omega_i}$ are random impulses affecting orientation, velocity and bias estimates. These are modeled as white Gaussian processes with mean zero and covariance obtained from integrating the covariances of the continuous time noise processes $\mathbf{a}_N, \boldsymbol{\omega}_N, \mathbf{a}_w$ and $\boldsymbol{\omega}_w$ over the timestep Δt, as

$$\boldsymbol{\Theta_i} = \sigma^2_{\boldsymbol{\omega}_N}\Delta t^2\boldsymbol{I} \qquad [rad^2] \tag{5.11a}$$
$$\boldsymbol{V_i} = \sigma^2_{\mathbf{a}_N}\Delta t^2\boldsymbol{I} \qquad [m^2/s^2] \tag{5.11b}$$
$$\boldsymbol{\Omega_i} = \sigma^2_{\boldsymbol{\omega}_w}\Delta t^2\boldsymbol{I} \qquad [rad^2/s^2] \tag{5.11c}$$
$$\boldsymbol{A_i} = \sigma^2_{\mathbf{a}_w}\Delta t^2\boldsymbol{I} \qquad [m^2/s^4] \tag{5.11d}$$

5.1.5 Jacobians and Perturbation Matrices

Summed up, the nominal state vector $\boldsymbol{x}$, error state vector $\delta\boldsymbol{x}$, the input vector $\boldsymbol{u}_m$ and the perturbations vector $\boldsymbol{i}$ are stated as

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \mathbf{v} \\ \mathbf{a}_B \\ \boldsymbol{\omega}_B \end{bmatrix}, \qquad \delta\mathbf{x} = \begin{bmatrix} \delta\mathbf{p} \\ \delta\mathbf{q} \\ \delta\mathbf{v} \\ \delta\mathbf{a}_B \\ \delta\boldsymbol{\omega}_B \end{bmatrix}, \qquad \mathbf{u}_m = \begin{bmatrix} \mathbf{a}_m \\ \boldsymbol{\omega}_m \end{bmatrix}, \qquad \boldsymbol{i} = \begin{bmatrix} \boldsymbol{\theta}_{\boldsymbol{i}} \\ \mathbf{v}_{\boldsymbol{i}} \\ \mathbf{a}_{\boldsymbol{i}} \\ \boldsymbol{\omega}_{\boldsymbol{i}} \end{bmatrix} \tag{5.12}$$

The error-state system is then

$$\delta\mathbf{x}_{k+1} = f_\delta(\mathbf{x}_k, \delta\mathbf{x}_k, \mathbf{u}_{m,k}, \boldsymbol{i}) = \boldsymbol{F}_{\mathbf{x}}(\mathbf{x}_k, \mathbf{u}_{m,k})\delta\mathbf{x}_k + \boldsymbol{F}_{\boldsymbol{i}}\boldsymbol{i} \tag{5.13}$$

From Table 5.1, the ESKF prediction step is given as

$$\boldsymbol{P}_k = \boldsymbol{F}_{\mathrm{x}}\boldsymbol{P}_{k-1}\boldsymbol{F}_{\mathrm{x}}^{\mathsf{T}} + \boldsymbol{F}_{\boldsymbol{i}}\boldsymbol{Q}_k\boldsymbol{F}_{\boldsymbol{i}}^{\mathsf{T}} \tag{5.14}$$

where $\boldsymbol{F}_{\mathbf{x}}$ and $\boldsymbol{F}_{\boldsymbol{i}}$ are the Jacobians of f_δ with respect to the error and perturbation vectors, and $\boldsymbol{Q}_k$ is the covariance matrix of the perturbation impulses i stated

$$\boldsymbol{Q}_k = \begin{bmatrix} \boldsymbol{\Theta}_{\boldsymbol{i}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{V}_{\boldsymbol{i}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Omega}_{\boldsymbol{i}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{A}_{\boldsymbol{i}} \end{bmatrix} \tag{5.15}$$

5.1.6 Observation of Error State

The measurements available are position $\boldsymbol{p}$ and ψ from respectively the GPS and GPS compass in the form

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_{t,k}) + \boldsymbol{v}_k \tag{5.16}$$

where $\boldsymbol{v}_k$ is white noise processes and $\boldsymbol{V}_k$ is the covariance matrix of the measurement noise. Since the filter is estimating the error state, the Jacobian matrix $\boldsymbol{H}$ needs to be defined with respect to the error state $\boldsymbol{\delta}\mathbf{x}$. To ease the computation, $\boldsymbol{H}$ is split up to represent the measurement with respect to the true state, and the true state with respect to the error state as follows

$$\boldsymbol{H} = \left.\frac{\partial \boldsymbol{h}}{\partial \delta\mathbf{x}}\right|_{\mathbf{x}} = \left.\frac{\partial \boldsymbol{h}}{\partial \mathbf{x}_t}\right|_{\mathbf{x}} \left.\frac{\partial \mathbf{x}_t}{\partial \delta\mathbf{x}}\right|_{\mathbf{x}} = \boldsymbol{H}_{\mathbf{x}}\mathbf{X}_{\delta\mathbf{x}} \tag{5.17}$$

where $\mathbf{X}_{\delta\mathbf{x}}$ is found in Solà (2017) to be

$$\mathbf{X}_{\delta\mathbf{x}} = \begin{bmatrix} \boldsymbol{I_3} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{Q}_{\delta\boldsymbol{\theta}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{I}_9 \end{bmatrix}, \qquad \boldsymbol{Q}_{\delta\boldsymbol{\theta}} = \frac{1}{2} \begin{bmatrix} -q_x & -q_y & -q_z \\ q_w & -q_z & q_y \\ q_z & q_w & -q_x \\ -q_y & q_x & q_w \end{bmatrix} \tag{5.18}$$

Furthermore, $\boldsymbol{H}_{\mathbf{x}}$ is found by calculating the Jacobian of the measurement vector with respect to the true state. Now with $h = \begin{bmatrix} \boldsymbol{p} & \psi \end{bmatrix}^\mathsf{T}$ this becomes

$$\frac{\partial \begin{bmatrix} \boldsymbol{p} \\ \psi \end{bmatrix}}{\partial \mathbf{x}_t}\Big|_{\mathbf{x}} = \begin{bmatrix} \frac{\partial \boldsymbol{p}}{\partial \mathbf{x}_t}\big|_{\mathbf{x}} \\ \frac{\partial \psi}{\partial \mathbf{x}_t}\big|_{\mathbf{x}} \end{bmatrix} \tag{5.19}$$

where the partial derivative of the position $\boldsymbol{p}$, with respect to the true state $\boldsymbol{x}_t$, is

$$\frac{\partial \boldsymbol{p}}{\partial \mathbf{x}_t}\Big|_{\mathbf{x}} = \begin{bmatrix} \boldsymbol{I}_3 & \mathbf{0}_4 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix} \tag{5.20}$$

Further, the angles in $\mathbf{x}_t$ are in quaternions, which means some manipulation is needed before ψ can be used. Since the rotation matrices of the two representations are equal (Vik, 2014)

$$\boldsymbol{R}(\Theta) = \boldsymbol{R}(\boldsymbol{q}) \tag{5.21}$$

the relationship between ψ and quaternions (Solà, 2017) are found to be

$$\psi = \mathsf{atan2}(2(q_x q_y + q_w q_z), q_w^2 + q_x^2 - q_y^2 - q_z^2) \tag{5.22}$$

By having

$$\mathbf{w} = \begin{bmatrix} \mathbf{w_1} \\ \mathbf{w_2} \end{bmatrix} = \begin{bmatrix} 2(q_x q_y + q_w q_z) \\ q_w^2 + q_x^2 - q_y^2 - q_z^2 \end{bmatrix} \tag{5.23}$$

we then use the chain rule to find

$$\frac{\partial \psi}{\partial \boldsymbol{q}} = \frac{\partial \psi}{\partial \mathbf{w}} \frac{\partial \mathbf{w}}{\partial \boldsymbol{q}} \tag{5.24}$$

which is split up and derivated in 3 parts

$$\frac{\partial \psi}{\partial \mathbf{w}} = \frac{1}{\mathbf{w}_1 + \mathbf{w}_2^2} \begin{bmatrix} \mathbf{w}_2 & \mathbf{w}_1 \end{bmatrix} \tag{5.25a}$$

$$\frac{\partial \mathbf{w}_1}{\partial \boldsymbol{q}} = 2 \begin{bmatrix} q_z & q_y & q_x & q_w \end{bmatrix} \tag{5.25b}$$

$$\frac{\partial \mathbf{w}_2}{\partial \boldsymbol{q}} = 2 \begin{bmatrix} q_w & q_x & -q_y & -q_z \end{bmatrix} \tag{5.25c}$$

Notice that $\frac{\partial \psi}{\partial \mathrm{w}}$ is the derivative of the atan2 function $\mathtt{atan2}(w_1, w_2)$. We can now write $\boldsymbol{H}_{\mathbf{x}}$ as

$$\frac{\partial h}{\partial \mathbf{x}_t} = \begin{bmatrix} I_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{\partial \psi}{\partial \boldsymbol{q}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \qquad \frac{\partial \psi}{\partial \boldsymbol{q}} \in \mathbb{R}^{1\times 4} \tag{5.26}$$

With these measurements the measurement noise matrix $\boldsymbol{V}_k$ is

$$\boldsymbol{V}_k = \begin{bmatrix} \boldsymbol{\Gamma} & 0 \\ 0 & \Pi \end{bmatrix} \tag{5.27}$$

where $\boldsymbol{\Gamma}$ is the covariance of the position measurements and Π is the covariance of the heading measurements from the GPS compass.

5.1.7 Discretization

The two different ways used to discretize the linearized error-state kinematics (5.8) are the ordinary Euler (5.10) and the use of exact discretization (Van Loan, 1978). The latter gives better computational stability (Moler and Loan, 2003), but requires the exponential of the state transition matrix. The exponential of a matrix $\boldsymbol{A}$ is

$$e^{\boldsymbol{A}} \tag{5.28}$$

which is approximated in Python, using the library *Scipy*, and in MATLAB .

5.1.8 Pseudocode ESKF

The ESKF can be implemented with the pseudocode found in Algorithm 5.1.

```
% Process noise Q
rAcc = sigmaAcc^2 * eye(3)
qBAccMat = sigmaAccBias^2/TAccBias * eye(3)
rGyro = sigmaAcc^2 * eye(3)
qBGyroMat = sigmaGyroBias^2/TGyroBias * eye(3)
qMatCont = blkdiag(rAcc,rGyro,qBAccMat,qBGyroMat)
% Measurement noise
rGps = sigmaGps^2 * eye(3)
rCompass = sigmaCompass^2 * eye(3)
V = blkdiag(rGps,rCompass) %V: Measurement noise
% Initialization of states
x(0) = x_init; P(0) = P_init
for measurement u %[IMU data, Measurement data]
    % Integration of nominal state x
    deltaT = currentTime-prevTime
```

```
    x_k+1 = x_k + deltaT * f_ns(x_k, u_k)
    x <- q <- q/norm(q) % Normalization
    % Do ESKF if measurement in u
    if Measurement data
        deltaT = currentTime-prevMeasurementTime
        % Perturbation Jacobian
        qI = discretize(qMatCont,deltaT) % Discretize Q
        qMat = fI*qI*fI'
        % Transition  Jacobian
        fCont = f(x,u) % StateSpace of f(x,u)
        fX = discretize(fCont,deltaT)
        % Measurement Jacobian
        Hx = parthpartx(Mea)
        Hdx = partXpartdx(x)
        H = Hx*Hdx %
        % Innovation
        innov = [       Mea_GPS - x(1:3)
            angularDifference(psi(x(4:7)) - Mea_Compass)]
        % Kalman
        P = fX*prevP*fX' + qMat% Covariance propogation
        K = P*H'*inv(H*P*H'+V) % KalmanGain
        P = (I-K*H)*P*(I-K*H)' + K*V*K' %ErrorCovariance
        dxHat = K*innov   %Error-state calculation
        x_k+1 = x_k+1 + f(dxHat) % Injection of error-state
        x_k+1 <- q <- q/norm(q) % normalization
        G = partgpartdx(dxHat) % Reset Jacobian
        P = G*P*G % NewErrorCovariance
        prevP = P % Save CovarianceMatrix
    end
end
```

Algorithm 5.1: Pseudocode for the Error-State Kalman Filter.

5.2 Wild Point Filter

When measuring position using a GNSS, the measurements can suddenly jump a great distance for one or more measurements and then jump back again. These wild points may occur when the GNSS acquires a new satellite, when it experiences interference from effects such as back scattering or when it looses a satellite. A wild point can be several hundred miles away and this can cause unexpected behavior in the system. Such wild points can be seen in Figure 5.2.

These wild points need to be removed and a filter has been created according to the following criteria:

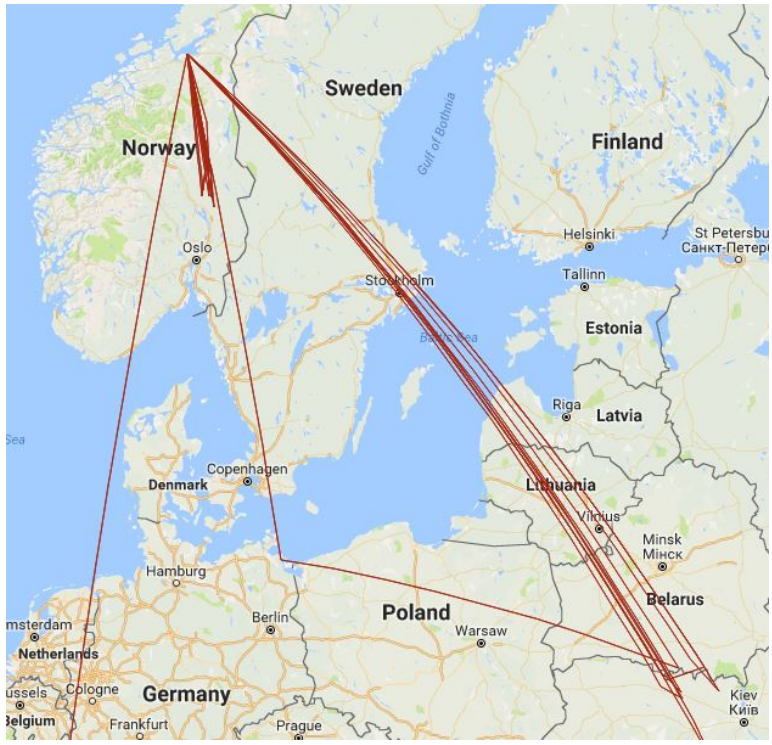


Figure 5.2: Wild points were recorded as far away as the Equator and Eastern Europe, while performing tests in Trondheim, Norway.

1. Compare the current position measurement with the previous approved measurement.

2. If a measurement is more than one meter away from the last measurement, this measurement is not approved.

3. If more than 20 measurements are not approved, the last measurement is approved anyways.

4. The initial position of the filter will be the median of the first 20 measurements.

The first criteria ensures that the filter does not approve a position if it is close to a wild point. This could happen if there for example are two wild points which occur after each other. The third criteria ensures that if there is a legit jump in position or if the filter is incorrectly initialized, the filter will be ”forced” to accept the new position measurement.

5.3 Lever-Arm Compensation

Since a GNSS antenna is not mounted in the same place as the IMU, forces measured by the IMU at the antenna position will not be accurate. It is therefore important to transform all the measurements to a common frame using lever-arm compensation. CG is often used as this common point, since CG does not move when the ship rotates (Vik, 2014). With a NED position measurement $\boldsymbol{p}_1^n$ and a lever-arm $\Delta\boldsymbol{p}^b$ the compensated position $\boldsymbol{p}_0^n$ becomes:

$$\boldsymbol{p}_0^n = \boldsymbol{p}_1^n + \boldsymbol{R}_b^n(\boldsymbol{\Theta})\Delta\boldsymbol{p}^b \tag{5.29}$$

5.4 Overall Navigation System

The ESKF, wild point filter and lever arm compensation make out the navigation system. The GNSS measurements are checked for wild points, before being transformed to the NED frame. These coordinates are lever-arm compensated and then used by the ESKF. See Figure 5.3 for a block diagram of detailing the navigation system.

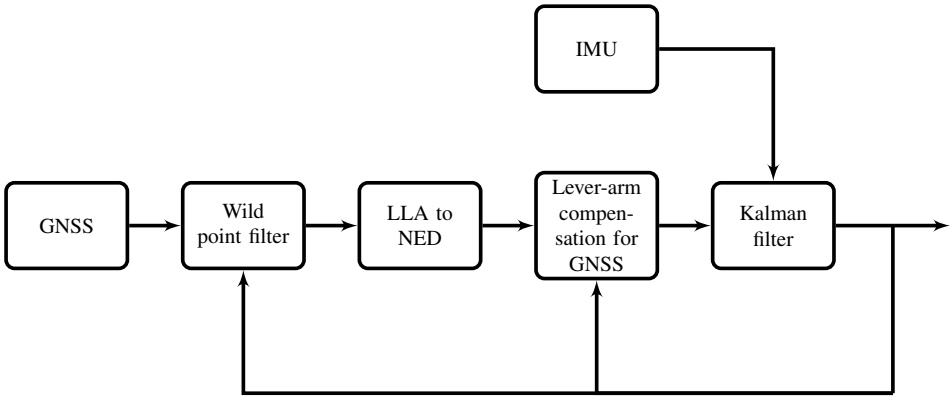


Figure 5.3: Block diagram detailing the navigation system.

Chapter 6

Experimental Platform

This chapter introduces the model ship ReVolt, it's background, and the components installed.

Figure 6.1: The model-scale ship ReVolt under way at Havnebasseng VI in Trondheim.

Some basic stats for the model ship ReVolt are found in Table 6.1.

Length	3m	Draft	0.23m
Width	0.72m	Battery Voltage	12V
Weight	257kg	Max engine power	360W
Top speed	2 knots	Battery capacity	900Wh

Table 6.1: Basic stats for the model ship ReVolt.

6.1 ReVolt's Background

The ReVolt is a concept ship which is designed by the international certification body and classification society, DNV GL in 2014 (DNV GL, 2015). The ship is designed to be an environmentally friendly solution, suitable for short sea shipping along the coast of Norway. The ReVolt is designed to be an unmanned vessel, giving it a futuristic look. The ship is designed and optimized for low energy demand which makes it optimal at speeds about 6 knots. The ReVolt's propulsion system is fully electric with two pods in the stern and an retractable azimuth thruster in the bow (DNV GL, 2015).

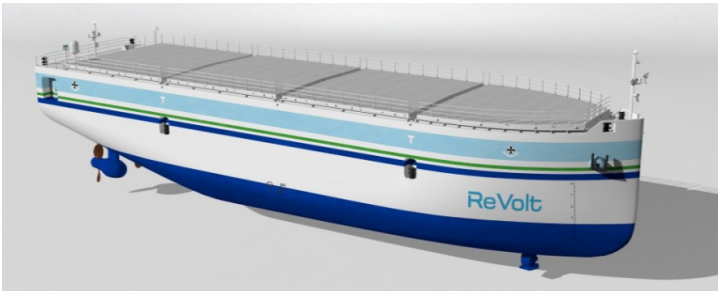
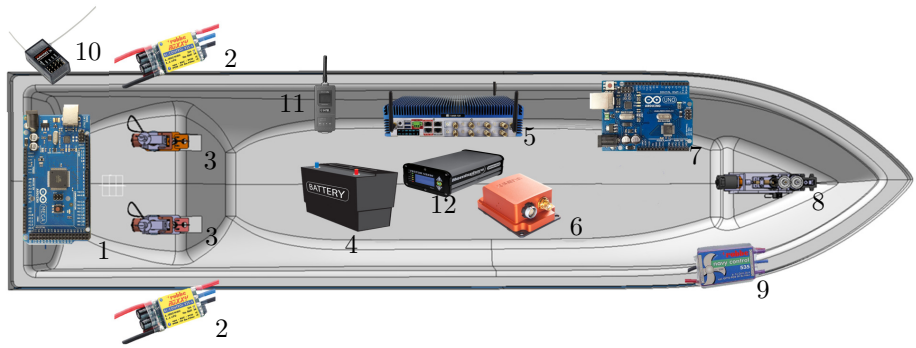


Figure 6.2: An illustration of the concept ship, ReVolt. Courtesy of DNV GL.

In 2014, DNV GL had a 1:20 scale model of the ship made by Stadt Towing Tank (STT), with the same thruster configuration as the concept ship. To avoid confusion, the model of the ReVolt is in the rest of the thesis referred to as just ReVolt.

6.2 Actuators

The actuators on ReVolt are mainly comprised of three thrusters, a bow thruster and two identical podded azimuth thrusters in the stern. These are comprised of several components found in Figure 6.4.



No.	Name	No.	Name
1	Arduino Mega	2	ESC - AC Robbe Roxxy
3	Stern Thruster	4	Batteries
5	Tank-720 PC	6	Xsens MTI-G-710
7	Arduino Uno	8	Bow Thruster
9	ESC - DC Robbe Roxxy	10	Spektrum AR400 RC Receiver
11	Satel Radio Link	12	Hemisphere Vector VS330

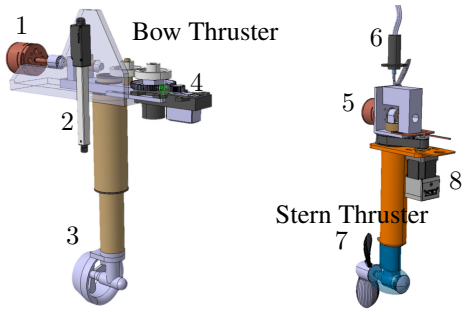
Figure 6.3: Main components and their placements on the ReVolt model. Adapted from STT.

Azimuth Stern Thruster

The two identical stern thrusters consists of an AC-motor to drive the propeller which are powered by an Electronic Speed Controller (ESC) rated at 40A, see no. 3 in Figure 6.3. The pod is rotated by a stepper motor via a belt system, allowing for indefinite rotation of the stern thrusters. The stepper motors are of the type Nanotec PD2-N41 motors which have an RS485 interface. The stepper motors use an encoder to ensure closed-loop position control, which is tuned and set up via the manufacturer's program called NanoPro.

Retractable Azimuth Bow Thruster

The bow thruster consists of a DC-motor to rotate the propeller which is powered by an ESC rated at 35A, see no. 8 in Figure 6.3. The propeller house is rotated by a servo motor, which is not capable of rotating more than $\pm 270°$. A linear actuator is mounted to a construction that holds the DC-motor and servo enabling the propeller house to be lowered or retracted in and out of the vessels hull.



Number	Name	Role
1	DC-Motor	Rotate the propeller.
2	Linear Actuator	Retract/lower the propeller house.
3	Propeller House	Houses propeller that creates thrust when rotated.
4	Servo	Rotate propeller house to get desired thrust direction.
5	AC-Motor	Rotate the propeller.
6	Slip Ring	Allows transmission of power from stationary to rotating structure.
7	Propeller	Rotates to create thrust.
8	Stepper Motor	Rotate propeller to get desired thrust direction.

Figure 6.4: Detailed overview of the thruster setups and their components. Adapted from STT.

6.3 Sensors

ReVolt has two navigational sensors and two auxiliary sensors. The auxiliary sensors consists of a water sensor and battery sensor omitted from this thesis. The two navigational sensors are presented below.

6.3.1 Xsens-MTI-G-710

The Xsens is an INS with a GNSS and IMU, see no. 6 in Figure 6.3. Xsens states: ”The MTi-G-710 GNSS/INS is a GNSS-aided, IMU-enhanced GNSS/INS” (Xsens, 2016) It provides position measurements in ECEF and also measures accelerations and angular velocity in the BODY frame. It also has an Attitude Heading Reference System (AHRS), which fuses the previous mentioned measurements to estimate its heading and velocity in NED. The Xsens is mounted in the roof of ReVolt, as close as possible to the CG. With SBAS corrections, the Xsens is able to provide a position measurement down to $\pm$2m. The orientation measurements have an accuracy of $\pm 0.3°$, $\pm 0.3°$ and $\pm 1.0°$ in roll, pitch and yaw, respectively.

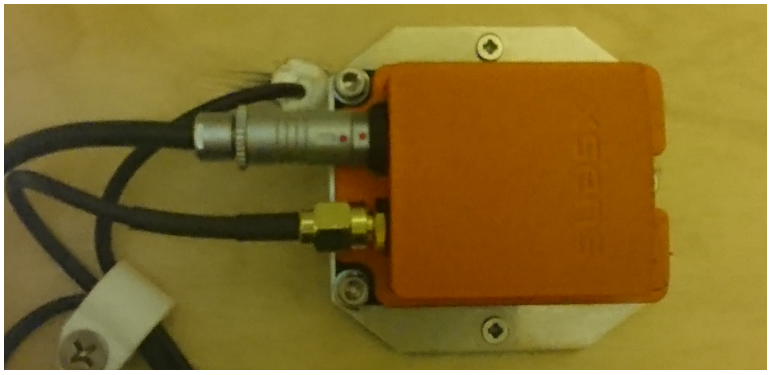


Figure 6.5: The Xsens-MTI-G-710 GNSS and IMU unit, mounted inside ReVolt.

Stats		Description	
GNSS Update rate	4Hz	IMU update rate	2000Hz
GPS codes	L1	Roll accuracy	0.3°
Horizontal accuracy	2.5m	Pitch accuracy	0.3°
Horizontal accuracy (with SBAS)	2.0m	Yaw accuracy	1.0°
Vertical accuracy	5.0m	Velocity accuracy(@30m/s)	0.05m/s

Table 6.2: Xsens specifications (Xsens, 2016).

6.3.2 Hemisphere Vector VS330

The Vector VS330, see no. 11 in Figure 6.3, is a dual antenna GNSS receiver, able to measure ReVolt's position and heading. Using the two antennas, the Vector measures the heading which is stabilized with an internal gyro. This is called a GNSS-compass. It also supports different correction data, such as SBAS and RTK, improving the accuracy and precision of the position measurements. A variety of outputs are supported, including the standard NMEA 0183 protocol used in this thesis. The mounted GNSS is shown in Figure 6.6a, with the antennas mounted on ReVolt's boat towers to minimize multipathing. The antennas are mounted with the forward antenna as the primary and a baseline of 2.05m.

With RTK correction data, the Vector VS330 is able to provide accurate heading and position measurement down to $\pm 0.2°$ and ± 1cm.

Satel Radio Link

The Satel radio link, see no. 12 in Figure 6.3, feeds RTK correction data, received from a local GNSS base station, to the Vector VS330 via a RS232 interface. Figure 6.6b shows the Satel radio link installed in ReVolt with the antenna mounted on the stern boat tower. The base station sends correction data using the following data:

Stats		Description	
GNSS Update rate	20Hz	Heading accuracy	0.05°
GPS codes	L1,L2		
Horizontal accuracy	0.30m	Vertical accuracy	0.60m
Horizontal accuracy (with SBAS)	0.30m	Vertical accuracy (with SBAS)	0.60m
Horizontal accuracy (with RTK)	0.01m	Vertical accuracy (with RTK)	0.02cm

Table 6.3: Specifications on the Hemisphere RTK GNSS (Hemisphere, 2017).

Protocol: RTCM3 Frequency: 441.00MHz Baud rate: 19200

(a) Hemisphere mounted in ReVolt. **(b)** Radio link for the RTK GNSS.

Figure 6.6: Hemisphere mounted in ReVolt and the Radio link connected.

6.4 Light Beacon

A light beacon has been implemented on ReVolt as a visual aid for the operators of ReVolt. The light beacon has three lights; red, yellow and green, as shown in Figure 6.7. If the water sensor fitted to ReVolt is triggered, the red light will shine. The yellow light will shine continuously when ReVolt is being manually controlled, and blink at a constant rate when ReVolt is in an automatic mode. The green light blinks at constant rate when the system is operating normally. The signal logic is summarized in Table 6.4. The light beacon is not intended as maritime navigational lights, only as a visual aid for the operators.

Light	Status	Description
Red	Constant	Water sensor triggered
Yellow	Blink	DP, heading control or manual thrust allocation
Yellow	Constant	Manual mode
Green	Blink	System OK
Green	Constant	System error

Table 6.4: Signal logic for the light beacon.

Figure 6.7: Close up of the stern boat tower. From the left: Light beacon, Wi-Fi antenna, GNSS antenna, Wi-Fi antenna, radio link antenna.

6.5 Onboard Computer and Microcontrollers

The Onboard Computer (OBC) in ReVolt is a Tank-720, see no. 5 in Figure 6.3, which is a fanless, robust, embedded computer running Linux Ubuntu, hereby called the OBC. On the OBC, a program called Robot Operating System (ROS) is installed, which is a open source ”operating system” providing low level device control, message passing and more. ROS is further described in Section 7.2.1.

For analog Input/Output (I/O)s there are two microcontrollers, an Arduino Mega and an Arduino Uno, see no. 1 and 7 in Figure 6.3. These handle low level communication signals such as Pulse Width Modulation (PWM) signals to the ESC and servomotor, and radio receiver, no 2, 9 and 10 in Figure 6.3.

Chapter 7

Embedded Computerized Control

It is important to create a good computerized control platform and hence create a solid foundation for the control algorithms and other code to reside. The embedded OBC, Arduinos and other surrounding hardware make up the hardware framework, while the code and its structure make up the software environment.

The rest of this chapter will address how the hardware is connected and interfaced with the OBC. The structure of the software environment is also described.

7.1 Hardware Framework

The physical components described in Chapter 6 have all been connected to the OBC using either Universal Serial Bus (USB) or RS232, depending on which was available as standard on the component. Figure 7.1 shows how the components are connected to the OBC.

The wiring on ReVolt is documented in the wiring schematic found in Appendix B.

7.2 Software Environment

The software environment in ReVolt consists of a variety of software. As stated in Section 6.5, the OBC runs the operating system Linux Ubuntu with the software framework ROS on top. Installed on the OBC is

- Linux Ubuntu 14.04 LTS
- ROS Indigo Igloo

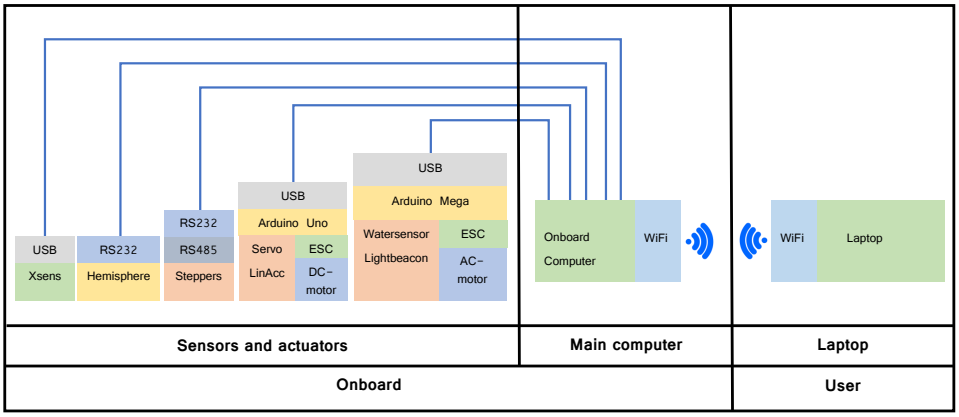


Figure 7.1: How the hardware is connected in ReVolt.

- Python 2.7
- Java 1.7.0_121

The ROS distribution has an End-of-Life (EOL) date set to April 2019 (Marguedas, 2017).

The main programming language used in this thesis is Python, which is widely used when developing and testing new systems. This is due to the fact that Python is designed for code readability and easy implementation. Python features a variety of libraries extensively used throughout this thesis, with useful features such as matrix operations and numerical solvers. However, Python is known to have a huge overhead implying that the code is not as efficient as other languages, but computation power is not an issue for this thesis. The nodes for the two Arduinos are programmed in Arduino's own C++ based language, in conjunction with a ROS library called rosserial. This handles the communication between the Arduinos and the OBC.

7.2.1 Robot Operating System

The entire control system is created on a software framework called Robot Operating System (ROS), which is a collection of open-source software libraries. These provide an operating system-like functionality and can be used to create a very modular system, using nodes. These nodes are executable files which are written in C++, Python or Lisp, with support for more programming languages being added.

The roscore is the main hub of the system, functioning as a registration service for other nodes. It keeps track of all nodes and messages published and subscribed to, such that nodes can communicate with each other. The messages in ROS are called topics and these are sent and received by nodes using a Transmission Control Protocol (TCP)/Internet Protocol (IP)-based message transport, called TCPROS. Nodes publish and subscribe to topics, as they are not addressed to a specific node. Nodes written in different languages

can communicate using topics as long as the structure of the data in the topic is known for both nodes. A short introduction to using ROS is found in Appendix C.

7.2.2 Graphical User Interface

The GUI, as seen in Figure 7.2, is developed using the rqt library and parameter server in ROS. This is made in order to ease the configuration of parameters for both the PID controller and reference filter during the field tests. The GUI also incorporates easy switching between the different operating modes, as seen in Figure 7.3. The GUI is accessed using a laptop connected to the OBC on ReVolt through Wi-Fi.

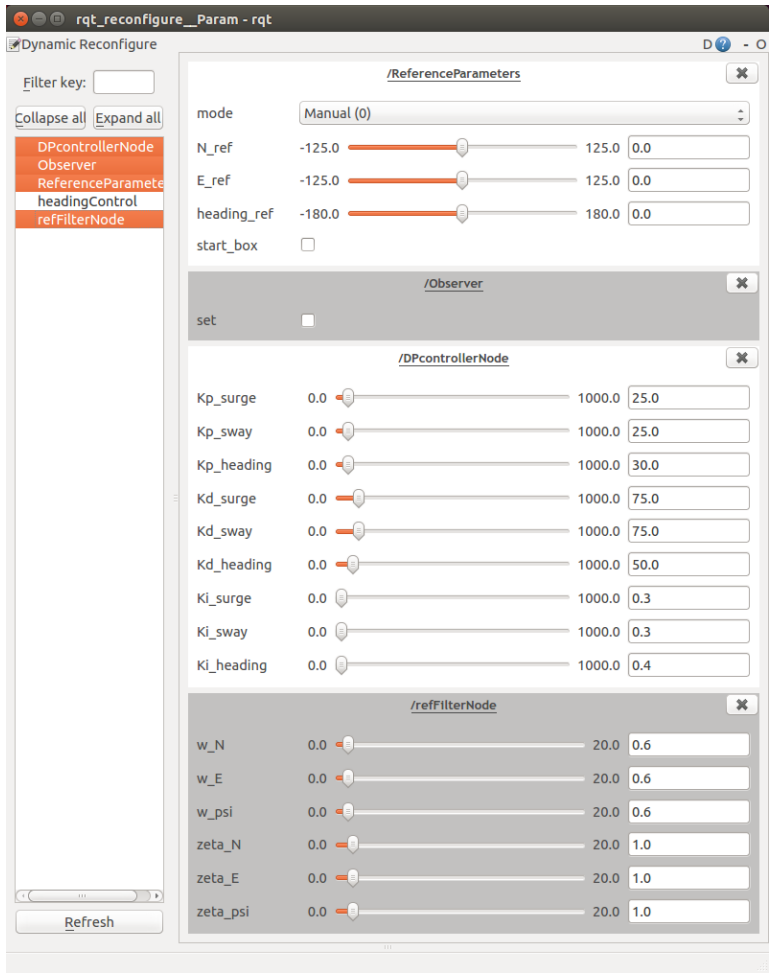


Figure 7.2: The Graphical User Interface programmed for the operators of ReVolt.

7.2.3 Software Structure

There are four operating modes which can be used to control ReVolt:

- Manual control
- Heading controller
- Manual thrust allocation
- Dynamic positioning

where the heading controller mode is omitted from this thesis.

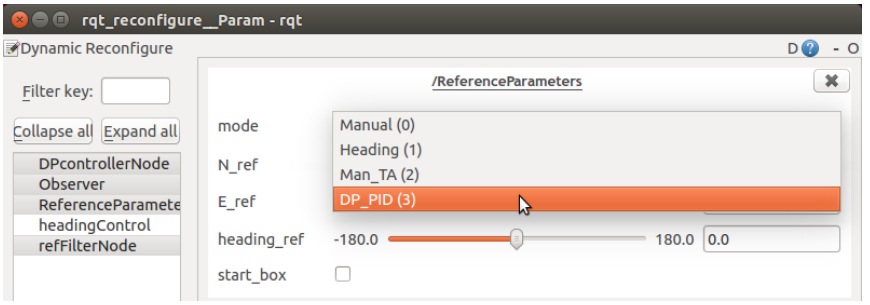


Figure 7.3: Choosing the operating mode of ReVolt using the laptop.

Manual Mode

The manual mode gives the user direct control of the stern thrusters, where their rotations are limited to $\pm 45°$. The bow thruster can also be controlled, as well as lowered and retracted into the hull. This is the default mode of ReVolt at launch and is used when performing basic checks of ReVolt. As there are no controllers related to the manual mode, it only maps the messages from the RC Remote into viable messages and is passed on to the thrusters. An overview is seen in Figure 7.4. If there is a problem or it is desired, it is possible for the operator to flip a switch on the remote control and ReVolt will revert to a safe mode with the same capabilities as the manual mode.

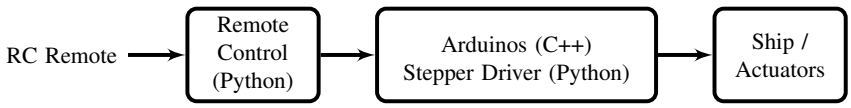


Figure 7.4: Block diagram of the nodes used for manual control and which programming language is used to program them.

Manual Thrust Allocation Mode

Figure 7.5 shows the structure of the manual thrust allocation mode, which utilizes all three thrusters with the bow thruster fixed at 90 degrees. Here the user demands a force relative to BODY for which the thrust allocation calculates a control input. This consists of a desired throttle and angle for each thruster, which is applied to the actuators. This mode is used for testing and tuning the thrust allocation, as described in Section 10.4.2.

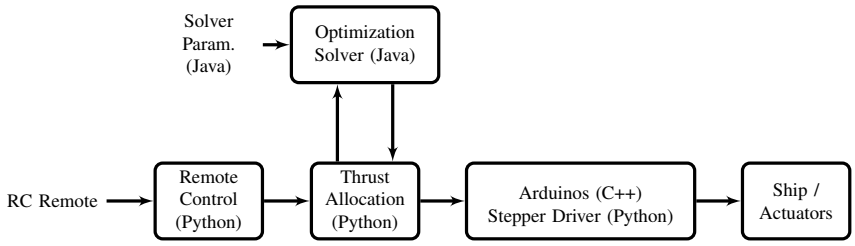


Figure 7.5: Block diagram of the nodes used for manual thrust allocation control and which programming language is used to program them.

DP Controller Mode

The DP mode is the most advanced mode involving the full DP system, seen in Figure 7.6. The mode utilizes feedback control as explained in Chapter 4 where the user, via a laptop, inputs a setpoint in 3-DOF through the parameter server. The reference filter feeds the PID with a realistic trajectory coinciding with the users input. The controller rate is set to 5Hz which should sufficiently handle the slow dynamics of ReVolt.

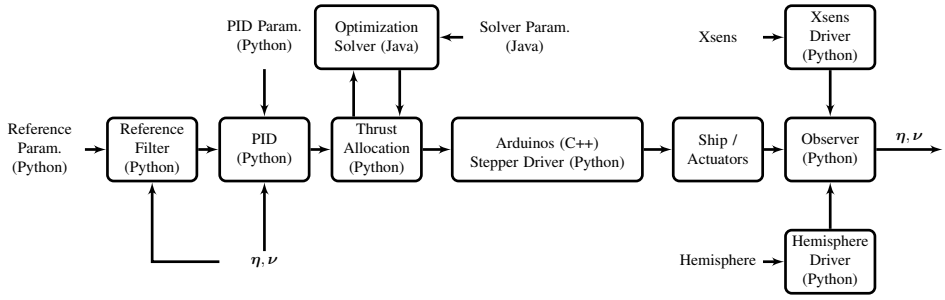


Figure 7.6: Block diagram of the nodes in the DP controller and which programming language is used to program them.

More detailed representations of the system structure for the different control modes can be found in Appendix D.

7.2.4 Thrust Allocation

The Thrust Allocation from DNV GL is written in Java. It is optimization based, much like the problem in (4.26). It punishes big steps in control inputs while seeking the optimum inputs to attain desired control force. The initialization parameters are listed in Table 7.1.

Parameter	Value	Description
numThr	3	Number of thrusters
tMax	$\{25, 25, 14\}$	Max thrust
tMin	$\{-25, -25, -6.1\}$	Minimum thrust
setFixedAngles	$\{0.0, 0.0, \text{PI}/2\}$	Angle for fixed thrusters
setFixedAngleEnables	$\{false, false, true\}$	Enables fixed thrusters if value $true$
Position	$\{\text{Position}(-1.65, -0.15)$ $\text{Position}(-1.65, 0.15)$ $\text{Position}(1.15, 0.0)\}$	Geometry of thrusters in (lx,ly)
setRateLimitDirections	$\{\text{PI}/6, \text{PI}/6, \text{PI}/36\}$	Radians per step $\Delta\alpha$
setRateLimitDirections	$\{10.0, 10.0, 4.0\}$	Newton per step Δf

Table 7.1: Thrust allocation parameters.

Implementation

A decent way to implement the thrust allocation in ROS is by having a Python script send and receive messages to the Java class. This is done with a open source library called PY4J which essentially enables sending messages between Python and Java classes. The ROS node sends an array with wanted control force $\boldsymbol{\tau}$ which the Gateway server implemented in PY4J forwards to the Java instance. The thrust allocation then returns a vector of corresponding forces and angles for each thruster. The ROS node then translates the forces into a suitable control input $\boldsymbol{u}$. This is illustrated in Figure 7.7.

Rotation Rate Constraint Identification

The parameter to limit the rotation rate, $\Delta\alpha$ in Section 4.2.4, of the stern steppers are obtained from running the PD2-n41 steppers using the included software. By setting the steppers to rotate 180°, the planned trajectory is presented in the manufacturers software. The trajectory is presented in Figure 7.8.

The acceleration phase and deceleration phase are equal, which implies that for bigger changes in angle, the rotation rate will be larger than for small changes in angle. By calculating the rotation rate for a 180°change in angle and for only the acceleration and deceleration phase, we get

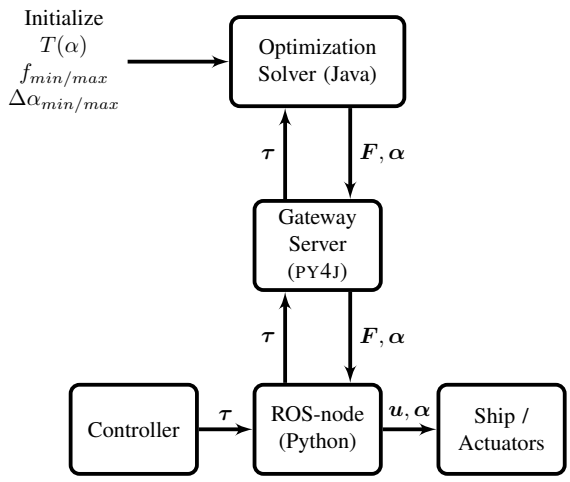


Figure 7.7: Communication flow between ROS and the optimization solver.

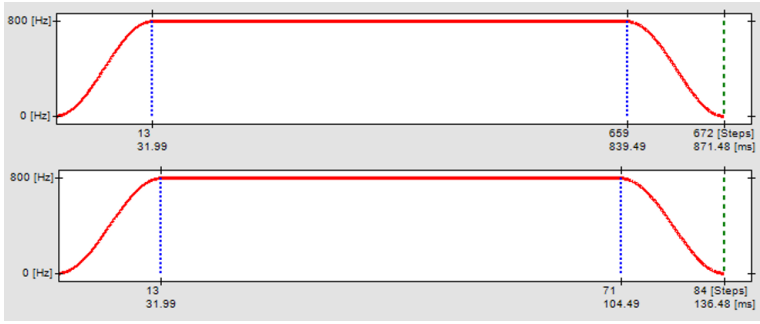


Figure 7.8: Plot of the manual run in Nanotec's software NanoPro of the PD2-N41 stepper motors.

$$180^\circ: \quad r = \frac{180^\circ}{0.871s} = 206^\circ/s \tag{7.1}$$

$$7^\circ: \quad r = \frac{7^\circ}{0.064s} = 108^\circ/s \tag{7.2}$$

where r is the rotation rate of the steppers and the 7° indicates the change in angle for the acceleration and deceleration part only. A compromise between the two rates are chosen at $150^\circ/s$ or $\frac{5\pi}{6}$, which translates to

$$\Delta\alpha = \frac{5\pi}{6}h = \pi/6, \qquad h = 0.2s \tag{7.3}$$

where h is the inverse of the controller rate.

7.3 Sensors

The two navigational sensors, the Xsens IMU/GNSS and Hemisphere GNSS, have both been setup and drivers have been implemented such that the data can be received from them. The sampling rate of the sensors have been set such that they will work optimally for the Kalman filter. The sampling rate which are set on the sensors can be found in Table 7.2.

Sampling rate [Hz]	Xsens IMU	Xsens GNSS	Hemisphere GNSS
Maximum	2000	4	20
Set	100	4	20

Table 7.2: The sampling rate of the two GNSSs and the IMU.

7.3.1 Xsens

To communicate with the Xsens in ROS, a third-party driver (Arunvydhya, 2017) was modified and used on ReVolt. This driver provides all the measurements wanted from the Xsens. The following measurements are provided by the Xsens:

- Orientation data: roll, pitch and yaw.
- Linear accelerations: surge acceleration, sway acceleration and heave acceleration.
- Angular velocities: roll, pitch and yaw rate.
- Position data: Latitude, longitude and altitude.
- Linear velocities: north, east and down velocity by combining GNSS and IMU to estimate velocity.

By default the Xsens is oriented to be mounted with the bottom facing down towards the Earth center. In ReVolt however, the Xsens is mounted in the ”roof” with the bottom facing up. Therefore the sensor output of the Xsens is rotated, in the Xsens setup, such that the measurements are correct with this new orientation. Therefore, using the provided Xsens software, the orientation is rotated by 180° around the x-axis.

7.3.2 Hemisphere

The Hemisphere driver, written during the Fall 2016, was prone to parsing errors, and not utilizing check sums and efficient serial port read functionalities. Therefore, a third party NMEA driver (Perko, 2017), is acquired and implemented into the ROS environment. The driver is modified to include parsing for all NMEA sentences used in this thesis, consisting of

- GPGGA - Latitude and Longitude, as well as altitude and location accuracy data.

- GPHDT - Heading data.
- GPVTG - Velocity and course data.

A part of the location accuracy information found in the GPGGA message, indicates whether or not the GNSS is receiving RTK correction data. This information is useful to determine the quality of the position data which is being received when undertaking experiments.

7.4 Logging System

When developing a new system, having the ability to log data and being able analyze this at a later time is of great importance. Some of the greatest benefits of having a logging system are:

- The user is able to analyze the performance of the system.
- The user can ”replay” a particular scenario which may arise during testing.
- Errors and bugs in the code can be analyzed.

To log data on ReVolt, the logging system which is a part of ROS has been used previously. This system records all the data into ROSbags, which have to be properly shut down in order to save the data. If the system looses power or the power is cut manually, it is very likely that the recordings will be corrupt and useless.

Therefore a new system to log data on ReVolt has been created. This system is written in Python and writes data directly to a Comma-Separated Values (CSV) file. For each timestep, the CSV file belonging to the data is opened, written too, saved and closed. If the system looses power the file is already saved and no data, other than the last signal, will be lost.

Every time data is written to a log file, the time is also recorded. The time stamp which is used has a resolution of 1 nano second, making the analysis of the data easier. Previously there was a problem with the time stamps only having a resolution of 0.01 seconds and this caused difficulties when analyzing the timestep of the system.

The two different ways of logging data have their pros and cons, which are listed below:

- ROSbag
  - Pro: Can easily replay logs in ROS
  - Con: File becomes corrupt if the system is shutdown improperly
- CSV-log
  - Pro: Logs will not become corrupt upon improper shutdown
  - Con: Logging of time stamps are not extensively tested and verified

Chapter 8

Modelling the ReVolt Model Ship

This chapter describes the modeling of ReVolt according to the mathematical model in Section 2.2 and how the necessary parameters were found.

8.1 Center of Gravity

To find the CG for ReVolt, a crane was used to lift ReVolt using two straps placed close together. The straps were placed close together to create a small surface area which lifted ReVolt, such that it could balance on the straps. The total length of ReVolt, or Length Between Perpendiculars (LPP) is 3 meters. Through trial and error it was found that ReVolt could balance on the straps when they were placed at 155cm from the stern. The experiment was performed when ReVolt was fully loaded with weights. The CG for ReVolt with weights is therefore at 155cm from the stern. See Figure 8.1.

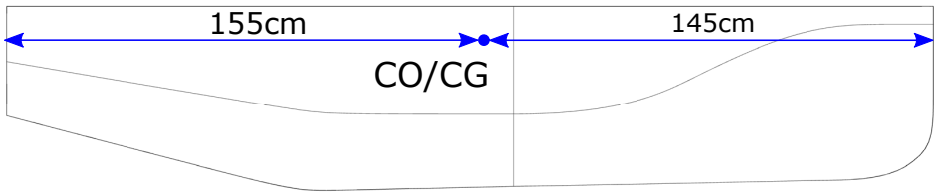


Figure 8.1: Center of Gravity on ReVolt.

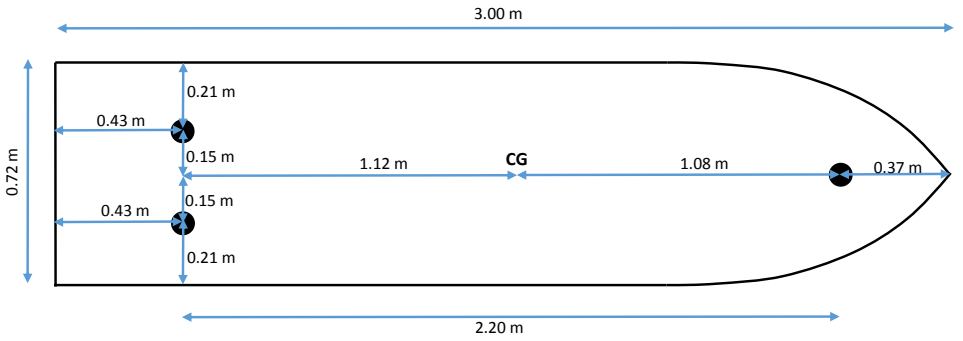


Figure 8.2: The dimensions of ReVolt, indicating the Center of Gravity and the thrusters placement.

8.2 Mathematical Model

For the mathematical model of ReVolt, the model and assumptions described in Section 2.2 has been used. With help from DNV GL, the added mass matrix $\boldsymbol{M}_A$ given in (2.8) and the linear damping matrix ($\boldsymbol{D}_L$) given in (2.11) were found through Wave Analysis by Diffraction and Morison (WADAM) simulations. The non-linear damping matrix ($\boldsymbol{D}(\boldsymbol{\nu})$) was not possible to attain from the computations and have been emitted from the model. The linear damping term will dominate over the non-linear term, therefore this should not be a problem. The relevant raw results can be found in Appendix E. The results from the computations are non-dimensional and need to be dimensionalized by multiplying the correct factors into the matrix values. The following values were used for the non-dimensional factors:

$$\rho = 1025kg/V \quad g = 9.807m/s^2 \quad LPP = 3m \tag{8.1}$$

where ρ is the density of sea water and g is gravity.

8.2.1 Inertia Matrix

The inertia matrix consists of the rigid-body inertia and the added mass. It is given by:

$$\boldsymbol{M} = \boldsymbol{M}_{RB} + \boldsymbol{M}_A \tag{8.2}$$

In order to find the rigid-body inertia matrix, ReVolt's weight and moment of inertia are required.

Mass

A crane and digital scale was used to find the total mass of ReVolt, see Figure 8.3. The scale showed a weight of 167 kg, but at the time of the experiment the Hemisphere GNSS

was not onboard. After adding the weight of the Hemisphere GNSS and subtracting the weight of the weighing equipment and two misplaced 5kg weights onboard, the total mass of ReVolt with out weights onboard, was

$$m_{\mathrm{ReVolt}} = 157kg \tag{8.3}$$

See Table 8.1 for a breakdown of the weights in the experiment .

Measured weight	167
Straps	-2
Planks	-1
GNSS and Satel	3
Weights	-10
ReVolt	157 kg

Table 8.1: Breakdown of the different weights in the experiment. All units are in kg.

With 100 kg of weights added to ReVolt, the total mass becomes

$$ma_{\mathrm{total}} = m_{\mathrm{ReVolt}} + m_{\mathrm{weights}} = 257kg \tag{8.4}$$

In ReVolt there are five rods where the weights can be mounted. These rods are separated along the x-axis of the ship and the weights are placed in such a manner that the trim of ReVolt is close to level. During the sea trails the weights were distributed as follows:

(Bow) 5 - 6 - 5 - 2 - 2 (Stern)

where the numbers indicate the number of 5 kg weights which were added to each rod.

Moment of Inertia

To find the rigid-body inertia matrix ($\boldsymbol{M}_{RB}$), the moment of inertia about the Z_b axis (I_Z) is required. The volume moment of inertia (V_{I_Z}) and volume (vol) was computed using a 3D computer model of the scale-model ship ReVolt and the program Rhino 5. The volume and V_{I_Z} were found to be:

$$vol = 0.268m^3 \quad V_{I_Z} = 0.310m^5 \tag{8.5}$$

The mass moment of inertia then becomes:

$$I_z = \frac{mass_{total}}{vol} V_{I_Z} = \frac{257}{0.268} 0.310 = 297.597kgm^2 \tag{8.6}$$

Rigid Body Inertia Matrix

With the mass of ReVolt, the moment of inertia and the CG being the same as the center of origin, and therefore the distance from the CO to the CG, $x_g = 0$, the rigid-body inertia matrix becomes:

$$\boldsymbol{M}_{RB} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & mx_g \\ 0 & mx_g & Iz \end{bmatrix} = \begin{bmatrix} 257 & 0 & 0 \\ 0 & 257 & 0 \\ 0 & 0 & 298 \end{bmatrix} \tag{8.7}$$

Figure 8.3: Weighing ReVolt using a crane and a digital scale.

Added Mass Matrix

The added mass matrix is given from the WADAM simulations:

$$\begin{aligned}\boldsymbol{M}_A &= \begin{bmatrix} -X_{\dot{u}} & 0 & 0 \\ 0 & -Y_{\dot{v}} & -Y_{\dot{r}} \\ 0 & -Y_{\dot{r}} & -N_{\dot{v}} \end{bmatrix} \\ &= \begin{bmatrix} 0.0253 \cdot \rho \cdot vol & 0 & 0 \\ 0 & 0.1802 \cdot \rho \cdot vol & 0.0085 \cdot \rho \cdot vol \cdot LPP \\ 0 & 0.0085 \cdot \rho \cdot vol \cdot LPP & 0.0099 \cdot \rho \cdot vol \cdot LPP^2 \end{bmatrix} \\ &= \begin{bmatrix} 6.930 & 0 & 0 \\ 0 & 49.440 & 7.007 \\ 0 & 7.028 & 24.556 \end{bmatrix} \end{aligned} \tag{8.8}$$

Total Inertia Matrix

With $\boldsymbol{M}_{RB}$ and $\boldsymbol{M}_A$ the total inertia matrix becomes:

$$\boldsymbol{M} = \boldsymbol{M}_{RB} + \boldsymbol{M}_A = \begin{bmatrix} 263.93 & 0 & 0 \\ 0 & 306.44 & 7.00 \\ 0 & 7.03 & 322.15 \end{bmatrix} \tag{8.9}$$

8.2.2 Coriolis-Centripetal Matrix

The Coriolis-Centripetal matrix $\boldsymbol{C}(\boldsymbol{\nu})$ given in (2.9) is restated below:

$$\begin{aligned}\boldsymbol{C}(\boldsymbol{\nu}) = \boldsymbol{C}_{RB}(\boldsymbol{\nu}) + \boldsymbol{C}_A(\boldsymbol{\nu}) = &\begin{bmatrix} 0 & 0 & -m(x_g r + v) \\ 0 & 0 & mu \\ m(x_g r + v) & -mu & 0 \end{bmatrix} + \\ &\begin{bmatrix} 0 & 0 & Y_{\dot{v}} v + Y_{\dot{r}} r \\ 0 & 0 & -X_{\dot{u}} u \\ -Y_{\dot{v}} v - Y_{\dot{r}} r & X_{\dot{u}} u & 0 \end{bmatrix} \end{aligned} \tag{8.10}$$

By using the values from the inertia matrix in Section 8.2.1, values for the Coriolis-centripetal matrix is found.

$$\boldsymbol{C}_{RB}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & -257v \\ 0 & 0 & 257u \\ 257v & -257u & 0 \end{bmatrix} \tag{8.11}$$

$$\boldsymbol{C}_A(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & -49.44v - 7.00r \\ 0 & 0 & 6.93u \\ 49.44v + 7r & -6.93u & 0 \end{bmatrix} \tag{8.12}$$

$$\boldsymbol{C}(\boldsymbol{\nu}) = \boldsymbol{C}_{RB}(\boldsymbol{\nu}) + \boldsymbol{C}_A(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & -207.56v + 7.00r \\ 0 & 0 & 250.07u \\ 207.56v - 7.00r & -250.07u & 0 \end{bmatrix} \tag{8.13}$$

8.2.3 Damping Matrix

The linear damping matrix given by the WADAM simulations is

$$\begin{aligned} \boldsymbol{D} &= \begin{bmatrix} -X_u & 0 & 0 \\ 0 & -Y_v & -Y_r \\ 0 & -N_v & -N_r \end{bmatrix} \\ &= \begin{bmatrix} 0.102 \cdot \rho \cdot vol \cdot \frac{g}{LPP} & 0 & 0 \\ 0 & 1.212 \cdot \frac{g}{LPP} & 0.056 \cdot \rho \cdot vol \cdot \sqrt{g \cdot LPP} \\ 0 & 0.056 \cdot \rho \cdot vol \cdot \sqrt{g \cdot LPP} & 0.0601 \cdot \rho \cdot vol \cdot \sqrt{g \cdot LPP} \end{bmatrix} \\ &= \begin{bmatrix} 50.66 & 0 & 0 \\ 0 & 601.45 & 83.05 \\ 0 & 83.10 & 268.17 \end{bmatrix} \end{aligned} \tag{8.14}$$

8.3 Bollard Pull Tests

As the propellers used on ReVolt are custom made, there are no available data on the force they generate at certain revolutions. As as solution to this, a crude bollard pull test was conducted in calm sea. It was conducted by having a rope tied to the boat in one end and a digital luggage weight on the other end, as seen in Figure 8.6. Each type of thrusters were mapped at different inputs and curve fitting techniques were used to find a suitable constant.

From Figure 8.4, the experimental data with the curve fitted model is plotted. The model used is

$$F_i = K_i |u_i| u_i \tag{8.15}$$

Notice that due to the bow thruster's propeller being asymmetrical, it does not produce the same thrust in both directions as opposed to the stern thrusters. The constants are

$$K_1^{\pm} = K_2^{\pm} = 2.7e-3 \qquad K_3^{-} = 6.172e-4 \qquad K_3^{+} = 1.518e-3 \tag{8.16}$$

where the subscript of K_i denotes the thruster number, given in Figure 4.4. With these values implemented in the thrust allocation, a map of available thrust for each angle can be seen in Figure 8.5.

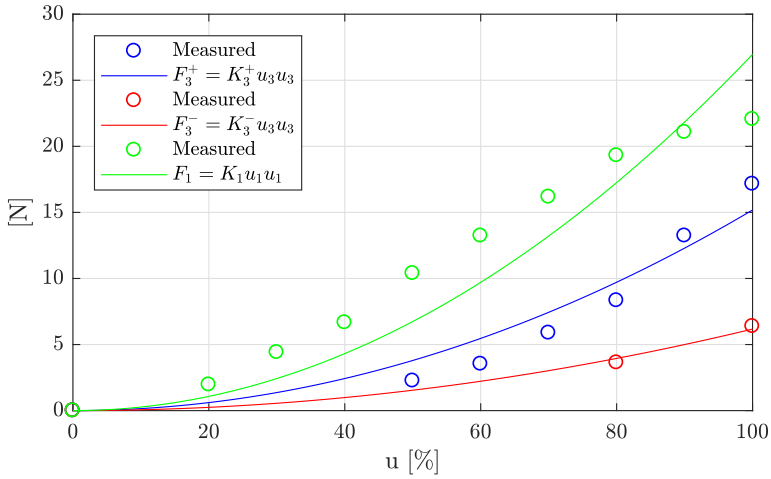


Figure 8.4: Measured and interpolated thrust force F_i for different inputs.

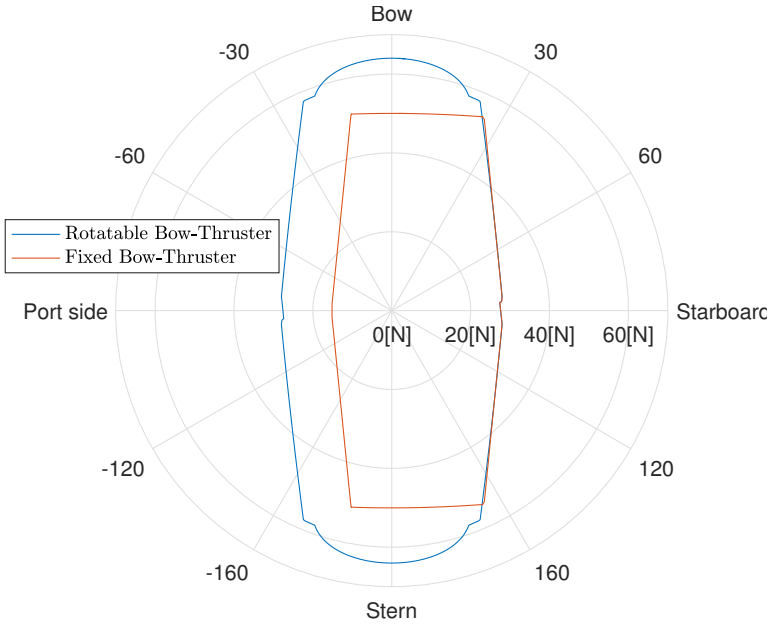


Figure 8.5: ReVolt's available thrust for all directions with $F_1^{max} = F_2^{max} = 25$, $F_3^{max} = 14$ and $F_3^{min} = -6.1$

Figure 8.6: Kjetil performing a bollard pull test with a luggage scale. Photo by Egil Eide.

Chapter 9

Simulation Results

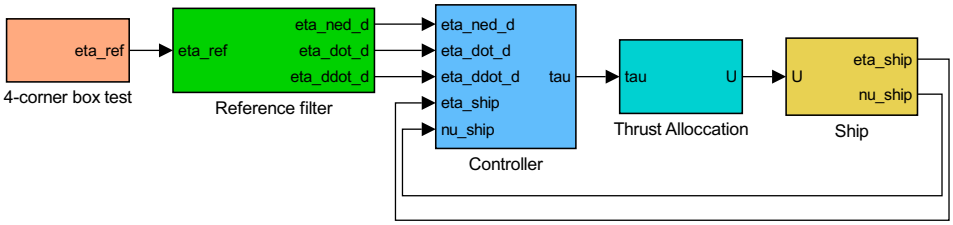


Figure 9.1: Simulation created in Simuliunk.

In order to assure that the new updates to the ReVolt software and algorithms will work as expected, a full closed loop simulation of the system has been created in MATLAB. The simulation has been made with the possibility to add a disturbance to mimic a force due to current. The specific algorithms tested are:

- Reference filter
- PID motion controller with feedforward
- DNV GL's thrust allocation

The observer is not implemented in this simulation.

Since there is no verified mathematical model of ReVolt, a model for CyberShip II is adapted from Skjetne et al. (2004) and used in the simulations. The dynamics of CyberShip II is kept the same, but with modified thruster setup to mimic ReVolt's three rotatable azimuth thrusters.

9.1 Test Scenario

To test the DP system, a 4-corner test used during the AMOS research cruise 2016 (Skjetne et al., 2017), has been adopted. This is shown in Figure 9.2. The 4-corner box test ensures that all motions of the vessel are tested and it is practical during real life experiments because the vessel returns to the initial position and heading, ready for another test.

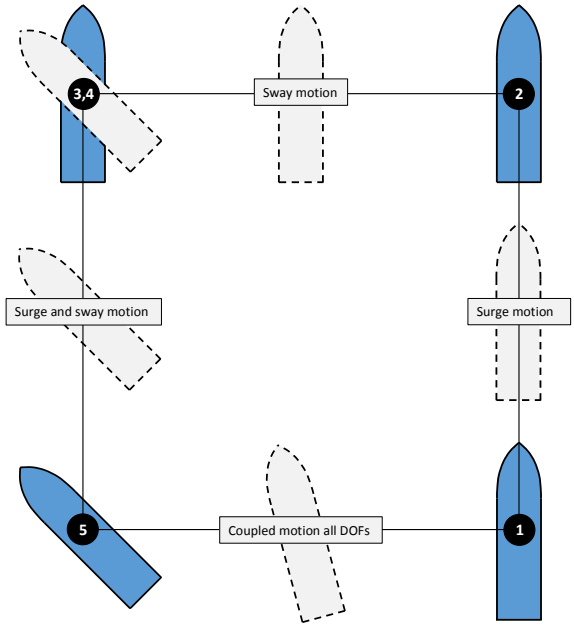


Figure 9.2: The 4-corner box test.

The test starts with the vessel pointing north at heading 0° and then the following reference changes are commanded:

1. Change position x meters due north. This test surge motion ahead.
2. Change position x meters due west. This test sway motion to port.
3. Change heading 45° counterclockwise. This test yaw motion.
4. Change position x meters due south. This test a coupled sway and surge motion to port and astern, respectively.
5. Change position x meters due east and change heading 45° clockwise. This test a coupled surge, sway and yaw motion.

9.2 Reference Filter Parameters

For the simulations, the sampling rate is set to 5 Hz, since this is a compromise between a fast, yet precise simulation. With a sampling time of $h = 0.2$s it is possible to compute the maximum values for the natural frequency of the reference filter, while keeping the filter stable as described in Section 3.1.3.

From (3.14), the smallest eigenvalue of the system must be

$$\begin{aligned} h &\leq -\frac{2}{\lambda} \\ \lambda &\geq -\frac{2}{h} \\ \lambda &\geq -\frac{2}{0.2} \\ \lambda &\geq -10 \end{aligned} \tag{9.1}$$

With (9.1) it can be shown that the largest natural frequency ω for the reference filter must be

$$\omega_{i,largest} \leq 10 \tag{9.2}$$

Therefore the natural frequency matrix $\mathbf{\Omega} > 0$ will be

$$\mathbf{\Omega} = \begin{bmatrix} 0 < \omega_{n,N} \leq 10 \\ 0 < \omega_{n,E} \leq 10 \\ 0 < \omega_{n,\psi} \leq 10 \end{bmatrix} \tag{9.3}$$

The velocity and acceleration limits for which the reference filter is tuned according to are:

$$\nu_u^b = 0.5m/s \quad -\nu_u^b = 0.2m/s \quad \pm\nu_v^b = 0.1m/s \quad \pm\nu_r^b = 3deg/s \tag{9.4}$$

$$\dot{\nu}_u^b = 0.1m/s^2 \quad -\dot{\nu}_u^b = 0.05m/s^2 \quad \pm\dot{\nu}_v^b = 0.05m/s^2 \quad \pm\dot{\nu}_r^b = 1.5deg/s^2 \tag{9.5}$$

The saturation limits in the reference filter are set conservatively to ensure that the vessel will be able to comfortably follow the desired acceleration and velocity. The saturation limits refer to the distance which the "carrot" is ahead of the vessel. Based on the velocity and acceleration limits given in (9.4) and (9.5) as well, the following saturation limits are used:

$$x_{max}^b = 1 \quad -x_{max}^b = 0.5 \quad \pm y_{max}^b = 0.3 \quad \pm\psi_{max}^b = 15^\circ \tag{9.6}$$

This resulted in the following tuning parameters:

$$\mathbf{\Delta} = \begin{bmatrix} \zeta_N \\ \zeta_E \\ \zeta_{psi} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad , \quad \mathbf{\Omega} = \begin{bmatrix} \omega_{n,N} \\ \omega_{n,E} \\ \omega_{n,\psi} \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.6 \\ 0.6 \end{bmatrix} \tag{9.7}$$

9.3 PID Parameters

For CyberShip II, the maximum thrust which the ship's thrusters can supply in each direction is:

$$\boldsymbol{\tau}_{max} = \begin{bmatrix} \tau_{X,max} \\ \tau_{Y,max} \\ \tau_{N,max} \end{bmatrix} = \begin{bmatrix} 8 \\ 1 \\ 2 \end{bmatrix} \tag{9.8}$$

Using the tuning method as described in Section 4.1.3, the following controller gains are obtained and used for the rest of the simulations:

$$\boldsymbol{K_p} = \begin{bmatrix} K_{p,surge} & 0 & 0 \\ 0 & K_{p,sway} & 0 \\ 0 & 0 & K_{p,yaw} \end{bmatrix} = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 12 \end{bmatrix} \tag{9.9}$$

$$\boldsymbol{K_i} = \begin{bmatrix} K_{i,surge} & 0 & 0 \\ 0 & K_{i,sway} & 0 \\ 0 & 0 & K_{i,yaw} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9.10}$$

$$\boldsymbol{K_d} = \begin{bmatrix} K_{d,surge} & 0 & 0 \\ 0 & K_{d,sway} & 0 \\ 0 & 0 & K_{d,yaw} \end{bmatrix} = \begin{bmatrix} 90 & 0 & 0 \\ 0 & 60 & 0 \\ 0 & 0 & 25 \end{bmatrix} \tag{9.11}$$

9.4 Results

The aim of the simulation is to show that the control algorithms work as expected and is able to control CyberShip II around the 4-corner box test. The distances in the 4-corner box test is set to 2 meters. CyberShip II is setup with a fixed bow thruster at 90° and unconstrained rotation rates of the two stern thrusters. Both the reference feedforward and feedback elements of the controller are in use. A constant disturbance is added to the simulation in order to see that the feedback part of the controller is working correctly. The constant disturbance is set to

$$\mathbf{w} = \begin{bmatrix} North \\ East \end{bmatrix} = \begin{bmatrix} 1.5 \\ 0 \end{bmatrix} \tag{9.12}$$

The disturbance values are in $[N]$ and indicate a disturbance coming from the North.

Figure 9.3 shows how CyberShip II follows the desired path for the 4-corner box test, from a starting position of $\boldsymbol{\eta} = [0, 0, 0]^\mathsf{T}$. With the constant environmental disturbance (9.12), there are small errors in the pose of the vessel. Further in Figure 9.4, the vessel tracks the desired trajectories, outputted by the reference filter, with high accuracy. This can be confirmed in Figure 9.5 where the vessel tracks the desired velocities, with almost no deviations. CyberShip II is able to track the reference filter output as a consequence of the filter being tuned conservatively to the vessel's motion capabilities.

The output from the thrust allocation as seen in Figure 9.6, shows that the control input $\boldsymbol{u}$ to the thrusters, is never utilized above 50%. This means that the vessel could have handled

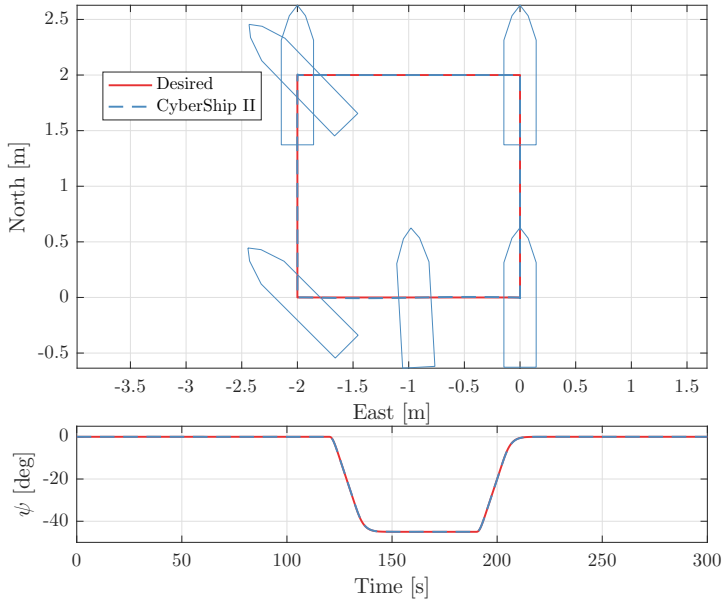


Figure 9.3: Simulation of the 4-corner box test with CyberShip II.

higher accelerations and speeds. A interesting observation is that the thrust allocation prefers to rotate the thrusters instead of reversing the thrust. This can be due to many propellers being designed to be most efficient in one direction, which the thrust allocation tries to adhere. The bow thruster being fixed means that the thrust allocation is forced to reverse the thruster to obtain a force in the opposite direction.

Figure 9.7 shows the total output of the controller, as well as the feedfoward and feedback elements. The control effort from the feedforward part is larger than the feedback part, indicating that the model in the feedforward is a accurate model, which is expected as it is the same as the model simulated. The feedback only has to counter minor position and heading error due to the environmental disturbances applied to the system. The high frequent noise in τ_{PID} is due to simulation jitter (Kyckelhahn and Forbus, 2004), which are tiny oscillations around setpoint. This however has no effect on the results of the simulation.

9.5 Concluding Remarks

From the results obtained in the simulation, it is clear that each part of the control system stated in the beginning of this chapter, are functioning properly. In the simulations, CyberShip II is able to accurately perform the 4-corner box test with a constant disturbance. The critical parts in field tests is be the accuracy of the mathematical model obtained in order to have a good feedforward, tuning and the effect of disturbances on the vessel.

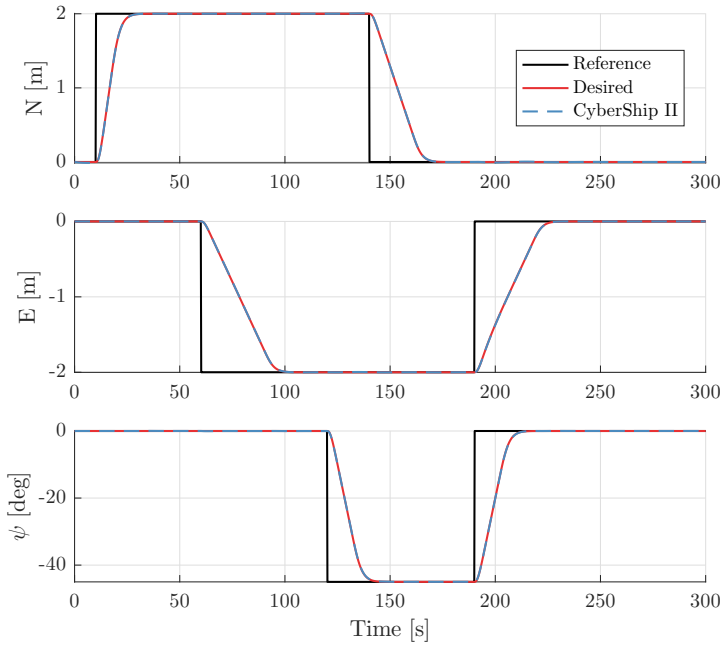


Figure 9.4: Reference, desired and actual pose of CyberShip II.

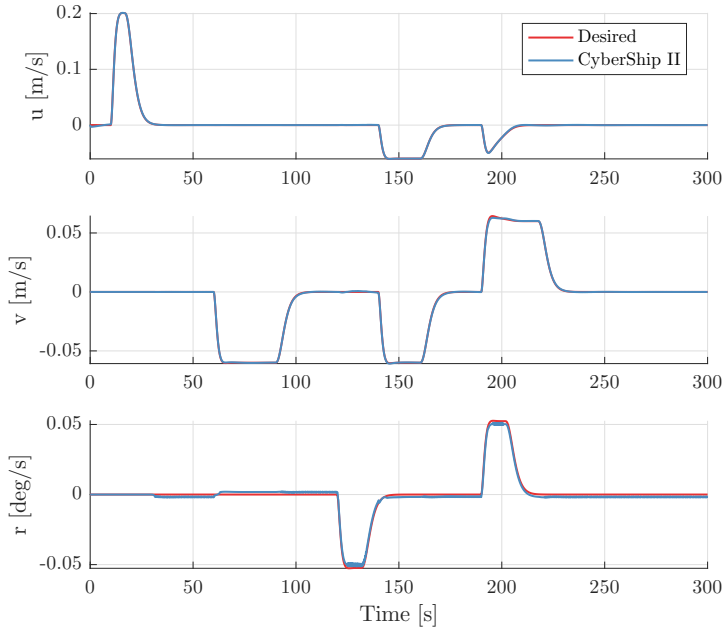


Figure 9.5: Desired and actual velocity of CyberShip II.

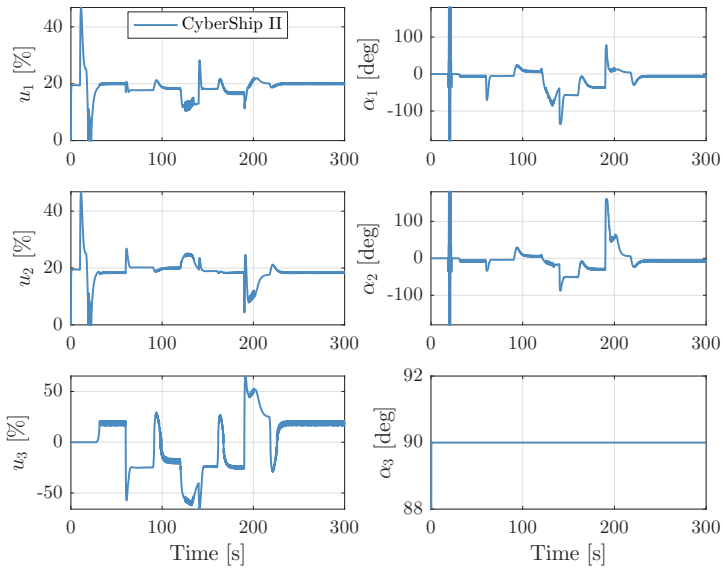


Figure 9.6: Desired thrust and thruster angle from the thrust allocation. The spikes in $\alpha_{1,2}$ at t=40s is due to wrapping around 180°.

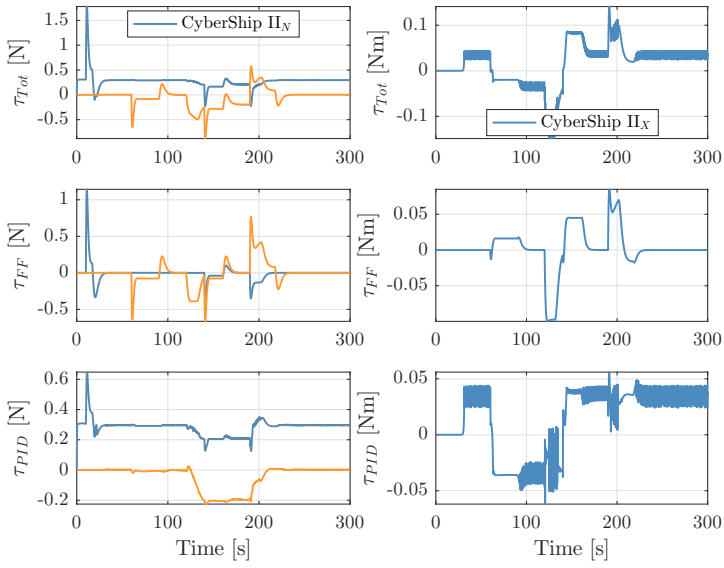


Figure 9.7: Controller output, divided into PID, feedforward and total control effort.

Chapter 10

Experimental Results

In this chapter, the results from the offline and online ESKF tests are presented as well as the results from the final DP tests.

10.1 Observer Performance

The ESKF is developed and tested in MATLAB before it is implemented on ReVolt. This is referred to as offline and online performance.

10.1.1 Offline Performance

The offline tests, with experimental data, are performed to see how the ESKF performs in a known software environment, MATLAB. This makes it easier to debug and analyze the ESKF. In conjunction with another experimental test, performed by PhD students at NTNU, the Xsens was brought to log experimental data for the ESKF. The test was conducted on the 6th April 2017, with Maritime Robotics' vessel Telemetron. Onboard there is an identical Vector VS330 to the one used in ReVolt (Section 6.3.2) without RTK corrections and Kongsberg's Seapath. The Seapath is a series of high grade heading, attitude and positioning sensors which provide high accuracy measurements (Kongsberg, 2016). The Seapath will serve as a reference for the offline results.

The IMUs of Xsens and Seapath are mounted next to each other, seen in Figure 10.1, and further assumed to be in the same frame. Due to a setting in the parser for the Vector, GNSS data was not published if the quality of the data dropped below a certain threshold. These dropouts occur often and last up to 12 seconds, which unintentionally leads to dead reckoning gaps.

Figure 10.1: Mounting of Seapath and Xsens IMUs on Telemetron.

The antennas are not mounted in the same place, as is evident from the stationary measurements shown in Figure 10.2. Furthermore, the position measurements from the Xsens are inaccurate and have a low precision, seen by the measurements slowly drifting. The Seapath and Vector have about the same precision with only a lever-arm between them.

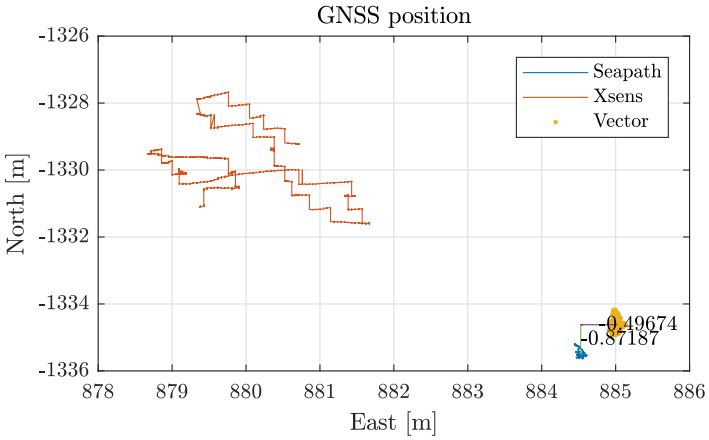


Figure 10.2: Stationary vessel measurements showing drift in Xsens position and mounting difference between Vector and Seapath antennas.

Estimate Quantification

To quantify the estimates, the error $\bar{e}$ is defined as

$$\bar{e} = \sqrt{\sum_{i=1}^{3} e_i^2}, \quad e_i = x_i - y_i \tag{10.1}$$

where x_i are elements of for example position estimates from the ESKF or Xsens, and y_i is the Seapath position vector.

Noise and Bias

The following parameters for the process noise $\boldsymbol{Q}_k$ and measurement noise $\boldsymbol{V}_k$ are used

$$\boldsymbol{Q}_k = \begin{bmatrix} 0.3^2\boldsymbol{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (0.4\frac{\pi}{180})^2\boldsymbol{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{15\frac{\pi}{180}}{3600}\boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{15\frac{\pi}{180}}{3600}\boldsymbol{I} \end{bmatrix} \tag{10.2}$$

$$\boldsymbol{V}_k = \begin{bmatrix} 1^2\boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & 0.4^2 \end{bmatrix} \tag{10.3}$$

Initialization

The ESKF is initialized using the navigation state of the Seapath and biases are set to zero. The ESKF use acceleration and angular rate measurements from the Xsens, and combines these with the position and heading angle measurements from the Vector.

Lever-Arm Compensation

The mounting of the Seapath IMU and Xsens IMU are assumed identical, see Figure 10.1, whereas the antenna for the Vector is lever-arm compensated using the method in Section 5.3 with

$$\Delta\boldsymbol{p}^b = \begin{bmatrix} 0.315 \\ 0.9525 \\ 2.07 \end{bmatrix} \tag{10.4}$$

Offline Results

In Figure 10.3, the ESKF seems stable and capable of utilizing the IMU to calculate new position estimates when dead reckoning, thus increasing the standard deviation σ. The error in estimated position, relative to the Seapath, are within reasonable range as they both have SBAS corrections which leaves them with an accuracy around $\pm 0.5m$. The slow dynamics in error are related to the difference in the Down direction, where the Vector measurements do not coincide with the Seapath. The fast dynamics in the error are due to dead reckoning, where the estimate drifts in wait for a measurement which, when received, is corrected.

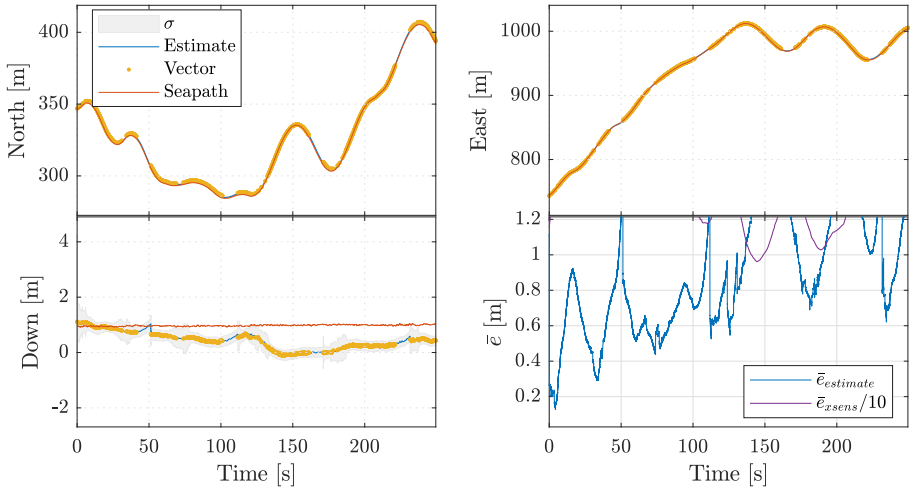


Figure 10.3: Estimated positions in NED compared to the Vector and Seapath. The error between the Seapath measurements and Xsens measurements are scaled down with a factor of 10 for comparability between the Xsens and Vector measurements.

In the orientation estimates seen in Figure 10.4, the standard deviation σ, represented as a grey shadow, is seen clearly for roll and pitch. These are prominent right after dead reckoning, which is expected as there has not been any position measurements for a small time period. This is because change in position is used to confirm whether or not a measured acceleration, in a given direction, is caused by movement or gravity. Overall, the dead reckoning of angles, especially yaw, are estimated with good precision.

For the velocity estimates seen in Figure 10.5, the errors are relatively small during continuous GPS measurements for example around t=150 seconds. During dead reckoning the error increases rapidly.

The bias estimates seen in Figure 10.6 looks to converge rapidly for acceleration measurements. The initial guess of the gravity, set in the ESKF, is corrected by the bias estimation. This is clearly visible in the z-axis of the acceleration, as this bias converges to -0.25. The standard deviation for the gyro rates in pitch and roll are around 10 times higher than for yaw. This is to be expected as yaw is updated by measurements from the Vector, while roll and pitch are estimated only.

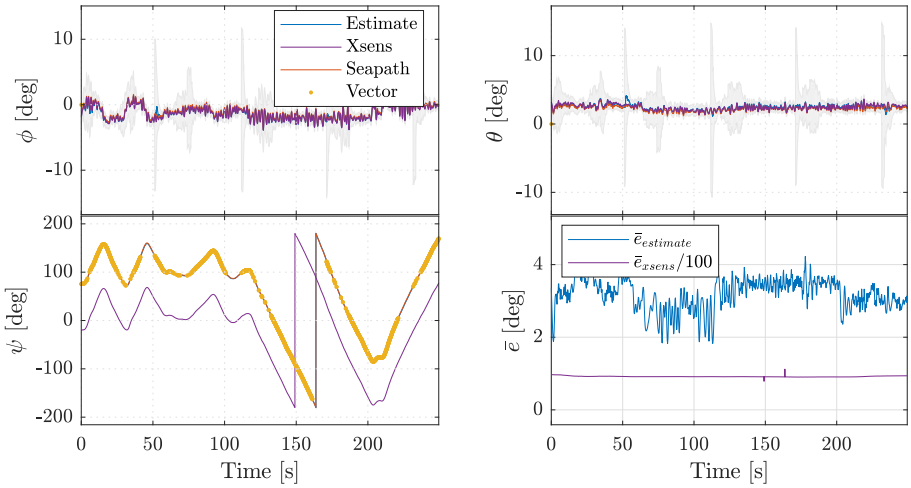


Figure 10.4: Estimated orientation compared to the Xsens, Vector's GNSS-compass and Seapath. Notice that the Xsens with AHRS enabled, fails to find true north and is therefore prone to a slowly varying bias.

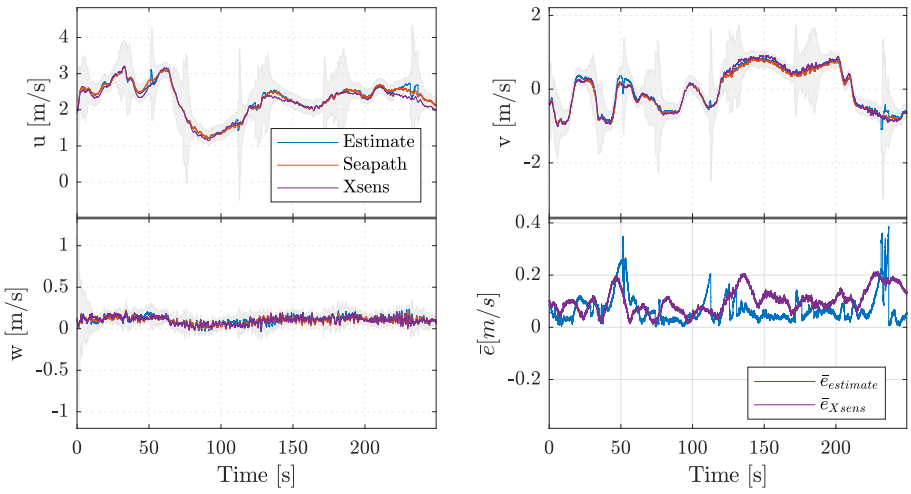


Figure 10.5: Body velocity estimate comparisons. Xsens velocity is rotated from NED using Seapath orientation.

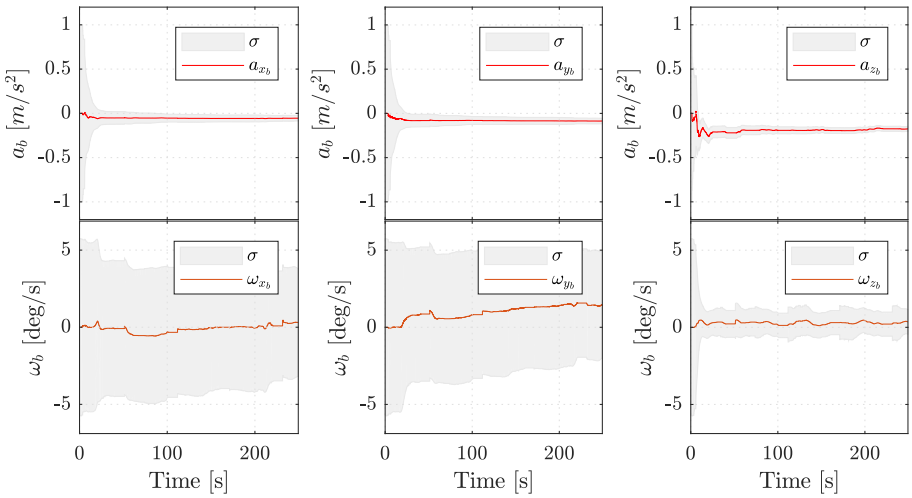


Figure 10.6: The estimated IMU biases, where a is the acceleration and ω is the angular rate.

Offline Results - Excerpt

With a focus on what happens between t=35-55 seconds, it is seen from Figure 10.7 that the boat turns slowly to starboard followed by turning to port. The plot also show that there is a loss of GPS data between roughly t=42-52 seconds. This immediately causes the velocity to drift (Figure 10.8) which in turn, causes the position estimate to drift (Figure 10.9). When the first GPS measurement is received after dead reckoning, the correction in position causes the velocity estimates to compensate by inducing a relatively high speed in surge and sway. This also disrupts the orientation estimates in roll and pitch due to the use of orientation in the determination of the acceleration direction.

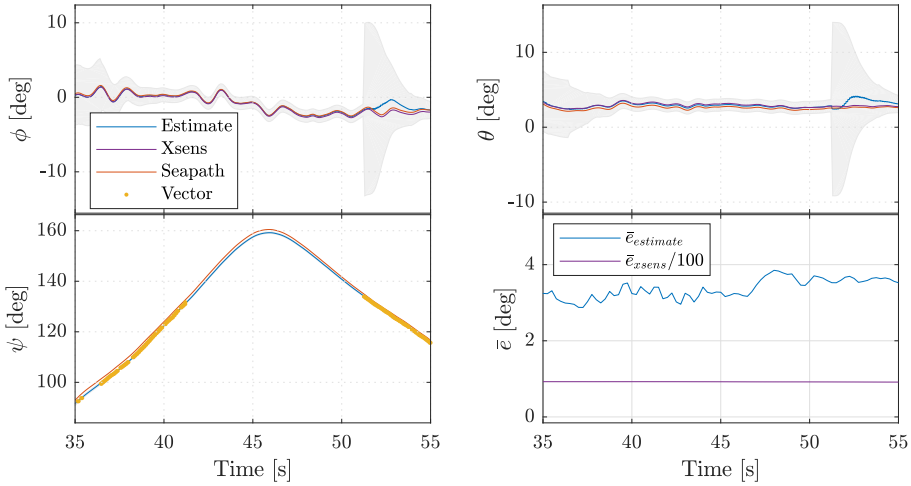


Figure 10.7: An extract of the orientation estimates comparison data, from t=35-55s. Xsens' estimate of ψ is excluded in bottom left plot as it's offset is too big.

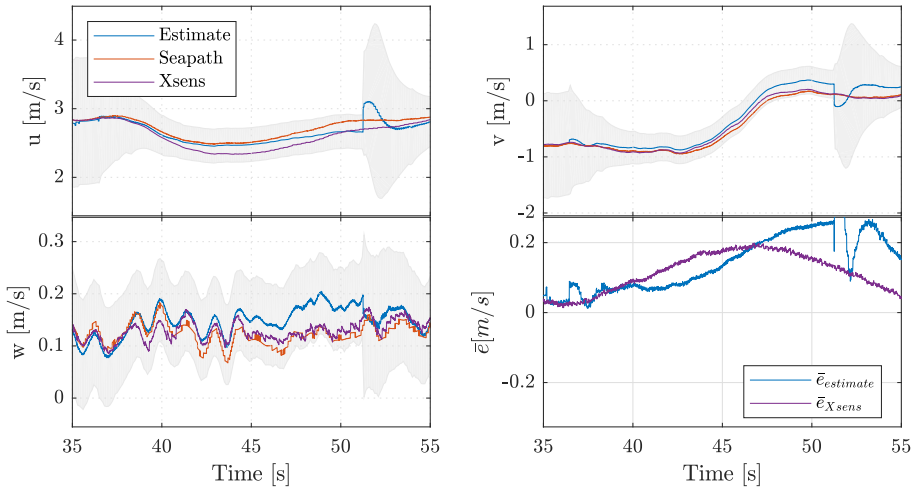


Figure 10.8: An extract of the body velocity estimates comparison data, from t=35-55s. Xsens' velocity is rotated from NED using Seapath angles, making the measurements more precise, rather than using the Xsens angle where yaw is not North referenced.

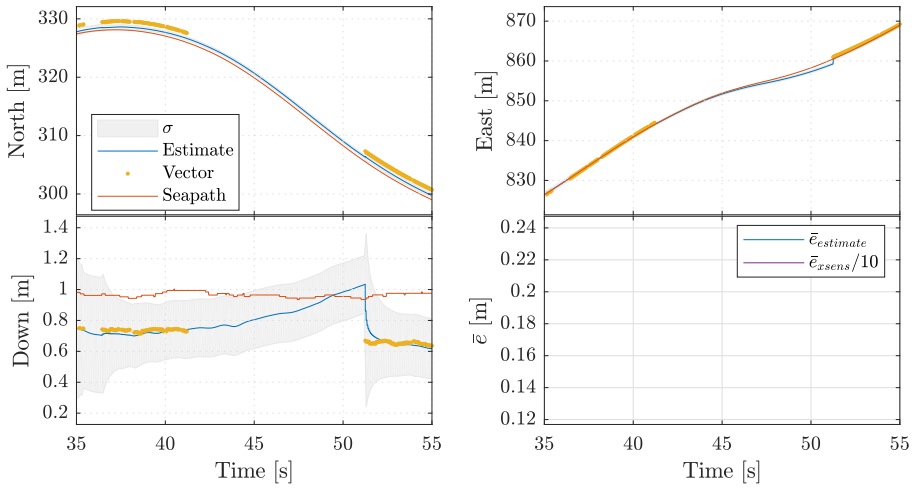


Figure 10.9: An extract of the estimated positions in NED compared to the Vector and Seapath, from t=35-55s.

10.1.2 Online Performance

The implementation of the ESKF was prone to stability issues at the final test, which was solved later. ”Online” thus means that the data recorded from the test, is played back in real time. Compared to offline in the previous section where the data was traversed until completion without any time limits, this tests implemented code on ReVolt, which will have a limited time for each iteration.

Online: Noise and Bias Parameters

The following parameters for the process noise $\boldsymbol{Q}_k$ and measurement noise $\boldsymbol{V}_k$ are used

$$\boldsymbol{Q}_k = \begin{bmatrix} 0.3^2\boldsymbol{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (0.4\frac{\pi}{180})^2\boldsymbol{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{15\frac{\pi}{180}}{3600}\boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{15\frac{\pi}{180}}{3600}\boldsymbol{I} \end{bmatrix} \tag{10.5}$$

$$\boldsymbol{V}_k = \begin{bmatrix} 0.05^2\boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & 0.4^2 \end{bmatrix} \tag{10.6}$$

Online Results

The ESKFs position estimates seen in Figure 10.10, follows the measurements from the Vector smoothly. Also, with the GNSS measurements set to a precision of $\pm$5cm, the standard deviation settles below 3.5cm. The Down measurements and estimates are prone to a high frequent disturbance which is likely due to errors in the GNSS measurements. This is also reflected in the bias for the accelerometer a_z trying to compensate for this. The heading estimate ψ seen in Figure 10.11, is smooth and follows the measurements from the Vector. The roll ϕ and pitch θ, which has no absolute measurements, are very plausible and are close to the Xsens' estimates. The deviations is likely due to the more advanced gravity model used in the Xsens than the one used in the ESKF. The velocities seen in Figure 10.12 coincides with the position plots. For the biases seen in Figure 10.13, the biases in angular rate converge rapidly to a stationary value. However, the acceleration biases a_x and a_y does not, which is most likely because the horizontal accelerometer biases are not observable without large rotations, in roll and pitch, to initialize the biases (Du et al., 2017).

This test reveals that the matrix exponential discretization mentioned in 5.1.7 is fast enough to be used in real time. This means that the ESKF can be implemented and used as an observer on ReVolt.

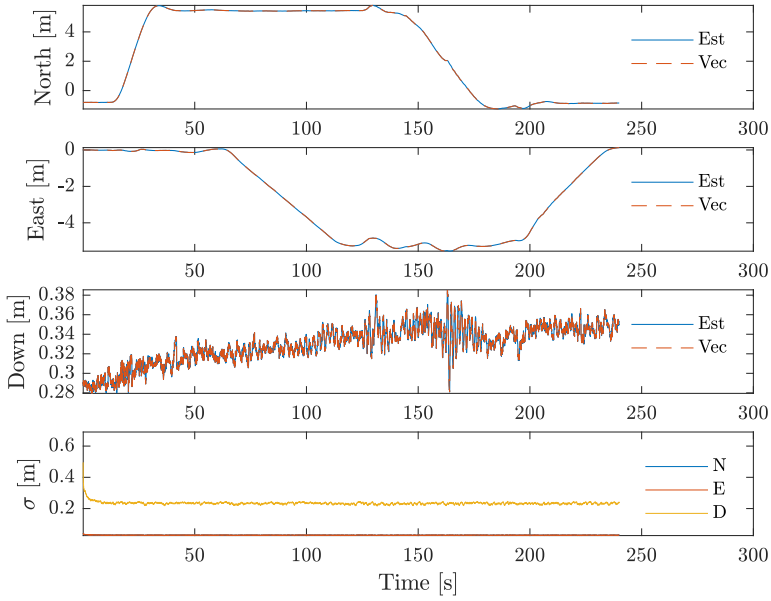


Figure 10.10: Online ESKF performance during a 4-corner box maneuver. Est is estimated, Vec is measurements from the Vector.

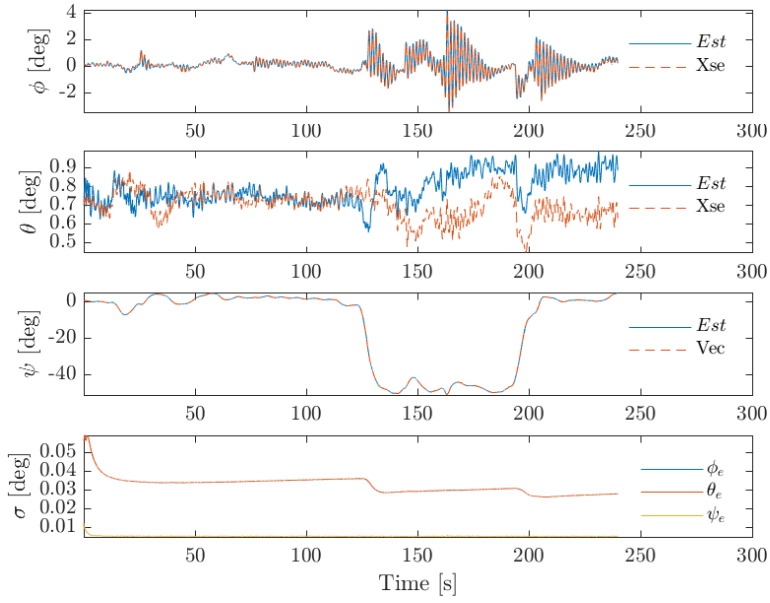


Figure 10.11: Online ESKF performance during a 4-corner box maneuver. Est is estimated, Xse and Vec is measurements from the Xsens and Vector respectively.

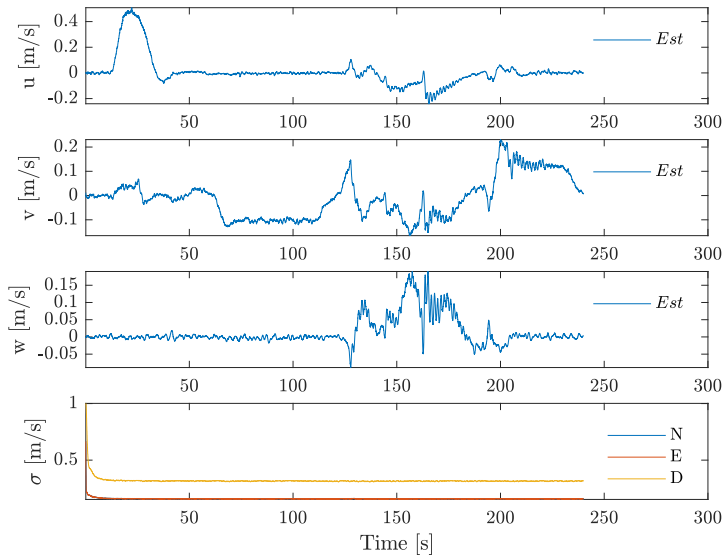


Figure 10.12: Online ESKF performance during a 4-corner box maneuver. Est is estimated.

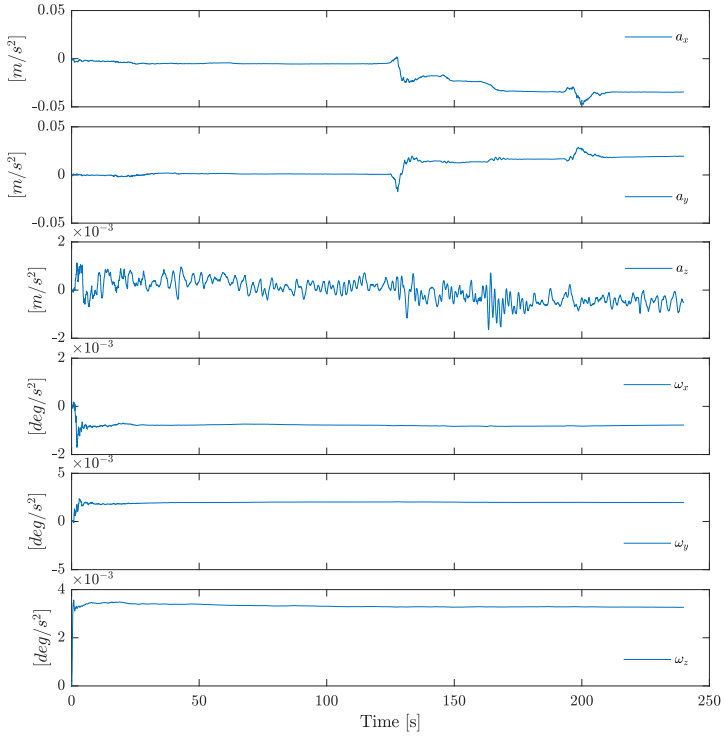


Figure 10.13: Online ESKF performance during a 4-corner box maneuver. Est is estimated.

10.2 Pose and Velocity Measurements

Since the Kalman Filter was not stable when implemented on ReVolt, the pose and velocity measurements used in the controller were taken from the Vector VS330.

The Xsens' velocity estimates were incorrect and therefore the velocities are obtained from low-pass differencing the pose data from the Vector. Low-pass differencing means that the difference between two position measurements is taken and passed through a low-pass filter to get smooth velocity measurements. Figure 10.14 shows the velocity data from the low-pass differencing and Xsens estimates as ReVolt is performing a 4-corner box maneuver.

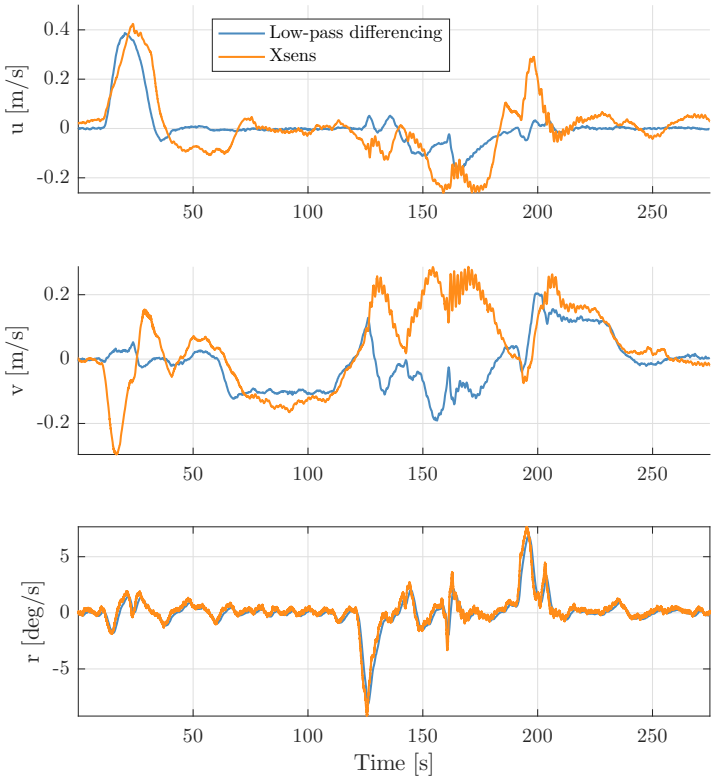


Figure 10.14: Comparison of vessel velocity estimated by the Xsens or computed by derivation of the position. The velocities are rotated to BODY using the yaw measurements from the Hemisphere Vector VS330.

With the Hemisphere's dual antennas, RTK corrections and a low-pass filter, the velocity measurements are smooth and behave as expected when performing a 4-corner box test. The Xsens provides a noisy surge speed estimate, with an unexpected positive surge speed at t=190s. The Xsens also provides positive measurements of sway speed around t=140s

which does not occur with the other measurement, nor was this observed during the test. The yaw rate from the Hemisphere and Xsens are very similar, but the low-passed Vector measurements provide a more smooth signal which is easier for the controller to use. The yaw rates are similar which is to be expected as this is a direct measurement from the gyro and this implies that the differencing is adequate.

The Xsens provides good roll and pitch measurements, seen in Figure 10.4. The deviations are small, with the pitch measurements deviating the most. This is however negligible and can be caused by how the Xsens is installed. The yaw measurement in addition to not being north referenced, drifts as time progresses. To estimate the heading, such a system needs a north seeking IMU able to differentiate the measurement of the Earth's angular velocity from other effects (Vik, 2014). The yaw measurements from the Hemisphere are obtained from differentiating the two antenna's relative position compared to north, thus providing a good heading measurement. The position measurements of the Xsens are, as seen in Figure 10.2, not as precise as the Vector VS330.

10.3 Test Area

Trondheimsfjorden was declared as a test site for autonomous ships on the 30 September 2016 (Meland, 2016). The sea state in the fjord is however too rough for ReVolt, therefore test sites at Dorabassenget and Havnebasseng III was devised in agreement with Trondheim Havn, where the final tests were performed in the latter, as marked in Figure 10.15. The test site was subject to slight wind from north-east and presumably no current. The experiments were performed in accordance with the guidelines for ReVolt which include safe operations, pre- and post operation procedures, as well as checklists for launch and recovery. The guidelines can be found in Appendix F.

10.4 Parameters

Before beginning with the 4-corner box tests, the reference filter, thrust allocation and PID controller were tuned. These values were kept the same for all the tests in this chapter.

10.4.1 Reference Filter Parameters

The reference filter parameters were, as in the simulations, set conservatively for ReVolt to easily follow the desired trajectory from the reference filter. The reference filter could have been more aggressive, making ReVolt move faster, but this was not the purpose. The following saturation limits were used:

$$x_{max} = 1.5 \quad -x_{max} = 1.0 \quad y_{max} = 0.6 \quad -y_{max} = 0.5 \quad \pm\psi_{max} = 35^\circ \quad (10.7)$$

Figure 10.15: The test area at *Havnebasseng III*. Adapted from Gulesider® Kart.

This resulted in the following tuning parameters:

$$\mathbf{\Delta} = \begin{bmatrix} \zeta_N \\ \zeta_E \\ \zeta_{psi} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad , \quad \mathbf{\Omega} = \begin{bmatrix} \omega_{n,N} \\ \omega_{n,E} \\ \omega_{n,\psi} \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.6 \\ 0.6 \end{bmatrix} \tag{10.8}$$

10.4.2 Tuning Thrust Allocation

When requesting forces from the thrust allocation, unwanted moments will be induced due to the ship's hull. This is caused by fact that the water's point of attack on the hull, not being the same as CG. By moving CG forwards or backwards, the moments of the thrust allocation can cancel out the induced moments. In practice, this is done by moving each thruster's position in the thrust allocation in the longitudinal direction l_{x_i}.

The thrust allocation was tuned as follows:

Request $\pm$ force in sway motion
if $\pm$ yaw motion
    move CG in $\mp$ direction
if $\mp$ yaw motion
    CG in $\pm$ direction

Which gave the results:

$$
\begin{aligned}
(l_{x_1}, l_{y_1}) &= (-1.12 - 0.53, -0.15) &&= (-1.65, -0.15) \\
(l_{x_2}, l_{y_2}) &= (-1.12 - 0.53, 0.15) &&= (-1.65, 0.15) \\
(l_{x_3}, l_{y_3}) &= (1.12 + 0.03, 0) &&= (1.15, 0)
\end{aligned}
$$

Notice that these are not the same as the measured distances, found in Figure 8.2.

10.4.3 Controller Gains

The saturation limits in the controller are set according to the maximum thrust the ship's thrusters can supply, found in Figure 8.5. Since the maximum thrust is different for ±sway and ±yaw rotation, a compromise was made and a suitable value has been chosen. The saturation limits are:

$$
\boldsymbol{\tau}_{max} = \begin{bmatrix} \tau_{X,max} \\ \tau_{Y,max} \\ \tau_{N,max} \end{bmatrix} = \begin{bmatrix} 50 \\ 20 \\ 32 \end{bmatrix} \tag{10.9}
$$

The controller was manually tuned directly on the physical system during the experiments, using the method described in Section 4.1.3.

The following controller gains were obtained:

$$
\boldsymbol{K_p} = \begin{bmatrix} K_{p,surge} & 0 & 0 \\ 0 & K_{p,sway} & 0 \\ 0 & 0 & K_{p,yaw} \end{bmatrix} = \begin{bmatrix} 25 & 0 & 0 \\ 0 & 25 & 0 \\ 0 & 0 & 30 \end{bmatrix} \tag{10.10}
$$

$$
\boldsymbol{K_i} = \begin{bmatrix} K_{i,surge} & 0 & 0 \\ 0 & K_{i,sway} & 0 \\ 0 & 0 & K_{i,yaw} \end{bmatrix} = \begin{bmatrix} 0.3 & 0 & 0 \\ 0 & 0.3 & 0 \\ 0 & 0 & 0.3 \end{bmatrix} \tag{10.11}
$$

$$
\boldsymbol{K_d} = \begin{bmatrix} K_{d,surge} & 0 & 0 \\ 0 & K_{d,sway} & 0 \\ 0 & 0 & K_{d,yaw} \end{bmatrix} = \begin{bmatrix} 75 & 0 & 0 \\ 0 & 75 & 0 \\ 0 & 0 & 50 \end{bmatrix} \tag{10.12}
$$

10.4.4 Lever-Arm

For ReVolt, the lever-arm is:

$$
\Delta \boldsymbol{p}^b = \begin{bmatrix} -0.81 \\ 0 \\ -0.2 \end{bmatrix} \tag{10.13}
$$

10.5 4-Corner Test

To fully test the DP control system and capabilities of ReVolt, a series of tests were performed on the 3rd, 4th and 5th of May 2017. During the tests, on all days, there was a slight north-east breeze and no significant current. The 4-corner box test as described in Section 9.1 was used for all tests, as this makes comparing the performance of the different scenarios easier. The plan for the test procedure is shown in Table 10.1.

<table>
<tr><td>Scenario</td><td>Thruster rotation rates</td><td>Bow thruster rotation</td><td>Controller</td></tr>
<tr><td>1</td><td>Unconstrained</td><td rowspan="2">Static</td><td rowspan="3">No FF</td></tr>
<tr><td>2</td><td>Constrained</td></tr>
<tr><td>3</td><td rowspan="4">Analyze scenario 1 and 2, proceed with the best scenario.</td><td>Rotating</td></tr>
<tr><td>4</td><td>Static</td><td rowspan="2">No FF</td></tr>
<tr><td>5</td><td rowspan="2">Analyze scenario 3 and 4, proceed with the best scenario.</td></tr>
<tr><td>6</td><td>With FF</td></tr>
</table>

Table 10.1: The test scenarios and in which order they were tested.

The different scenarios were tested at least two times, in order to determine the repeatability of the scenario.

During testing it was found that the model of ReVolt might not be accurate, due to ReVolt's behavior with the feedforward activated. An on-site adjustment to the feedforward element of the controller was made and the tuned feedforward is also compared to the original feedforward. This is described in more detail in Section 10.5.4.

10.5.1 Performance Metrics

To compare controller performance, the errors in position and heading are normalized between 0 and 1, where it is assumed that the greatest errors are 5m and 50° in north/east position and heading, respectively.

$$\bar{\boldsymbol{p}}(t) = \frac{\boldsymbol{p}(t)}{5}, \qquad \bar{\boldsymbol{p}}_d(t) = \frac{\boldsymbol{p}_d(t)}{5}, \qquad \bar{\psi} = \frac{\psi}{50}, \qquad \bar{\psi}_d = \frac{\psi_d}{50} \tag{10.14}$$

These normalized values are then integrated over time. The equation for calculating the Integrated Absolute Error (IAE) is given as

$$IAE(t) = \int_0^t |\bar{e}(\sigma)| d\sigma \tag{10.15}$$

where $\bar{e}(t)$ is the error at time t and is defined as

$$\bar{e}(t) = \sqrt{(\bar{\boldsymbol{p}}(t) - \bar{\boldsymbol{p}}_d(t))^2 + (\bar{\psi}(t) - \bar{\psi}_d(t))^2)} \tag{10.16}$$

The Integral of Absolute Differentiated Control (IADC) adapted from Eriksen and Breivik (2017), and is used to assess each scenarios wear and tear on the actuators. The change in actuator input is integrated to to get the accumulated difference in actuator input over time. The modified IADC is then given as

$$IADC(t) = \int_0^t |\bar{\boldsymbol{u}}|(\sigma) + |\bar{\boldsymbol{\alpha}}|(\sigma) d\sigma \tag{10.17}$$

where $\bar{\boldsymbol{u}}$ and $\bar{\boldsymbol{\alpha}}$ is the normalized difference in actuator input at a given time and is defined as

$$\bar{\boldsymbol{u}}(t) = \frac{\bar{\boldsymbol{u}}(t) - \bar{\boldsymbol{u}}(t-h)}{h}, \qquad \bar{\boldsymbol{u}} = \frac{\boldsymbol{u}}{100} \tag{10.18}$$

$$\bar{\boldsymbol{\alpha}}(t) = \frac{\boldsymbol{\alpha}(t) - \boldsymbol{\alpha}(t-h)}{h}, \qquad \bar{\boldsymbol{\alpha}} = \frac{\boldsymbol{\alpha}}{90^\circ} \tag{10.19}$$

10.5.2 Unconstrained vs. Constrained

The purpose of this test is to compare the performance of Revolt with and without constraints on the thruster rotation rates in the thrust allocation. More specifically, the only change is whether the stern thrusters rotation rate is constrained or not. The maximum thruster rotation rate for the two stern thrusters was set to $\Delta\alpha_{1,2} = 30^\circ$ according to Section 7.2.4. The test was performed with a static bow thruster, set at 90°, and without feedforward.

Figure 10.16 and Figure 10.17, show the 4-corner box tests performed with unconstrained thruster rotation and constrain thruster rotation, respectively. It can be seen that in both cases the vessel trajectories are similar, indicating that the tests are repeatable. Figure 10.18 shows the performance metrics of all the trajectories in this test. From this it is seen that the total error is similar in both setups, but the change in control output is larger with the unconstrained setup as expected since the thrust allocation is unconstrained. To compare these two setups, the best trajectory of both tests are chosen.

Figures 10.19 and 10.20 show the desired- and vessel-trajectory from the two setups. In the first, surge motion the unconstrained thrust allocation overshoots more than the constrained. This is due to the thrust allocation, with unconstrained rotation rates, rotating the thrusters faster than physically achievable, see Figure 10.22, and applying thrust. This causes thrust in unwanted directions, resulting in velocities in sway, as seen in Figure 10.21 at t=40s. This also causes the PID to compensate which is seen as oscillations in Figure 10.23. During the coupled surge-sway motion, from point 4 to 5 in the box maneuver, the unconstrained setup surprisingly performs better.

The unconstrained setup seems to tracks the desired trajectory better as shown by the IAE in Figure 10.18. However, the constrained suffers less from big errors and roll motions, due to rapid change of the thruster angles. The effect of these roll motions can be seen as oscillations in the control effort and sway speed of the unconstrained setup. Also it can clearly be seen that the IADC for the unconstrained setup is higher. This is not favourable as this causes extra wear and tear unnecessarily on the actuators. It is therefore decided to go forward with the constrained setup.

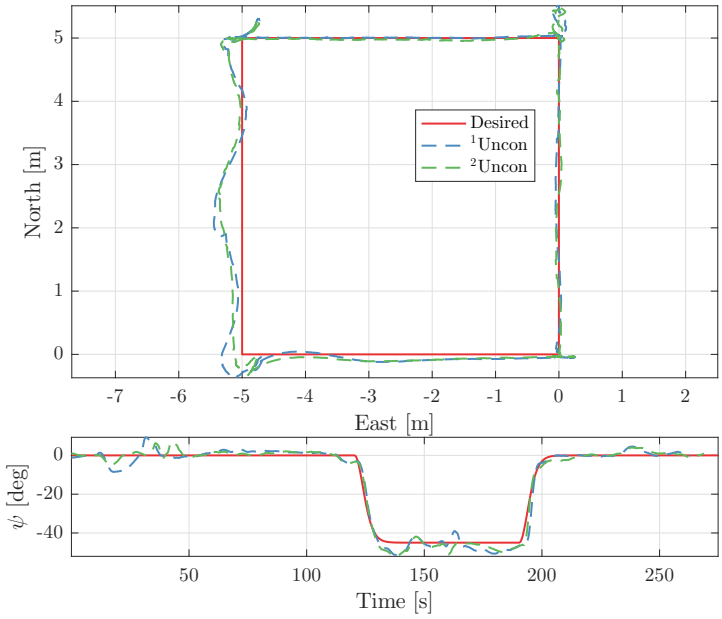


Figure 10.16: All vessel trajectories for static bow thruster with no constraints on $\boldsymbol{\Delta\alpha}$ in the thrust allocation.

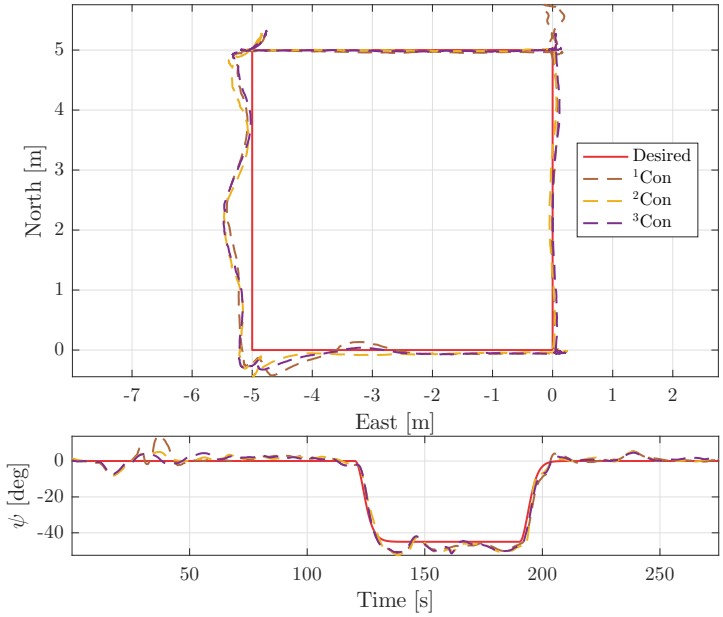


Figure 10.17: All vessel trajectories for static bow thruster with constraints on $\Delta\alpha$ in the thrust allocation.

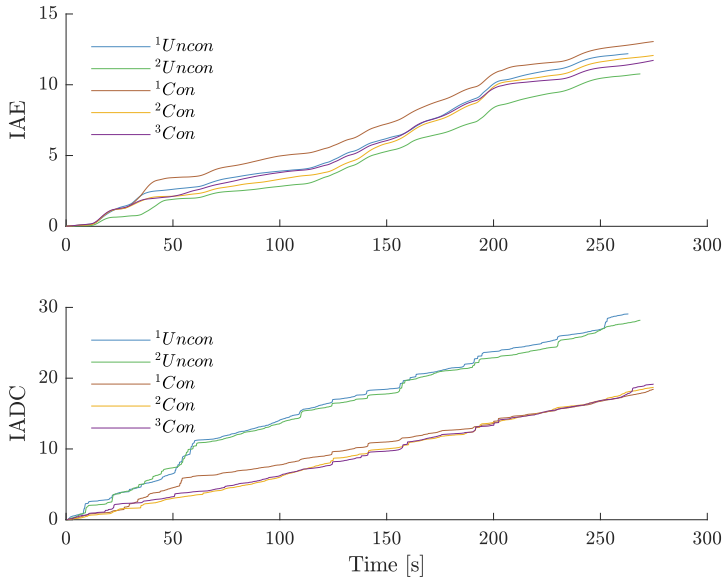


Figure 10.18: Performance metrics - static bow thruster with no constraints on $\Delta\alpha$ in the thrust allocation.

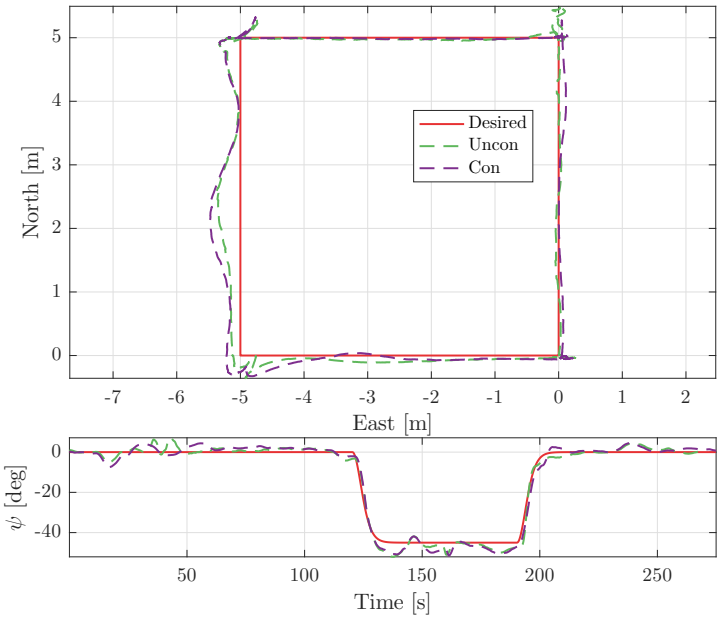


Figure 10.19: Comparison of the two best vessel trajectories with unconstrained and constrained $\boldsymbol{\Delta\alpha}$ in the thrust allocation.

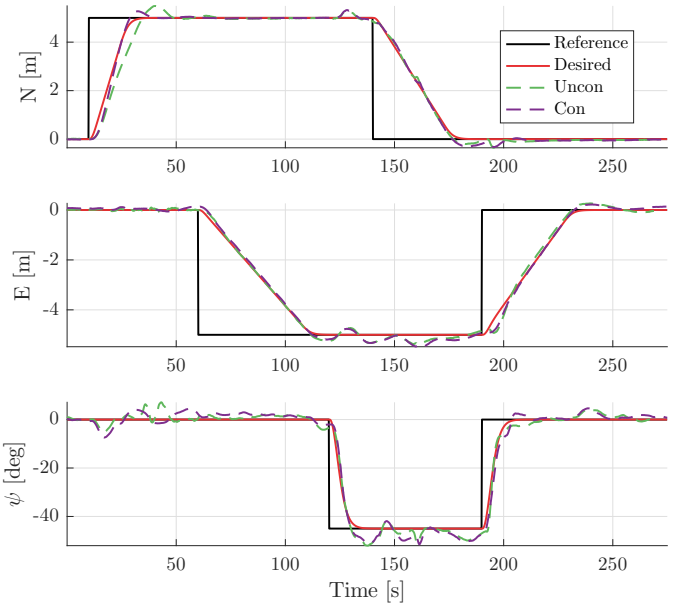


Figure 10.20: Comparison of the two best vessel trajectories over time using unconstrained and constrained $\boldsymbol{\Delta\alpha}$ in the thrust allocation.

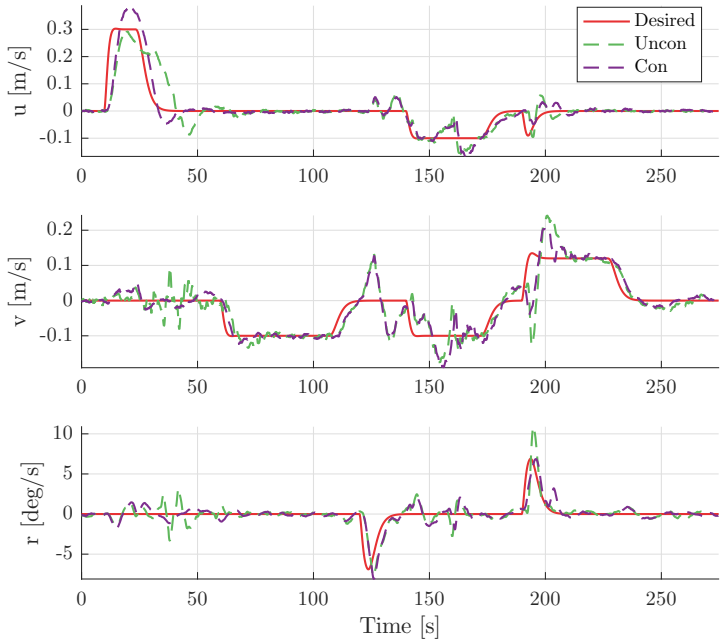


Figure 10.21: Comparison of the velocity for the two best vessel trajectories with unconstrained and constrained $\boldsymbol{\Delta\alpha}$ in the thrust allocation.

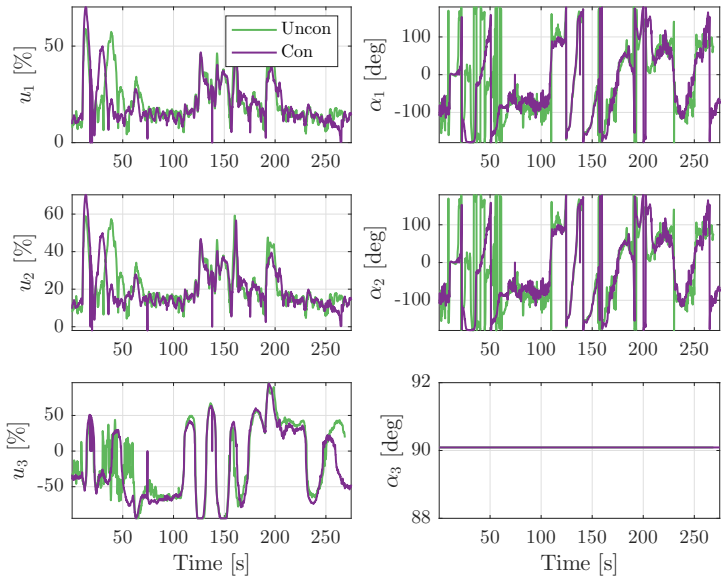


Figure 10.22: Comparison of the thrust allocation for the two best vessel trajectories with unconstrained and constrained $\boldsymbol{\Delta\alpha}$ in the thrust allocation.

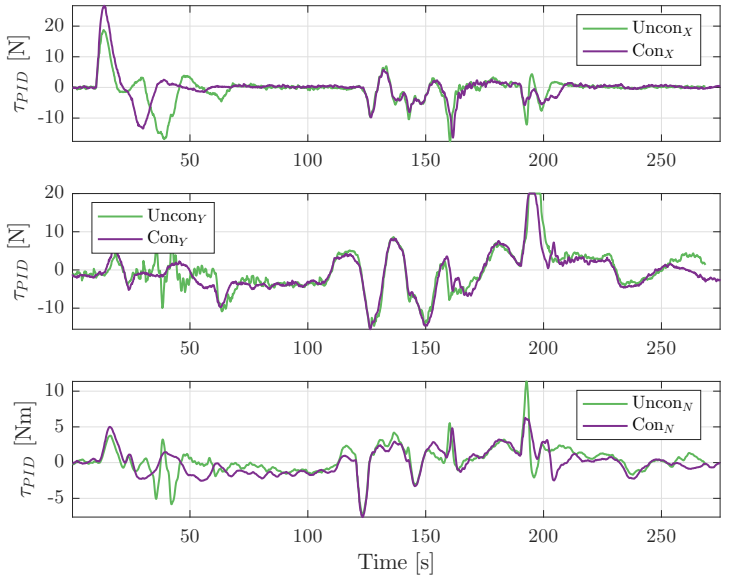


Figure 10.23: Comparison of the controller output for the two best vessel trajectories with unconstrained and constrained $\boldsymbol{\Delta\alpha}$ in the thrust allocation.

10.5.3 Rotation vs. Static

The purpose of this test is to compare whether or not allowing the bow thruster to rotate will improve the performance of Revolt. The maximum thruster rotation rate for the bow thruster was set to $\Delta\alpha_3 = \pi/36$. All the tests were performed with constrained thruster rotation rates and using only the PID controller. The only change is whether the bow thruster rotates or is static.

Figures 10.24 and Figure 10.25 show the 4-corner box tests performed with the bow thruster rotating and being static, respectively. From the rotating thruster plot, it is seen that there are some differences in the trajectories. This is presumably caused by environmental disturbances. In the case of the static bow thruster, the vessel trajectories are similar, again indicating that the tests are repeatable. From the performance metrics in Figure 10.26, the IADCs are similar, however the IAEs for rotating bow thruster suffers more. The two best trajectories from each setup, are further compared.

Figures 10.27 and 10.28 show the desired- and vessel-trajectory from the two setups. The setup with the rotating bow thruster has a larger overshoot with every setpoint change of the box maneuver, compared to the setup with the static bow thruster. The reason for this can be seen in figures 10.30 and 10.31 where the controller wants a desired force, and the thrust allocation prioritizes to turn the bow thruster instead of reversing thrust which is faster. This also induces roll motions similar to those with the unconstrained setup, mentioned in Section 10.5.2. This propagates into the controller, the demanded thrust and the velocities in Figure 10.29, weakening the performance. It is therefore decided to go forward with the static bow thruster setup.

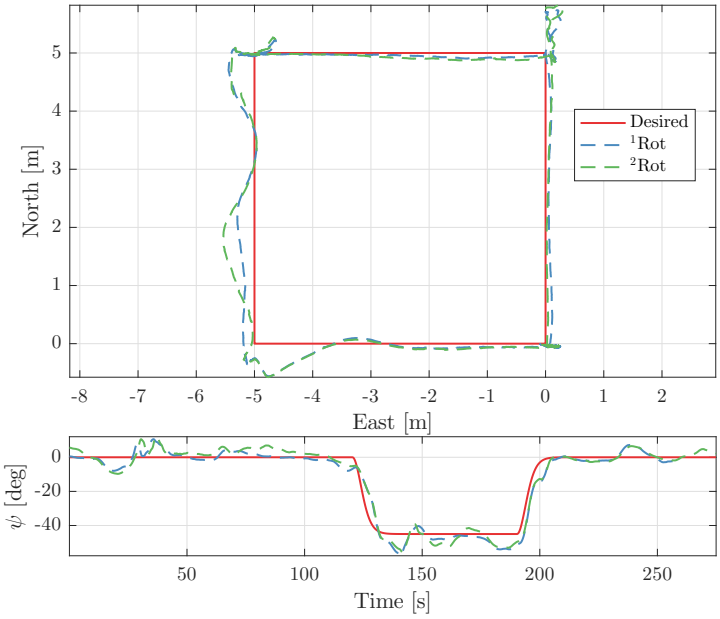


Figure 10.24: All vessel trajectories with rotating bow thruster $\dot{\alpha_3} \neq 0°$.

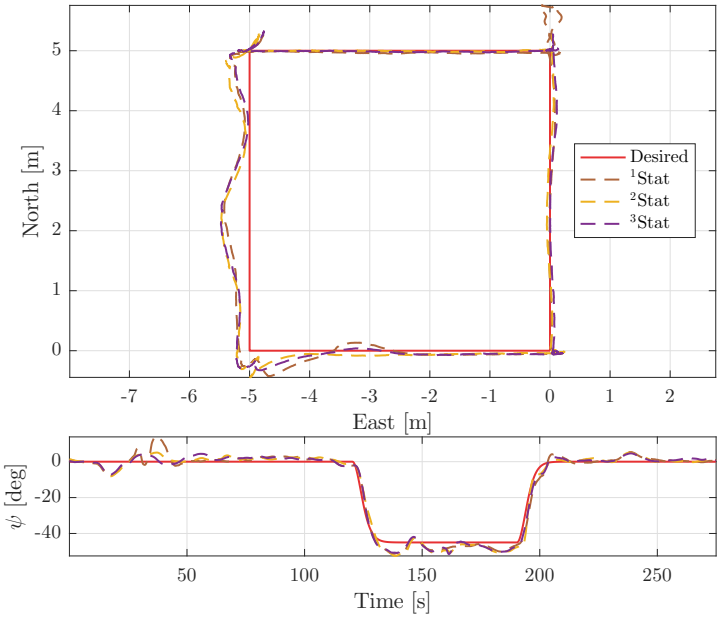


Figure 10.25: All vessel trajectories with static bow thruster $\alpha_3 = 90°$.

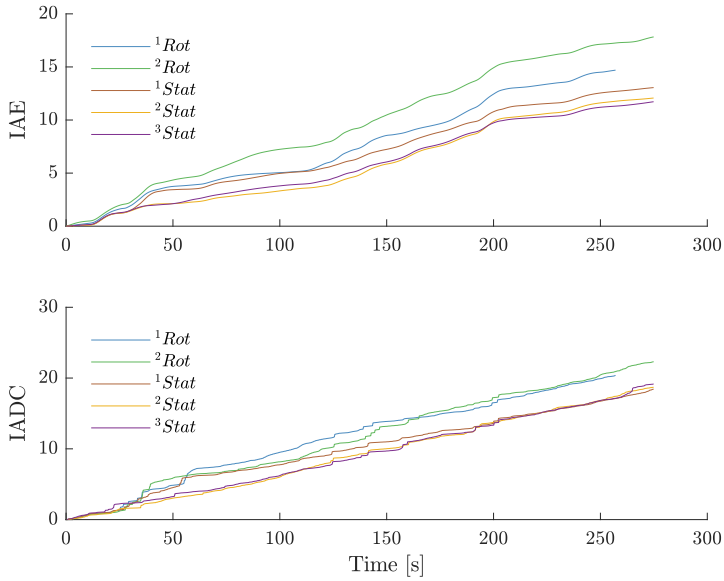


Figure 10.26: Performance metrics - Rotating bow thruster $\dot{\alpha_3} \neq 0$ vs static bow thruster $\alpha_3 = 90°$.

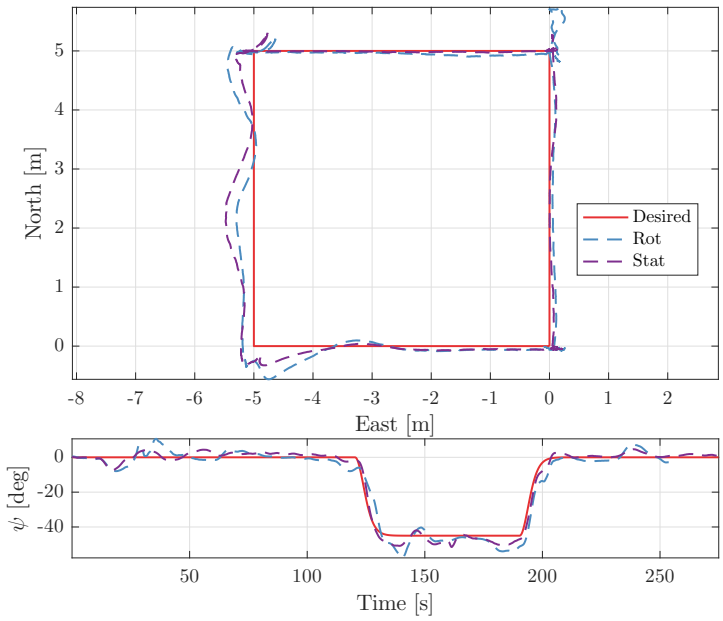


Figure 10.27: Comparison of the two best vessel trajectories with rotating bow thruster $\dot{\alpha_3} \neq 0$ vs static bow thruster $\alpha_3 = 90°$.

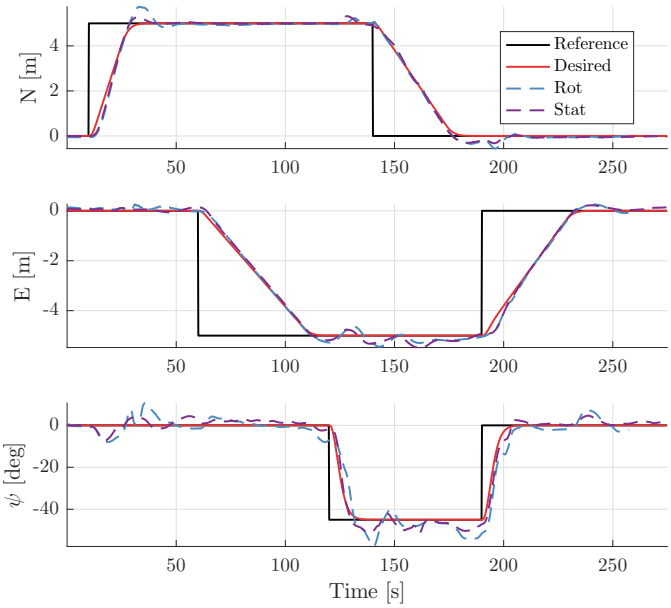


Figure 10.28: Comparison of the two best vessel trajectories over time using rotating bow thruster $\dot{\alpha_3} \neq 0$ vs static bow thruster $\alpha_3 = 90°$.

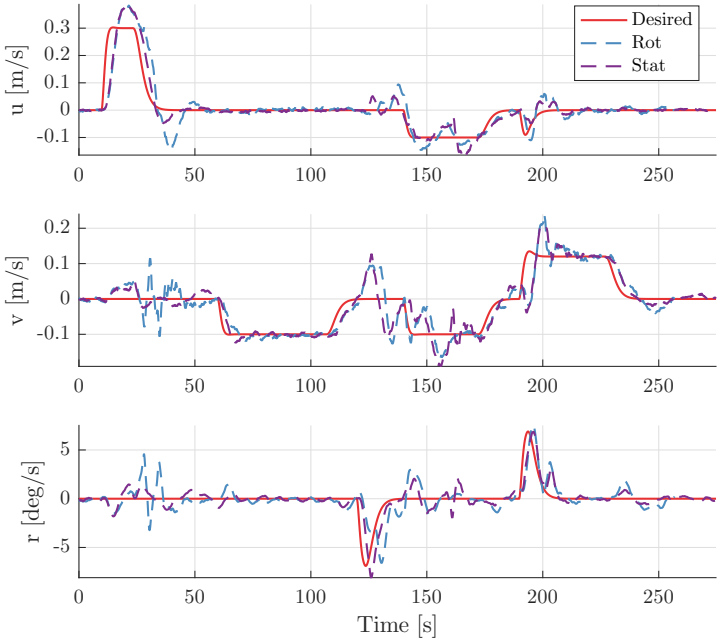


Figure 10.29: Comparison of the velocity for the two best vessel trajectories with rotating bow thruster $\dot{\alpha_3} \neq 0$ vs static bow thruster $\alpha_3 = 90°$.

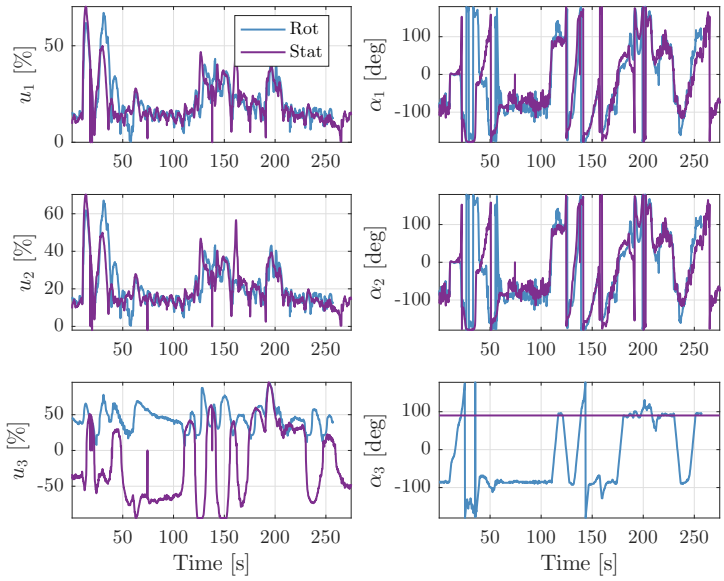


Figure 10.30: Comparison of the thrust allocation of the two best vessel trajectories with rotating bow thruster $\dot{\alpha_3} \neq 0$ vs static bow thruster $\alpha_3 = 90°$.

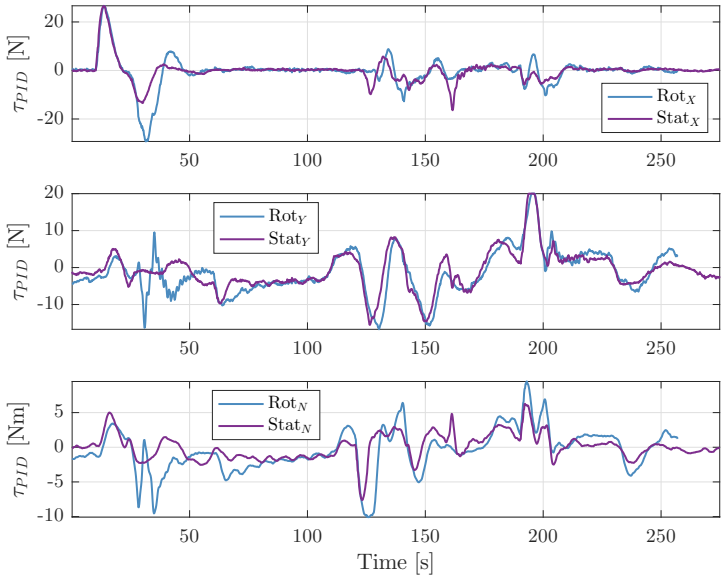


Figure 10.31: Comparison of the controller output for the two best vessel trajectories with rotating bow thruster $\dot{\alpha_3} \neq 0$ vs static bow thruster $\alpha_3 = 90°$.

10.5.4 PID-FF vs. Tuned PID-FF

From Figure 10.32, it is obvious that the mathematical model of ReVolt used in the feedforward part of the controller is inaccurate. It was therefore decided to limit the control effort created by the feedforward term, by multiplying a constant $K_{FF,i}$, with each of the elements in the feedforward control effort vector $\boldsymbol{\tau}_{FF}$. The tuned control effort is:

$$\boldsymbol{\tau}_{FF} = \begin{bmatrix} K_{FF,1}\tau_{FF,X} \\ K_{FF,2}\tau_{FF,Y} \\ K_{FF,3}\tau_{FF,N} \end{bmatrix} = \begin{bmatrix} 1.0 \cdot \tau_{FF,X} \\ 0.05 \cdot \tau_{FF,Y} \\ 0.2 \cdot \tau_{FF,N} \end{bmatrix} \tag{10.20}$$

The purpose of this test is to compare the tuned feedforward with the original feedforward. All the tests were performed with constrained thruster rotation rates and with a static bow thruster rotated to 90°.

Figure 10.33 shows the 4-corner box tests performed with the tuned feedforward. One trajectory differs greatly from the two other, caused by a wave from west. From the performance metrics in Figure 10.34, the IAE for the tuned feedforward is obviously lower than the original and has a slightly lower IADC, which is mainly due to a smaller overshoot to correct for.

Figures 10.35 and 10.36 show the desired- and vessel-trajectory from the two setups. The setup with the original feedforward tracks the desired path poorly and has large overshoots in position and heading. This is also the case for the velocities in Figure 10.37. The controller reaches its saturation limit for force in sway and heading as seen in Figure 10.38. The tuned feedforward is chosen for further use.

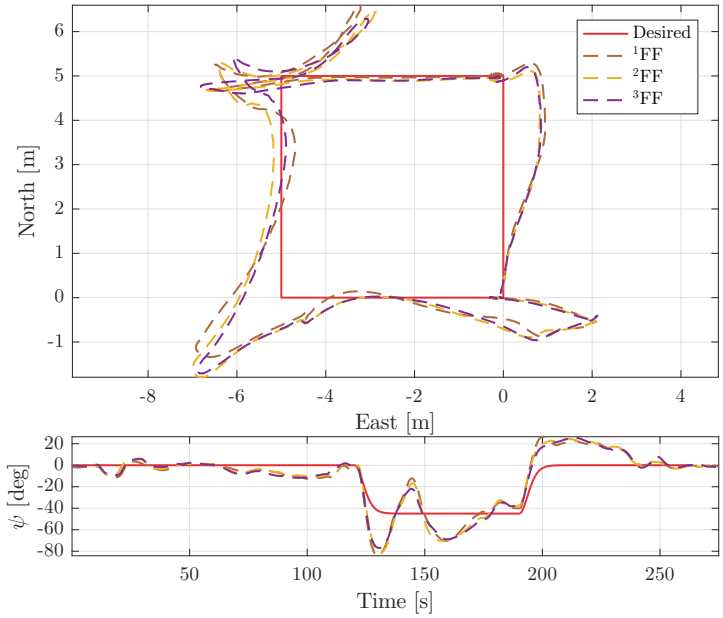


Figure 10.32: All vessel trajectories for PID with original feedforward.

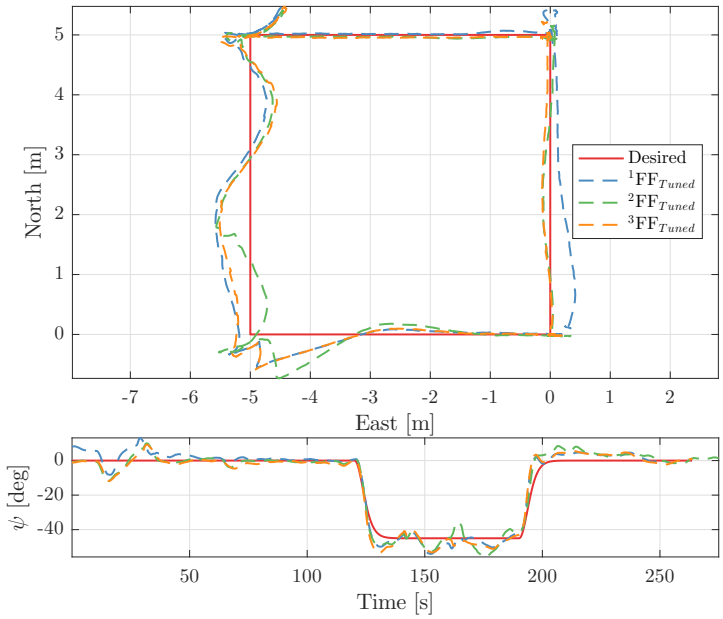


Figure 10.33: All vessel trajectories for PID with tuned feedforward.

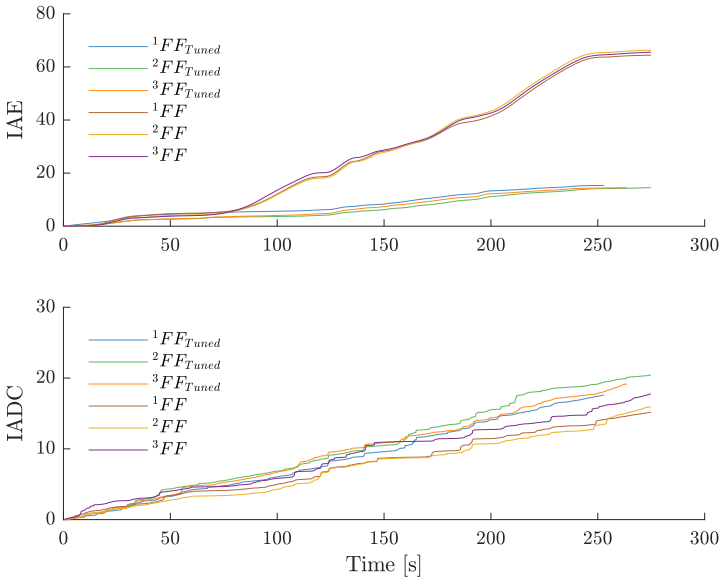


Figure 10.34: Performance metrics - PID with original feedforward and tuned feedforward.

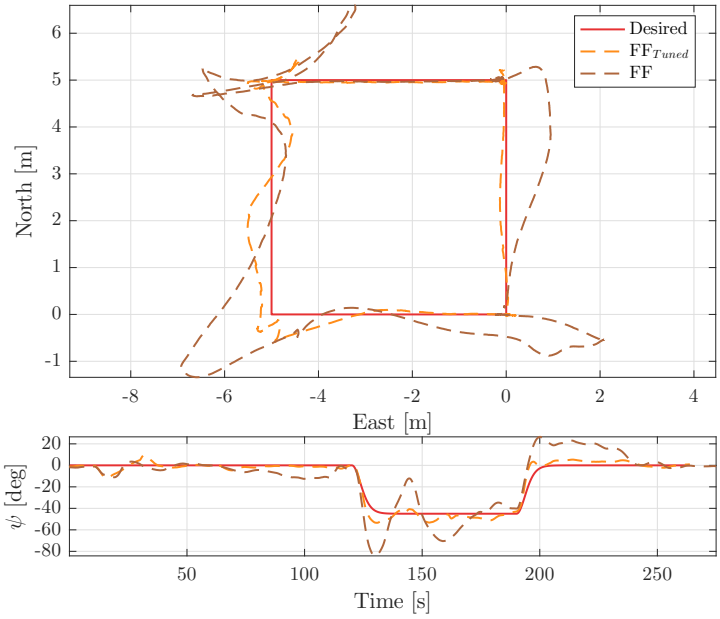


Figure 10.35: Comparison of the two best vessel trajectories with PID with original feedforward and tuned feedforward.

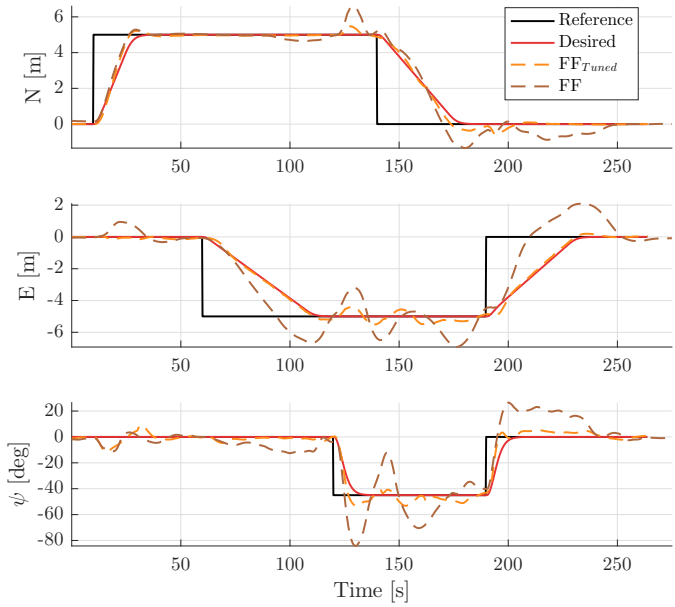


Figure 10.36: Comparison of the two best vessel trajectories over time using PID with original feedforward and tuned feedforward.

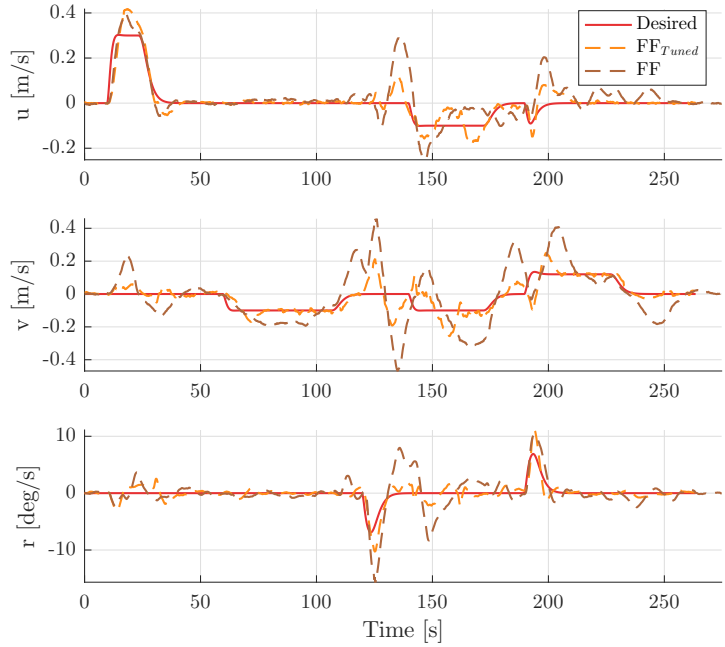


Figure 10.37: Comparison of the velocity for the two best vessel trajectories with PID with original feedforward and tuned feedforward.

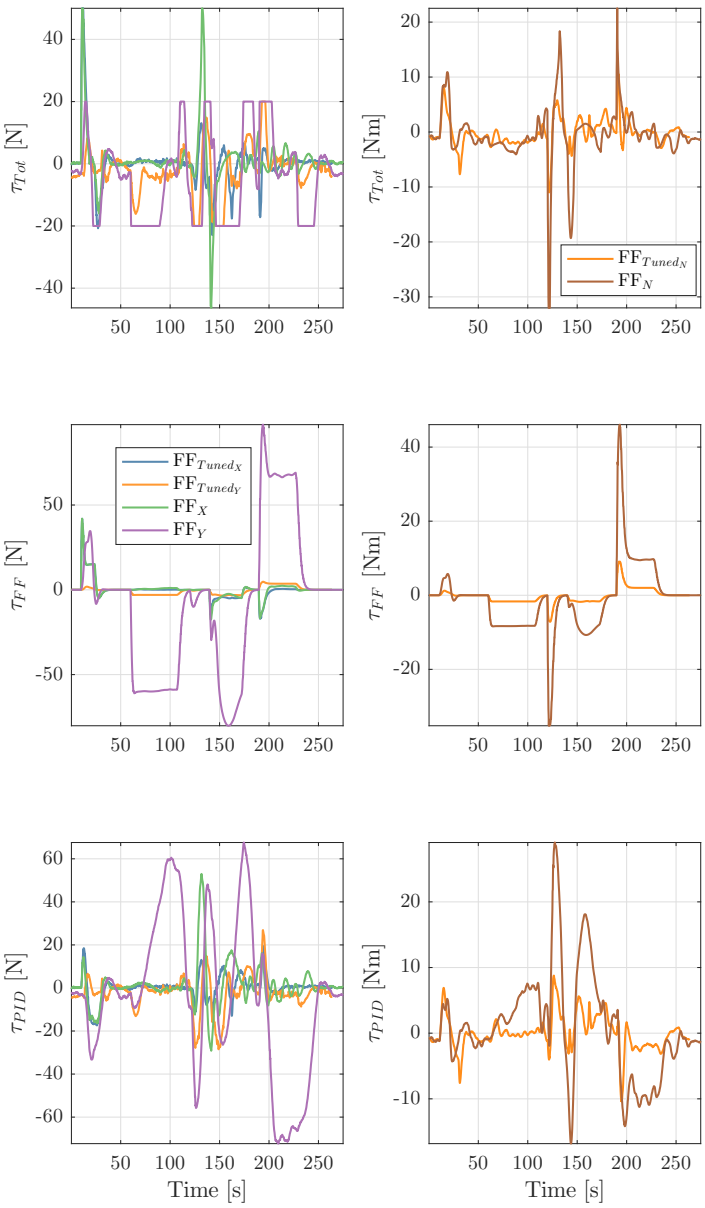


Figure 10.38: Comparison of the controller output for two best vessel trajectories with PID with original feedforward and tuned feedforward.

10.5.5 PID vs. Tuned PID-FF

The purpose of this test is to determine whether the PID controller with or without feedforward performs better. This test will have constrained thruster rotation rates, a static bow thruster set at 90° and a tuned feedforward as these setups proved to be best.

The performance metrics of the trajectories from the constrained PID run in Figure 10.17 and the tuned feedforward run in Figure 10.33, are shown in Figure 10.39. The IAE is slightly larger with the feedforward, whereas the IADC similar for the two setups. The two best trajectories from each setup are further compared.

ReVolt is able to track the desired trajectory in the pure surge and sway motions fairly well with both setups, as seen in figures 10.40 and 10.41. However, the feedforward makes ReVolt respond faster, since the desired trajectory is passed forward in the controller. This rapid response does, however, cause ReVolt to have larger overshoots in pose and velocity which can be seen in Figure 10.42. Both setups are struggling with the coupled surge-sway motion, which is especially evident from the vessel's poor ability to track the desired velocities for this motion. The PID controller performs better at this motion. The expected result from using the feedforward is for the PID part to contribute less to the total control effort. However, this is not the case in Figure 10.44. The PID controller needs to work more with the feedforward active to correct for the errors caused, which propagates into the thrust allocation. This is seen in Figure 10.43.

The setup with no feedforward is concluded to be the best. It has a smaller pose error, follows the desired trajectory and velocity better, is easier on the actuators and does not saturate the controller. The benefits of the feedforward did not compensate for the drawbacks caused by using an inaccurate model. The importance of having a correct feedforward model cannot be stressed enough. This will otherwise introduce errors into the system.

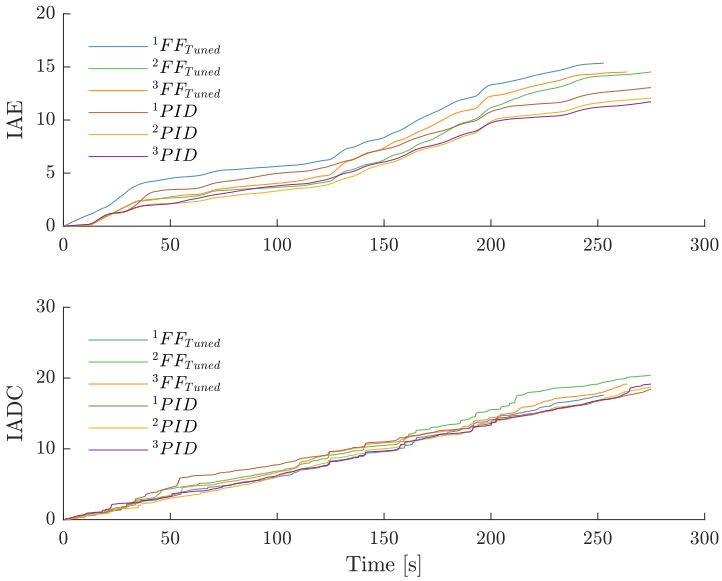


Figure 10.39: Performance metrics - PID vs PID with tuned feedforward.

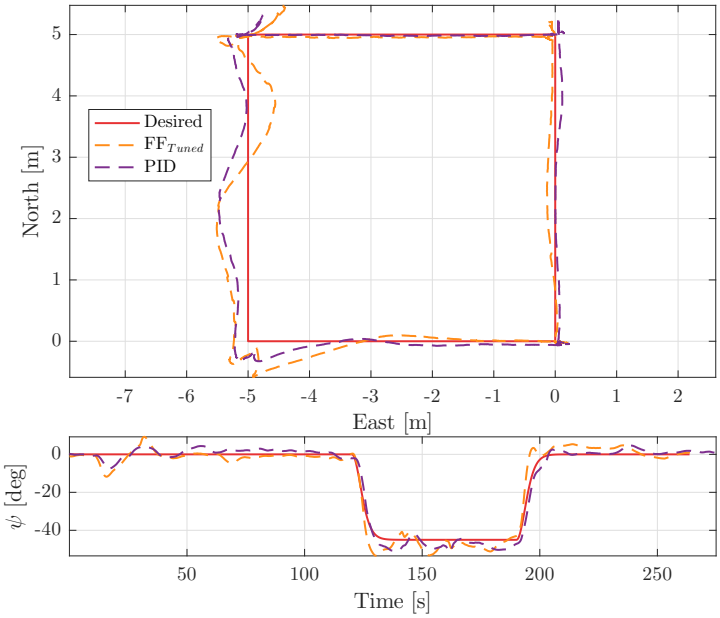


Figure 10.40: Comparison of the two best vessel trajectories with PID and PID with tuned feedforward.

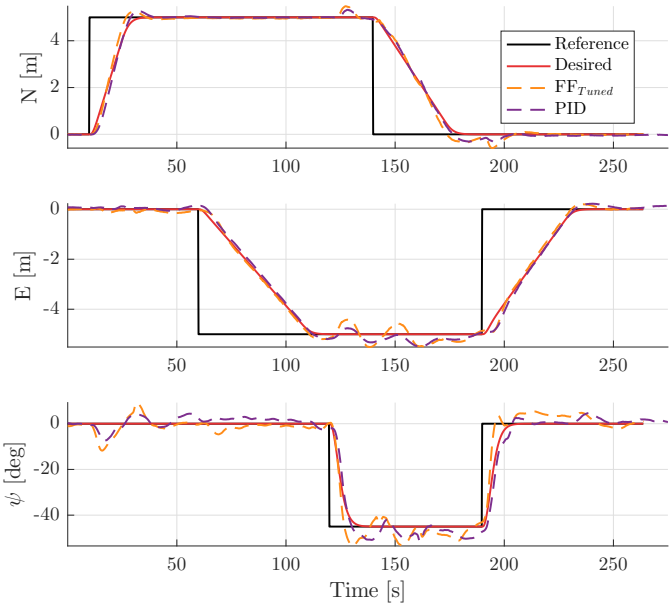


Figure 10.41: Comparison of the two best vessel trajectories over time using PID and PID with tuned feedforward.

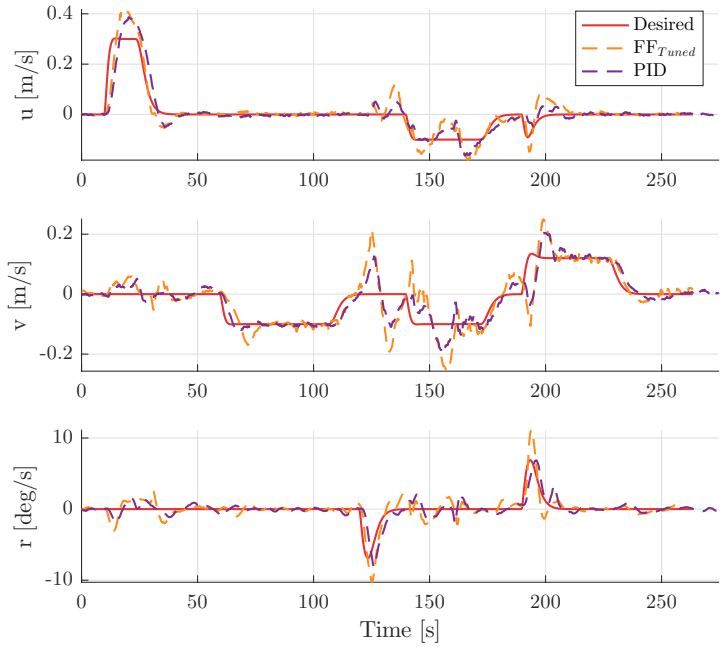


Figure 10.42: Comparison of the velocity for the two best vessel trajectories with PID and PID with tuned feedforward.

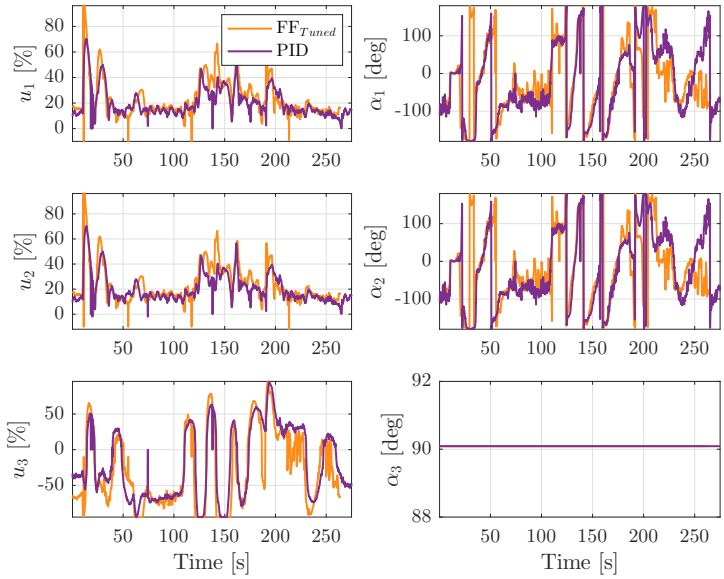


Figure 10.43: Comparison of the thrust allocation for the two best vessel trajectories with PID and PID with tuned feedforward.

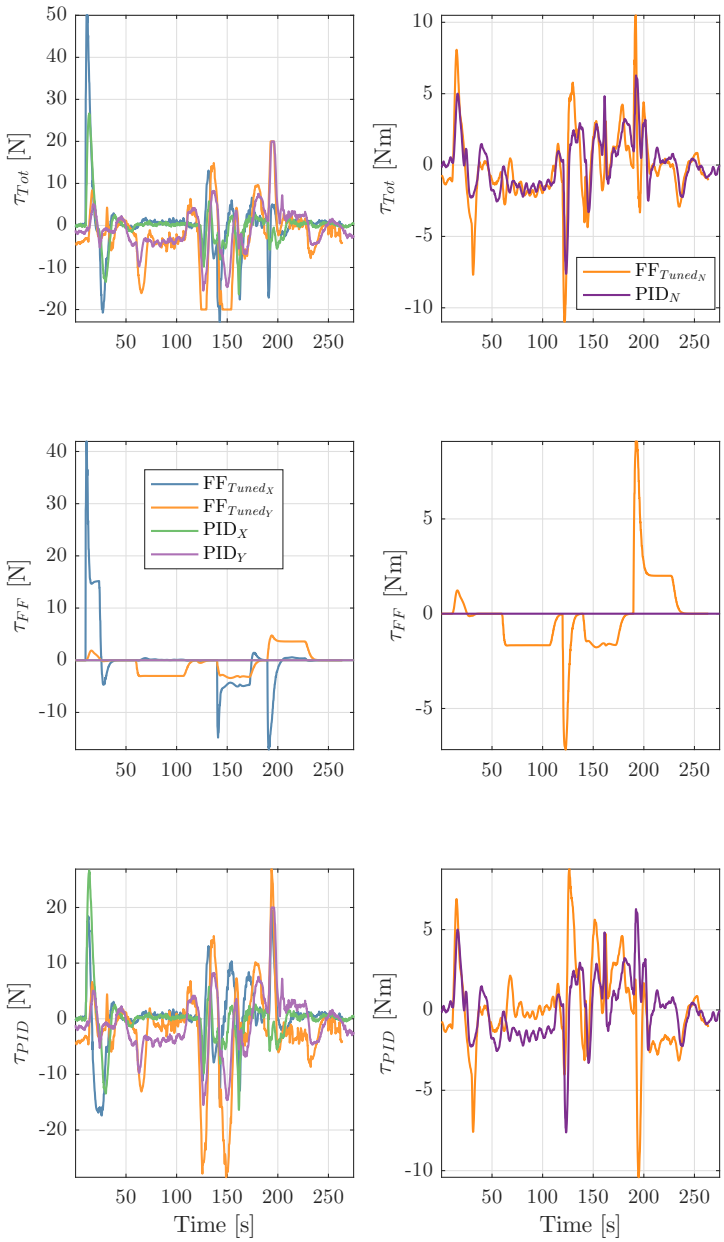


Figure 10.44: Comparison of the controller output for the two best vessel trajectories with PID and PID with tuned feedforward.

10.6 Application to Docking

To get an idea if this control system could be applied to docking an autonomous ferry, such a test was performed. The test consisted of setting the position reference of ReVolt close to a berth with fenders. ReVolt performed the maneuver with no interference from the operators. Figure 10.45 shows the position and heading of ReVolt in this attempt, as well as the position of the berth. The maneuver was a coupled surge-sway motion, with a desired heading of -40°. This is as mentioned previously, seemingly the hardest maneuver for ReVolt, which may have impaired the results, as opposed to having a pure sway motion. The sway speed was too high as ReVolt approached the berth, seen in Figure 10.46, which caused ReVolt to lightly bounce of the fenders before it came to a stop alongside the berth. The surge and sway speed was less than 0.15m/s as ReVolt made contact with the fenders, which would result in a deceleration unpleasant for passengers. However, by spreading the impact force on several fenders, it would not cause damage to the vehicle itself.

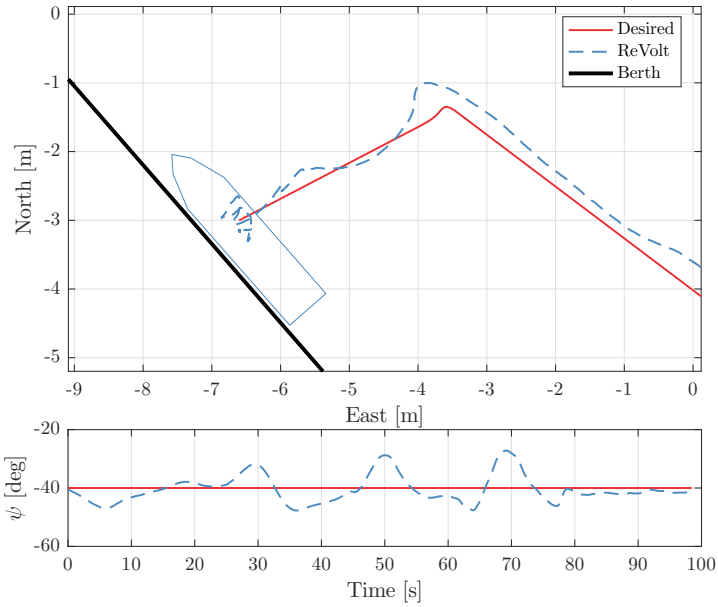


Figure 10.45: Position and heading of ReVolt when performing a docking maneuver.

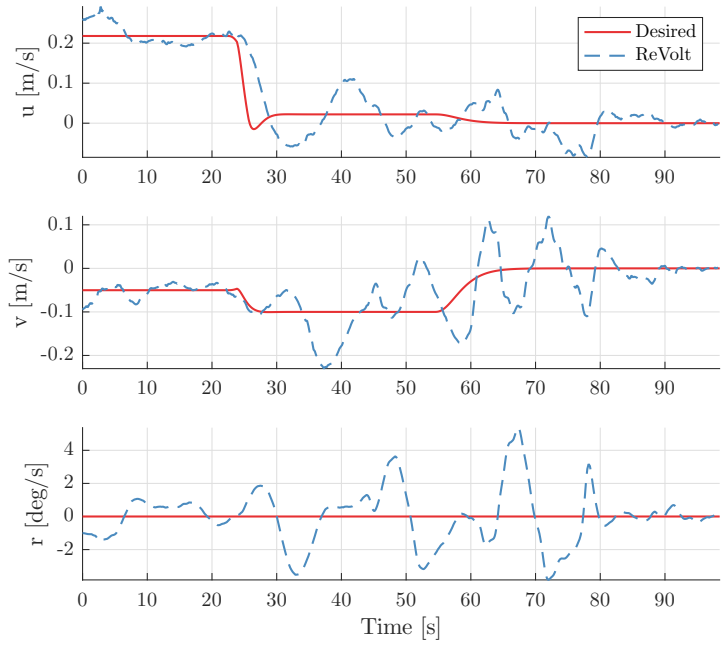


Figure 10.46: Velocity of ReVolt when performing a docking maneuver. Slight oscillations occur due disturbances and imperfect controller tuning.

Chapter 11

Discussion

This chapter includes a discussion of the choices made in this thesis, the experimental results and some remarks regarding the execution of the project.

Figure 11.1: ReVolt in sunset at Trondheim harbour 17 November 2016.

11.1 Dynamic Positioning System

A reference filter was chosen as the guidance system due to its simplicity and its easy implementation. The thought was for ReVolt to perform simple maneuvers and implement more advanced guidance with path following at a later time. The reference filter became more complex than first intended as saturating the output of the filter in Fossen (2011), caused the acceleration, velocity and position to become inconsistent. Therefore a saturation of the input was developed to get consistent position, velocity and acceleration trajectories.

As this thesis' goal was station keeping and low-speed maneuvering of ReVolt, a PID with feedforward was proposed. The PID controller is simple yet robust, which is crucial for an unmanned vessel. Adding the model-based feedforward gives some of the advantages of more advanced controllers which potentially can improve the maneuvering capabilities of ReVolt. In the case of an inaccurate model, it can simply be removed, leaving a functioning control system in the form of a PID controller.

Since this thesis is in association with DNV GL, they made their thrust allocation algorithm available to us. The implementation was not straight forward as it was written in a different programming language than used in ReVolt. Nonetheless, a big portion time was saved as opposed to developing one from scratch. The need to implement a bow thruster with non-symmetrical thrust characteristics was unexpected. This had to be dealt with in the thrust allocation as well as the reference filter, which increased the complexity of the DP system.

An accurate measurement of ReVolt's navigation state, pose and velocity, was desired for the PID controller. An observer based on ESKF was chosen due to its dead reckoning abilities and the possibility to add position measurements from other source than a GNSS. This is relevant in ReVolt's future as more navigational sensors are planned to be implemented.

11.2 Simulations

The different parts of the DP system was implemented and tested in MATLAB. However, this should have been done directly in ROS. This would test the actual implemented system and could prove helpful in finding bugs before experimental tests are conducted. This will save time as well as algorithms will not need to be implemented twice, which again will save more time.

11.3 Station Keeping and Low-Speed Maneuvering

One of the goals in Chapter 1 was for ReVolt to achieve DP and low-speed maneuvering. The optimal setup for this was found after several executions of the 4-corner box test with different setups. The best result, as seen in Figure 11.2, was achieved with constrained thrust allocation, a static bow thruster and no feedforward due to an inaccurate model.

ReVolt was able to follow the desired trajectory, with a maximal error of 0.3m in north, 0.5m in east and 6.5° in heading. These errors occurred during the coupled surge-sway motion from point 4 to 5 in the 4-corner box test. This coupled maneuver is in itself difficult for ReVolt, but also since the bow thruster produces the least amount of thrust in negative sway. Setting the bow thruster the opposite direction to -90° could have yielded a better result for that maneuver, but impaired the coupled movement in positive sway direction.

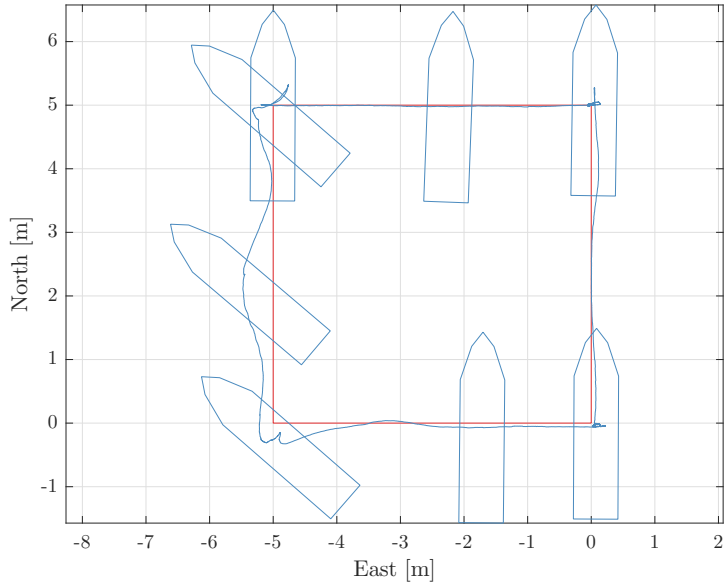


Figure 11.2: The best 4-corner box maneuver achieved by ReVolt. The outline has the same dimensions as ReVolt.

Due to an inaccurate model, the feedforward did not live up to its potential. A quick solution during the sea trial, was to limit the control effort from the feedforward. A better result was obtained, but with more time, tuning the model's added mass and dampening matrices could have given better results. It is evident from box plots such as Figure 11.2 that the center of rotation is not the same as CG. When ReVolt rotates during step 3 to 4 in the 4-corner box maneuver, it is seen in Figure 11.3 that its position curves upwards indicating that the center of rotation is in front of CG.

The ESKF did show promising results in the offline as well as the "online" tests. Further experiments with the implemented ESKF are desired to see if the acceleration biases stabilize by for example conducting a roll and pitch motion upon initialization. Also, a gravity state could be added in the ESKF to prevent initial errors in gravity which are now compensated for in the accelerometer biases. This can cause instability of the ESKF when exposed to roll motions, as the gravity component will be rotated to the horizontal plane and induce sway velocity.

Finally, using the estimates from the ESKF in conjunction with the controller and a feed-

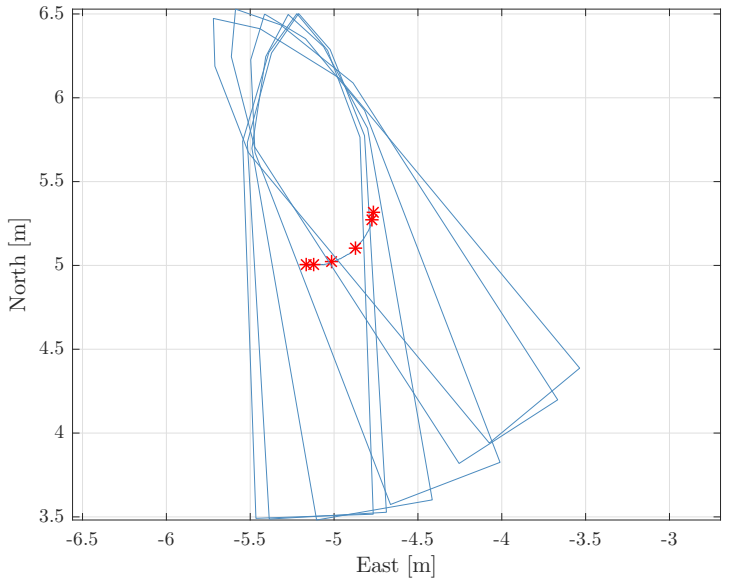


Figure 11.3: Enhanced portion of the North-West corner of the 4-corner box maneuver, with the Center of Gravity of marked. The outline has the same dimensions as ReVolt.

forward with an accurate model, is desirable to see if it improves the control performance of ReVolt.

11.4 Docking

ReVolt was set to dock with its port side towards a berth with fenders. This allowed for a experimental docking scenario not much unlike the autonomous ferry. This was a improvised test leading to a coupled surge-sway motion towards the berth. As mentioned previously, this is seemingly the hardest maneuver for ReVolt which may have impaired the results.

However, the vessel managed decently and with the following small changes to the DP system, the results could have been improved:

- Changing the reference parameters, for a more conservative desired trajectory.
- Setting the last setpoint closer/onto the berth, such that the thrusters keep the vessel pushed toward the berth.

For an autonomous ferry, the results may be adequate if a system is built in order to guide and moor the vessel, as well as extra sensors to provide close quarter navigation.

11.5 Practical Considerations

During this thesis there are particularly two practical considerations which stand out.

11.5.1 ”Behind the Scenes” Work

This thesis has been a very practical thesis with a lot of hands on work. Therefore a lot of work ”behind the scenes”, which has no value being mentioned in this thesis, is omitted. This includes for example all the hours spent debugging, overcoming implementation problems and converting experimental data to be used in MATLAB. It also includes the time spent debugging the RTK system as well as moving and installing it closer to the harbor, for better coverage at the test sites. Also omitted from the report is working with the in-house workshop at NTNU to create a mounting bracket for the Xsens, and improve the boat trolley to accommodate for easier launch of ReVolt.

11.5.2 Threshold for Field Tests

Everything done in this thesis has revolved around prototyping a DP control system for ReVolt and getting it to work. The only way to verify that the DP system works as expected is to perform field tests on land and sea. With ReVolt being 3 meters long, weighing a total of 257kg including weights and with the other necessary equipment, the work before and after every sea trial has been significant. Each sea trial has required good planning and coordination with NTNU’s transportation service and finding a support vessel for ReVolt. This had to be planned several days in advance. With each sea trial being a big operation, the threshold for field tests has been high and it has been crucial that each sea trial has yielded good results. It has been essential being two candidates collaborating, due to the size of the thesis and the size of ReVolt.

Chapter 12

Conclusion and Future Work

The main objective of this thesis was to develop a DP control system to achieve accurate and precise low-speed navigation using an IMU and GNSS with RTK corrections as navigational sensors. Section 12.1 presents the conclusion for this thesis and Section 12.2 lists potential future work on ReVolt.

12.1 Conclusion

The control system of the model ship ReVolt has been immensely improved, simulated, implemented and tested with experimental sea trials. ReVolt is now able follow a desired trajectory using a GNSS with RTK corrections and an IMU as navigational sensors.

Additionally, a mathematical model of ReVolt has been proposed. This model is based on physical measurements of ReVolt and simulations of the hull in WADAM, and the model is used in the feedforward part of the controller, utilizing changes in the reference. However, due to modeling errors, the feedforward did not improve the control performance of ReVolt.

ReVolt is able to achieve station keeping and low-speed maneuvering capabilities, with the best setup being a constrained thrust allocation, a static bow thruster at 90° and with the PID controller. ReVolt is able to follow the desired trajectory, with a maximal error of 0.3m in north, 0.5m in east and 6.5° in heading. The maximal errors occur during the coupled surge-sway motion, which is especially challenging for ReVolt to perform.

An observer consisting of an ESKF has been proposed for ReVolt, and gives promising results in offline and ”online” tests. However when implemented on ReVolt, it has issues with bias estimation. More tests were desired, but not possible due to limited time. Additionally, some prominent weaknesses and strengths of the navigational sensors on ReVolt, were revealed.

12.2 Future Work

This is an extensive thesis with great potential for improvements and future work. Following are some ideas for further work on ReVolt.

Improving the Error-state Kalman Filter

As the ESKF did not function properly, additional tests should be performed to test its stability. Looking into adding a gravity state, or implementing a more advanced gravity model to remove initial errors in gravity, can help improve the ESKF. If the navigation sensor package on ReVolt is extended, with for example a Light Detection and Ranging (LIDAR), can be added in a modular fashion to aid the ESKF during GNSS loss.

Implement a simulator in ROS

A simple simulator should be implemented in ROS for thorough Hardware In the Loop (HIL)-testing of the implemented system before real life experiments are conducted.

Mathematical model

As the mathematical model used is incorrectly, experimentally identifying these would render the feedforward useful, and probably improve the performance of ReVolt during maneuvering. Such a tests are planned conducted at the end of this thesis by by DNV GL at Marintek in Trondheim.

Waypoint tracking and path following

With a functioning DP control system, the guidance system should be expanded to include waypoint tracking and path following capabilities. This will enable ReVolt to more efficiently navigate a predefined route.

Sensor implementation

ReVolt is equipped with sensors to precisely understand its current location, but it has no ability to sense the surrounding environment. Additional sensors, such as a LIDAR, camera and radar, should be considered for ReVolt to increase its awareness of the environment and to aid in navigation. These sensors would need to be fused with the existing sensors in order to obtain the whole picture.

Docking

Further studies should be conducted to assess the existing control system's capabilities to perform docking maneuvers. It is expected that docking can be improved with only minor changes.

Collision avoidance

With an expansion of navigational and environmental sensors to detect obstacles, implementation of collision avoidance capabilities can be added to ReVolt.

Bibliography

Alfheim, H. L. and Muggerud, K. (2016). Dynamic positioning of the revolt model-scale ship. Project thesis.

Arunvydhya (2017). Xsens_mti_ros_node. http://github.com/xsens/xsens_mti_ros_node. Last Accessed: 3 March 2017.

Breivik, M. (2010). *Topics in Guided Motion Control of Marine Vehicles*. PhD thesis, Norwegian University of Science and Technology, Trondheim, Norway.

Breivik, M., Kvaal, S., and Østby, P. (2015). From Eureka to K-Pos: Dynamic Positioning as a Highly Successful and Important Marine Control Technology. *IFAC-PapersOnLine*, 48 (16):313 – 323. 10th IFAC Conference on Manoeuvring and Control of Marine Craft MCMC 2015.

Brown, R. G. and Hwang, P. Y. C. (2012). *Introduction to Random Signals and Applied Kalman Filtering: With MATLAB Exercises and Solutions*. John Wiley & Sons, Inc.

DNV GL (2015). ReVolt main report. Technical Report 2015-0170, Rev.1, DNV GL, Høvik, Norway.

Du, S., Sun, W., and Gao, Y. (2017). Improving observability of an inertial system by rotary motions of an imu. *Sensors*, 17(4):698.

Egeland, O. and Gravdahl, J. T. (2002). *Modeling and Simulation for Automatic Control*. Marine Cybernetics Trondheim, Norway.

Eriksen, B.-O. H. and Breivik, M. (2017). *Modeling, Identification and Control of High-Speed ASVs: Theory and Experiments*, pages 407–431. Springer International Publishing, Cham.

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

Grip, H. F., Fossen, T. I., Johansen, T. A., and Saberi, A. (2015). Globally exponentially

stable attitude and gyro bias estimation with application to GNSS/INS integration. *Automatica*, 51:158–166.

Hemisphere (2017). *Vector VS330 GNSS Receiver Datasheet*. Hemisphere.

Hoberock, L. (1976). A survey of longitudinal acceleration comfort studies in ground transportation vehicles. Research Report 40, The University of Texas at Austin, Austin, Texas.

Kongsberg (2016). *Kongsberg Seatex Seapath 330+ Datasheet*. Kongsberg Seatex.

Kyckelhahn, B. A. and Forbus, K. D. (2004). Jitter in self-explanatory simulation. *Northwestern University*. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.497.706&rank=1.

Marguedas (2017). ROS distributions. http://wiki.ros.org/Distributions. Last Accessed: 23 May 2017.

Meland, S. I. (2016). Først i verden med testområde for ubemannede fartøyer. http://www.adressa.no/nyheter/sortrondelag/2016/09/30/F%C3%B8rst-i-verden-med-testomr%C3%A5de-for-ubemannede-fart%C3%B8yer-13572359.ece. Last Accessed: 01 December 2016.

Moler, C. and Loan, C. V. (2003). Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, New York, 2nd edition.

Parkinson, B. W. (1996). *Progress in astronautics and aeronautics: Global positioning system: Theory and applications*, volume 2. AIAA.

Pavlis, N. K., Holmes, S. A., Kenyon, S. C., and Factor, J. K. (2012). The development and evaluation of the earth gravitational model 2008 (egm2008). *Journal of Geophysical Research: Solid Earth*, 117(B4).

Perko, E. (2017). Nmea_navsat_driver. http://wiki.ros.org/nmea_navsat_driver. Last Accessed: 1 April 2017.

Rolls-Royce (2016). Rolls-Royce unveils a vision of the future of remote and autonomous shipping. http://www.rolls-royce.com/media/press-releases/yr-2016/pr-12-04-2016-rr-unveils-a-vision-of-future-of-remote-and-autonomus-shipping.aspx. Last Accessed: 06 December 2016.

Skjetne, R., Smogeli, . 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.

Skjetne, R., Sørensen, M. E. N., Breivik, M., Sørensen, A. J., Brodtkorb, A. H., Værnø, S. A. T., Kjerstad, Ø. K., Vinje, B. O., and Calabrò, V. (2017). AMOS DP research cruise 2016: Academic full-scale testing of experimental dynamic positioning control

algorithms onboard R/V gunnerus. In *Proceedings of OMAE, Trondheim, Norway, June 2017*.

SNAME (1950). Nomenclature for treating the motion of a submerged body through a fluid. Technical report, The Society of Naval Architects and Marine Engineers.

Solà, J. (2017). Quaternion Kinematics for the Error-State Kalman Filter. Technical report, Institutde Robtica i Informtica Industrial. Available: http://www.iri.upc.edu/people/jsola/JoanSola/objectes/notes/kinematics.pdf.

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

Sørensen, M. E. N., Bjørne, E. S., and Breivik, M. (2016). Performance comparison of backstepping-based adaptive controllers for marine surface vessels. In *Control Applications (CCA), 2016 IEEE Conference on*, pages 891–897. IEEE.

Stensvold, T. (2016). Verdens frste frerlse passasjerferge kan g over en kanal i Trondheim. https://www.tu.no/artikler/verdens-forste-forerlose-passasjerferge-kan-ga-over-en-kanal-i-trondheim/363790.

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

Vik, B. (2014). *Integrated Satellite and Inertial Navigation Systems*. Booklet, Norwegian University of Science and Technology, Trondheim, Norway.

Wit, C. D. (2009). Optimal thrust allocation methods for dynamic positioning of ships. Master's thesis, Delft University of Technology, Delft, the Netherlands.

Xsens (2016). *Mti User Manual - Mti 10 Series and MTi 100-Series*. Xsens Technologies B.V.

Yara (2017). YARA and KONGSBERG enter into partnership to build world's first autonomous and zero emissions ship. yara.com/media/press_releases. Last Accessed: 14 May 2017.

Appendices

Appendix A

Media Coverage

Since the start of the academic project with ReVolt in the fall of 2016, there has been a lot of media attention related to the project. The purpose of these media appearances have been to promote the concept of unmanned ships and show the public how far technology has come in this field. The different articles and movies related to ReVolt due to this master thesis are listed below.

Figure A.1: Henrik and Kjetil operating ReVolt while being filmed by the German TV-channel ZDF. Courtesy of Trondheim Havn

Articles and videos:

- Article about the concept ship ReVolt and how the scale model ship will be tested in

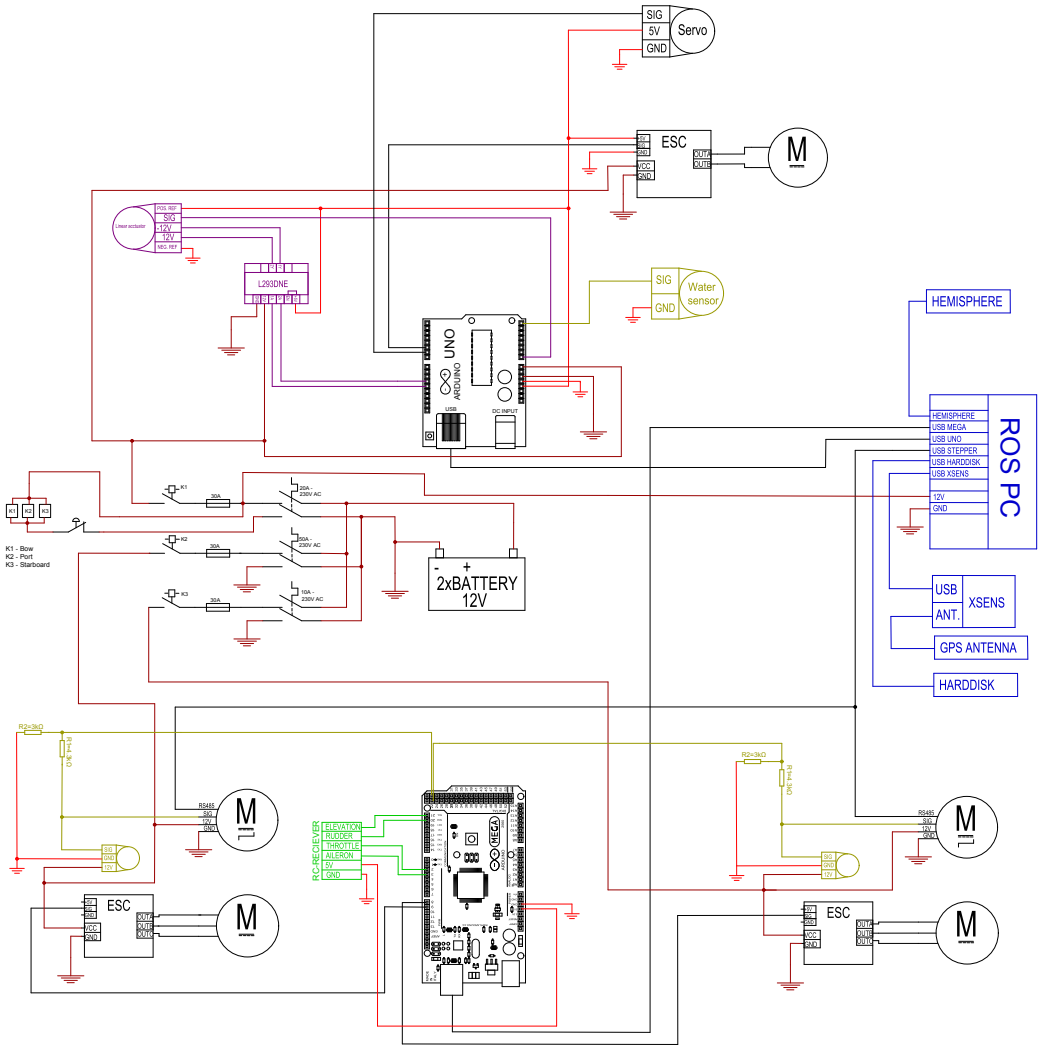
Trondheim (15.11.2016): www.trondheimhavn.no/nyhet/tester-forerlose-skip-i-nyhavna-1216.aspx

- Article about the first sea trials with ReVolt in Trondheim (21.11.2016): www.trondheimhavn.no/nyhet/tester-trondheims-forste-forerlose-ferge-1221.aspx
- Video report by Der Spiegel about ReVolt (20.01.2017): video.spiegel.de/flash/33/54/1734533_1024x576_H264_HQ.mp4
- Video about the specialization project during the fall created by Kindergarden Media for NTNU, and filmed by Henrik Alfheim and Kjetil Muggerud (06.02.2017): www.youtube.com/watch?v=AyFkOfD1II8
- Article about the filming day with the German news channel ZDF (16.03.2017): trondheimhavn.no/nyhet/norsk-skipsteknologi-pa-tysk-tv-1248.aspx
- German TV-channel 3-Sat (16.03.2017): www.3sat.de/mediathek/?mode=play&obj=65614
- German TV-channel ZDF (20.03.2017): www.zdf.de/nachrichten/heute-plus/videos/autonome-schiffe-100.html
- Article about unmanned ships and ReVolt in SHZ (04.04.2017): www.shz.de/deutschland-welt/netzwelt/der-kapitaen-verlaesst-das-fahrende-schiff-id16511641.html
- Article about unmanned ships in the German newspaper Sächsische Zeitung (04.04.2017): www.sz-online.de/nachrichten/autonome-schifffahrt-rueckt-naeher-3652317.html
- News segment about NTNU Ocean Week on NRK's local news (03.05.2017): tv.nrk.no/serie/distriktsnyheter-midtnytt/DKTL98050317/03-05-2017#t=1m1s
- Article about unmanned maritime transport in Universitetsavisa (03.05.2017): www.universitetsavisa.no/forskning/2017/05/03/Tror-p%C3%A5-automatisert-maritim-transport-66069.ece
- Article about NTNU Ocean Week in Teknisk Ukeblad (04.05.2017): www.tu.no/artikler/her-er-kongsbergs-to-nyeste-autonome-bater/382484
- Article about ReVolt and NTNU Ocean Week in Adressa (04.05.2017): www.adressa.no/pluss/okonomi/2017/05/04/B%C3%A5ten-styres-fra-land-14676736.ece

Appendix B

ReVolt Wiring Schematic

ReVolt Wiring schematic
Date: 01.06.2017



Appendix C

Introduction to ROS

Introduction to ROS

1 Introduction

This introduction to Robot Operating System (ROS), written by Henrik Lemcke Alfheim and Kjetil Muggerud, as a guide to those starting with ROS. It is recommended to go through the tutorial at http://wiki.ros.org/ROS/Tutorials for a more in-depth insight to ROS.

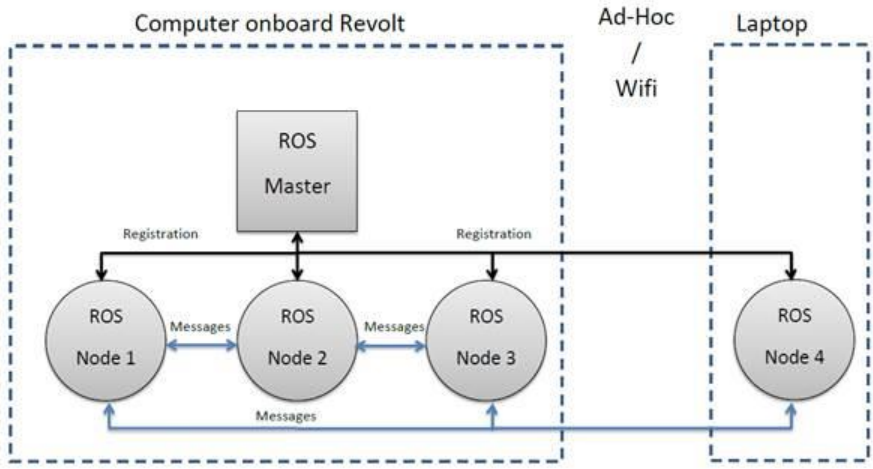


Figure 1 - Simple ROS network

2 Basics 101: (assuming ROS is correctly installed)

1. Create a "workspace"

```
user@computer:~/$ mkdir –p ~/workspace/src
user@computer:~/$ cd workspace/src
user@computer:~/workspace/src/catkin_init_workspace
user@computer:~/workspace/src/$ cd ..
user@computer:~/workspace/catkin_make
```

The above commands do the following:
- Make a directory
- Move into workspace/src
- Initiate the workspace
- Step back to workspace/
- Build

You will now have a workspace with essentially nothing in it. It is now necessary to source this workspace with:

```
user@computer:~/workspace/ source devel/setup.bash
```

2. Creating a package:

```
user@computer:~/workspace/src/ catkin_create_pkg example std_msgs rospy roscc
```

An example package is now created that also depends on other packages:
std_msgs: is a package containing many different structs for sending and receiving “topics”
rospy: makes the package able to understand Python language
roscc: makes the package able to understand C++ language

These dependencies can be edited in the CMakelists.txt and package.xml file.

3. Now let us create a node example in Python.
First we create a publisher node that publishes the topic chatter

File location: workspace/src/example/talker.py

```
#!/usr/bin/env python
# license removed for brevity
import rospy
from std_msgs.msg import String

def talker():
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        hello_str = "hello world %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

Then we create a listener node that runs a function every time it picks up data on the topic chatter.

File location: workspace/src/example/listener.py

```
#!/usr/bin/env python
import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)

def listener():

    # In ROS, nodes are uniquely named. If two nodes with the same
    # node are launched, the previous one is kicked off. The
    # anonymous=True flag means that rospy will choose a unique
    # name for our 'listener' node so that multiple listeners can
```

```
    # run simultaneously.
    rospy.init_node('listener', anonymous=True)

    rospy.Subscriber("chatter", String, callback)

    # spin() simply keeps python from exiting until this node is stop
ped
    rospy.spin()


if __name__ == '__main__':
    listener()
```

4. Then we build this workspace by running catkin_make from the bottom of the workspace directory and starting roscore:

```
user@computer:~/workspace/$catkin_make
user@computer:~/workspace/$ roscore
```

Open two new terminals, source devel/setup.bash inside your workspace and run both nodes like this:

```
user@computer:~/workspace/$ rosrun example publisher.py
```

and

```
user@computer:~/workspace/$ rosrun example listener.py
```

5. Open a new terminal and try out these handy commands:

Command	**Info**
Rosnode list	Lists all active nodes
Rostopic list	Lists all initiated topics
Rostopic echo "topic"	Outputs data on the specified "topic"
Rostopic pub topic msg_type args example rostopic pub chatter std_msgs/String -- Hello	Manually writes to the specified topic

6. Launching both nodes from a launch file. First press Ctrl-C in all three windows to shutdown roscore, and both example nodes.

7. **Writing a launch file**

   File location: workspace/src/tutorial.launch

```
<launch>

  <node name=''listener'' pkg=''example'' type=''listener.py''>
  <node name=''publisher'' pkg=''example'' type=''publisher.py''>

</launch>
```

```
user@computer:~/workspace/src/$ roslaunch tutorial.launch
```

   This will start roscore and the two nodes in one command.

8. Recording or “bagging”

   To bag, make a new directory anywhere suited to store the bag files, and run the command “rosbag record –a” which will record everything.

```
user@computer:~/workspace/bags/$ rosbag record -a
```

   To play them use:
   Rosbag play “name.bag”

9. Some different graphical user interfaces that are handy for plotting and viewing ROS messages

   rqt_plot
   rqt_console
   rqt_bag

3 ROS on ReVolt

	Operating system	ROS version	Password
Embedded computer	Ubuntu 14.04 LTS	Indigo	******
Laptop	Ubuntu 14.04 LTS	Indigo	******

3.1 General information

- The workspace is named revolt and is located in the user directory: /home/ros/**revolt**
- The workspace setup.bash file is automatically sourced in .bashrc (/home/ros/.bashrc)
- The launch file is located in /src/revolt.launch and will launch everything needed to run the ReVolt
    - It is automatically launched from roscore_startup at bootup (comment to stop this)
    - The bootup script can be found at /usr/locan/bin/roscore_startup
- For debugging and finding errors, it's recommended to not start the launch, but run each node individually with rosrun as this will show errors and log information.

3.1.1 Roscore_startup

Roscore_startup is a script that is run at start up, it is located at /etc/init.d

```
Source /opt/ros/indigo/setup.bash
Source /home/ros/revolt/devel/setup.bash

export ROS_WORKSPACE=/home/ros/revolt
export ROS_IP=10.0.0.1
export ROS_MASTER_URI=http://10.0.0.1:11311

(cd /home/ros/revolt && source devel/setup.sh)                    #Might not be needed

(cd /home/ros/revolt && catkin_make)            #Builds the workspace
(cd /home/ros/revolt && catkin_make actuators_firmware_stern-upload) #upload Arduino code
(cd /home/ros/revolt && catkin_make actuators_firmware_bow-upload) #upload Arduino code

(cd /home/ros/revolt/src && roslaunch revolt.launch)              #launces the launch file
```

- This file is run at bootup
    - It will source the installed directories where ROS is installed
    - Setting the right IP addresses (see section **Error! Reference source not found.**)
    - Builds the workspace
    - Upload code to Arduinos
    - Execute launch file

3.2 Adding custom messages

In the package custom_msgs you can easily add new type of custom messages.
Look in /home/revolt/src/custom_msgs/msg/threeFloats.msg

```
float64 setpoint
float64 state
float64 error
```

It consists of three float64 types named setpoint, state and error.

This message needs to be built so we have to tell ROS to add this message. This is done in the CMakeLists.txt for this package, located at /home/revolt/src/custom_msgs/CMakeLists.txt

```
## Generate messages in the 'msg' folder
Add_message_files(
FILES
…
###### ADDED CUSTOM msgs ####
podAngle.msg
threeFloats.msg
)
```

When adding new messages, build the workspace with the command catkin_make from workspace folder (in this case /revolt).

New packages needs to depend on the custom_msgs package in order to use these messages.
This is done by adding dependencies in the /workspace/src/package/ (CMakelists and package.xml)

File location: /revolt/src/actuators/CMakeLists.txt

```
Find_package( catkin RQUIRED COMPONENTS
        roscpp
        rospy
        …
        custom_msgs                    # added dependency on custom_msgs package
)
```

File location: /revolt/src/actuators/package.xml

```
<build_depend>…..</build_depend>
<build_depend>custom_msgs</build_depend>        #added build dependency on custom_msgs
<run_depend>……..</run_depend>
<run_depend>custom_msgs</run_depend>            # added run dependency on custom_msgs
```

Now the message can be imported in any Python or C++ script.

In Python:

```
from custom_msgs import threeFloats
```

Appendix D

ROS Graphs

A ROS graph is a graphical representation of the system's software structure which is generated in ROS. The representation includes which nodes are active and indicate which topics are being sent between these nodes. Figure D.1 shows a simple example of information sharing between two nodes. The node named /ns/node1, which is a part of the namespace ns, subscribes to the topic /topic2 and publishes the topic /ns/topic1. /topic2 is published by the node /node2, which is not a part of a name space. /node2 subscribes to the topic /ns/topic1.

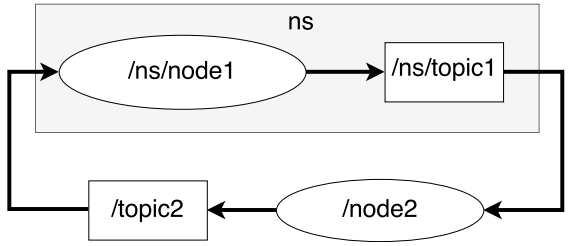


Figure D.1: A simple ROS graph showing the sharing of information between two nodes.

The same visualization of namespaces, nodes and topics is used in the ROS graphs.

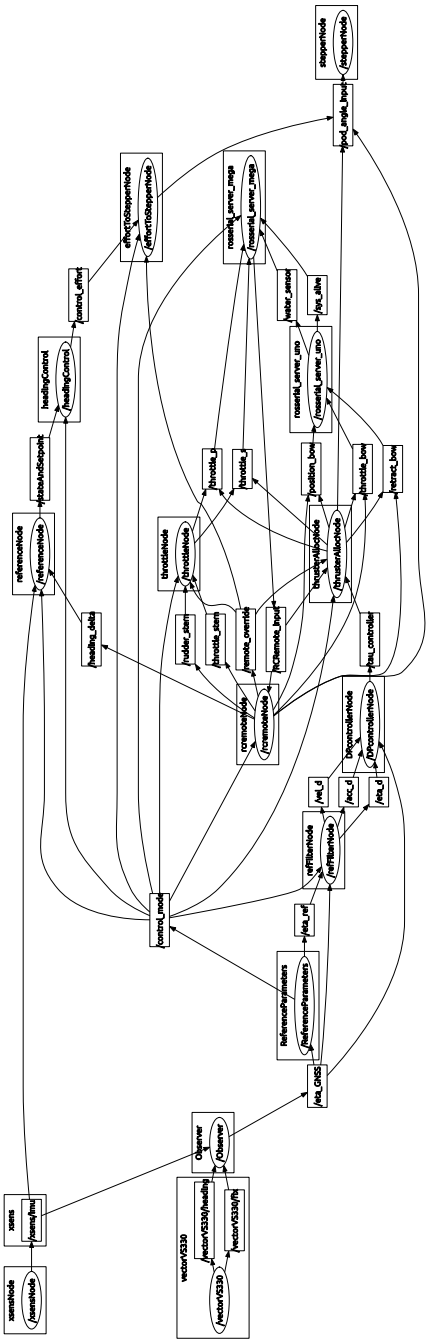


Figure D.2: ROS graph showing the system structure of the entire system on ReVolt.

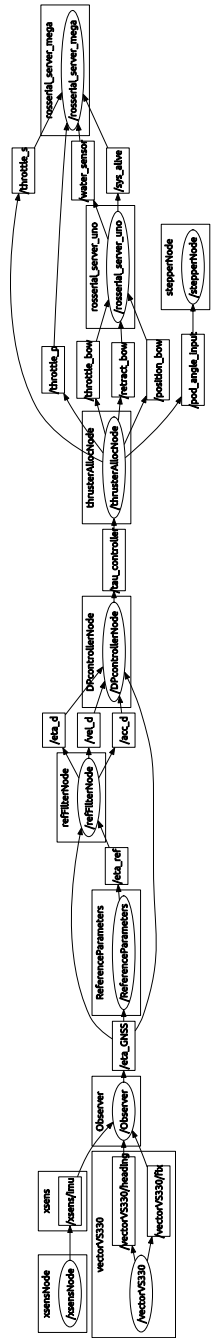


Figure D.3: ROS graph showing the system structure of the necessary elements to control ReVolt in dynamic positioning mode.

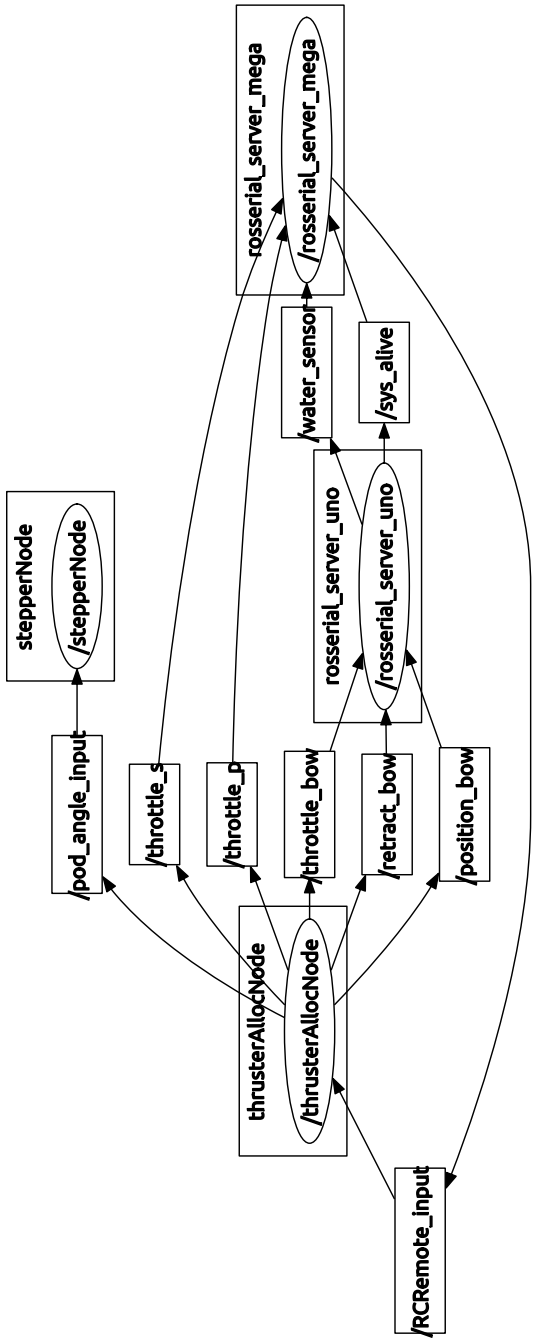


Figure D.4: ROS graph showing the system structure of the necessary elements to control ReVolt in manual thrust allocation mode.

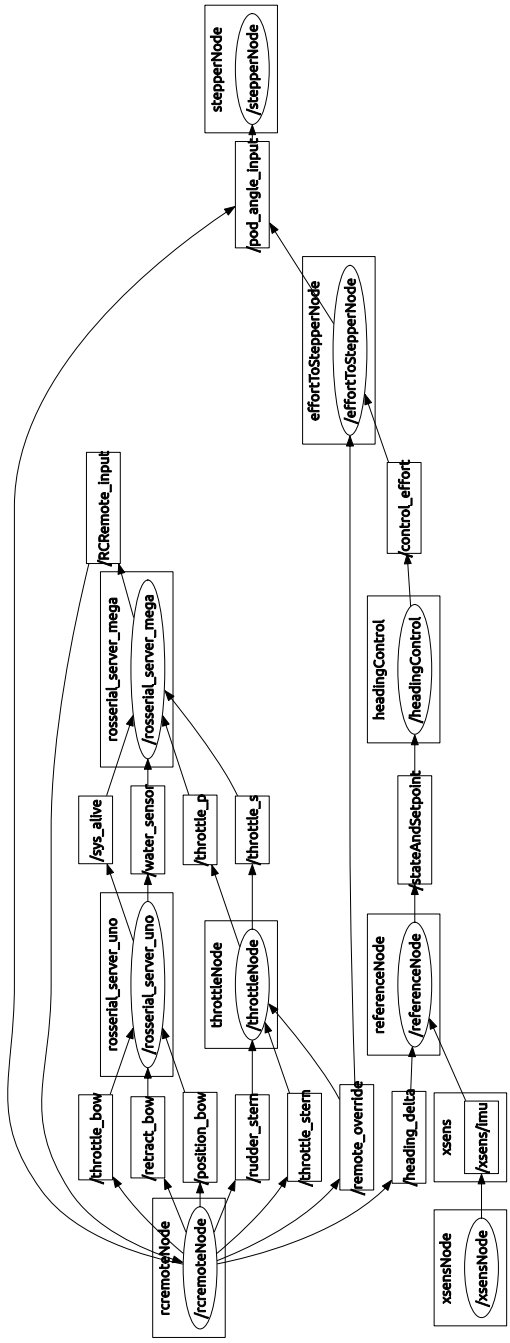


Figure D.5: ROS graph showing the system structure of the necessary elements to control ReVolt in heading controller mode.

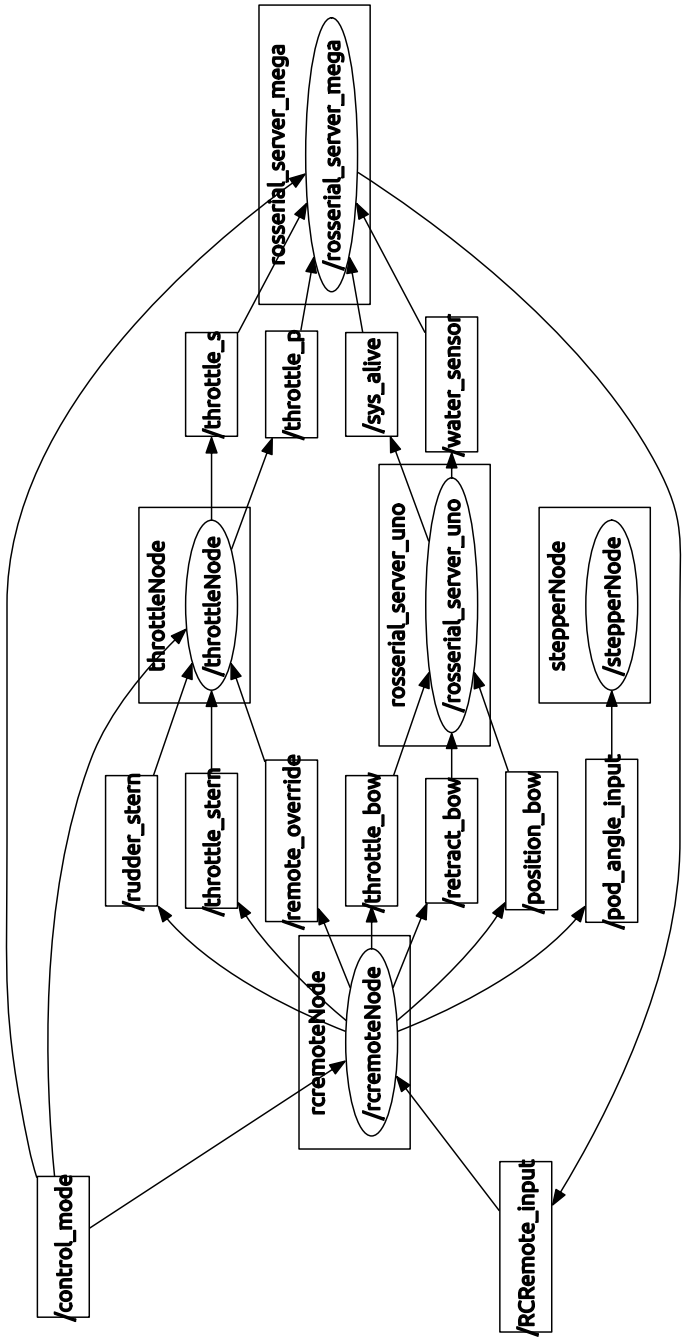


Figure D.6: ROS graph showing the system structure of the necessary elements to control ReVolt in manual mode.

Appendix E

Relevant Output From WADAM Simulations

Below is the relevant data from WADAM simulations of ReVolt's hull. This is the raw data used in the added mass matrix and the linear damping matrix. The values in the matrices are dimensionless and the dimensionalizing factors are found in the top of the document.

```
     4.1 EXPLANATION OF THE RESULTS
-------------------------------


NON-DIMENSIONAL DEFINITIONS:
----------------------------                          I=1-3              I=1-3  I=4-6       I=4-6
                                                      J=1-3              J=4-6  J=1-3       J=4-6

ADDED MASS MATRIX    NON-DIMENSIONALIZED BY:    RO*VOL,            RO*VOL*L           RO*VOL*L*L
DAMPING MATRIX       NON-DIMENSIONALIZED BY:    RO*VOL*SQRT(G/L)   RO*VOL*SQRT(G*L)   RO*VOL*L*SQRT(G*L)


NON-DIMENSIONALIZING FACTORS:
-----------------------------

THE OUTPUT IS NON-DIMENSIONALIZED USING -

RO  = DENSITY OF THE FLUID
G   = ACCELERATION OF GRAVITY
L   = CHARACTERISTIC LENGTH, AS GIVEN IN THE INPUT
VOL = DISPLACED VOLUME OF BODY 1 (COMBINED MORISON AND PANEL MODEL)
RO  =   0.1025E+04
G   =   0.9807E+01
VOL =   0.2270E+04
L   =   0.6000E+03


        INDICES:
        --------

1  INDICATES  SURGE  AND  X-COMPONENT OF FORCE
2  INDICATES  SWAY   AND  Y-COMPONENT OF FORCE
3  INDICATES  HEAVE  AND  Z-COMPONENT OF FORCE
4  INDICATES  ROLL   AND  ROLL  MOMENT
5  INDICATES  PITCH  AND  PITCH MOMENT
6  INDICATES  YAW    AND  YAW   MOMENT


ADDED MASS MATRIX
-----------------
            1             2             3             4             5             6

1     2.5253E-02    0.0000E+00    4.3714E-02    0.0000E+00    3.0520E-02    0.0000E+00
2     0.0000E+00    1.8016E-01    0.0000E+00   -9.0145E-03    0.0000E+00    8.5114E-03
3     4.3225E-02    0.0000E+00    1.1046E+00    0.0000E+00    4.2323E-02    0.0000E+00
4     0.0000E+00   -9.0228E-03    0.0000E+00    7.3716E-04    0.0000E+00    6.2726E-04
5     3.0550E-02    0.0000E+00    4.2979E-02    0.0000E+00    5.0995E-02    0.0000E+00
6     0.0000E+00    8.5367E-03    0.0000E+00    6.2633E-04    0.0000E+00    9.9408E-03


TOTAL DAMPING MATRIX
--------------------
            1             2             3             4             5             6

1     1.0210E-01    0.0000E+00    2.1229E-02    0.0000E+00    8.7190E-02    0.0000E+00
2     0.0000E+00    1.2122E+00    0.0000E+00   -4.4273E-02    0.0000E+00    5.5793E-02
3     1.9315E-02    0.0000E+00    1.3380E+00    0.0000E+00    1.1508E-01    0.0000E+00
4     0.0000E+00   -4.4312E-02    0.0000E+00    2.9163E-03    0.0000E+00    1.8756E-03
5     8.6963E-02    0.0000E+00    1.1856E-01    0.0000E+00    1.0066E-01    0.0000E+00
6     0.0000E+00    5.5825E-02    0.0000E+00    1.8811E-03    0.0000E+00    6.0053E-02
```

Appendix F

Instructions for Users of ReVolt

Instructions for users of ReVolt, ITK, NTNU

General

1. All users must read and use ReVolt in accordance with the contract between NTNU and DNVGL[1]
2. Only persons with permission from NTNUs contact person[1], are allowed to access or use ReVolt and its equipment.
3. The model ship ReVolt is DNVGLs property and is lent to NTNU for educational and research purposes. Any other use must be approved by DNVGL.
4. Equipment and documentation that is taken out of ReVolt shall be used and stored in a secure way such that unauthorized access is prevented.
5. Documentation of equipment and any source code shall be treated as classified unless it is clear that it is open.

Familiarization of risks and equipment

6. All users shall be familiar with the risks of operating ReVolt (i.e. rotating equipment).
7. All users shall be familiar with the User Manual for ReVolt.

Before operation

8. Scope of work and plan for missions (objectives, location, resources, etc.) shall be documented and used as a basis for risk assessment and risk mitigation.
9. Mission Risk Analysis of the operation with the ReVolt system has to be conducted. The external factors (such as weather, boat traffic, population, and infrastructure) has to be taken into account.
10. A full system check of hardware
    1. Batteries.
    2. Sensors.
    3. Actuators.
    4. Emergency Stop switch.

During operation

11. Life vests must be worn at all times while working close to and on the water.
12. Checklist for sea-launch
    1. Inspect the hull for any damage, inside and outside. Check around the azimuth thrusters for any damage.
    2. Turn on the controller, with the throttle in the neutral position.
    3. Turn on all the circuit breakers
    4. Check that the system and actuators operate as expected.
    5. Check that the front azimuth is in the up position.
    6. Close all hatches.
    7. Attach a guide rope.
    8. Roll the trailer into the water, unhook the boat and pull the trailer back on land while still keeping ReVolt safe.
    9. Place the weights into the boat (Recommended for stability). Check for leaks.

[1] UTLÅNSAVTALE – REVOLT MODELLBÅT v4

10. Check again if the system operates as expected, before setting off.

13. Checklist for sea-recovery
    1. Remove weights from the boat.
    2. Ensure azimuth is retracted into the hull.
    3. Roll the trailer fully into the water and guide revolt onto the trailer. Attach the hook and pull the trailer with ReVolt out of the water.
    4. End operations (logging, etc.) and turn off the circuit breakers.
    5. Document any damage/water-intrusion.

After operation

14. ReVolt and its equipment must be thoroughly rinsed with fresh water after use to avoid corrosion.
15. Reporting of incidents and accidents in NTNU's HSE system, and other reporting as required.
16. Report any damage to your professor/project-manager.
17. Maintenance of ReVolt if needed.