NTNU
Norwegian University of
Science and Technology

Dynamic Positioning for Small Autonomouse Surface Vessels

Stein-Inge Torset Øien

Master of Science in Cybernetics and Robotics
Submission date: June 2016
Supervisor: Tor Arne Johansen, ITK
Co-supervisor: Thor Inge Fossen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Abstract

The thesis seeks to prove that it is possible to implement dynamic positioning on small autonomous surface vessels by means of simple control and thrust allocation algorithms. Doing so without the need for observers and state estimation, wave filtering and detailed system identification and modeling. The thesis focuses primarily on functionality and practical implementation.

A DP control algorithm was developed, consisting of a 3-DOF nonlinear PID with a linear thrust allocation algorithm. A model of the vessel was also derived, approximated as a 3-DOF mass damper system. The DP control algorithm was tuned with pole placement based on the approximated model. Both the vessel model and control algorithm was prototyped and simulated in Matlab. A DP application, containing the DP control algoritm, was developed in Qt (C++). The application interfaces with the Telemetron vessel via TCP/IP, with a TCP client application implemented in the DP application. The functionality of the application was simulated locally before testing on the Telemetron vessel.

Three tests where completed, with focus on achieving DP and LSM objectives. The tests consisted of performing station keeping, translational and rotational maneuvers, in both a light and a moderate sea state. All objectives where accomplished when testing in a light sea state, with minor oscillations due to too aggressive proportional tuning. The test in a moderate sea state achieved the station keeping objective, but failed to perform successfull low speed maneuvers, due to thrust allocation saturation. This was due to the thrust allocation output limitations being set too low, limiting the thrusters to less than 25% of their maximum thrust output.

A re-test is proposed, with less proportional gain and increased maximum thrust output, to correct the errors in the last test in the thesis.

Samandrag

Denne avhandlingen søker å bevise at det er mogleg å implementere dynamisk posisjonering på små autonome overflatefartøyer ved bruk av enkle kontroll- og thrust allokeringsalgoritmer. Dette utan behov for tilstandsestimering, bølgefiltering, detaljert system identifikasjon og modellering, og komplekse kontroll- og thrust allokeringsalgoritmer for å oppnå dette målet. Avhandlingen fokuserer primært på funksjonalitet og praktisk gjennomføring.

Ei DP-kontrollalgoritme blei utvikla, beståande av ein 3-DOF ulineær PID regulator med en lineær thrust allokeringsalgoritme. En modell av fartøyet blei også utleda, tilnærma som eit 3-DOF masse-dempar system. DP-kontrollalgoritma blei stilt inn med polplassering, basert på den tilnærma modellen av fartøyet. Både fartøys modellen og DP-kontrollalgoritmen blei prototypa og simulert i Matlab. Ein DP-applikasjon, beståande av blant anna DP-kontrollalgoritma, blei utviklet i Qt (C++). DP-applikasjonen interfacer med Telemetron fartøyet via TCP/IP, med ein TCP-klientapplikasjon implementert i DP-applikasjonen. DP-applikasjonens funksjonalitet blei simulert lokalt før testing på Telemetron fartøyet.

Tre testar vart utført, med fokus på å oppnå DP og LSM mål. Testane bestod av å utføre stasjonær posisjonering, translasjons- og rotasjonsmanøvrer, både i lett og en moderat sjøtilstand. Alle måla vart oppnådd ved testing i lett sjø, med mindre svingninger på grunn av fór aggressiv proporsjonal innstilling. Testen i moderat sjø klarte å oppnå stasjonær posisjonering, men klarte ikke å utføre vellykka lavhastighet manøvrar, på grunn av at thrust allokeringa gikk i metning. Dette var grunna i at begrensingar i utgangen til thrust allokeringa vart satt for lavt, noko som gir ei begrensing til thrusterane på mindre enn 25 % av maksimal kraft tilgjengelig.

En ny test er foreslått, med mindre proporsjonal forsterking og auka maksimal kraft begrensing til thrustarane.

Preface

The work described in this thesis is carried out in the Department of Engineering Cybernetics at the Norwegian University of Science and Technology, during the spring of 2016, as part of the 2 year masters program Cybernetics and Robotics. The thesis is part of a larger research collaboration at NTNU, called Autosea.

I would like to thank my responsible professor and supervisor, Tor Arne Johansen, for his guidance and helpful supervision, as well as my second NTNU supervisor Tor Inge Fossen. I would also like to thank Vegard Evje Hovstein, Kenan Trnka and Thomas Ingebretsen at Maritime Robotics for all their help with making this project come to life on the Telemetron vessel.

A special thanks goes out to my loving wife, Solveig Moss Kolseth, and our darling children, Ebba and Gustave. Thank you for being there for me and giving me the time, space and support to work when it was needed the most.

Trondheim, 2016-06-01

Stein-Inge Torset Øien

Contents

List of Tables

List of Algorithms

List of Figures

Abbreviations

DP Dynamic Positioning

LSM Low Speed Maneuvering

PID Proportional-Integral-Derivate

ASV Autonomous Surface Vessel

USV Unmanned Surface Vessel

SOG Speed Over Ground

COG Course Over Ground

IMU Inertial Measurement Unit

INS Inertial Navigation System

GPS Global Positioning System

NED North-East-Down

ECEF Earth-Centered, Earth-Fixed

LLH	Latitude, Longitude and Height
DOF	Degrees of Freedom
GUI	Graphical User Interface
HMI	Human Machine Interface
TCP	Transmission Control Protocol
IP	Internet Protocol
TT	Tunnel Thruster
OE	Outboard Engine
OEA	Outboard Engine Angle
HP	Horse Power
RPM	Revolutions Per Minute
LQR	Linear Quadratic Regulator
MPC	Model Predictive Controller

Chapter 1

Introduction

This thesis seeks to solve the problem of dynamic positioning (DP) of an under actuated small autonomous surface vessel (ASV). The hypothesis is that achieving DP objectives should be possible by means of conventional PID control algorithms with linear control laws, along with GPS and compass measurements, and without the need for state observers, wave filtering or measuring any type of disturbance or external forces.

The thesis focuses primarily on physical implementation and functionality, but a simulated solution will also be derived and tested prior to physical implementation.

This chapter contains the following topics:

- Section 1.1 provides a short introduction to the background and motivation behind the project, and a brief overview of previous contributions.

- Section 1.2 presents the problem formulation and the main objectives of the project, along with approach and limitations.

- Section 1.3 presents an outline of the contents in this thesis.

1.1 Historical View and Motivation

Dynamic positioning was developed in the 1970's as a way of controlling and maintaining a fixed heading and position of a ship or surface vessel. This was done by means of conventional PID controllers in cascade with low-pass and/or notch filters to suppress the wave-induced motion components [4]. Up until now, DP has been mainly used for large offshore vessels, and lately also cruise ships, military vessels, and other large commercial and industrial vessels. DP for smaller surface vessels have, up until recently, not been given much attention. Nowadays, as software and hardware expenses related to DP for smaller vessels are decreasing, DP technology is becoming more accessible to smaller vessels, such as recreational boats. The increasing interest in DP applications for small surface vessels is the main motivator for this study, and the possibility to achieve DP control objectives with the bare minimum with regards to sensors and control algorithms needed. The main control objectives are being able to hold a fixed position and heading, as well as low speed maneuvering.

1.1.1 Previous Work

Dynamic Positioning has been researched and developed since the 70's, but up until very recently, the main focus has been for large vessels like supply ships, barges and rigs, mostly for the oil and gas industry. Although other industries have also started using DP, like ocean liners and cargo ships, the research for DP in small surface vessels have been limited.

DP systems for small surface vessels have mainly been developed for commercial use, one being the Volvo Penta EVC system for dynamic positioning[8]. The system has two stern mounted pod or azimuth thrusters located on each side of the surge axis (x-axis in the BODY frame) of the vessel. The thrusters are able to rotate 360 degrees, making the system fully actuated, as shown in figure 1.1.

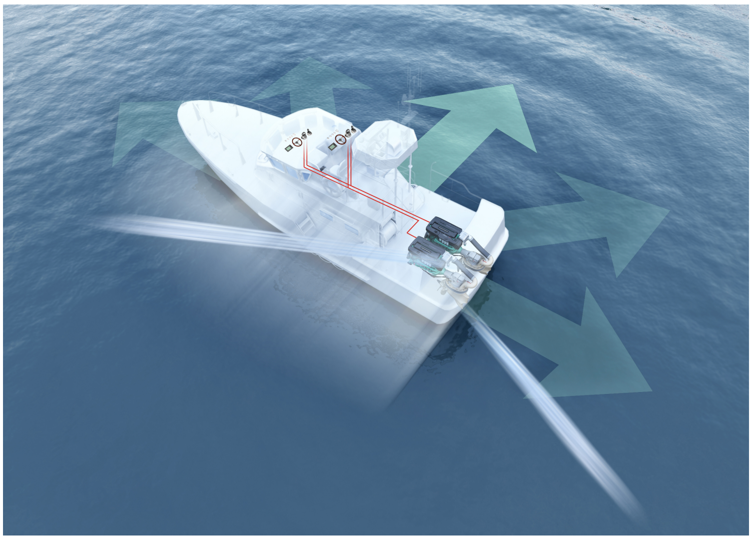


Figure 1.1: Volvo Penta EVC System - Dynamic Positioning

This thruster configuration is the only one used for dynamic positioning on small commercial vessels (based on the authors research), as it provides full actuation with only two thrusters. Another similar system is the Cummins Zeus Engine system, using pod thrusters positioned as in figure 1.1. The Cummins Zeues DP system is called *Skyhook*, and have station keeping and low speed maneuvering capabilities [3].

A master thesis has also been written on the subject of dynamic positioning of small unmanned surface vehicles, by Håvard Halvorsen here at NTNU. The master thesis looks into solving the DP problem on a fully actuated vessel of approximately the same size as the Telemetron, using LQR control laws. The thesis's focus is more theoretical, and does not move beyond hardware-in-the-loop simulation for testing the solution[5]. The main difference from the two being practical vs theoretical implementation, under actuated vs fully actuated vessel, as well as complex control laws with state estimation and wave filtering vs simple control laws and no state estimation or wave filtering. Both are however written for NTNU in collaboration with Maritime Robotics.

1.2 Problem Formulation

A dynamic positioning (DP) system for a small underactuated surface vessel shall be developed. The system shall be fully developed and tested on the vessel Telemetron in collaboration with Maritime Robotics and NTNU, as a part of the larger Autosea research project. The DP system will interface to an existing ASV system architecture, that sends GPS and compass data for position and heading feedback, and receives thruster set points to control a center stern mounted outboard engine and a center bow fixed tunnel thruster in order to control the vessel. The DP system must therefor be developed as a stand-alone application, to be run on a commercial computer (i.e laptop), interfacing with the vessel with an external connection.

1.2.1 Objectives

The main objectives of this project are to

1. Design a DP control algorithm for the Telemetron vessel, hereby named the vessel
2. Design an approximated 3-DOF numerical model of the vessel
3. Implement, Simulate and verify the DP control algorithm on the vessel model in Matlab

4. Design a stand-alone application, hereby named the DP application
5. Port the DP control algorithm and vessel model to the DP application
6. Design a GUI for the DP application for operating the DP control algorithm
7. Implement a communication scheme between the DP system and the vessel
8. Simulate and verify the DP system and communication scheme on the vessel model
9. Test and verify the communication between the DP system and the vessel
10. Test and achieve DP and LSM objectives with the DP application connected to the vessel

1.2.2 Approach

The DP algorithm will first be prototyped, tested and simulated on an approximated vessel model in Matlab. The goal of the simulation will be to achieve the basic DP control objective of station keeping, holding a fixed position and heading, when perturbed by external forces (wind, waves and current). Low speed maneuvers such as translational and rotational maneuvers will also be tested.

The DP control algorithm will then be ported to a DP application to be developed in Qt (C++) to interface with the Telemetron vessel. This DP application will consist of a GUI, the DP control algorithm, and a communication scheme for communicating with the Telemetron vessel framework. A vessel model application will also be developed for simulation purposes. The DP application's communication scheme will be tested and verified before moving on to test the application against DP control objectives.

1.2.3 Limitations

In order to implement a DP system, then control of surge, sway and yaw (3-DOF) must be possible. The Telemetron vessel has one outboard engine located at center stern, and a tunnel thruster located center bow. This means that the vessel is an underactuated vessel due to the nonholonomic constraint imposed by the outboard engine's angle limit[11]. That is, the vessel cannot accelerate in a direction outside the combined thrust vector of the outboard engine and tunnel thruster. This in turn means that there are several configurations (position and heading) that can not be achieved, given a specific range of external force vectors perturbing the vessel relative to the vessels combined thrust force vector.

An example of this is if an ocean current's velocity vector stands perpendicular to the vessels surge direction, and the vessel is required to maintain its position and heading at this time. Since the vessel can't move in the sway direction without also moving in the surge direction, only one of the two control objectives (position or heading) can be achieved.

The DP application is also constrained to rely on just position and heading feedback. This limitation is chosen specifically to see if it possible to achieve DP control objectives under these constraints, by means of simple control laws.

1.3 Structure of the Report

The rest of the report is organized as follows:

- Chapter 2 gives an introduction to the theory that serves as the basis for the work presented in the report

- Chapter 3 presents the thesis's methods and work needed to achieve the thesis objectives

- Chapter 4 presents the results made possible by work and methods presented in Chapter 3, as well as a short discussion after each results
- Chapter 5 presents a longer discussion of the results and the thesis as a whole, ending with a conclusion of the thesis along with recommendations for future work

Chapter 2

Theory

In this chapter we will look into

- Reference Frames
- Equations of Motion
- Thrust Configuration
- Control Algorithm
- Sensors

The theory and derivations in this chapter will lay the foundation for the work and methods needed to solve and meet this projects challenges and goals.

Table 2.1 below contain all the definitions and different notations used in the sections to come.

Table 2.1: Notations and definitions - 1/2

Name	Description
$\{e\}$	ECEF frame notation
$\{n\}$	NED frame notation
$\{b\}$	BODY frame notation
N_n	x position in the NED frame, $\{n\}$
E_n	y position in the NED frame, $\{n\}$
D_n	z position in the NED frame, $\{n\}$
$\dot{N}_n$	Linear velocity along the NED frame x-axis, $\{n_x\}$
$\dot{E}_n$	Linear velocity along the NED frame y-axis, $\{n_y\}$
u	Surge velocity along the BODY frame x-axis, $\{b_x\}$ (aft to fore)
v	Sway velocity along the BODY frame y-axis, $\{b_y\}$ (port to starboard)
r	Yaw rate of the BODY frame, $\{b\}$, with respect to the NED frame, $\{n\}$, expressed in the BODY frame, $\{b\}$
ψ	Yaw angle, BODY frame x-axis, $\{b_x\}$, relative to NED frame x-axis, $\{n_x\}$
$\dot{\psi}$	Yaw rate of the BODY frame, $\{b\}$, with respect to the NED frame, $\{n\}$, expressed in the inertial NED frame, $\{n\}$
X_f	Surge force along the BODY frame x-axis, $\{b_x\}$ (aft to fore)
Y_f	Sway force along the BODY frame y-axis, $\{b_y\}$ (port to starboard)
N_m	Yaw moment around the BODY frame origin, $\{b\}$
$\boldsymbol{p}^e_{b/e} = [x, y, z]^T$	Position of the BODY frame origin, $\{b\}$, with respect to the ECEF frame, $\{e\}$, expressed in the ECEF frame, $\{e\}$
$\boldsymbol{p}^n_{b/n} = [N_n, E_n, D_n]^T$	Position of the BODY frame origin, $\{b\}$, with respect to the NED frame, $\{n\}$, expressed in the inertial NED frame, $\{n\}$
$\boldsymbol{v}^b_{b/n} = [u, v]^T$	Linear velocity of the BODY frame, $\{b\}$, with respect to the NED frame, $\{n\}$, expressed in the BODY frame $\{b\}$
$\boldsymbol{\eta} = \left[N_n, E_n, \psi\right]^T$	Position and orientation of the BODY frame, $\{b\}$, with respect to the NED frame, $\{n\}$, expressed in the intertial frame $\{n\}$
$\dot{\boldsymbol{\eta}} = \left[\dot{N}_n, \dot{E}_n, \dot{\psi}\right]^T$	Linear and angular velocity of the BODY frame, $\{b\}$, with respect to the NED frame, $\{n\}$, expressed in the inertial NED frame $\{n\}$
$\boldsymbol{\nu} = [u, v, r]^T$	Linear and angular velocity of the BODY frame, $\{b\}$, with respect to the NED frame, $\{n\}$, expressed in the BODY frame $\{b\}$
$\dot{\boldsymbol{\nu}} = [\dot{u}, \dot{v}, r]^T$	Linear and angular acceleration of the BODY frame, $\{b\}$, with respect to the NED frame, $\{n\}$, expressed in the BODY frame $\{b\}$
$\boldsymbol{\tau} = \left[X_f, Y_f, N_m\right]^T$	Force and moment vector given in the BODY frame, $\{b\}$

Table 2.2: Notations and definitions - 2/2

Name	Description
l	Longitude
μ	Latitude
$\boldsymbol{R}_b^n(\psi)$	Rotation matrix from BODY to NED frame, $\mathbb{R}^{2x2}$
$\boldsymbol{J}_b^n(\psi)$	Transformation matrix from BODY to NED frame, $\mathbb{R}^{3x3}$
$\boldsymbol{\Theta}_{en} = [l, \mu]^T$	Longitude and latitude vector
$\boldsymbol{R}_e^n(\boldsymbol{\Theta}_{en})$	Rotation matrix from ECEF to NED frame
l_1	Distance from CR to tunnel thruster
l_2	Distance from CR to outboard engine
T_{2x}	Outboard engine surge force
T_{2y}	Outboard engine sway force
T_{1y}	Tunnel thruster sway force
f_1	Tunnel thruster force magnitude
f_2	Outboard engine force magnitude
α	Outboard engine angle, relative to the body frame

2.1 Reference Frames

In navigation, a reference frame is predefined coordinate system of either two or three dimensions that is used to define the motion of said coordinate system relative to another coordinate system. In navigation, these coordinate systems are often fixed to the navigated objects and to the world the object is navigating in. These reference frames then form a hierarchy in which one frame has a relation to the next frame via rotational and translational transformations.

2.1.1 ECEF, NED and BODY

For a surface vessel, we define three frames of reference in order to navigate by means of GPS and compass data.

ECEF

The first reference frame, which is the parent frame in our previously mentioned hierarchy, we have the *ECEF* frame, denoted as $\{e\}$, which has its origin fixed at Earth's center, thus spinning along with its rotation. The *z-axis* points north, the *x-axis* point to where the prime meridian at 0° longitude intersects with the equator at 0° latitude, and the *y-axis* completes the right hand rule for coordinate frames and stands perpendicular to the *xz-plane*.

NED

The second frame is called the *NED* frame, denoted as $\{n\}$, which in our case will be fixed at the starting point of our navigation with its *x-axis* pointing towards true north, the *z-axis* points down toward earth's center, and the *y-axis* points east, following the right hand rule. The NED frame then acts as a local inertial frame of reference in which we are navigating in.

BODY

The third frame is called the *BODY* frame, denoted as $\{b\}$, where the origin of the frame is located at the center of the vessel body. The BODY frame uses the same with the *x-axis* pointing in the forward surge direction, the *z-axis* points downward into the water, and the *y-axis* points in the sideways sway direction (portside of the vessel), thus completing the right hand rule. The frame moves with the rigid body of the vessel, and when the vessel points towards true north, the BODY frame is parallel to the NED frame.

In dynamic positioning, we are only interested in controlling the motion in surge, sway and yaw. This means that we operate in a two dimensional workspace. Therefor, when expressing reference frames with the x- and *y-axis*, the missing *z-axis* would then follow regular right hand rule for coordinate frames, pointing into the image, and any yaw rotation, ψ, will then follow the same rule for positive rotation about said axis. In contrast to conventional representation of the *xy-plane*, the *x-axis* stands vertical and the *y-axis* horizontal, giving us a "from above" two dimensional view of the system like the north-east representation of a map, as presented in figure 2.1 below.

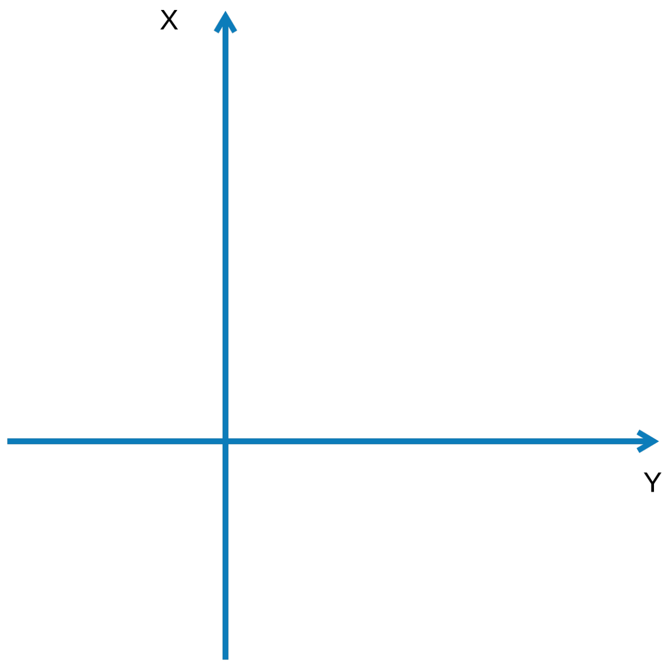


Figure 2.1: xy-plane in the NED frame convention

2.1.2 Transformations

With our frames of reference defined, we can then go on to defining the transformations between these frames and how they relate to DP and navigation by GPS and compass. A transformation between coordinates is usually done by means of translation and rotation of one frame relative to another. These can be achieved by composing the translation and rotation as vectors and matrices, which then simplifies the arithmetic involved in defining and calculating these transformations.

Rotation Matrix

A basic rotation can be seen as a rotation about the axis of a coordinate system, in our case about the *z-axis* of the NED frame, following the right hand rule for positive rotation. This rotation can be described by a rotation matrix[12].

The rotation matrix in general is an element of the *special orthogonal group of order 3*, which is defined as [11]

$$\mathrm{SO}(3) = \left\{\boldsymbol{R} | \boldsymbol{R}^T \in \mathbb{R}^{3\times3}, \quad \boldsymbol{R} \text{ is orthogonal and } det(\boldsymbol{R}) = 1\right\} \tag{2.1}$$

which again is a subset of the *orthogonal group of order 3* which is defined as

$$\mathrm{O}(3) = \left\{\boldsymbol{R} | \boldsymbol{R}^T \in \mathbb{R}^{3\times3}, \quad \boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}\right\} \tag{2.2}$$

We can then define the two rotation matrices needed to go from ECEF to NED to BODY frame and back, by means of latitude, longitude and yaw angles.

LLH to ECEF

Earth is not perfectly spherical, so when transforming latitude, longitude and height to ECEF coordinates, we have to use approximations of its shape to achieve accurate enough results that can be used for navigation. The most common way of doing this is by using the WGS-84 standard, which gives a very good approximation of earth's curvature by defining it as an ellipse with a specific set of parameters. These parameters are defined as follows

Table 2.3: WGS-84 parameters

Name	Description
$r_e = 6378137$	Equatorial radius of ellipsoid (semi-major axis)
$r_p = 6356752$	Polar axis radius of ellipsoid (semi-minor axis)
$e = 0.08181979099211$	Eccentricity of ellipsoid
h_e	Ellipsoidal height from vessel to ellipsoid surface
N_e	Radius of curvature in the prime vertical

From table 2.3 we can calculate the radius of curvature in the prime vertical, N_e, as

$$N_e = {r_e}^2 \sqrt{\left(r_e \cos(\mu)\right)^2 + \left(r_p \sin(\mu)\right)^2} \tag{2.3}$$

, which can in turn be used to calculate the ECEF coordinates, $\boldsymbol{p}^e_{b/e}$, as

$$\boldsymbol{p}^e_{b/e} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (N_e + h_e)\cos(\mu)\cos(l) \\ (N_e + h_e)\cos(\mu)\sin(l) \\ N_e\left(\left(\frac{r_p}{r_e}\right)^2 + h_e\right)\sin(\mu) \end{bmatrix} \tag{2.4}$$

ECEF to NED

ECEF is a coordinate system that is fixed to the earth, rotating along with it, at an origin at the center mass of the earth. *ECEF* has the z-axis pointing north, and the x-axis intersects the sphere of the earth at zero degree latitude and longitude, and the y-axis completes the right hand rule.

As we can see from section 2.1.2, the longitude, l, and latitude, μ, can then be used to calculate the position in *ECEF* coordinates, which again can be transformed into *NED* frame coordinates by using the rotation matrix, $\boldsymbol{R}_e^n$. By first defining

$$\boldsymbol{R}_{y,-\mu-\frac{\pi}{2}} = \begin{bmatrix} \cos(\mu+\frac{\pi}{2}) & 0 & -\sin(\mu+\frac{\pi}{2}) \\ 0 & 1 & 0 \\ \sin(\mu+\frac{\pi}{2}) & 0 & \cos(\mu+\frac{\pi}{2}) \end{bmatrix} \tag{2.5}$$

$$= \begin{bmatrix} -\sin(\mu) & 0 & -\cos(\mu) \\ 0 & 1 & 0 \\ \cos(\mu) & 0 & -\sin(\mu) \end{bmatrix} \tag{2.6}$$

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

, we can then define $\boldsymbol{R}_e^n$ from

$$\boldsymbol{R}_n^e = \boldsymbol{R}_{z,l}\boldsymbol{R}_{y,-\mu-\frac{\pi}{2}} = \begin{bmatrix} -\sin(\mu)\cos(l) & -\sin(l) & -\cos(\mu)\cos(l) \\ \sin(\mu)\sin(l) & \cos(l) & -\cos(\mu)\sin(l) \\ \cos(\mu) & 0 & -\sin(\mu) \end{bmatrix} \tag{2.8}$$

$$\boldsymbol{R}_e^n = \left(\boldsymbol{R}_n^e\right)^T = \begin{bmatrix} -\sin(\mu)\cos(l) & \sin(\mu)\sin(l) & \cos(\mu) \\ -\sin(l) & \cos(l) & 0 \\ -\cos(\mu)\cos(l) & -\cos(\mu)\sin(l) & -\sin(\mu) \end{bmatrix} \tag{2.9}$$

We can then transform the position from *ECEF* to *NED*

$$\boldsymbol{p}_{b,n}^n = \boldsymbol{R}_e^n(\boldsymbol{p}_{b,e}^e - \boldsymbol{p}_{b,e_0}^e) \tag{2.10}$$

where $\boldsymbol{p}_{b,e_0}^e$ is the position origin expressed in *ECEF* coordinates.

2.1.3 BODY to NED

We can transform vectors and coordinates from BODY to NED frame or vice-versa, by using a rotation matrix composed of the yaw, ψ, rotation. Since the *z-axis* of both frames are in parallel (in a 3-DOF approximation), we can then define the rotation matrix, $\boldsymbol{J}_b^n(\psi)$, as

$$\boldsymbol{J}_b^n(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

, and by following the rules for matrix arithmetic, we can simply transpose this rotation matrix to represent a rotation of the NED frame relative to the BODY frame as

$$\left(\boldsymbol{J}_b^n(\psi)\right)^T = \boldsymbol{J}_n^b(\psi) \tag{2.12}$$

With this we can, for instance, transform the linear and angular velocity given in the BODY frame, $\boldsymbol{\nu}$, to the linear and angular velocity given in the NED frame, $\dot{\boldsymbol{\eta}}$

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_b^n(\psi)\boldsymbol{\nu} \tag{2.13}$$

and vice-versa

$$\boldsymbol{\nu} = \boldsymbol{J}_n^b(\psi)\dot{\boldsymbol{\eta}} \tag{2.14}$$

The above equations are known as the kinematic equations of motion for DP, leaving only the kinetic equations of motion to describe the full vessel dynamics in 3-DOF, which will be covered in section 2.2. The relation between the two frames can be seen in the figure below.

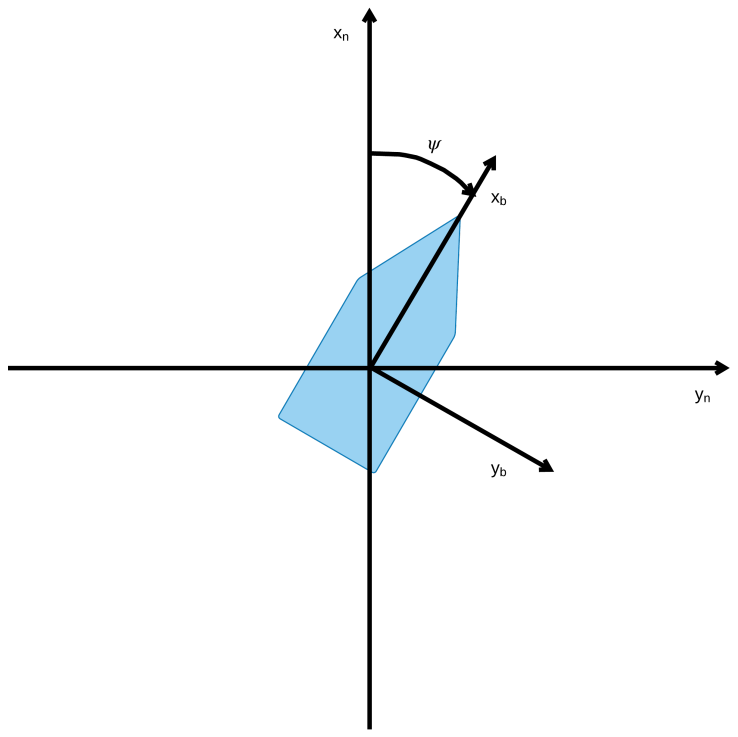


Figure 2.2: BODY frame and inertial NED frame

2.1.4 ECEF to LLH

When going back from ECEF coordinates to LLH coordinates, we can solve longitude directly, but latitude and height are given implicitly from the equations presented in section 2.1.2, given as

$$l = \arctan(\frac{y}{x}) \tag{2.15}$$

$$\mu = \arctan\left(\frac{z}{p}\left(1 - e^2\frac{N}{N+h_e}\right)^{-1}\right) \tag{2.16}$$

$$h_e = \frac{p}{\cos(\mu)} - N \tag{2.17}$$

where e is given as

$$e = \sqrt{1 - \left(\frac{r_p}{r_e}\right)^2} \tag{2.18}$$

We can solve these iteratively by using an algorithm presented in [7], defined as

Algorithm 2.1: Transform ECEF coordinates to LLH

#1	Compute p	$p = \sqrt{x^2 + y^2}$
#2	Compute the approximate value, $\mu_{(0)}$	$\mu_{(0)} = \arctan\left(\frac{z}{p}\left(1-e^2\right)^{-1}\right)$
#3	Compute an approximate value, $N_{e(0)}$	$N_{e(0)} = {r_e}^2\sqrt{\left(r_e\cos(\mu_{(0)})\right)^2 + \left(r_p\sin(\mu_{(0)})\right)^2}$
#4	Compute the ellipsoidal height, h_e	$h_e = \frac{p}{\cos(\mu_{(0)})} - N_{e(0)}$
#5	Compute an improved value for the latitude, μ	$\mu = \arctan\left(\frac{z}{p}\left(1 - e^2\frac{N_{e(0)}}{N_{e(0)}+h_e}\right)^{-1}\right)$

#6	Check for another iteration step	`if` $\lVert\mu - \mu_{(0)}\rVert < tol$ (where tol is a small number
		    `goto END algorithm`
		`else`
		    `goto step #3`

2.2 Vessel Model

In dynamic positioning, we are only interested in controlling the surge, sway and yaw motions. This gives us three degrees of freedom in a two dimensional workspace. Therefor, when visualizing two dimensional reference frames with the *x*- and *y-axis*, the "missing" *z-axis* would then follow regular right hand rule and any yaw rotation will then follow the same rule for positive rotation about this axis, as can be seen in figure 2.2 above.

2.2.1 Equations of Motion

For simulation purposes, an approximated model of the vessel is derived in order to test the DP control system. The model consist of the kinematic and kinetic equations of motion, with the former having been defined in equation 2.13.

Since the model is only to be used for simulation and verification of the functionalities of the DP system, a linearized DP model has been considered to be more than sufficient for these purposes[4].

A linearized DP model for 3-DOF is basically a mass damper system with linearized viscous damping[4]. The kinetic equations for such a model can, from [4], be defined as

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{D}\boldsymbol{\nu} = \boldsymbol{\tau} \tag{2.19}$$

We can then write the complete equations of motion as

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_b^n(\psi)\boldsymbol{\nu} \tag{2.20}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{D}\boldsymbol{\nu} = \boldsymbol{\tau} \tag{2.21}$$

We can then augment this model to account for external forces, such as wind, current and waves, defined as

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_b^n(\psi)\boldsymbol{\nu} \tag{2.22}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{D}(\boldsymbol{\nu} - \boldsymbol{\nu}_c) = \boldsymbol{\tau} + \boldsymbol{\tau}_{ext} \tag{2.23}$$

, where $\boldsymbol{\nu}_c$ is defined as the current velocity with respect to the BODY frame, and $\boldsymbol{\tau}_{ext}$ are the sum of external forces combined for wind and waves. This leaves only the actuator models and thrust configuration, expressed as $\boldsymbol{\tau}$, and the model can be used for simulating and testing the solution.

2.2.2 Thrust Configuration

The thrust configuration of the vessel is derived from the forces and moments from the equations of motion, which is defined as

$$\boldsymbol{\tau} = \begin{bmatrix} X_f \\ Y_f \\ N_m \end{bmatrix} \tag{2.24}$$

From figure 2.3, 2.4 and 2.5 below, we can see that the tunnel thruster provides sway force and angular yaw momentum, and the outboard engine, as a function of its angle relative to the BODY frame, provides surge and sway force as well as angular yaw momentum.

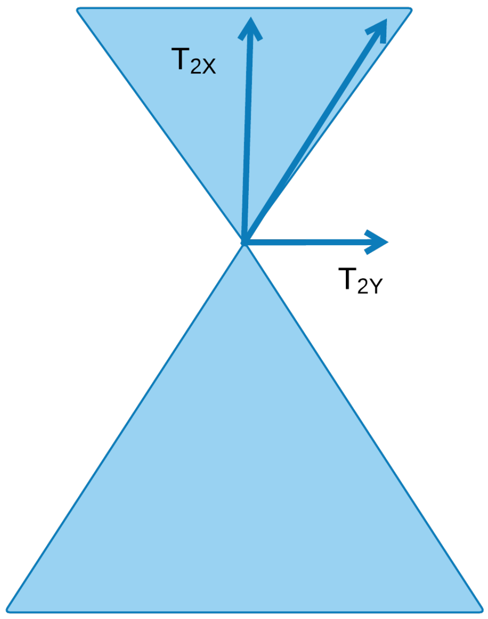


Figure 2.3: Thrust region of outboard engine

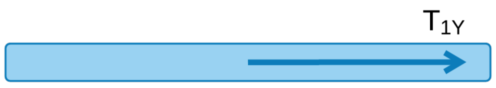


Figure 2.4: Thrust region for tunnel thruster

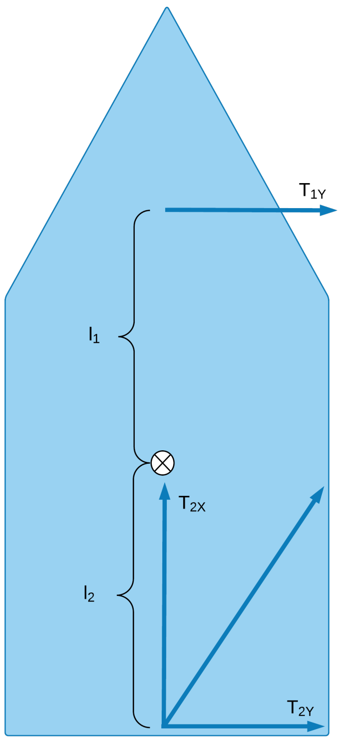


Figure 2.5: Vessel thrust vectors

The angular yaw momentum is a product of the sway forces, T_{1y} and T_{2y}, and the moment arms, l_1 and l_2, from the vessels center of rotation, as seen in figure 2.5. From this we can define the 3-DOF torque, $\boldsymbol{\tau}$, as

$$\boldsymbol{\tau} = \begin{bmatrix} f_2 \cos(\alpha) \\ f_2 \sin(\alpha) + f_1 \\ f_1 l_1 + f_2 l_2 \sin(\alpha) \end{bmatrix} = \begin{bmatrix} T_{2x} \\ T_{2y} + T_{1y} \\ T_{1y} l_1 + T_{2y} l_2 \end{bmatrix} \tag{2.25}$$

2.3 Control Algorithm

The usual objective of a control algorithm is to control a system (i.e a boat), so that its output value (states) equals a desired value (reference). There are several control algorithms that achieve this objective, but the most common one is the PID controller [1].

2.3.1 PID

The purpose of the PID algorithm is to drive the difference between a reference state, x_r, and the current state of a system (vessel), x, towards zero. The PID algorithm continuously calculates an error value as the difference between the desired reference state and the system's current state, given as

$$e(t) = x_r(t) - x(t) \tag{2.26}$$

This current state is typically given to us from sensor readings, i.e GPS or compass. The algorithm then tries to minimize this error state over time by perturbing the system by changing the value of a control element, i.e engine torque or rudder angle. The size of this control signal, u, is calculated from the weighted sum

$$\tau_{pid}(t) = K_p e(t) + K_d \frac{d}{dt} e(t) + K_i \int_{t_0}^{t} e(t) dt \tag{2.27}$$

which is then discretized in order to be executed on a computer, given as

$$\tau_{pid}(k) = K_p e(k) + K_d \frac{1}{\Delta_k} (e(k) - e(k-1)) + K_i \sum_{k=0}^{k} e(k) \Delta_k \tag{2.28}$$

where Δ_k is the timestep between each increment, k, of the algorithm.

K_p, K_d and K_i, are known as the proportional, derivate and integral control gains, respectively, where

- K_p accounts for present values of the error. For example, if the error is large and positive, the control output will also be large and positive.
- K_d accounts for possible future values of the error, based on its current rate of change.
- K_i accounts for past values of the error. For example, if the current output is not sufficiently strong, error will accumulate over time, and the controller will respond by applying a stronger action. This is to remove steady state error in the system.

As a PID controller relies only on the measured process variable, not on knowledge of the underlying process, it is easy to implement, given that the PID gains are tuned properly, which is the biggest challenge. It is possible to tune it purely based on heuristic methods like Zeigler-Nichols by trial and error, or by using a mathematical and model based approach, such as pole placement[1].

2.3.2 Dynamic Positioning

Dynamic positioning is a specific set of control objectives for a control algorithm, a so called DP algorithm. There are several DP control objectives, of which some require a more complex DP control algorithms and added sensors to work. The most common objectives are

- Hold a fixed position and heading above the sea floor
- Hold a fixed heading

- Hold a fixed distance and angle to another vessel/object
- Positioning relative the combined weather force vector, called weathervaning
- Low speed maneuvering from a point to another

with the first objective being the primary objective for most applications. We can achieve this objective by augmenting the PID algorithm shown in the previous section.This is done by replacing x with $\boldsymbol{\eta}$, where $\boldsymbol{\eta}_e = \boldsymbol{\eta} - \boldsymbol{\eta}_r$. We can then write it as

$$\boldsymbol{\tau}_{pid}(k) = -\left(\boldsymbol{K_p}\boldsymbol{\eta}_e(k) + \boldsymbol{K_d}\frac{1}{\Delta_k}\left(\boldsymbol{\eta}_e(k) - \boldsymbol{\eta}_e(k-1)\right) + \boldsymbol{K_i}\sum_{k=0}^{k}\boldsymbol{\eta}_e(k)\Delta_k\right) \tag{2.29}$$

However, by looking at the vessel's equations of motion from section 2.2, we see that the vessel is controlled through the force vector $\boldsymbol{\tau}$ in the kinetic equation. This means that in order to use the PID algorithm above to control the vessel, we first have to transform the output from NED to the BODY frame, in which the kinetic equation is represented. By using the rotation matrix, $\boldsymbol{J}_b^n(\psi)$, shown in section 2.1.3, we can write this as

$$\boldsymbol{\tau}(k) = \boldsymbol{J}_b^n(\psi(k))^T\boldsymbol{\tau}_{pid}(k) \tag{2.30}$$

, which makes our PID nonlinear, and is then defined as a nonlinear PID algorithm, as presented in [4].

2.4 Sensors

In surface vessel navigation, the most commonly used sensors for measuring the state of a vessel are the compass and gps, but also inertial sensors, such as an IMU is used. For our vessel, only the two former are used.

2.4.1 Compass

The magnetic compass, which is strapped down to the vessel body, is a sensor that measures earths magnetic field (and other magnetic field sources), which can be used to determine the angle relative to magnetic north. The magnetic compass carries with it some noise, and the output angle can therefor be defined as

$$\psi^n_{\text{cmp}} = \psi^n_{b/n} + w_{\text{cmp}_\psi} \tag{2.31}$$

where w_{cmp_ψ} is Gaussian white noise.

2.4.2 GPS

The GPS outputs height, and latitude and longitude angles with respect to the ECEF frame, as explained in section 2.1.2. This gives us

$$\boldsymbol{\Theta}^n_{\text{gps}} = \boldsymbol{\Theta}_{en} + \boldsymbol{w}_{\text{gps}_\Theta} \tag{2.32}$$

$$h_{gps} = h_e + w_{gps,h} \tag{2.33}$$

where $\boldsymbol{w}_{\text{gps}_\Theta}$ is Gaussian white noise.

Chapter 3

Methods

This chapter will give an insight to the methods and work that has been done in this project, based on the theory and goals of the prior chapters.

The chapter is divided in to four sections:

- DP system
  - Vessel Model
  - DP control algorithm
  - Graphical User Interface
  - Communication
- Simulation
- Experiment

All parameters and values in this chapter can be found in appendix A.2 and in the digital file appendix B.

3.1 Vessel Model

To derive an accurate model of a vessel, there is a need for system identification to find the physical parameters that describes the system via a set of equations of motion. The Telemetron Vessel has however not been parametrized in such a way, nor is the focus of this thesis to accurately identify a dynamic model of the vessel. For this reason, and to save time, the model parameters of a similar vessel, the *Viknes 830* , will be used as a base for the model of the Telemetron vessel[5][9].

Although there isn't an accurate model of the Telemetron vessel, some parameters are available to us. We have the mass, length from stern to bow, beam lenght, draft height and position of the actuators relative to the vessels origin. By augmenting the Viknes model with these parameters, we end up with a sufficient approximation of the Telemetron vessel, which will serve as model for simulation and initial tuning of the DP control algorithm.

3.1.1 Linearized DP Model

The linearized model presented in equation 2.19 consist of two 3×3 matrices, $\boldsymbol{M}$ and $\boldsymbol{D}$, which describe the system dynamics. From the maneuvering theory presented in [4], we know that the $\boldsymbol{M}$ matrix represents the mass and inertia of the system which includes not only the vessel but also added mass from the body of water it sits in. These are the rigid-body mass matrix, $\boldsymbol{M}_{RB}^{CO}$, and the hydrodynamic system inertia matrix, $\boldsymbol{M}_{A}^{CO}$, both given with respect to the center of origin, *CO*. We can write this as

$$\boldsymbol{M} = \boldsymbol{M}_{RB}^{CO} + \boldsymbol{M}_{A}^{CO} \tag{3.1}$$

These matrices,$\boldsymbol{M}_{RB}^{CO}$ and $\boldsymbol{M}_{A}^{CO}$, are constructed by transforming them from their definition in the center of gravity, *CG*, and center of buoyancy, *CB*, respectively. These are defined as

$$\boldsymbol{M}_{RB}^{CG} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I_z^{cg} \end{bmatrix}, \quad \boldsymbol{M}_{A}^{CB} = -\begin{bmatrix} X_{\dot{u}} & 0 & 0 \\ 0 & Y_{\dot{v}} & 0 \\ 0 & 0 & N_{\dot{r}} \end{bmatrix} \tag{3.2}$$

This is done by pre and post multiplying them with the transformation matrices, $\boldsymbol{H}(\boldsymbol{r}_{cg}^b)$ and $\boldsymbol{H}(\boldsymbol{r}_{cg}^b)$, where $\boldsymbol{r}_{cg}^b$ is the translation vector going from CO to the CG, and $\boldsymbol{r}_{cb}^b$ is the translation vector going from CO to the CB. These are defined as

$$\boldsymbol{r}_{cg}^b = \begin{bmatrix} x_g \\ y_g \\ z_g \end{bmatrix}, \quad \boldsymbol{r}_{cb}^b = \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} \tag{3.3}$$

The transformation matrices are then defined as

$$\boldsymbol{H}(\boldsymbol{r}_{cg}^b) = \begin{bmatrix} 1 & 0 & -y_g \\ 0 & 1 & x_g \\ 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{H}(\boldsymbol{r}_{cb}^b) = \begin{bmatrix} 1 & 0 & -y_b \\ 0 & 1 & x_b \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

The transformations from CG and CB can then be written as

$$\boldsymbol{M}_{RB}^{CO} = \boldsymbol{H}(\boldsymbol{r}_{cg}^b)^T \boldsymbol{M}_{RB}^{CG} \boldsymbol{H}(\boldsymbol{r}_{cg}^b) \tag{3.5}$$

$$\boldsymbol{M}_{A}^{CO} = \boldsymbol{H}(\boldsymbol{r}_{cb}^b)^T \boldsymbol{M}_{A}^{CB} \boldsymbol{H}(\boldsymbol{r}_{cb}^b) \tag{3.6}$$

which then give us

$$\boldsymbol{M}_{RB}^{CO} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & m x_g \\ 0 & m x_g & I_z^{cg} + m {x_g}^2 \end{bmatrix}, \quad \boldsymbol{M}_{A}^{CO} = -\begin{bmatrix} X_{\dot{u}} & 0 & 0 \\ 0 & Y_{\dot{v}} & Y_{\dot{r}} \\ 0 & Y_{\dot{r}} & N_{\dot{r}} \end{bmatrix} \tag{3.7}$$

Since I_z^{cg} is not known, a good approximation of the term $I_z^{cg} + m{x_g}^2$ can be found from the following equation

$$I_z^{cg} + m{x_g}^2 = I_z^{co} = m(0.25L_{pp})^2 \tag{3.8}$$

, where L_{pp} is the length from stern to bow. This approximation comes from the hydrostatic theory presented in [4].

The elements in $\boldsymbol{M}_A^{CO}$ are known as the hydrodynamic derivatives, where ,for instance, $Y_{\dot{r}}$ is the added mass force, Y, along the y-axis due to an angular acceleration, $\dot{r}$, around the z-axis. It is also known as the hydrodynamic system inertia matrix or matrix of added mass and moments. This added mass is due to when a vessel is moving through water, it has to "push" the water out of the way when moving through it, since the vessel and the water cannot occupy the same volume at the same time [4]. The elements in this matrix are based on the Viknes model parametrization, modified with Telemetron parameters.

The $\boldsymbol{D}$ matrix is the linear viscous damping matrix due to potential damping and possible skin friction. This matrix, according to seakeeping theory in [4], is defined in CB as

$$\boldsymbol{D}^{CB} = \begin{bmatrix} \frac{m+A_{11}(0)}{T_{\text{surge}}} & 0 & 0 \\ 0 & \frac{m+A_{22}(0)}{T_{\text{sway}}} & 0 \\ 0 & 0 & \frac{I_z+A_{66}(0)}{T_{\text{yaw}}} \end{bmatrix} \tag{3.9}$$

We can transform it to the CO by multiplying it with the transformation matrix, $\boldsymbol{H}(\boldsymbol{r}_{cb}^b)$, written as

$$\boldsymbol{D}^{CO} = \boldsymbol{H}(\boldsymbol{r}_{cb}^b)^T \boldsymbol{D}^{CB} \boldsymbol{H}(\boldsymbol{r}_{cb}^b) \tag{3.10}$$

This results in

$$\boldsymbol{D}^{CO} = -\begin{bmatrix} X_u & 0 & 0 \\ 0 & Y_v & Y_r \\ 0 & N_v & N_r \end{bmatrix} = \begin{bmatrix} \frac{m+A_{11}(0)}{T_{\text{surge}}} & 0 & 0 \\ 0 & \frac{m+A_{22}(0)}{T_{\text{sway}}} & x_b\frac{m+A_{22}(0)}{T_{\text{sway}}} \\ 0 & x_b\frac{m+A_{22}(0)}{T_{\text{sway}}} & \frac{{x_b}^2[m+A_{66}(0)]}{T_{\text{sway}}} + \frac{{x_b}^2[m+A_{22}(0)]+I_z+A_{66}(0)}{T_{yaw}} \end{bmatrix} \tag{3.11}$$

Our linearized DP model can then be defined as

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_b^n(\psi)\boldsymbol{\nu} \tag{3.12}$$

$$\boldsymbol{M}_{RB}^{CO}\dot{\boldsymbol{\nu}} + \boldsymbol{D}^{CO}\boldsymbol{\nu} = \boldsymbol{\tau} \tag{3.13}$$

The time constants, T_{surge}, T_{sway} and T_{yaw} are based on the Viknes vessel model parameters, as well as $\boldsymbol{r}_g$ and $\boldsymbol{r}_b$.

Environmental Forces

Since the vessel model is an approximation of the real vessel, so will the environmental forces be modeled by either linear or constant perturbation. The vessel can't compensate for sudden wind gusts, and we will therefor ad together the ocean current and wind into one combined force expressed as a velocity given in the body frame, $\boldsymbol{\nu}_{c,w}$. Waves can be modeled with complex wave spectrums of many orders, but for an approximated model on such a small vessel, it is sufficient to model it as a sinusoidal change in position given in the NED frame, defined as $\boldsymbol{\omega}_w$. From[13] and [6], we see that average wave period within feasible weather conditions for the DP control algorithm is roughly $1 < T_w < 6$ seconds.

The vessel model is then augmented with environmental forces, given as

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_b^n(\psi)\boldsymbol{\nu} + \boldsymbol{\omega}_w \tag{3.14}$$

$$\boldsymbol{M}_{RB}^{CO}\dot{\boldsymbol{\nu}} + \boldsymbol{D}^{CO}\left(\boldsymbol{\nu} - \boldsymbol{\nu}_{c,w}\right) = \boldsymbol{\tau} \tag{3.15}$$

For larger vessels (i.e supply vessel, mariners, cruisers, tanker), it is not possible to compensate directly for the sudden change in position that a wave imposes. And for waves up to a given size, the waves will not move the ship directly but instead have an integral effect, similar to how ocean currents perturb the vessel [4], due to the ship's large inertia. However, For a smaller vessel, such as the Telemetron, even relatively small waves (larger than 0.5 meter in amplitude) will cause oscillatory changes in the vessels position, and will therefor have an effect on the position data that is fed back to the controller.

An objective will then be to see if the vessel can compensate for these changes, or if the vessel and controller dynamics will naturally filter them out, or if a filter needs to be implemented.

3.1.2 Actuator Dynamics

The dynamics of for both electric and combustion engines are nonlinear. For modeling and simulation, according to [4], these dynamics can be approximated with a first order model. The dynamics is based on the difference between the actuator set points and its current state, defined as

$$\dot{u}_i = -\frac{1}{T_i}\left(u_i - u_{i,r}\right) \tag{3.16}$$

, where u_i is the actuator state, $u_{i,r}$ is the actuator set point, and T_i is the actuator time constant, describing its dynamics. Discretized, this becomes

$$u_i(k+1) = u_i(k) - h\frac{1}{T_i}\left[u_i(k) - u_{i,r}(k)\right] \tag{3.17}$$

where h is the timestep in seconds. We then choose a time constant $T_i, i \in 1,2,3$ for our three actuators. These time constants are based on the actuator dynamics derived for the Viknes vessel[5] actuators (stern outboard engine and bow tunnel thruster).

Outboard engines also have to change gears when going from forward to reverse thrust and vice versa. This dynamic can be modeled by checking the sign of the actuator state, implementing a delay, t_{gear} whenever the state goes below the minimum RPM in both directions, and set a gear change to either neutral or forward/backward accordingly. This delay is based on empirical results provided by Maritime Robotics, defined as

$$t_{\text{gear}} = 3, \quad \text{(gear change timedelay, sec)} \tag{3.18}$$

, which means every time there is a gear change (thrust allocated from forward to reverse in surge or vise versa), there is a 3 second delay before the actuator starts producing thrust in the new gear direction.

The tunnel thruster also changes thrust direction from starboard to port and vice versa, but as it is a proportional electric engine, it doesn't have any gears, only a shift in voltage or phase to make the engine change thrust direction, which is negligible.

The outboard engine angle is also constrained to $\alpha = \pm\frac{2\pi}{9}$, which equals ±40 degrees, causing the vessel to be underactuated. This causes a problem when, for instance, the external force vector (wind, current, waves) stands perpendicular to the vessels x-axis given in the BODY frame. In order to hold a fixed position and angle, the vessel would then have to "zig zag" back and forth, which with our thruster configuration, means that everytime it "zigz" or "zags" the outboard engine has to, not only change gear, but also turn the angle from $\alpha = +40$ degrees to $\alpha = -40$ degrees, and vice versa. This process, combined with the gear change will ultimately give a very slow response, and it can therefor be hard to accomplish if the external forces are large enough. More on this in the chapter 4.

The combined actuator dynamics vector, $\boldsymbol{u}$, for the Telemetron vessel is defined as

$$\boldsymbol{u} := \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} \text{Tunnel thruster thrust value, given in RPM} \\ \text{Outboard engine thrust value, given in RPM} \\ \text{Outboard engine angle, } \alpha \text{, given in degrees} \end{bmatrix} \tag{3.19}$$

The thrust force produced by the thrusters have a quadratic relationship in RPM, and is defined as

$$F_{TT} = k_1 u_1 |u_1| \tag{3.20}$$

$$F_{OE} = k_2 u_2 |u_2| \tag{3.21}$$

$$\tag{3.22}$$

where F_{TT} is the tunnel thruster force, F_{OE} is the outboard engine force, and k_1 and k_2 are the respective thruster coefficients. These values are taken from the Viknes actuator parameters[5], but modified to account for the difference in HP (Viknes's 140 HP vs Telemetron's 225 HP).

The thruster forces can then be transformed into BODY frame coordinate forces as

$$\boldsymbol{\tau} = \begin{bmatrix} F_{OE}\cos(\alpha) \\ F_{TT} + F_{OE}\sin(\alpha) \\ l_1 F_{TT} + l_2 F_{OE}\sin(\alpha) \end{bmatrix} \tag{3.23}$$

where l_1 and l_2 are the distance from CO to the tunnel thruster and outboard engine position, respectively.

With this, the vessel model is ready for simulation purposes, and the implementations in both *Matlab* and *C++* can be found in the digital file appendix.

3.2 DP Control Algorithm

The nonlinear PID algorithm presented in equation 2.30 is discretized and is therefor ready to be implemented for running on a computer/processor. However, when programming the algorithm, the equation has to be divided up and executed in sections.

3.2.1 Nonlinear PID

The algorithm is then produced from the discretized nonlinear PID, presented in equation 2.30 as

$$\boldsymbol{\tau}(k) = -\boldsymbol{J}_b^n(\psi)^T \left(\boldsymbol{K_p}\boldsymbol{\eta}_e(k) + \boldsymbol{K_d}\frac{1}{h}\big(\boldsymbol{\eta}_e(k) - \boldsymbol{\eta}_e(k-1)\big) + \boldsymbol{K_i}\sum_{k=0}^{k}\boldsymbol{\eta}_e(k)h \right) \tag{3.24}$$

where h is the timestep between each iteration, k, of the equation.

By first defining the integral term as

$$\boldsymbol{I}_{\eta_e}(k) = \sum_{k=0}^{k}\boldsymbol{\eta}_e(k)h \tag{3.25}$$

, we can then define the discrete nonlinear PID algorithm as

Algorithm 3.1: Discrete nonlinear PID Algorithm

```
#0  Initialize algorithm                              $k = 0$ , $\boldsymbol{I}_{\eta_e}(k) = \boldsymbol{I}_{\eta_e}(k-1) = [0,0,0]$ ,
                                                      $\boldsymbol{\eta}_e(k) = \boldsymbol{\eta}_e(k-1) = [0,0,0]^T$
#1  Compute $\boldsymbol{\eta}_e$                     $\boldsymbol{\eta}_e(k) = \boldsymbol{\eta}(k) - \boldsymbol{\eta}_r(k)$
#2  Compute $\dot{\boldsymbol{\eta}}_{\boldsymbol{e}}$ by numerical differentiation    $\dot{\boldsymbol{\eta}}_{\boldsymbol{e}} = \frac{1}{h}\left(\boldsymbol{\eta}_e(k) - \boldsymbol{\eta}_e(k-1)\right)$
#3  Compute integral term, $\boldsymbol{I}_{\eta_e}(k)$    $\boldsymbol{I}_{\eta_e}(k) = \boldsymbol{I}_{\eta_e}(k-1) + h\boldsymbol{\eta}_e(k)$
#4  Compute the rotation matrix, $\boldsymbol{J}_b^n(\psi(k)$    $\psi(k)) = \boldsymbol{\eta}[\mathbf{3}](k)$,  $\boldsymbol{J}_b^n(\psi(k))$
#5  Compute the PID output, $\boldsymbol{\tau}(k)$    $\boldsymbol{\tau}(k) = -\boldsymbol{J}_b^n(\psi(k))^T \big(\boldsymbol{K_p}\boldsymbol{\eta}_e(k) +$
                                                      $\boldsymbol{K_d}\dot{\boldsymbol{\eta}}_{\boldsymbol{e}}(k) + \boldsymbol{K_i}\boldsymbol{I}_{\eta_e}(k)\big)$
#9  Save values for next iteration                    $\boldsymbol{\eta}_e(k-1) = \boldsymbol{\eta}_e(k)$
                                                      $\boldsymbol{I}_{\eta_e}(k-1) = \boldsymbol{I}_{\eta_e}(k)$
                                                      k++
                                                      goto step #1
```

This algorithm will now produce a thrust vector, $\boldsymbol{\tau}$, to drive the deviation from our reference state towards zero. However, this thrust is given with respect to the CO in the BODY frame, and has to be allocated to the thrusters in order to move the actual vessel appropriately. There are also several other issues with this algorithm that makes it nonfunctional given a specific set of states and inputs.

3.2.2 Thrust Allocation

The thrust vector, $\boldsymbol{\tau}$, from the nonlinear PID algorithm is given in the BODY frame with respect to the CO. However, in order to move the vessel, it has to be allocated to the vessel's actuators. From section 2.2, we have that

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_n \end{bmatrix} = \begin{bmatrix} T_{2x} \\ T_{2y} + T_{1y} \\ T_{1y}l_1 + T_{2y}l_2 \end{bmatrix} \tag{3.26}$$

We can see that these are three equations with three unknowns T_{2x}, T_{2y}, T_{1y}, and can therefor be solved by gaussian elimination, which results in

$$\begin{bmatrix} T_{2x} \\ T_{2y} \\ T_{1y} \end{bmatrix} = \begin{bmatrix} \tau_x \\ \frac{\tau_y l_1 - \tau_n}{l_1 - l_2} \\ \frac{\tau_y l_2 - \tau_n}{l_2 - l_1} \end{bmatrix} \tag{3.27}$$

By examining figure 2.5, we can see how the thrust is then allocated to their respective actuators.

From equation 3.27 we will then define an algorithm to produce the outboard actuator thrust level, outboard actuator angle, and tunnel thruster thrust level.

Algorithm 3.2: Thrust Allocation Algorithm

#1	Allocate the thrust vector from the PID algorithm, $\boldsymbol{\tau}$, to T_{2x}, T_{2y} and T_{1y}	See equation 3.27
#2	Saturate, *sat()*, the allocated thrust, T_{2x}, T_{2y} and T_{1y}, to limit thruster levels within $\pm u_{1,max}$, $\pm u_{2,max}$, $\pm u_{3,max}$	$sat(T_{2x})$, $sat(T_{2y})$, $sat(T_{1y})$
#3	Allocate and scale thrust value output	$u_1 = \sqrt{\frac{T_{1y}}{k_{1y}}}\frac{1}{u_{1,max}}$ $u_2 = \sqrt{\frac{T_{2x}}{k_2}}\frac{1}{u_{2,max}}$ $u_3 = atan2\left(\frac{T_{2x}}{k_2}\right)\frac{u_{3,max}}{2\pi}$

3.2.3 Practical Algorithmic Functionalities

As it is, algorithm 3.1 is more theoretical than practical, and a set of functionalities have to implemented in order to make the algorithm usable for controlling the Telemetron vessel based on GPS and compass data, in addition to the thrust allocation derived above. This section will cover the most important functionalities.

Yaw wrapping

When navigating using NED frame coordinates, we want our yaw angle to be between $\pm\pi$, with respect to the *x-axis*, pointing north. The reason for this is that, say if a vessel turns around 5 times, which adds up to a yaw angle of 10π. Now, say that the heading reference is set to, $\psi_{ref} = \pi$, this would give me a heading error of $10\pi - \pi = 9\pi$, which would cause the vessel to turn around almost 5 times before reaching its reference when it would only have had to turn around half a turn to reach it. To keep this from happening, the yaw angle can be wrapped within the formerly mentioned limits. This can be done in a few ways, but the easiest way, and the most robust way, is to use the modulo operator, %.

The modulo operator gives the residual after a division, and yields zero if the modulo dividend is any whole number multiple of the modulo divisor. So by taking the modulo of 2π for the yaw angle, the result will always stay within $0 \rightarrow 2\pi$. To then wrap it further within $\pm\pi$, we use a simple if statement to check whether it is larger than π or smaller than π , and then subtract or add 2π respectively. We can then define the complete yaw wrapping algorithm as

Algorithm 3.3: Yaw wrapping algorithm

```
#1   Perform modulo operator yaw wrap,%, s.t $0 \leq \psi \leq 2\pi$     $\psi(k) = \psi(k)\%2\pi$
#2   Perform conditional yaw wrap, such that $-\pi \leq \psi \leq \pi$   if $(\psi(k) > \pi)$
                                                                             $\psi(k) = \psi(k) - 2\pi$
                                                                         else if $(\psi(k) < -\pi)$
                                                                             $\psi(k) = \psi(k) + 2\pi$
```

Saturation

The maximum engine force/torque and RPM is based on the Force/RPM engine profile for the 140 HP outboard engine for the Viknes vessel, derived in [5]. The results are adjusted for the difference in HP, as the Telemetron vessel has a 225 HP outboard engine. Since the DP application is for station keeping and low speed maneuvering, a limit at 1000 RPM is set for the outboard engine, resulting in a maximum force thrust of approximately 10 kN (kilo Newton) in surge, $\tau_{x,max} = \pm 10kN$, and sway, $\tau_{y,max} = \pm 10kN$, and 55 kNm in yaw. The latter calculated from

$$\tau_{\psi,max} = \tau_{y,max} r_{gr} \tag{3.28}$$

, where r_{gr} is an approximation of the radius of gyration.

We can define

$$\boldsymbol{\tau}_{max} = \begin{bmatrix} \tau_{x,max} \\ \tau_{y,max} \\ \tau_{\psi,max} \end{bmatrix} \tag{3.29}$$

After each iteration of the discretized PID algorithm, 3.1, the output, $\boldsymbol{\tau}$, is then saturated within these limits, $\boldsymbol{\tau}_{max}$.

Anti-Windup

Another issue is when the state won't reach the set point fast enough (i.e if the setpoint is very large), causing the integral term, $\boldsymbol{I}_{\eta_e}(k)$, to continue integrating the error, causing the integral value to go far above the saturation limits. When the error finally goes to zero, or if another reference is set, the integral term will then take a long time to reduce the large value already integrated. With the output saturation implementation above, this will cause the output to be saturated for a long time after error has been reduced to zero, causing the system to overshoot its setpoint, which in turn could make the system unstable. This phenomenon is known as integral windup, and can be fixed in a couple of ways, by anti-windup methods.

One way is to saturate the integral term within the output saturation limits or a fraction of these limits, known as clamping. Another solution, and a more dynamic solution, is to back-calculate the integral windup. This will cause dynamic decay of the windup term. The windup term calculation algorithm can be defined as

Algorithm 3.4: Anti-Windup: back-calculation

```
#1  Reset the anti-windup term ,$\boldsymbol{\eta}_{aw}(k)$     $\boldsymbol{\eta}_{aw}(k) = [0,0,0]$
#2  Calculate the anti-windup term, $\boldsymbol{\eta}_{aw}(k)$  if $\left(\boldsymbol{I}_{\eta_e}(k) > \boldsymbol{\tau}_{max}(k)\right)$
                                                        $\boldsymbol{\eta}_{aw}(k) = \boldsymbol{\tau}_{max} - \boldsymbol{I}_{\eta_e}(k)$
                                                    else if $\left(\boldsymbol{I}_{\eta_e}(k) < -\boldsymbol{\tau}_{max}(k)\right)$
                                                        $\boldsymbol{\eta}_{aw}(k) = -\boldsymbol{\tau}_{max} - \boldsymbol{I}_{\eta_e}(k)$
```

The dynamics of the back-calculation defined by the anti-windup gain matrix, $\boldsymbol{K}_{aw}$, which can be tuned based on the PID controller gains. Specifically it is tuned based on the derivate and integral gains, defined as

$$\boldsymbol{K}_{aw} = \begin{bmatrix} \frac{\boldsymbol{K}_i(1,1)}{\boldsymbol{K}_d(1,1)} & 0 & 0 \\ 0 & \frac{\boldsymbol{K}_i(2,2)}{\boldsymbol{K}_d(2,2)} & 0 \\ 0 & 0 & \frac{\boldsymbol{K}_i(3,3)}{\boldsymbol{K}_d(3,3)} \end{bmatrix} \tag{3.30}$$

With this and algorithm 3.4, we can calculate the anti-windup and correct for the windup in the integral term, $\boldsymbol{I}_{\eta_e}(k)$. This can be written as

$$\boldsymbol{I}_{\eta_e}(k) = \boldsymbol{I}_{\eta_e}(k) + h\boldsymbol{K}_{aw}\boldsymbol{\eta}_{aw}(k) \tag{3.31}$$

Manual/Auto mode

The DP application is ment for station keeping and low speed maneuvering, so for any other activity there is a need for a separate control algorithm or autopilot that can handle that specific activity. It is also necessary to be able to control the vessel manually, in case of emergency, control algorithm failure, or for any other reason. It should also be possible to control the actuators directly, or to manually set the output, $\boldsymbol{\tau}$, which is then allocated to the thrusters by the thrust allocation algorithm. An operator of the DP application would then be able to switch from auto (DP and LSM) to manual mode and vice versa, to control the vessel directly when needed.

Bumpless Transfers

Whenever the nonlinear PID algorithm as activated, or when going from manual to auto mode or back, there has to be a reinitialization of the nonlinear PID algorithm. This is to avoid keeping the integral term from the last saved iteration to cause a large output when there for instance is no error when the algorithm is activated. It is also important to set the manual output equal the last output from the nonlinear PID output, to avoid too sudden changes. This is known as having a bumpless transfer between mode change.

Another important implementation of the bumpless transfer is when the PID gains are changed. As it is now, the integral term, $\boldsymbol{I}_{\eta_e}(k) = \boldsymbol{I}_{\eta_e}(k)$, is first integrated, and the total sum is then multiplied with the integral gain, $\boldsymbol{K}_i$. If the integral term is large and the integral gain is doubled, this will cause a large spike in the output of the algorithm, possibly causing it to become unstable. A way around this is to instead of multiplying the entire integral, the residual integral increment for each iteration is multiplied with the integral gain instead. When changing the integral gain, only the following iterations will be affected, and not the entire integral at the time of the change. We can write this as

$$\boldsymbol{I}_{\eta_e}(k) = \boldsymbol{I}_{\eta_e}(k-1) + h\boldsymbol{K}_i\boldsymbol{\eta}_e(k) \tag{3.32}$$

Derivate Kick

The nonlinear PID algorithm's derivate term, $\boldsymbol{K_d}\dot{\boldsymbol{\eta}}_{\boldsymbol{e}}(k)$, takes the derivate of both the state and the reference. This means whenever there is a non-differentiable change in reference, like a step, will cause the derivate to spike. This is known as a "Derivate Kick". This spike causes an unwanted response in our output. A simple way to fix this is by removing the reference term altogether, and only correcting for changes in the state, written as $\boldsymbol{K_d}\dot{\boldsymbol{\eta}}(k)$.

Reference Model

For low speed maneuvering (and maneuvering in general), it is necessary to generate paths which the vessel should follow. If the desired position is far away, a large change in reference can cause a initial large output from the PID algorithm, which could cause unwanted behaviour. When maneuvering, the vessel should move smoothly towards the reference. A simple way of generating such paths, is to derive a reference model, which in essence works like a low pass filter for the reference. In order to avoid sudden changes in the reference, it needs to be continuously differentiable, which we get from a second order reference model.

A second order reference model for position and heading can then be defined as

$$\ddot{\boldsymbol{\eta}}_d + 2\boldsymbol{\zeta}\boldsymbol{\omega}_{n,d}\dot{\boldsymbol{\eta}}_d + \boldsymbol{\omega}_{n,d}^2\boldsymbol{\eta}_d = \boldsymbol{\omega}_{n,d}^2\boldsymbol{\eta}_{ref} \tag{3.33}$$

which we can rewrite as

$$\dot{\boldsymbol{\eta}}_{\boldsymbol{d}} = \boldsymbol{v}_d \tag{3.34}$$

$$\dot{\boldsymbol{v}}_d = \boldsymbol{\omega}_{n,d}^2\left(\boldsymbol{\eta}_{ref} - \boldsymbol{\eta}_d\right) - 2\boldsymbol{\zeta}\boldsymbol{\omega}_{n,d}\boldsymbol{v}_d \tag{3.35}$$

where $\boldsymbol{\zeta}$ is the relative damping ratio, $\boldsymbol{\omega}_{n,d}$ is the natural frequency and $\boldsymbol{\eta}_{ref}$ is the reference set point. Since we want our reference to change slower than our system, to give a smooth motion for the vessel, the natural frequency is set to be 5 times lower than the vessel model's natural frequency, giving us

$$\boldsymbol{\omega}_{n,d} = \frac{\boldsymbol{\omega}_n}{5} \tag{3.36}$$

Discretized, this becomes

$$\begin{aligned} \boldsymbol{\eta}_d(k+1) &= \boldsymbol{\eta}_d(k) + h\boldsymbol{v}_d(k) \\ \boldsymbol{v}_d(k+1) &= \boldsymbol{v}_d(k) + h\left[\boldsymbol{\omega}_{n,d}^2\left(\boldsymbol{\eta}_{ref}(k) - \boldsymbol{\eta}_d(k)\right) - 2\boldsymbol{\zeta}\boldsymbol{\omega}_{n,d}\boldsymbol{v}_d(k)\right] \end{aligned} \tag{3.37}$$

where h is the timestep.

With this we can also redefine our error term as

$$\boldsymbol{\eta}_e(k) = \boldsymbol{\eta}(k) - \boldsymbol{\eta}_d(k) \tag{3.38}$$

With all these functionalities defined in this section, the discretized nonlinear PID algorithm, along with the thrust allocation can be rewritten as the DP control algorithm

Algorithm 3.5: DP Control Algorithm

```
#0   Initialize algorithm                                   $k = 0$ , $\boldsymbol{I}_{\eta_e}(k) = \boldsymbol{I}_{\eta_e}(k-1) = [0,0,0]$ ,
                                                            $\boldsymbol{\eta}_e(k) = \boldsymbol{\eta}_e(k-1) = [0,0,0]^T$
#1   Compute $\boldsymbol{\eta}_e$                          $\boldsymbol{\eta}_e(k) = \boldsymbol{\eta}(k) - \boldsymbol{\eta}_r(k)$
#2   Compute $\dot{\boldsymbol{\eta}}_{\boldsymbol{e}}$ by numerical differentiation    $\dot{\boldsymbol{\eta}}_{\boldsymbol{e}} = \frac{1}{h}\left(\boldsymbol{\eta}_e(k) - \boldsymbol{\eta}_e(k-1)\right)$
#3   Compute the rotation matrix, $\boldsymbol{J}_b^n(\psi(k)$    $\psi(k)) = \boldsymbol{\eta}\mathbf{[3]}(k)$,  $\boldsymbol{J}_b^n(\psi(k))$
#4   Compute the PID output, $\boldsymbol{\tau}(k)$         $\boldsymbol{\tau}(k) = -\boldsymbol{J}_b^n(\psi(k))^T\left(\boldsymbol{K_p}\boldsymbol{\eta}_e(k)+\right.$
                                                            $\left.\boldsymbol{K_d}\dot{\boldsymbol{\eta}}(k) + \boldsymbol{I}_{\eta_e}(k)\right)$
#5   Calculate anti-windup term                             See algorithm 3.4
#6   Saturate output, $\boldsymbol{\tau}(k)$                $sat(\boldsymbol{\tau}(k))$
#7   Compute integral term, $\boldsymbol{I}_{\eta_e}(k)$    $\boldsymbol{I}_{\eta_e}(k) = \boldsymbol{I}_{\eta_e}(k-1) + h\left(\boldsymbol{K}_i\boldsymbol{\eta}_e(k)+\right.$
                                                            $\left.\boldsymbol{K}_{aw}\boldsymbol{\eta}_{aw}(k)\right)$
#9   Compute reference model output, $\boldsymbol{\eta}_d(k)$    See equation 3.37
#10  Allocate thrust to actuators from $\boldsymbol{\tau}(k)$    See algorithm 3.2
#11  Save values for next iteration                         $\boldsymbol{\eta}_e(k-1) = \boldsymbol{\eta}_e(k)$
                                                            $\boldsymbol{I}_{\eta_e}(k-1) = \boldsymbol{I}_{\eta_e}(k)$
                                                            k++
                                                            goto step #1
```

This algorithm, along with the vessel model, together form a closed feed-back loop, as presented in figure 3.1 below.

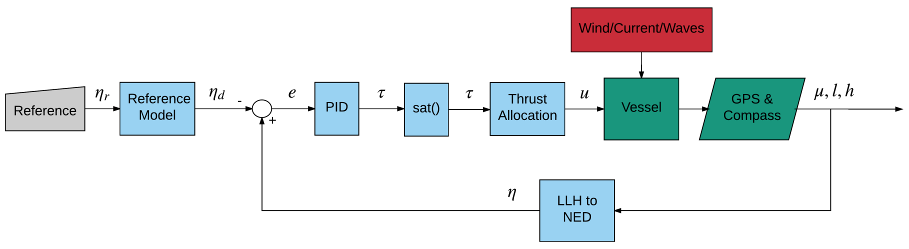


Figure 3.1: Block diagram of DP control system

3.2.4 Pole Placement

There are several ways to tune a control algorithm. A common approach is to perform manual tuning on the actual system, for instance if a model of the system isn't available but the physical system is. Another approach is the tune the controller based on a model of the vessel. For PID algorithms this is known as *pole placement*.

This thesis's focus is physical implementation and testing, and we only have a crude approximation for the vessel model. Because of this, the control algorithm will be tuned initially by pole placement based on the approximated model, and then re-tuned based on initial test results on the vessel.

By using the pole placement algorithm presented in [4], defined as

Algorithm 3.6: Pole Placement Algorithm

#1	Specify the bandwidth, $\boldsymbol{\omega}_{b,pid}$ and the relative damping ratio, $\boldsymbol{\zeta}_{pid}$	$\boldsymbol{\omega}_{b,pid} > 0, \quad \boldsymbol{\zeta}_{pid} > 0$
#2	Compute the natural frequency, $\boldsymbol{\omega}_{n,pid}$	$\boldsymbol{\omega}_{n,pid} = \frac{1}{\sqrt{1-2\boldsymbol{\zeta}_{pid}^2+\sqrt{4\boldsymbol{\zeta}_{pid}^4-4\boldsymbol{\zeta}_{pid}^2+2}}}\boldsymbol{\omega}_{b,pid}$
#3	Compute the proportional gain, $\boldsymbol{K}_p$	$\boldsymbol{K}_p = \boldsymbol{M}\boldsymbol{\omega}_{n,pid}{}^2$
#4	Compute the derivate gain, $\boldsymbol{K}_d$	$\boldsymbol{K}_d = 2\boldsymbol{\zeta}_{pid}\boldsymbol{\omega}_{n,pid}\boldsymbol{M}$
#5	Compute the integral gain, $\boldsymbol{K}_i$	$\frac{\boldsymbol{\omega}_{n,pid}}{10}\boldsymbol{K}_p$

, where $\boldsymbol{M} = \boldsymbol{M}^{\boldsymbol{CO}}$ only contains the diagonal elements in the matrix. The PID algorithm can then be tuned based on the model of the vessel.

We start by finding the eigenvalues of our model, as to find the poles and natural frequencies of our model. The eigenvalues, λ_i, are calculated in *Matlab*, but can be calculated manually. Since our model can be written as

$$\dot{\boldsymbol{\nu}} = -\boldsymbol{M}_{RB}^{CO^{-1}}\left(\boldsymbol{D}^{CO}\boldsymbol{\nu} + \boldsymbol{\tau}\right) \tag{3.39}$$

we can calculate the eigenvalues by solving

$$\left| \boldsymbol{I}\lambda_i - \left(-\boldsymbol{M}_{RB}^{CO^{-1}}\boldsymbol{D}^{CO}\right)\right| = 0, \quad i \in 1,2,3, \quad \boldsymbol{I} = \text{Identity matrix} \tag{3.40}$$

where we, on a generalized form [2], can write the connection between eigenvalues, natural frequencies, relative damping ratios and time constants as

$$\lambda_i = \alpha_i + i\beta_i, \qquad \text{(eigenvalue)} \tag{3.41}$$

$$\omega_{ni} = |\lambda_i|, \qquad \text{(natural frequency)} \tag{3.42}$$

$$\zeta_i = \frac{-\alpha_i}{\sqrt{{\alpha_i}^2 + {\beta_i}^2}}, \qquad \text{(relative damping ratio)} \tag{3.43}$$

$$T_i = \frac{1}{\omega_{ni}\zeta_i}, \qquad \text{(time constant)} \tag{3.44}$$

The model is a stable mass-damper with only real eigenvalues ($\alpha_i < 0, \beta_i = 0, \zeta_i = 1$), and therefor our poles/eigenvalues will of equal magnitude to our natural frequencies, $\lambda_i = -\omega_i$.

When choosing the bandwidth, $\boldsymbol{\omega}_{b,pid}$, for our PID controller, tuned by pole placement, a good rule of thumb is to have the controller dynamics be 2-10 times faster than the system it is controlling. As a starting point for this thesis, the bandwidth, and therefor the natural frequency of the PID controller, will be 5 times larger,

$$\boldsymbol{\omega}_{n,pid} = 5\boldsymbol{\omega}_n \tag{3.45}$$

, making the controller 5 times faster than the vessel dynamics.

All parameters and values can be found in appendix A.2, and in the digital file appendix B.

3.3 Human Machine Interface

The DP control algorithm on its own won't do anything else but fill variables with values. To control an actual vessel with it, an interface between the vessel and the DP control algorithm as to be developed, as well as an interface between the DP control algorithm and an operator. These interfaces together form a human machine interface.

This can be achieved in various ways and there are plenty of software, frameworks and programming languages that are suited for the job. To simplify the development to some extent, and to avoid reinventing the wheel, *Qt* is chosen as the HMI development framework, as it provides a rich library for both visualization and communication.

3.3.1 Qt

Qt is a cross-platform application framework that is widely used for developing application software that can be run on various software and hardware platforms with little or no change in the underlying codebase, while still being a native application with the capabilities and speed thereof.

Qt uses standard C++ with extensions including signals and slots that simplifies handling of events, and this helps in development of both GUI and server applications which receive their own set of event information to process accordingly.

It runs on the major desktop platforms and some of the mobile platforms. Non-GUI features include SQL database access, XML parsing, JSON parsing, thread management and network support, of which to two latter will be used extensively in this application. The DP application is developed with Qt v5.5 on a laptop running Windows 10.

The DP application will feature

- GUI
  - DP control panel
  - Manual Controller
  - Map
- DP control algorithm
- TCP client

, as well as a vessel model application for simulation purposes. The application is object-oriented, and each item in the list above is implemented as a seperate object, where all the elements communicate with each other by means of signals and slots, as shown in figure 3.2.

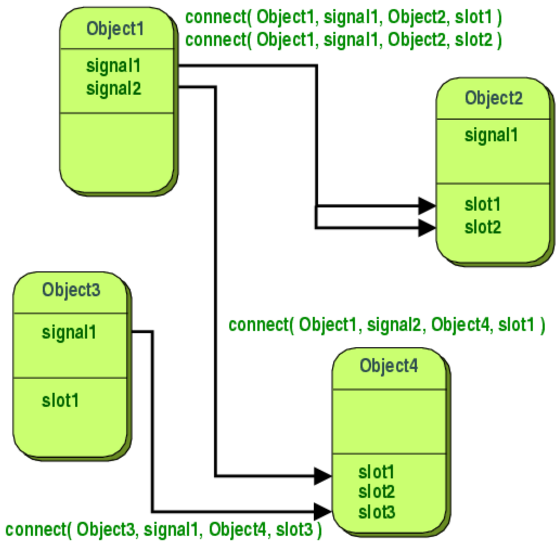

Object1
signal1
signal2
connect(Object1, signal1, Object2, slot1)
connect(Object1, signal1, Object2, slot2)
Object2
signal1
slot1
slot2
Object3
signal1
slot1
connect(Object1, signal2, Object4, slot1)
Object4
slot1
slot2
slot3
connect(Object3, signal1, Object4, slot3)

Figure 3.2: Qt: Signals and Slots

DP Control Algorithm

The DP control algorithm, defined in algorithm 3.5, is programmed and implemented in C++. It is programmed as an object, able to communicate with other objects and threads in the application by means of signals and slots, as shown in figure 3.2. The object is designed to run on a separate thread with a fixed frequency, freeing up processing power to handle other events and processes.

Communication

The DP application and the Telemetron vessel have to be able to communicate with each other. Since the Telemetron vessel has a pre-existing TCP/IP communication framework, this will be used to communicate between the two. TCP/IP communication makes it possible to connect to the vessel not only on a local network but also over the internet.

A TCP client is therefor implemented in the DP application, programmed as an object, enabling it to communicate between other objects with signals and slots within the application (see figure 3.2).

The Telemetron vessel is running a TCP server application that opens a TCP socket between it and any client trying to connect. On this socket the client and server can both send and receive data until the connection is lost, broken or disconnected, causing the socket to be terminated. The Telemetron vessel aquires GPS and compass data, which is formated into a navigation message, containing

- Timestamp
- GPS data (latitude, longitude and altitude)
- Compass heading

- External Control flag

The data is sent over the TCP socket as a comma delimited string with the prefix "$NAV", ending with a line change. The navigation message is then received by the TCP client, and is then parsed and fed into their respective variables in the DP control algorithm running in the DP application.

The TCP client in the DP application generates a comma separated control message with the prefix "$CTRL" from the output of the DP control algorithm, containing

- Tunnel Thruster RPM (normalized between -1 and 1)

- Outboard Engine RPM (normalized between -1 and 1)

- Outboard Engine Angle (between -40 and 40 degrees)

The control message is then received and parsed by the server and another process allocates the setpoints to the actuators.

The communication flow between the vessel and the DP application can be seen in figure 3.3.

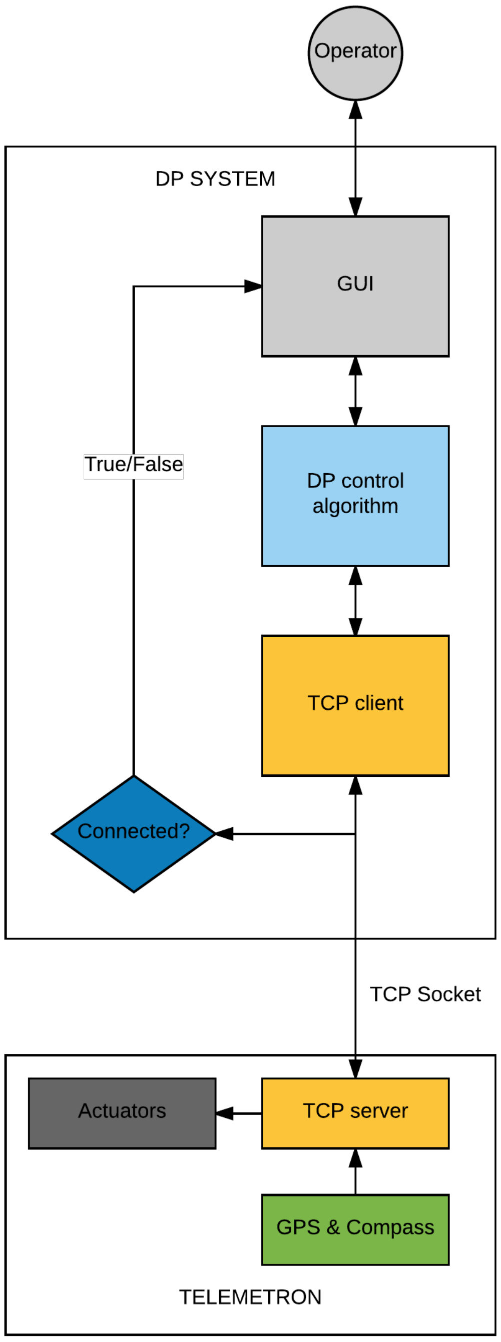


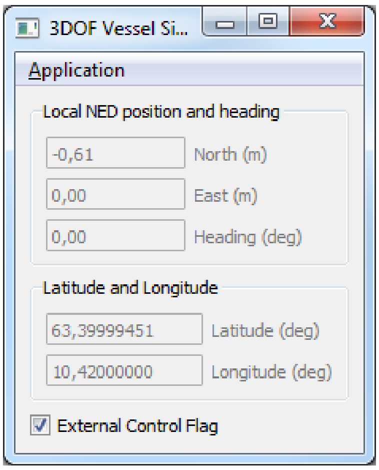
Figure 3.3: Flow diagram of complete DP system

Vessel Model

The DP application has to be tested and verified locally before testing it on the actual vessel. For this reason, a separate application is developed, also in Qt, that runs a TCP server and a vessel model simulator.

The vessel model is programmed based on the model derived in section 3.1. The vessel simulator is set to run at produce data at a fixed frequency, and takes the local NED coordinates from the vessel model and generates simulated GPS data (latitude, longitude and altitude). The simulated GPS data is produced by algorithm 2.1, presented in chapter 2. The GPS coordinates are initialized with real coordinates approximately at the same location as the Telemetron vessel is located in Trondheim.

The vessel simulator also has a GUI, that gives us information on its state, as well as giving is the possibility to set the external control flag in a similar manner as the Telemetron vessel (see figure 3.4. With this, we can produce the same navigation message as the Telemetron vessel, and we can test the TCP/IP communication locally between two computers.



Local NED position and heading
-0,61 North (m)
0,00 East (m)
0,00 Heading (deg)
Latitude and Longitude
63,39999451 Latitude (deg)
10,42000000 Longitude (deg)
☑ External Control Flag

Figure 3.4: Vessel Simulator

Graphical User Interface

To interface with the DP application, a GUI is developed. The GUI is built as an application for a technical operator, where there will be possibilities for

- Manual Control of the vessel
- Connect/Disconnect to vessel
- Setting the reference
- Read data
- Record data
- Tune DP control algorithm
- 2D vessel state visualization

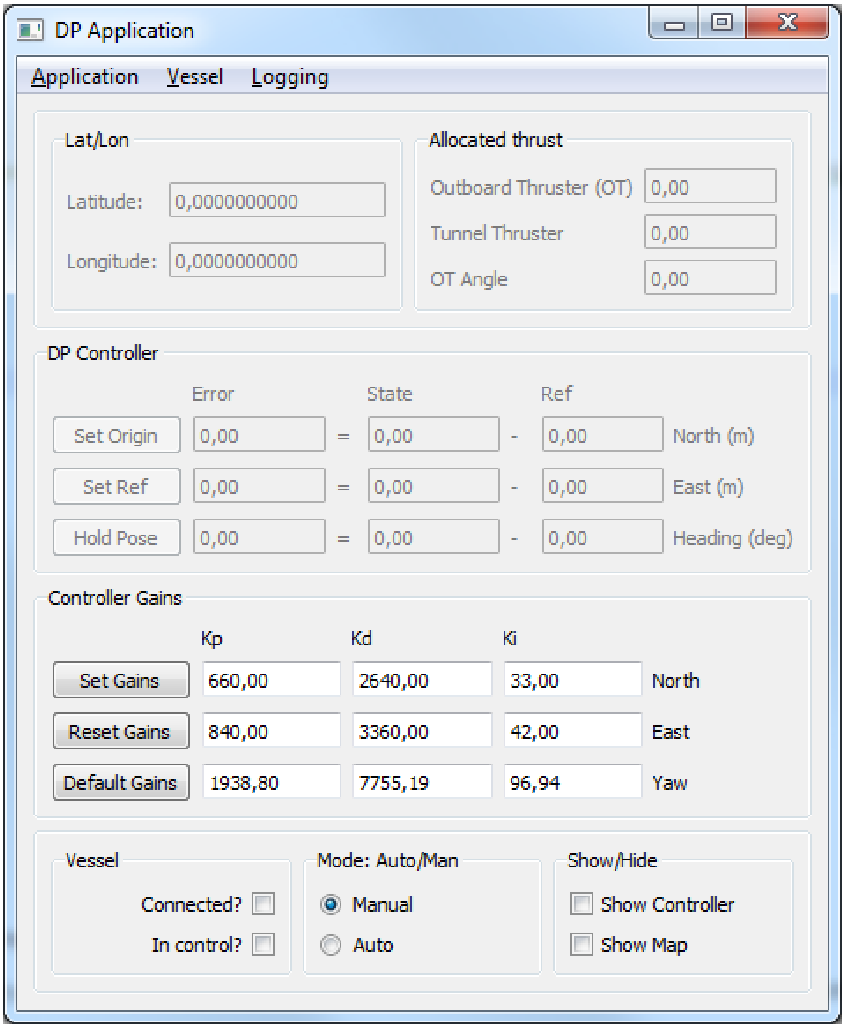
The GUI is a simple design with basic building blocks found in most native Windows 10 applications, such as

- Radio buttons
- Check boxes
- Boolean switches/buttons
- Numerical fields (Read, Read/Write)
- Menu bar
- String field

, making the application look similar to its native OS.

The GUI is programmed as an object and can then communicate with the other objects in the DP application by means of signals and slots, as mentioned earlier.

The application launches as a multi window application, with a main window start screen. The main window launches in a disconnected and disabled state, where on can only tune the PID gains, as shown in figure 3.5.



Lat/Lon

Latitude:	0,0000000000
Longitude:	0,0000000000

Allocated thrust

Outboard Thruster (OT)	0,00
Tunnel Thruster	0,00
OT Angle	0,00

DP Controller

	Error		State		Ref	
Set Origin	0,00	=	0,00	-	0,00	North (m)
Set Ref	0,00	=	0,00	-	0,00	East (m)
Hold Pose	0,00	=	0,00	-	0,00	Heading (deg)

Controller Gains

	Kp	Kd	Ki	
Set Gains	660,00	2640,00	33,00	North
Reset Gains	840,00	3360,00	42,00	East
Default Gains	1938,80	7755,19	96,94	Yaw

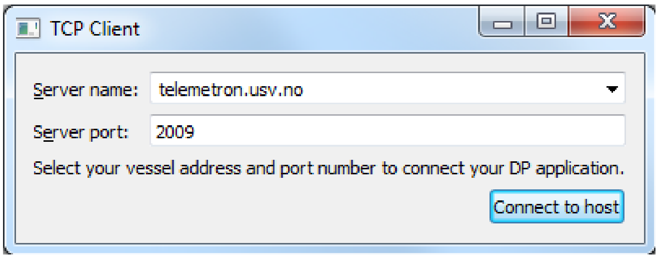
Vessel
Connected? ☐
In control? ☐

Mode: Auto/Man
[✓ Manual]
Auto

Show/Hide
☐ Show Controller
☐ Show Map

Figure 3.5: DP Application: Default window

By going to the "Vessel" menu bar and clicking "connect", a pop-up window appears for the operator to fill in the vessel IP address and port number, for connecting the application to the vessel (see figure 3.6).



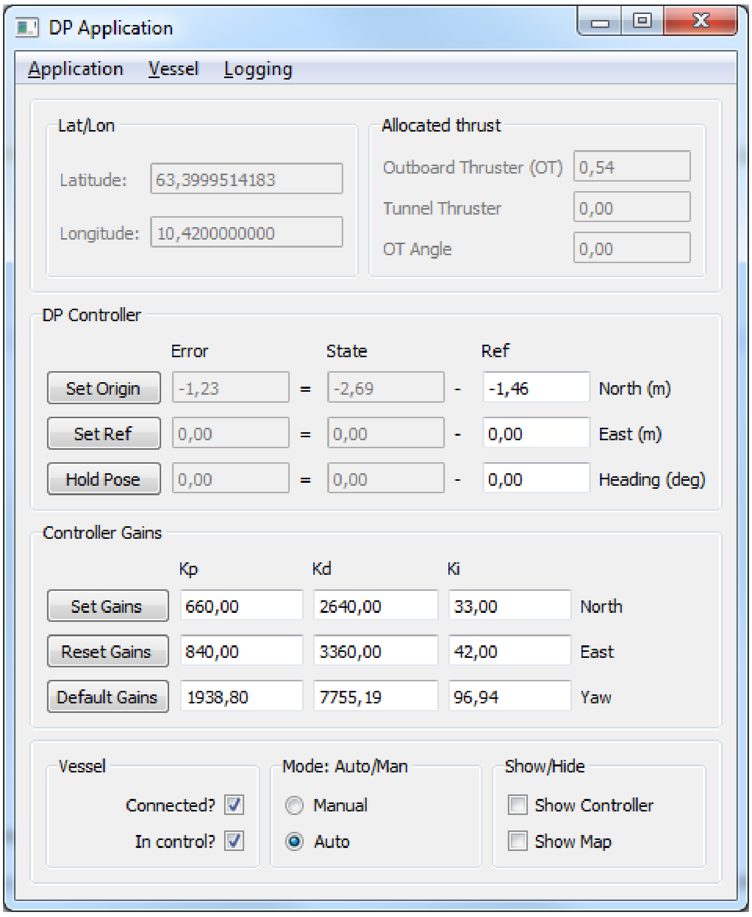
Server name: telemetron.usv.no

Server port: 2009

Select your vessel address and port number to connect your DP application.

Figure 3.6: DP Application: TCP client pop-up

The main window will then indicate if the connection is active, in the lower left corner. It also gives the operator an indication to show if the vessel has given the DP application external control access, as shown in figure 3.7; a safety measure made by Maritime Robotics for emergency takeover. The DP application will be disabled if these flags are not active.



DP Application

Application Vessel Logging

Lat/Lon

Latitude:	63,3999514183
Longitude:	10,4200000000

Allocated thrust

Outboard Thruster (OT)	0,54
Tunnel Thruster	0,00
OT Angle	0,00

DP Controller

	Error		State		Ref	
Set Origin	-1,23	=	-2,69	-	-1,46	North (m)
Set Ref	0,00	=	0,00	-	0,00	East (m)
Hold Pose	0,00	=	0,00	-	0,00	Heading (deg)

Controller Gains

	Kp	Kd	Ki	
Set Gains	660,00	2640,00	33,00	North
Reset Gains	840,00	3360,00	42,00	East
Default Gains	1938,80	7755,19	96,94	Yaw

Vessel

Connected? ✓

In control? ✓

Mode: Auto/Man

○ Manual

● Auto

Show/Hide

☐ Show Controller

☐ Show Map

Figure 3.7: DP Application: Connected and in "Auto" mode

When connected, the user can either control the vessel manually by means of sliding controllers, avaiable by clicking the "Show Controller" check box, or using the DP algorithm, activated by toggling from manual to auto mode.

The manual controller lets the user control each actuator separately, as shown in figure 3.8.

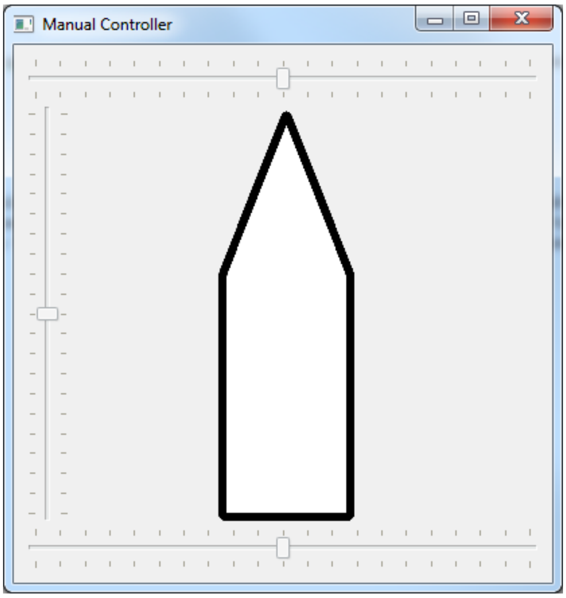


Figure 3.8: Manual Controller

The 2D pose and reference can be visualized in both modes by clicking the "Show Map" check box. The map is scaled within ±50 meters, but can have its origin moved by clicking the "Set Origin" button in the main window, which sets the origin to the current position of the vessel (see figure 3.9).

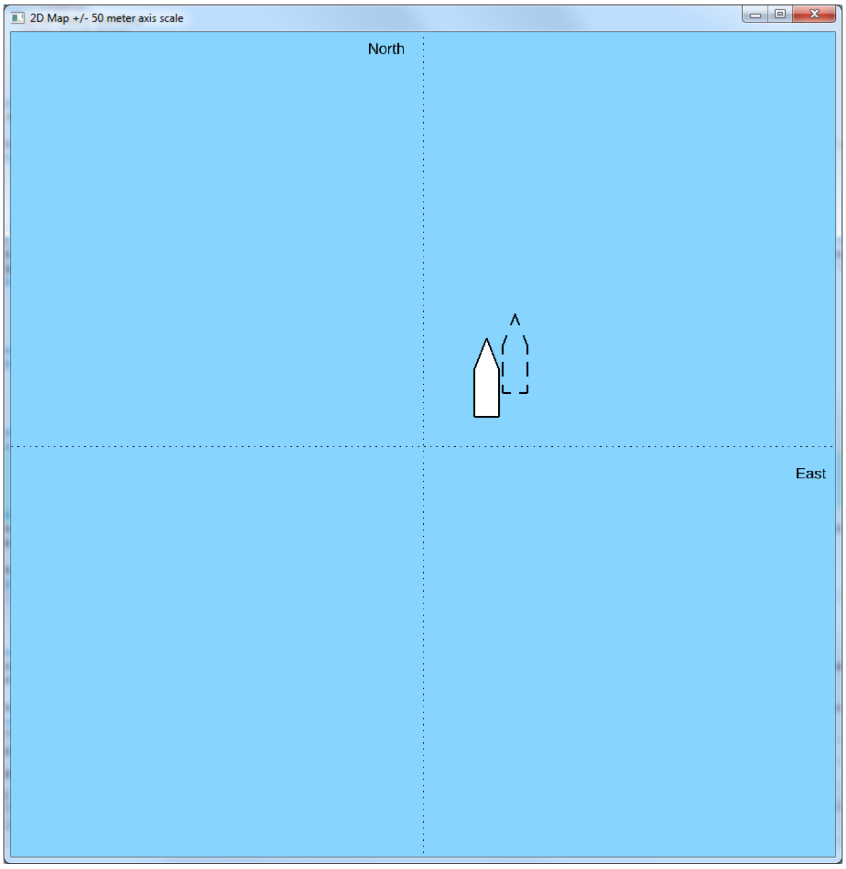


Figure 3.9: 2D Map

The dottet black outline is the reference pose, and the white object is the actual pose, in local NED coordinates.

The DP controller can also be tuned by adjusting the PID gains, where the "Set Gains" button apply the gains from the numerical fields, the "Reset Gains" button rolls back the gains to the previously set gains, and the "Default Gains" button will roll the gains back to their default value, given by the pole placement, initialized on startup.

3.4 Simulation

The DP control system, as shown in figure 3.1 has to be tested and verified through simulations before being tested on the Telemetron vessel. The simulations objectives will include

- Dynamic positioning, with and without external forces
- Low speed maneuvering (translation and rotation), with and without external forces

perturbation by external forces (wind, current, waves), low speed maneuvering and dynamic positioning.

3.4.1 Matlab

Matlab is a numerical computing environment as well as a programming language. It has a *C* like syntax, and it is very quick and easy to perform complex computations and simulations with, making it ideal for rapid prototyping and testing of algorithms and code in general.

The DP control algorithm (see algorithm 3.5) and the vessel model (see equation 2.19), connected as shown in figure 3.1, is implemented in Matlab, where the simulation objectives presented above are tested.

3.4.2 Qt - C++

The DP control system will also have to be simulated and tested with the DP application. Here the DP control algorithm will tested against the same simulation objectives prestented above.

In addition the communication scheme will be tested locally between two separate computers, one running the vessel model application and the other running the DP application, both presented in sectionsec:hmi. The test will verify that the communication scheme works as intended, as well is verifying that the front panel of the GUI is functional and without errors.

3.5 Experiment

The full scale experiment will take place on the Telemetron vessel in *Trondheimsfjorden* and in the docks outside *Maritime Robotics* headquarters.

The laptop running the DP application will be connected to the Telemetron vessel server, running on a real-time linux system, by TCP/IP via network cabel. Maritime Robotics provide a captain to navigate the vessel to and from the test location, as well as being a safety measure for taking over control of the vessel, if need be. The test setup with personel can be seen in the picture below.

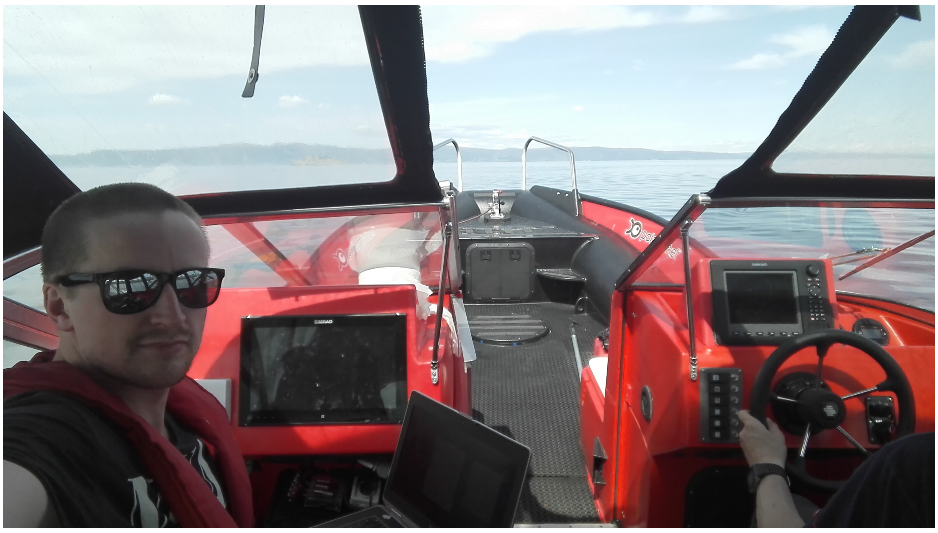


Figure 3.10: Test setup with pilot at test location

The tests objectives are

- Verify communication and actuator control
- Test DP and LSM objectives in calm weather
- Retest DP and LSM in moderate weather

, where the weather requirements are secondary, due to small time windows and long term planning issues that might occur.

Chapter 4

Results

This chapter presents the results from the methods and work presented in chapter 3, along with a discussion and conclusion of the thesis, along with recommendations for future work. The results are divided in two. The first being a presentation of the simulated results, along with a short discussion. The second being a presentation of the full scale experiment on the Telemetron vessel, along with a short discussion. These results will then be the basis of a longer discussion regarding the objective of the thesis in the following chapter.

4.1 Simulation

The simulations are divided in to stages. First the DP control algorithm will be prototyped and tested in Matlab, verified against all test objectives before moving on the second stage. The second stage is to test the DP application created in Qt. This is done by using the vessel model application connected locally to the DP application via ethernet cable. The DP application will then run the same tests as in Matlab, to verify the control algorithm ,as well as verifying the functionality of the GUI and communication of the DP application itself.

4.1.1 Matlab

The objectives for the simulation will be

- Dynamic Positioning
- Low Speed Maneuvering: Translation
- Low Speed Maneuvering: Rotation

, where the three objectives will all be given four tests each,

- Not perturbed by environmental forces
- Perturbed by wind and current , within the feasible thrust region
- Perturbed by wind, current and waves, within the feasible thrust region
- Perturbed by wind and current, outside the feasible thrust region
- Perturbed by wind, current and waves, outside the feasible thrust region

where we define the angle of the environmental force vector as β_{wc}, relative to the NED frame.

Perturbations outside the feasible thrust region will be applied such that the environmental force vector, $\boldsymbol{v_{wc}}$, stands perpendicular to the vessel's reference x-axis, forcing the vessel in the sway direction, relative to the reference. The ocean current speed will be set at $v_c = 0.5$ meters per second for all simulations, and the wave time period will be set at $T_w = 3$ seconds.

Although all of the test cases above where run, only the cases including all environmental forces, both inside and outside the feasible thrust region, will be presented and discussed, as these are the tests of most interest and of highest importance.

Dynamic Positioning

The first test objective is dynamic positioning, station keeping.

The reference is set at $\boldsymbol{\eta}_r = [0,0,0]$, and the external force vector angle is set at $\beta_{wc} = 0$

From figures 4.1 , 4.2 and 4.3, we can see that the vessel maintains its reference position and heading within reasonable limits. The largest standard deviation, along the x-axis, is calculated to $\bar{x}_n = 0.1457$. Apart for the minor oscillations caused by the waves, the system is stable.

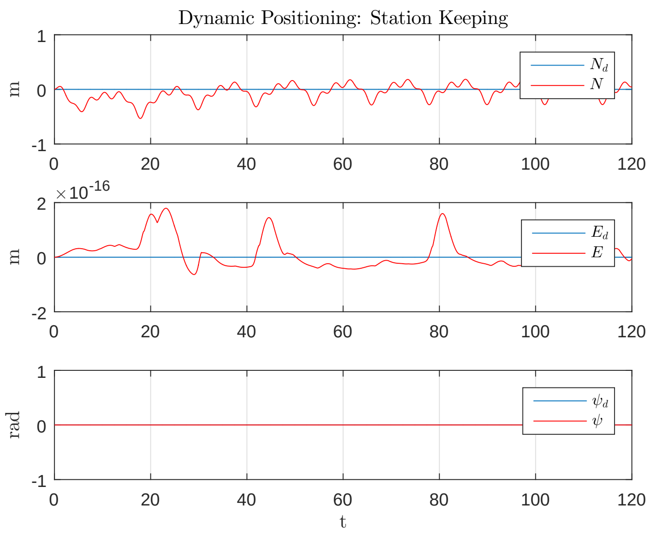


Figure 4.1: DP: $\boldsymbol{\eta}$ - Perturbed by current, wind and waves ($\beta_{wc} = 0$)

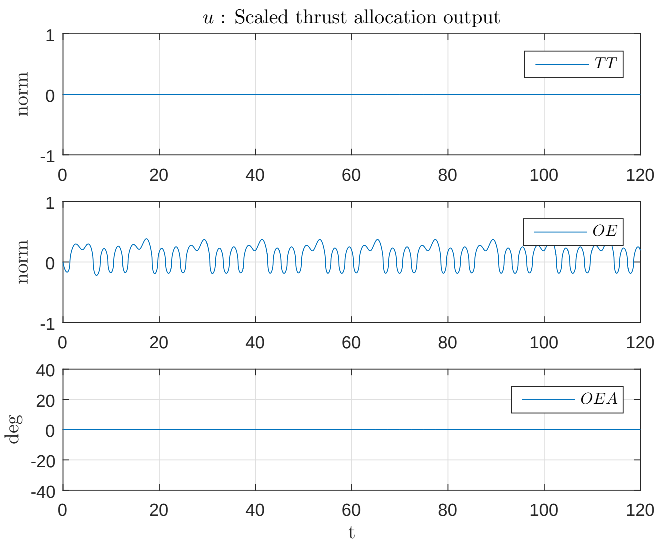


Figure 4.2: DP: $\boldsymbol{u}$ - Perturbed by current, wind and waves ($\beta_{wc} = 0$)

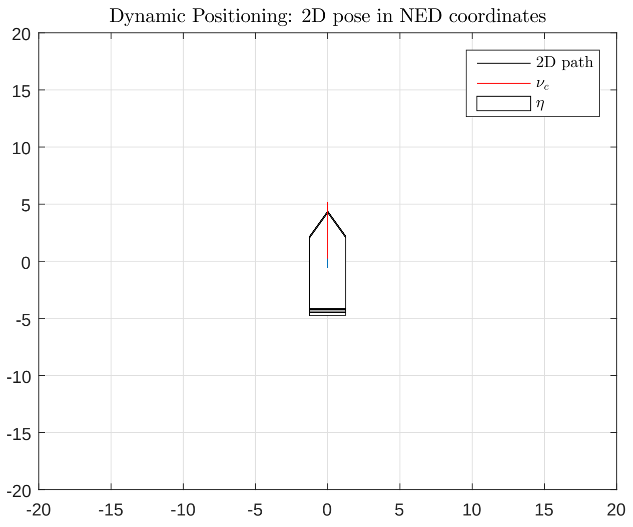


Figure 4.3: DP: 2D Pose - Perturbed by current, wind and waves ($\beta_{wc} = 0$)

The next test sets the reference again at $\boldsymbol{\eta}_r = [0, 0, 0]$, and the external force vector angle is set at $\beta_{wc} = -\frac{\pi}{2}$.

From figures 4.4 , 4.5 and 4.6, we can see that the vessel maintains its reference position and heading but with slightly larger deviations than before, most so in heading and position along the y-axis. The largest standard deviation in position, along the y-axis, is calculated to $\bar{y}_n = 0.1334$.

The oscillations on both position and heading is caused by the fact the outboard engine angle is limited to ±40 degrees, making it impossible to drive the vessel only in the sway direction. This forces the vessel to "zig-zag" back and forth, to be able to maintain its position. This in turn makes the outboard engine angle switch from +40 degrees to −40 degrees continuously. This, along with the three second delay in gear change and wave perturbation, causes the vessel to oscillate slightly. However, it maintains stability, and doesn't diverge from its reference.

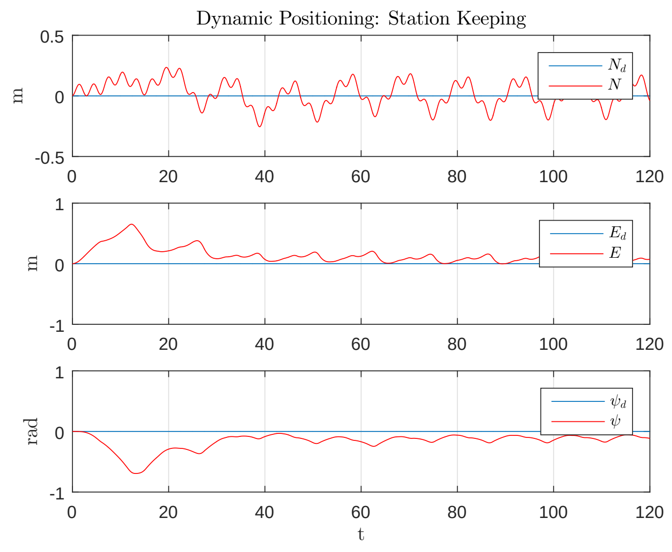


Figure 4.4: DP: $\boldsymbol{\eta}$ - Perturbed by current, wind and waves ($\beta_{wc} = -\frac{\pi}{2}$)

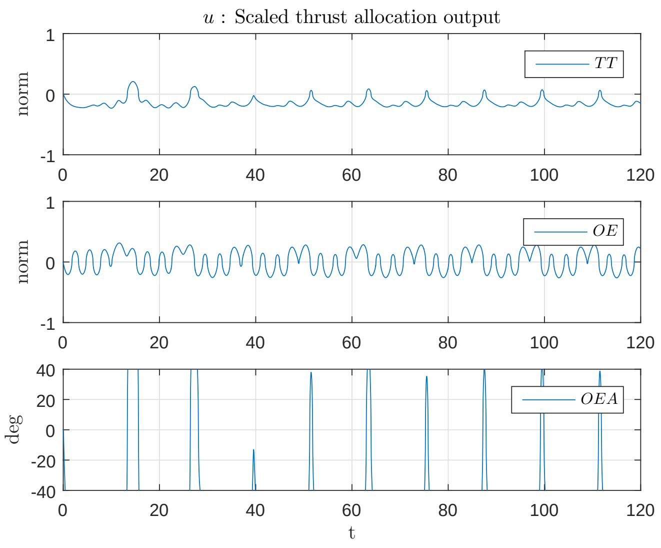


Figure 4.5: DP: $\boldsymbol{u}$ - Perturbed by current, wind and waves ($\beta_{wc} = -\frac{\pi}{2}$)

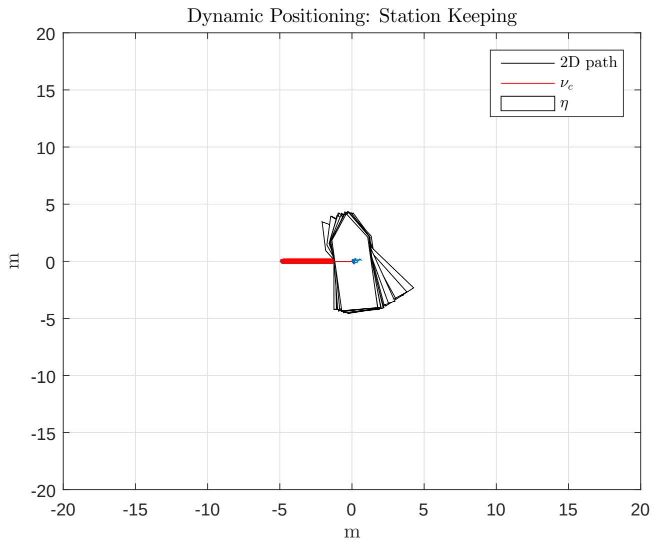


Figure 4.6: DP: 2D Pose - Perturbed by current, wind and waves ($\beta_{wc} = -\frac{\pi}{2}$)

Low Speed Maneuvering: Translation

The second test objective is low speed maneuvering, translation maneuver.

The reference is set at $\boldsymbol{\eta}_r = [10, 10, 0]$, and the external force vector angle is set at $\beta_{wc} = 0$

From figures 4.7 , 4.8 and 4.9, we can see that the vessel converges on the reference, after some initial delay. This delay is most likely caused by the environemental forces acting as a step on the vessel, as it is at rest with inactive actuators at the start. The initial fluctuations in yaw are relatively small, below 0.05 radians at peak value. The largest standard deviation in position, along the x-axis, is calculated to $\bar{x}_n = 0.5552$.

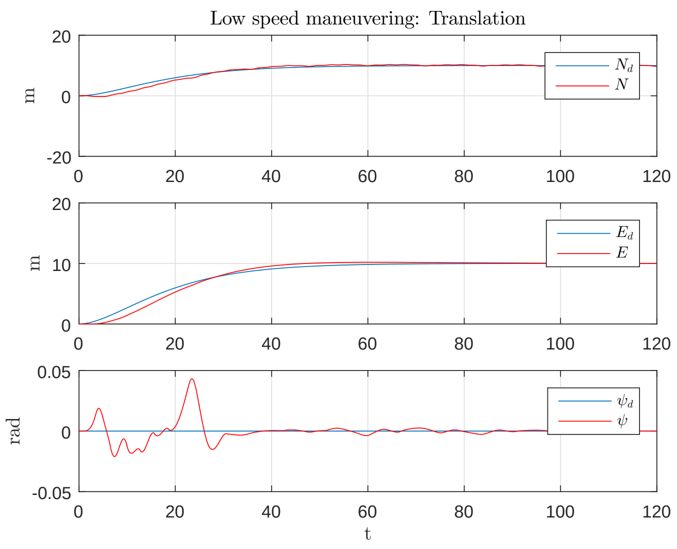


Figure 4.7: LSM: $\boldsymbol{\eta}$ - Perturbed by current, wind and waves ($\beta_{wc} = 0$)

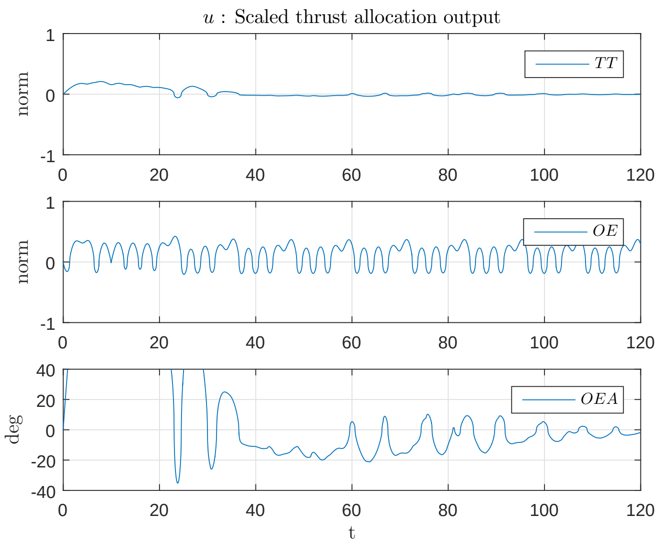


Figure 4.8: LSM: $\boldsymbol{u}$ - Perturbed by current, wind and waves ($\beta_{wc} = 0$)

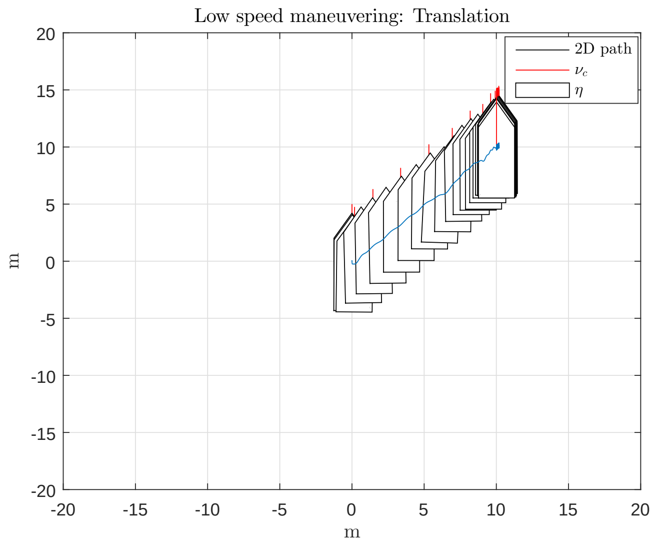


Figure 4.9: LSM: 2D Pose - Perturbed by current, wind and waves ($\beta_{wc} = 0$)

The next test sets the reference again at $\boldsymbol{\eta}_r = [10, 10, 0]$, and the external force vector angle is set at $\beta_{wc} = -\frac{\pi}{2}$.

From figures 4.7 , 4.8 and 4.9, we see again that the vessel converges on the reference, after some initial delay. Again, the delay is most likely caused by the environemental forces acting as a step on the vessel, as it is at rest with inactive actuators at the start. The deviation in yaw is more noticable this time, having a standard deviation of $\bar{\psi} = 0.0532$ small. The largest standard deviation in position, along the y-axis, is calculated to $\bar{y}_N = 0.5533$.

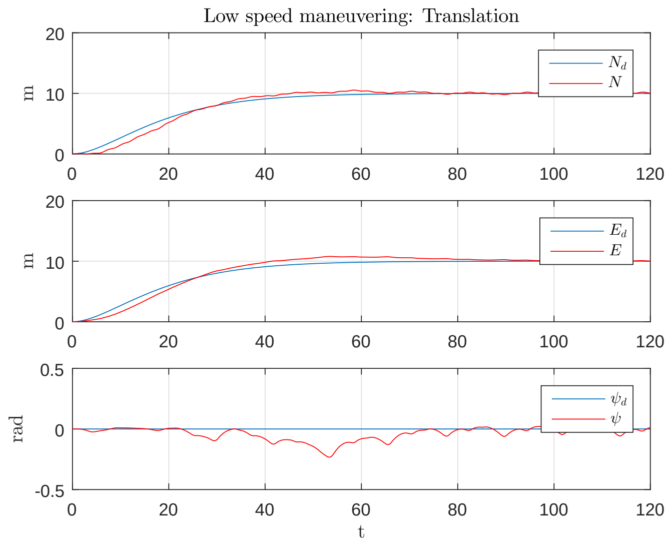


Figure 4.10: LSM: $\boldsymbol{\eta}$ - Perturbed by current, wind and waves ($\beta_{wc} = -\frac{\pi}{2}$)

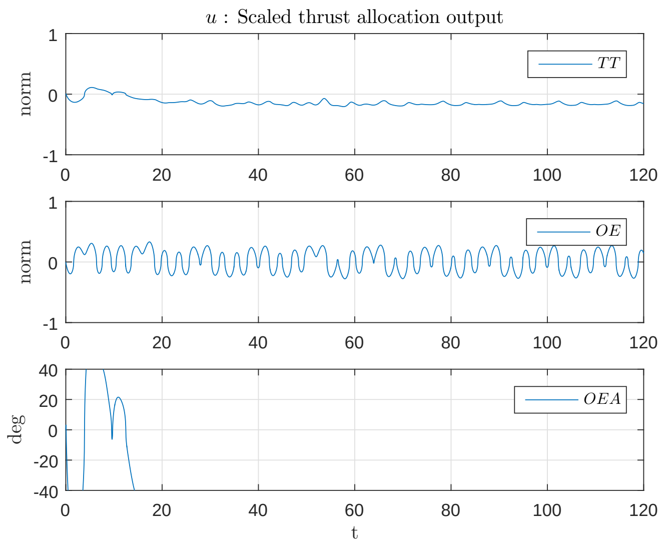


Figure 4.11: LSM: $\boldsymbol{u}$ - Perturbed by current, wind and waves ($\beta_{wc} = -\frac{\pi}{2}$)

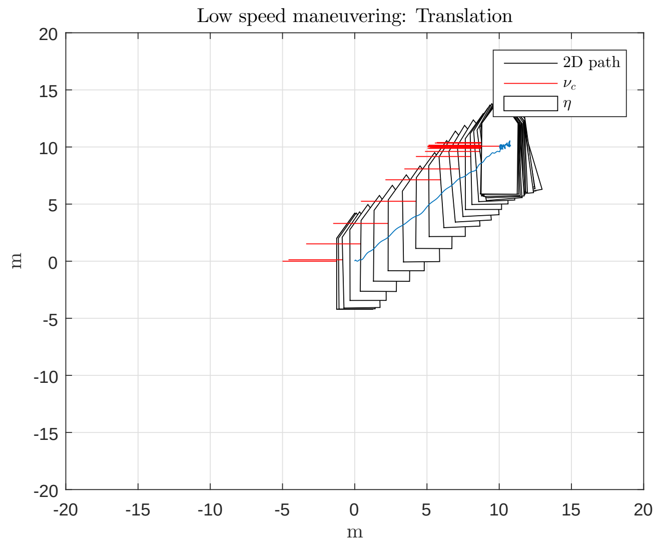


Figure 4.12: LSM: 2D Pose - Perturbed by current, wind and waves ($\beta_{wc} = -\frac{\pi}{2}$)

Low Speed Maneuvering: Rotation

The third test objective is low speed maneuvering, rotation maneuver.

The reference is set at $\boldsymbol{\eta}_r = [0, 0, \pi]$, and the external force vector angle is set at $\beta_{wc} = 0$

Here the test objective with environmental forces without waves are shown, due to the algorithm not converging on its reference. From figures 4.13 , 4.14 and 4.15, we see again that the vessel stops rotating when the environmental forces are perpendicular to the vessel's x-axis. The reason for this is shown in figure 4.16, which showes the output from the PID to the thrust allocation algorithm.

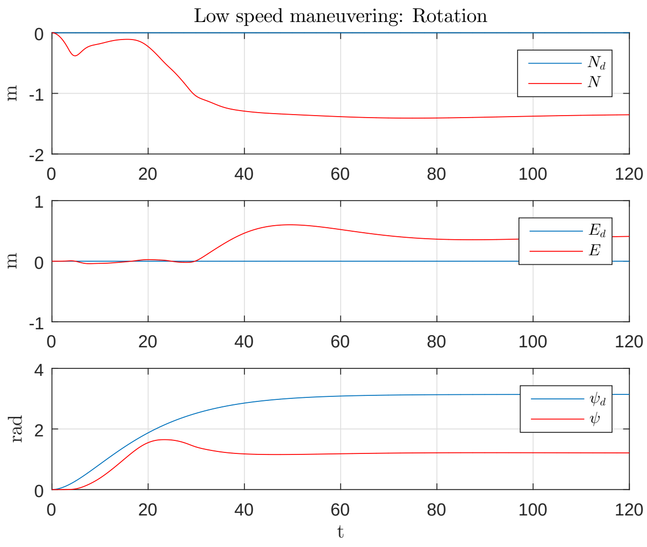


Figure 4.13: LSM: $\boldsymbol{\eta}$ - Perturbed by current, wind and waves ($\beta_{wc} = 0$)

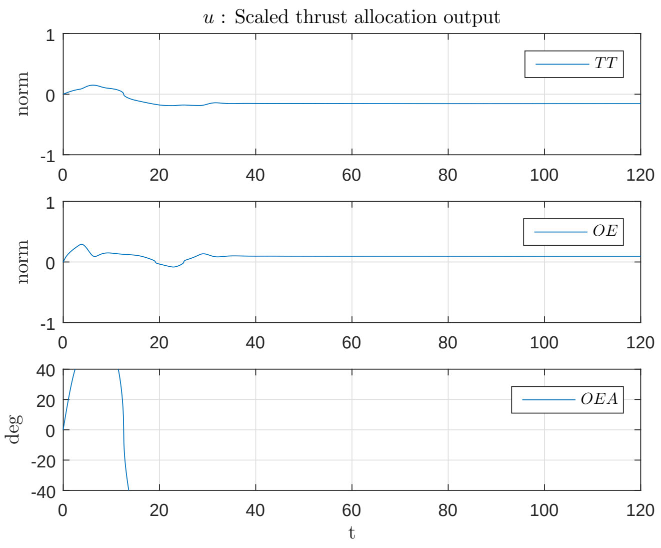


Figure 4.14: LSM: $\boldsymbol{u}$ - Perturbed by current, wind and waves ($\beta_{wc} = 0$)

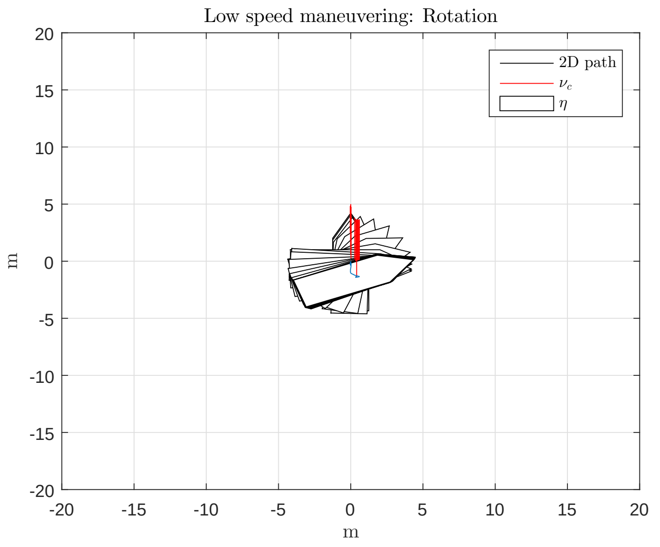


Figure 4.15: LSM: 2D Pose - Perturbed by current, wind and waves ($\beta_{wc} = 0$)

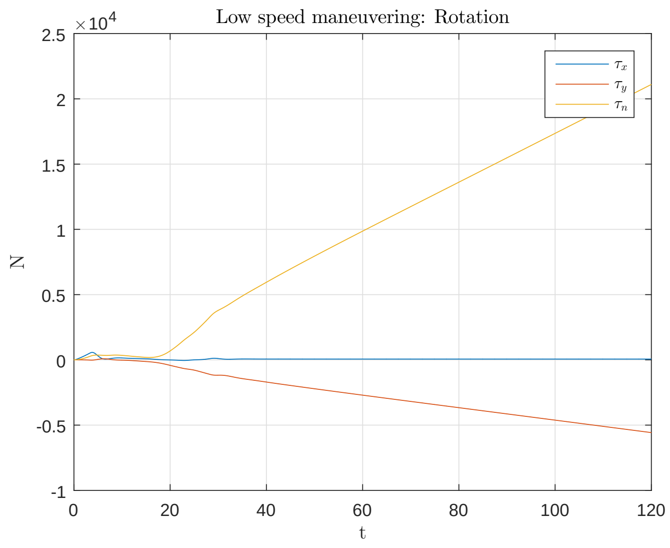


Figure 4.16: LSM: $\boldsymbol{\tau}$ - Perturbed by current, wind and waves ($\beta_{wc} = 0$)

Due to the way thrust is allocated, defined in algorithm 3.2, the thrust in sway and yaw cancel each other out when allocating thrust to the tunnel thruster, given as

$$T_{y1} = \frac{\tau_y l_2 - \tau_n}{l_2 - l_1} \tag{4.1}$$

As the thrust values in sway and yaw both changes, the relationship between them are almost constant, given as

$$\tau_y l_2 \approx \tau_n \tag{4.2}$$

This causes the thrust allocation algorithm to allocate very little thrust to the tunnel thruster (see figure 4.14), making it impossible to turn around any further. A way to fix this is to tune the yaw gain, so that yaw gets a higher priority when allocating thrust.

By doubling the natural frequency for yaw in a new pole placement, $\omega_{n,\psi}$, we get the results shown in figures 4.17 , 4.18 and 4.19 below, which also include wave perturbation.

From figures 4.17 , 4.18 and 4.19 we can now see that the vessel converges on its reference, and adequate thrust is located to the thunnel thruster. From figure 4.17 it might appear to be a large deviation, but this is just the yaw wrapping within $\pm\pi$. The largest standard deviation in position, along the x-axis, is calculated to $\bar{x}_n = 0.1685$.

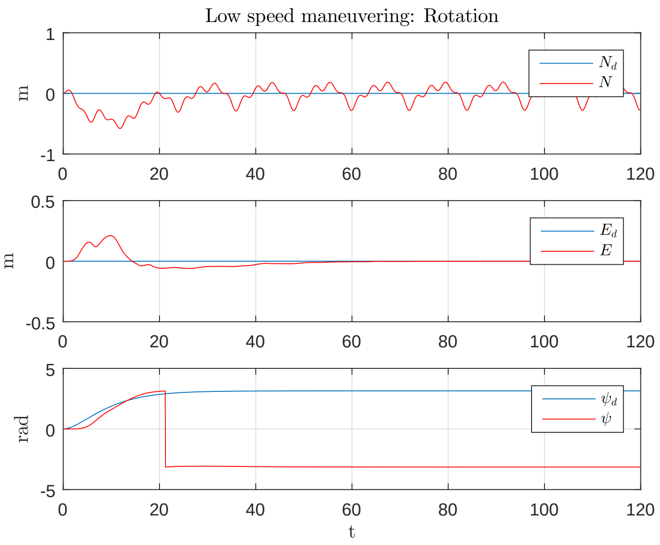


Figure 4.17: LSM: $\boldsymbol{\eta}$ - Perturbed by current, wind and waves ($\beta_{wc} = 0$)

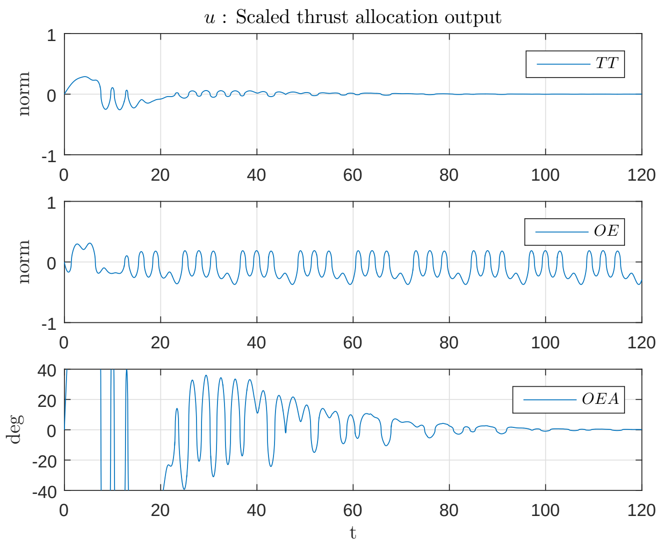


Figure 4.18: LSM: $\boldsymbol{u}$ - Perturbed by current, wind and waves ($\beta_{wc} = 0$)

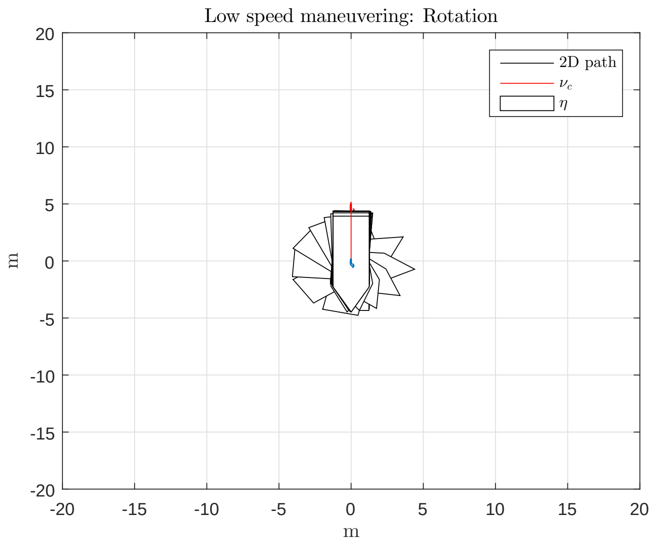


Figure 4.19: LSM: 2D Pose - Perturbed by current, wind and waves ($\beta_{wc} = 0$)

The next, and final, test sets the reference again at $\boldsymbol{\eta}_r = [0, 0, \pi]$, and the external force vector angle is set at $\beta_{wc} = -\frac{\pi}{2}$.

From figures 4.20 , 4.21 and 4.22 we can now see that the vessel converges on its reference, with some transient delay and initial stationary overshoot. The deviation in yaw is therefor more noticable this time, having a standard deviation of $\bar{\psi} = 0.3956$. The largest standard deviation in position, along the y-axis, is calculated to $\bar{x}_E = 0.5368$.

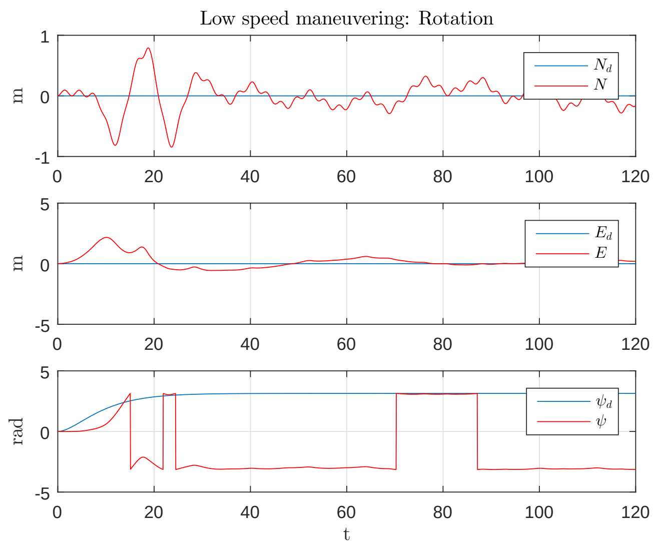


Figure 4.20: LSM: $\boldsymbol{\eta}$ - Perturbed by current, wind and waves ($\beta_{wc} = -\frac{\pi}{2}$)

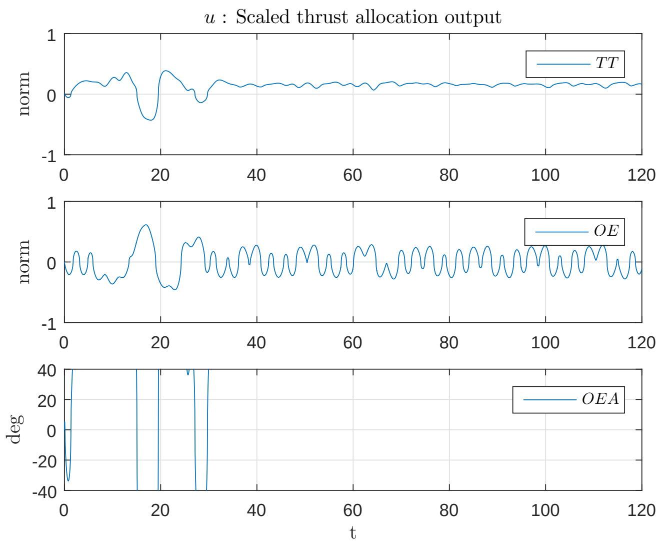


Figure 4.21: LSM: $\boldsymbol{u}$ - Perturbed by current, wind and waves ($\beta_{wc} = -\frac{\pi}{2}$)

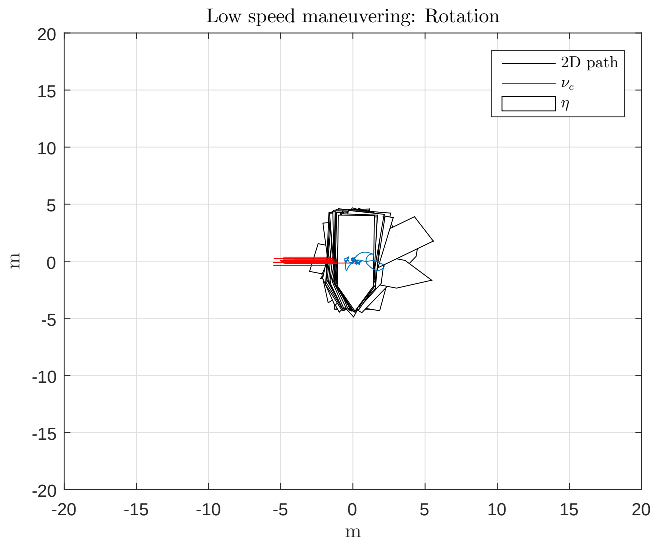


Figure 4.22: LSM: 2D Pose - Perturbed by current, wind and waves ($\beta_{wc} = -\frac{\pi}{2}$)

Discussion

The simulations, after retuning the yaw gain, has yielded adequate results. Since the DP control algorithm was tuned with pole placement based on the vessel model used in the simulations, it was expected that the system would be stable. However, actuator dynamics was not a part of this model, which was the main cause of the small deviations seen in the simulations, particularly with the environmental force vector forcing the actuators ouside their thrust region.

4.1.2 DP Application

The DP Application along with the vessel model application, presented in chapter 3, 3.3, where set up on separate computers, connected via ethernet cable.

The simulated results for the DP application were exact replicas of the results presented in the Matlab simulations (after retuning of the yaw gain). For this reason the resulting plots will not be shown.

Besides this, the DP Application was tested for the following functionalities

- Connecting and disconnecting to a vessel
- Transition from Manual to Auto mode, and vice versa
- Setting the reference
- Redefining the origin
- Online PID tuning
- Manual control with the manual controller
- 2D visualization of the local NED pose
- External control activation/deactivation

Minor bugs regarding closing and opening of pop-up windows where detected and fixed.

The DP application was developed by continuous integration and testing, which is the main reason for the lack of issues and bugs in the final simulation of the system.

4.2 Experiment

The experiments where conducted on the Telemetron vessel. The DP application was set up on a laptop and connected to the Telemetron framework with an ethernet cabel, aboard the vessel. A skipper was present for setting up the system for testing, as well as taking over control if the control algorithm became unstable or if anything else should happen.

All tests at sea are given a *Beaufaurt scale* number, indicating mean wind speed and wave amplitude (see appendix A.1).

Three tests were completed, each with different objectives.

- Test 1: Test DP application communication and functionality
- Test 2: Test DP and LSM objectives in light weather
- Test 3: Test DP and PSM objectives in moderate weather

Luckily, the weather turned out as planned, and all three tests were completed in full.

The actuator control set point limits for the Telemetron vessel actuators gives the control algorith maximum thrust from all actuators. As maximum thrust is not wanted, and is also not modelled and simulated, the thrust allocation output is rescaled where the tunnel thruster can only receive 25% of its maximum thrust, and the outboard engine can give 12.5% thrust in the forward direction and 25% in the reverse direction.

The reason for the difference in reverse and forward thrust for the outboard engine is due to the thrust available in reverse is significantly lower than forward thrust.

4.2.1 First Test: Communication and Functionality

Before testing the vessel in open ocean, a systems check to verify communication and actuator control must be performed. This was done with the vessel docked at Maritime Robotics's marina, as shown in the picture below.

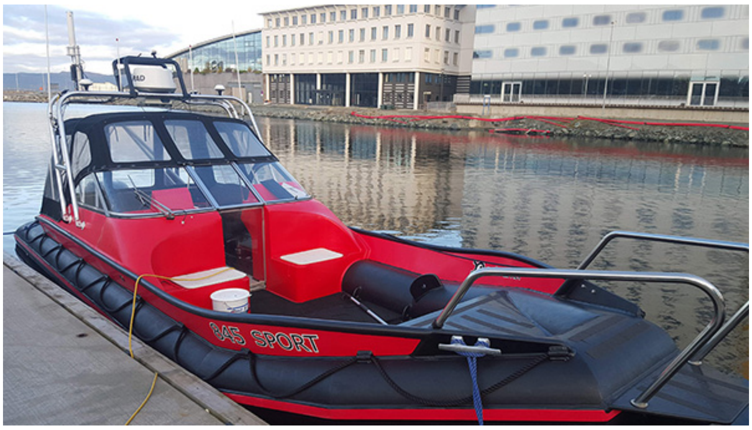


Figure 4.23: Test of communication and actuator control with docked vessel

Initial TCP/IP communcation test revealed an error in the TCP client, which didn't present itself in the simulation. The error caused the TCP client to open two sockets, one for sending data and one for receiving. This wasn't a problem when simulating, because the TCP server did the same thing, as they where designed in pair. This isn't the correct way to have bidirectional communcation. Instead, this should be done by using just one socket for both.

After resorting the client side problem, another issue occured, this time caused by the TCP server on the Telemetron vessel. Maritime Robotics fixed the problem, and a successfull connection was made between the DP application and the Telemetron vessel.

The actuator control allocation, by manual control, was tested. It was discovered that the outboard engine angle went the opposite way of what was expected. After changing the sign of the output from the allocated outboard engine angle, the issue was sorted.

A low speed test inside the marina was performed, to log data and verify that the data was correct. The vessel was controlled by the skipper, and the DP application only logged data. When analysing the data, it was discovered that there was a delay accumulating in the measurements received. The DP application therefor was debugged and tested. The error was found to be the TCP client. The TCP server and the TCP client run on different rates, the latter is slower than the former. This causes the buffer in the TCP socket to fill up with messages. Since the server was sending data at a higher frequency than we received it, this accumulated delay was proportional to the rate between these two frequencies. This was not discovered in the simulation, due to the server and client running at the same rate. To avoid this, every time the TCP client receives a navigation message, the buffer is then cleared and only the last message added to the buffer is used in the DP control algorithm each iteration, as it is the newest message from the vessel. The problem was fixed and the system was ready for testing DP and LSM functionalities.

4.2.2 Second Test: DP & LSM - Light Weather

After having resolved all the issues from the communication test, the DP application is ready for testing DP objectives in open ocean.

The tests performed are

- Dynamic Positioning: Station Keeping
- Low Speed Maneuvering: Translation and Rotation

The tests are done in Beaufaurt scale 1 weather, with wind speeds between 1-5 km/h and wave heights between 0-0.2 meters.

Dynamic Positioning

The first test objective is dynamic positioning, station keeping.

The reference is set at $\boldsymbol{\eta}_r = [0.32, 0.58, 0.55]$

From figures 4.24 , 4.25 and 4.26, we can see that the vessel maintains its reference position and heading, with some oscillations, but remaining stabile. The thrust allocations contains a lot of spikes and noise. The largest standard deviation, along the x-axis, is calculated to $\bar{x}_n = 0.4247$.

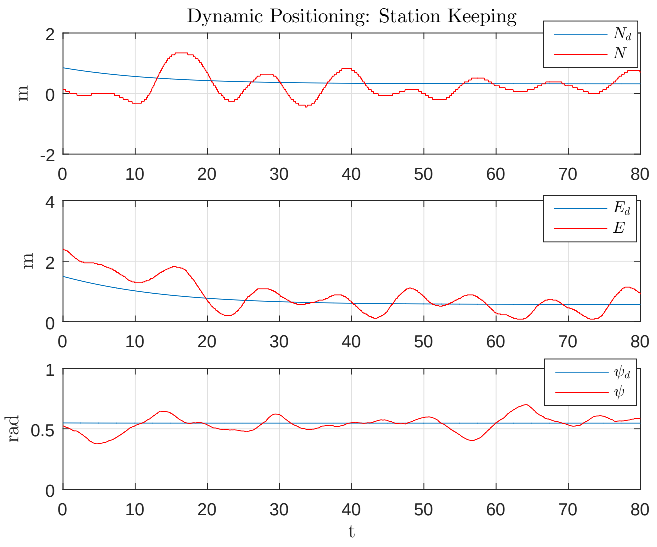


Figure 4.24: $\boldsymbol{\eta}$: Dynamic Positioning

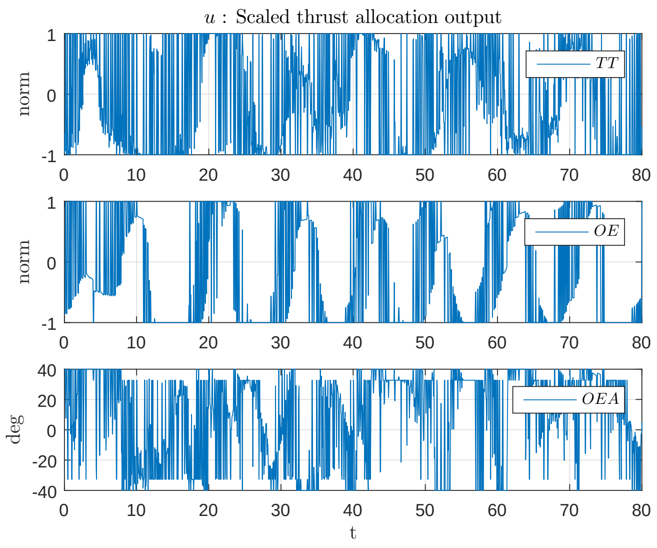


Figure 4.25: $\boldsymbol{u}$: Dynamic Positioning

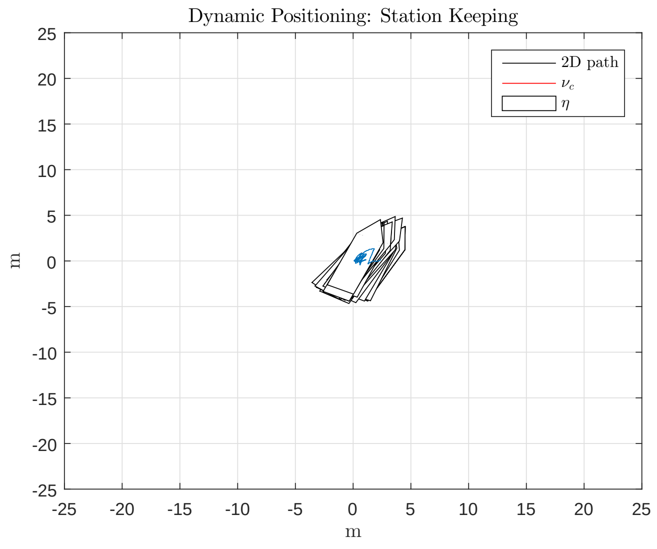


Figure 4.26: 2D Pose: Dynamic Positioning

Low Speed Maneuvering: Translation

The second test objective is low speed maneuvering, translation.

The reference is set at $\boldsymbol{\eta}_r = [10, 0, 0.3]$

From figures 4.27 , 4.28 and 4.29, we can see that the vessel follows its reference position and heading, with minor error in position and larger error in heading, but the system is stabile. The thrust allocations contains a lot of spikes and noise. The largest standard deviation in position, along the x-axis, is calculated to $\bar{x}_n = 1.0413$. The standard deviation in heading is calculated to $\bar{\psi}_n = 0.2874$.

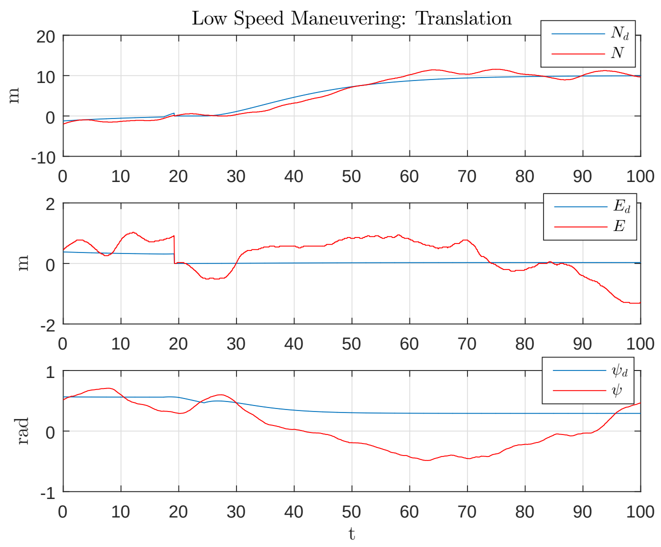


Figure 4.27: $\boldsymbol{\eta}$ Low Speed Maneuvering, translation

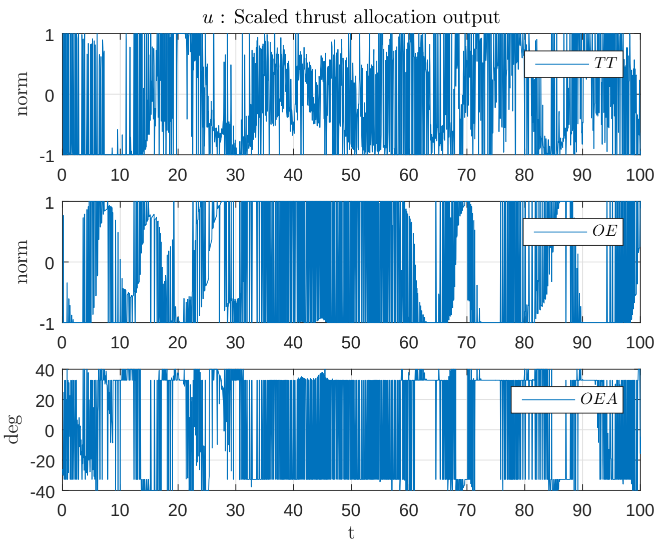


Figure 4.28: $\boldsymbol{u}$: Low Speed Maneuvering, translation

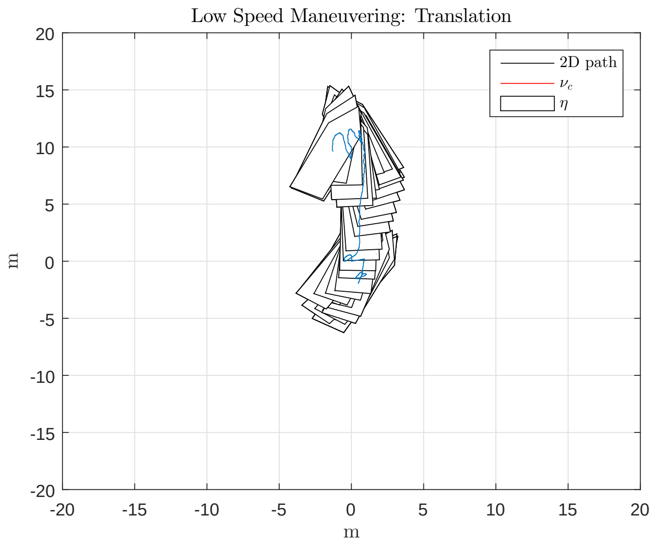


Figure 4.29: 2D Pose: Low Speed Maneuvering, translation

Low Speed Maneuvering: Rotation

The third test objective is low speed maneuvering, rotation.

The reference is set at $\boldsymbol{\eta}_r = [0, 0, \pi]$

From figures 4.30 , 4.31 and 4.32, we can see that the vessel follows its reference position and heading, with minor oscillations, but the system is stabile. The thrust allocations contains a lot of spikes and noise. The largest standard deviation in position, along the x-axis, is calculated to $\bar{x}_n = 0.6841$. The standard deviation in heading is calculated to $\bar{\psi}_n = 0.2898$.

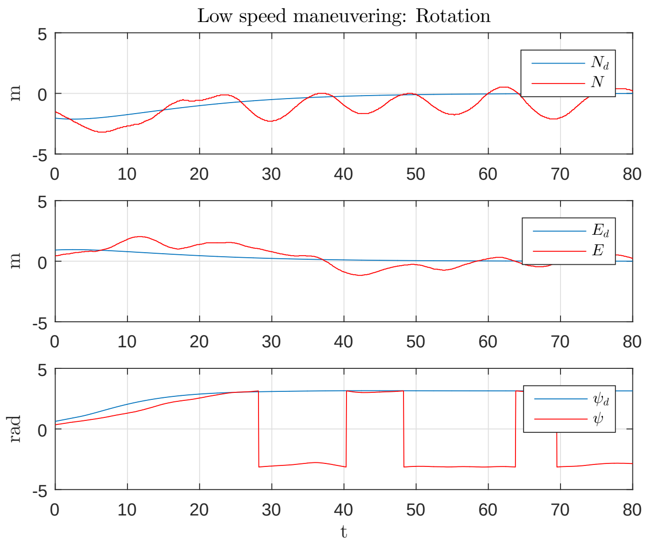


Figure 4.30: $\boldsymbol{\eta}$ Low Speed Maneuvering, rotation

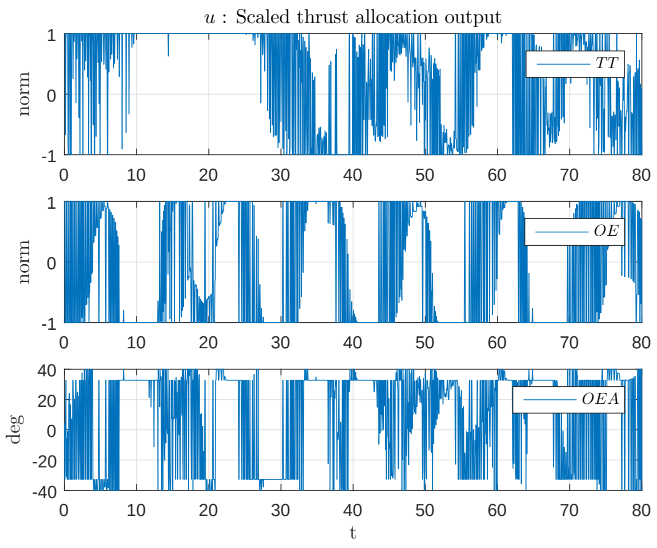


Figure 4.31: $\boldsymbol{u}$: Low Speed Maneuvering, rotation

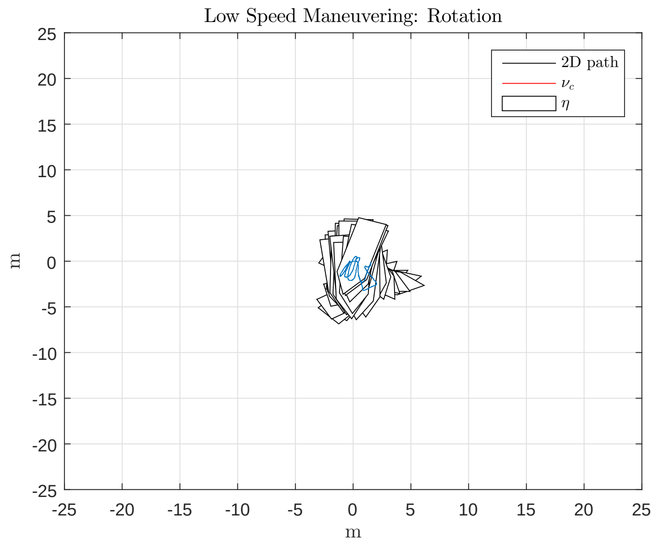


Figure 4.32: 2D Pose: Low Speed Maneuvering, rotation

Discussion

We can see from the figures 4.24, 4.27 and 4.30 that the oscillations have an approximate time period of $T_o = 10$ seconds. which is the same time period as our PID controllers time constants, $T_{t,x}$, $T_{t,y}$ and $T_{t,\psi}$. When a control loop oscillates with the same time constant as the controller, it is a sign that the proportional gain is too large [10]. Oscillations that have a larger time constant is a sign of the integral gain being too large, and the derivative gain for smaller time constants.

From the thrust allocation, $\boldsymbol{u}$, seen in figures 4.25, 4.28 and 4.31, we can see that there are alot of spikes and noise. By analysing the velocity, we can see that its the differentiation of the position to velocity that causes spikes in the output, as it is multiplied with derivate gain in the controller, as can be seen in figure 4.33.

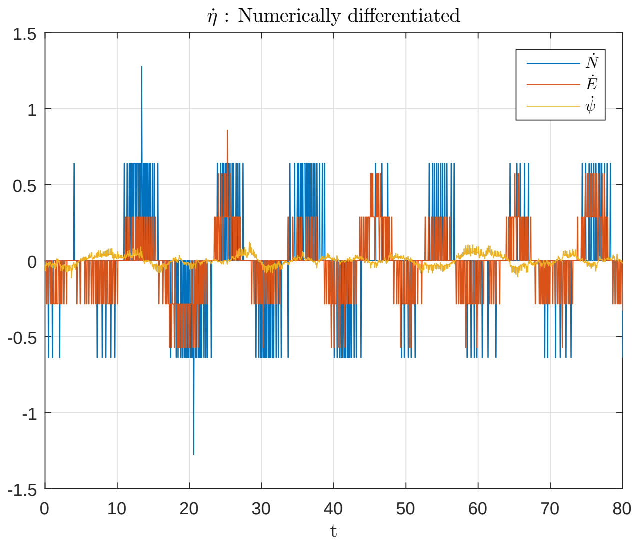


Figure 4.33: $\dot{\boldsymbol{\eta}}$ Dynamic Positioning

This is due to the velocity being numerically differentiated at each timestep, defined as

$$\dot{\boldsymbol{\eta}}(k) = \frac{\boldsymbol{\eta}(k) - \boldsymbol{\eta}(k-1)}{h} \tag{4.3}$$

where h is the timestep between each iteration of the PID algorithm. A solution to this is to low pass filter the differentiation.

A first order low pass filter, known as an "exponentially weighted moving average" filter, will be used. By first defining the filtered velocity state as $\hat{\dot{\boldsymbol{\eta}}}(k)$, we can define the low-pass filter as

$$\hat{\dot{\boldsymbol{\eta}}}(k) = \hat{\dot{\boldsymbol{\eta}}}(k-1) + \alpha_f \left[\dot{\boldsymbol{\eta}}(k) - \hat{\dot{\boldsymbol{\eta}}}(k-1) \right] \tag{4.4}$$

where α_f is the filter constant known as the *smoothing factor*, where $0 < \alpha_f < 1$. The filter constant is calculated based on the cut-off frequency of the filter, ω_c, and the timestep, h, of the filter, where a small value will give more noise filtering and a large value will give less. This relationship can be defined as

$$\alpha_f = \frac{\omega_c h}{\omega_c h + 1} \tag{4.5}$$

When choosing a cut-off frequency, it is important to have the cut-off frequency higher than the natural frequency of the system, to avoid filtering out the dynamics of the vessel and the controller. By setting the cut-off frequency five times higher than the natural frequencies off the controller, we can ensure that no dynamics in the system will be damped out by the filter, defined as

$$\omega_c = 5\omega_{n,pid} \tag{4.6}$$

One caveat of using a low-pass filter, is that they induce a delay to the signal it's filtering, which in turn could induce delay and instability into the system. The smaller the filter constant, the larger the delay will become. So when tuning the filter, one has to find the trade-off between good filtering with small enough delay.

By filtering the numerically differentiated velocity in post-processing of the DP test, we get a smoothed velocity estimate. This can be compared to indirectly calculated velocity, by using the course over ground (COG) and speed over ground (SOG) provided by the GPS, seen in figure 4.34.

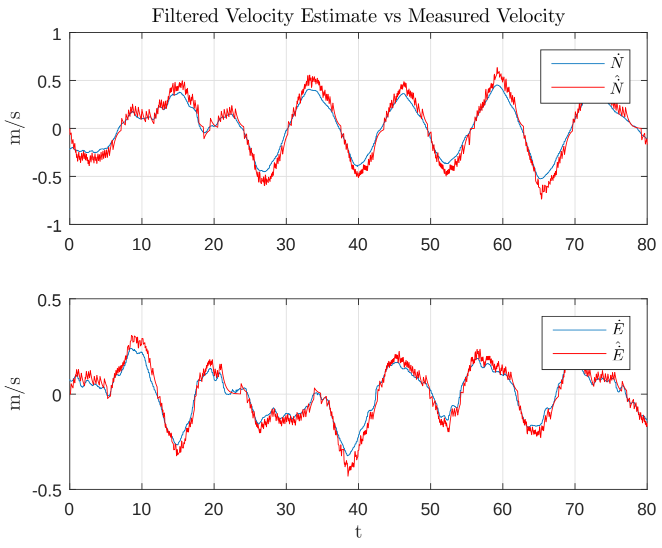


Figure 4.34: Filtered velocity estimate vs Measured velocity

We can see from the figure above that the smoothed velocity gives an accurate estimate of the measured velocity, with little noise and delay.

4.2.3 Third Test: DP & LSM - Moderate Weather

After having implemented the low pass filter for velocity estimation, the DP application is ready for testing DP objectives in open ocean. The former test discussion suggested that the proportional gain was too large, and should therefor be tuned accordingly. Since the former test had alot of noise induces into the system due to bad velocity estimates, this test will instead use the same controller gains as the first test, given from the pole placement (with retuned yaw gains). The reason being to see the difference in performance from the filtering, before making further alterations to the control algorithm dynamics.

The tests performed are

- Dynamic Positioning: Station Keeping
- Low Speed Maneuvering: Translation and Rotation

The tests are done in Beaufaurt scale 4 weather, with wind speeds between 20-28 km/h and wave heights between 1-2 meters.

Dynamic Positioning

The first test objective is dynamic positioning, station keeping.

The reference is set at $\boldsymbol{\eta}_r = [-5.2, 0, 1.85]$

From figures 4.35 , 4.36 and 4.37, we can see that the vessel maintains its reference position and heading, with some oscillations in position, but remaining stabile. The largest standard deviation in position, along the y-axis, is calculated to $\bar{y}_n = 0.7875$. The standard deviation in heading is calculated to $\bar{\psi}_n = 0.0433$

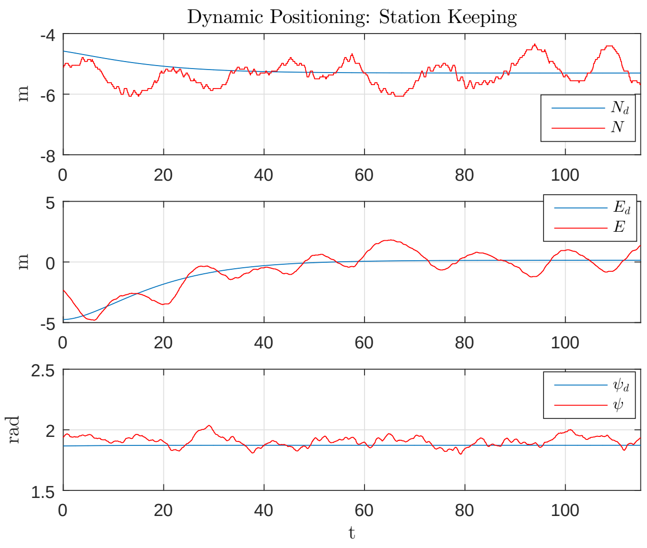


Figure 4.35: $\boldsymbol{\eta}$: Dynamic Positioning

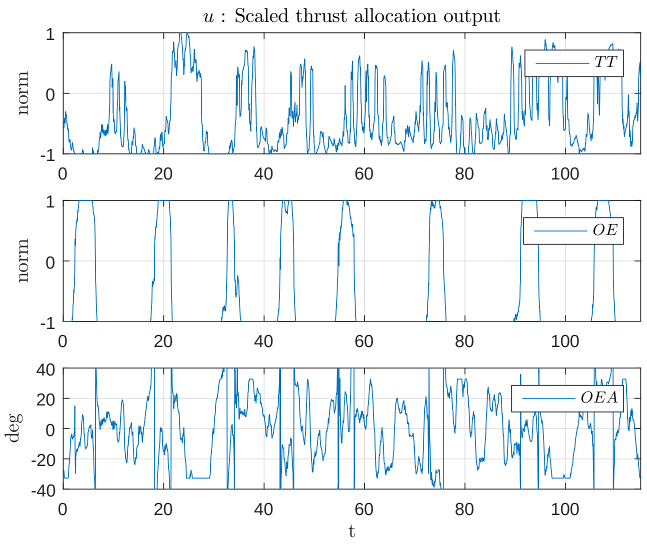


Figure 4.36: $\boldsymbol{u}$: Dynamic Positioning

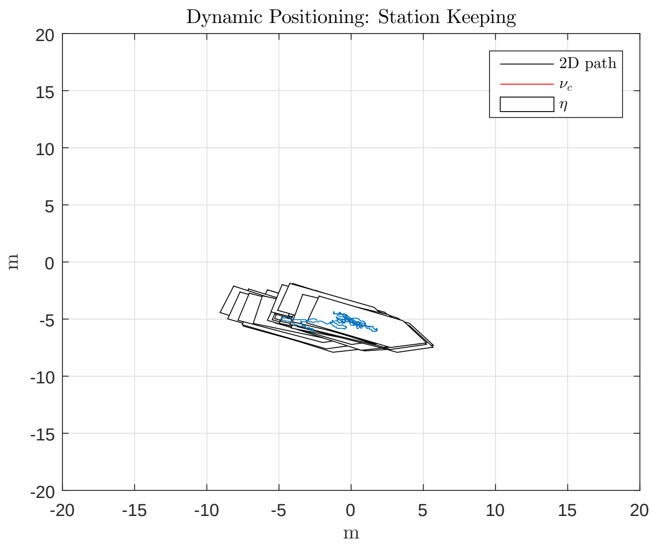


Figure 4.37: 2D Pose: Dynamic Positioning

Low Speed Maneuvering: Translation

The second test objective is low speed maneuvering, translation.

The reference is set at $\boldsymbol{\eta}_r = [20, 20, 0]$

From figures 4.38 , 4.39 and 4.40, we can see that the vessel follows its reference position, but with large deviations. The deviation in heading is large and stationary, never moving towards its reference. Despite this, the system is stabile. The largest standard deviation in position, along the y-axis, is calculated to $\bar{y}_n = 3.31$. The standard deviation in heading is calculated to $\bar{\psi}_n = 1.353$, however the stationary heading error is $\psi_e = -\pi$, which is the opposite direction of the heading reference.

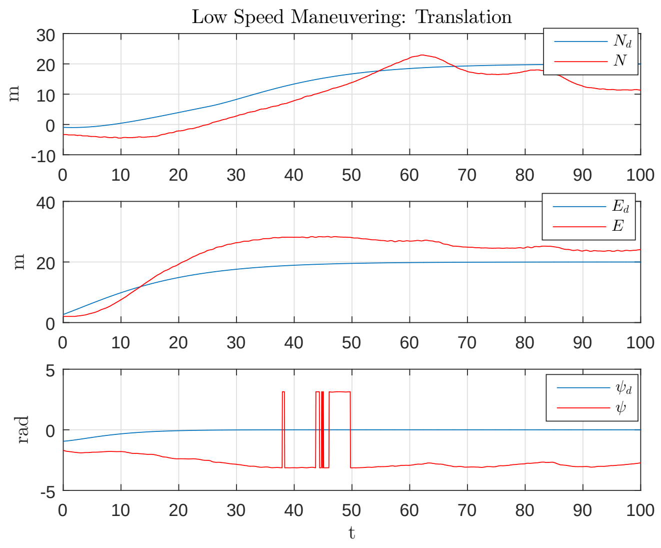


Figure 4.38: $\boldsymbol{\eta}$: Low Speed Maneuvering, translation

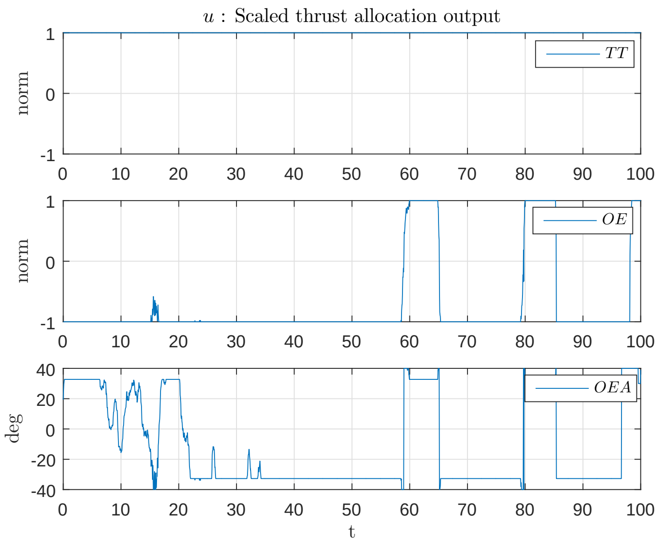


Figure 4.39: $\boldsymbol{u}$: Low Speed Maneuvering, translation

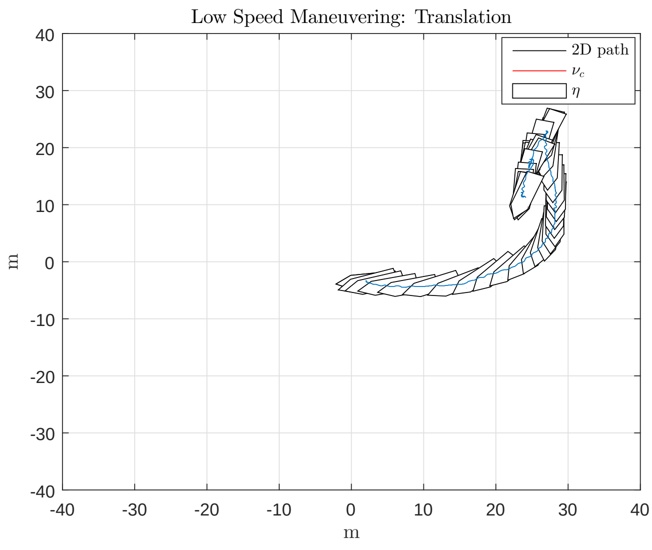


Figure 4.40: 2D Pose: Low Speed Maneuvering, translation

Low Speed Maneuvering: Rotation

Due to lack of fuel, the rotation maneuver was not tested.

Discussion

From figure 4.35, we now see two types of oscillations; small amplitude with short time time period, and large amplitude with longer time period. The latter is the oscillations we saw from the second test run, where we got standing proportional gain oscillations, due to the proportional gain being too large. The smaller oscillations, due to rougher weather, are caused by the waves perturbing the system. From figures 4.39 and 4.36, we see that these smaller oscillations caused by the waves are inducing some spikes in the thrust allocation output, although not not of any significance when compared to when the velocity was unfiltered.

The performance was also reduced in general due to the heavier weather conditions, clearly seen in the low speed translation maneuver in figure 4.38, where the heading ends up with a stationary error close to the maximum possible error, at $\pm\pi$. From figure 4.39, we can see that the outputs are saturated, and the vessel isn't able overcome the external forces perturbing the system. This can be fixed by scaling up available output to both the tunnel thruster and the outboard engine thrust. The tunnel thruster was limited to 25% if its maximum output, and the outboard engine was limited to 12.5% of its maximum value in the forward surge direction, and 25% in the reverse surge direction. The reason being that the engine produces more torque in forward gear than in reverse gear. This problem was, however, not discovered in the second test run, due to external forces being close to non-existing.

Unfortunately the third test was the final test for this thesis, and there was no re-test performed with the proportional gain re-tuned, to remove the standing oscillations in the pose measurements. Nor was there a re-test with the thrust allocation output limits scaled up, to achieve better performance with large external forces perturbing the system.

Chapter 5

Discussion and Conclusion

This chapter will feature a discussion of the results in chapter 4, followed by a conclusion, and recommendations for future work.

5.1 Discussion

A small discussion for every results was given in chapter 4, due to iterative testing that required analysis of the results in between iterations. In this chapter, the discussion will focus more on the big picture, bringing together all the results for a final analysis, shedding light on the strenghts and limitations the results provided, set up against the thesis objectives.

5.1.1 Simulation

Explained in section 3.2.4, the DP control algorithm was tuned after the vessel model. For this reason, when simulating the system with the vessel model, a certain prediction could be made about the simulation yielding positive results. However, as the vessel was perturbed by simulated environmental forces, along with the delay when changing gear, the algorithm had to be retuned to provide a large enough yaw moment to overcome the environemental forces when turning the vessel past an environmental force vector standing perpendicular to the surge direction, as shown in simulation result 4.1.1. After re-tuning the DP control algorithm, all the control objectives were achieved.

We could see from the simulation results that the wave perturbation affected the vessel, which in turn caused minor oscillations in the thrust allocation output. From[13] [6], we know that the average wave period within feasible weather conditions for the DP control algorithm is $1 < T_w < 6$ seconds. As the time constant for the vessel model is larger than this (see appendix A.2), the waves will be slightly damped by the vessel itself. However, as the the controler time constants being 5-10 times larger than the vessel time constants (see appendix A.2), this puts the time constants of the controller right in the span of average wave time periods, making it very hard to filter out the waves by moving the poles, and consequently the time constants, of the controller, as anything lower than the system dynamics will cause the system to become unstabile [2].

The waves could be filtered to some extent with a low pass filter, but since the cut-off frequency of the filter would have to be somewhere between the natural frequency of the vessel and the frequency of the waves. Since the time constants of the vessel is 10 times larger than the time constants of the smallest wave time period, and a first order filter reduces the wave amplitude by -20 dB per decade, the largest reduction would then be -20dB. Another consern is the delay induced by a filter. However, a wave filter is not within the scope of this thesis, but for future work it would be worth looking in to.

5.1.2 Experiment

The light sea state experiment gave good results for all control objectives. Although there where spikes and noise present in the thrust allocation output, induced by unfiltered velocity calculations. There where also standing oscillations with a time period equal to the time constant of the control system, which indicates a proportional gain oscillation [10], as mentioned in section 4.2. A good rule of thumb is to change one thing at a time in a control system, to be sure of which change is causing what [10]. For this reason the controller was not re-tuned to compensate for the proportional gain oscillations, due to the velocity calculations being filtered and tested first.

The moderate sea state experiment, accomplished the station keeping objective, but failed to accomplish the translational low speed maneuver. The reason for this could be seen from thrust allocation plots, which showed saturated outputs. As explained the output of the thrust allocation was scaled down, as the default limits gave us the maximum thrust available for both thrusters on the vessel. This problem was not induced in the light sea state experiment, as the environmental forces was close to non-existing, with no wind, waves and little current. This could be fixed by scaling up the output of the thrust allocation limit. These limits could also be found by running tests on the actuators prior to the experiment, as to identify their dynamic properties with greater precision than the approximations made in this thesis.

The increase in wave height also had an effect on the performance, as one could clearly see small oscillations in the position and heading, which in turn induced some noise in the thrust allocation output. Even though the mean wave height was between

1 to 2 meters, the effect on the measurement was less noticable than the wave perturbations made in our simulated results, shown in section 4.1.1. Which goes to show that the dynamics of the ship dampens out the waves to some extent, as it's natural frequency is lower than the average frequencies of the waves. Another reason for this could also be how the GPS and compass processes its data before sending it, where it is often typical to filter out noise from the measurements.

An issue that wasn't discussed in the results chapter, due to not being noticed until after all the experiements where over, is that there is a bug in the reference model in the DP application, causing the reference to converge towards the reference set point when station keeping is attempted. The reference should instad have been fixed at the reference set point defined in each test. The bug is caused due to never resetting the velocity term in the reference model when changing the reference. This makes the reference in position and heading to converge towards the new reference from the old reference state in the reference model, instead of converging to the new reference from the current state of the vessel. This can be seen in figures 4.24, 4.35. The bug is easely fixed by simply setting the velocity term in the reference model to zero everytime the reference is changed. In this way, the reference model will converge towards the reference specified from the position and heading the vessel is currently at, instead from the positon and heading reference it the reference model is currently at.

5.2 Conclusion

The main objectives for this thesis was to develop a stand-alone DP application that could interface with and control the Telemetron vessel, a small underactuated surface vessel, to achieve dynamic positioning and low speed manvuering control objectives. A DP application was made in Qt, programmed in C++, able to communicate with and control the Telemetron vessel via TCP/IP connection. A DP control algorithm was prototyped and tested in Matlab, then ported to the the DP Application. The DP application was tested in both simulated and real experiments, achieving all DP and LSM control objectives in the simulated results, as well as achieving most DP and LSM control objectives in the experimental results, with minor proportional gain oscillations. It failed to achieve LSM objectives in a moderate sea state due to limitations in the thrust allocation output being set too low.

The DP control algorithm was tuned based on an approximated model of the Telemetron vessel, based on the Viknes vessel parametrization, augmentated with parameteres from the Telemetron vessel. The Telemetron parameters used was physical parameters, such as weight, dimensions, thruster locations, which makes the approximated modelling possible without the need for system identification by running dynamic response testing on the vessel. Despite using a simplified model, the performance of the initial pole placement yielded acceptable results in the experiment.

The waves effect on the measurements and its effect on the controller was minimial, leading to the conclusion that there is no need for wave filtering, nor is it possible to compensate for these effects with the controller, as both the dynamics of the vessel and the controller are slower than the dynamics of the waves.

5.3 Recommendations for Future Work

Although the solution in this thesis was proven relatively successful, there are still a lot that can be done to improve upon this solution. Below is an overview of all proposed improvements and recommendations for future work, each with their own section explaining the concept in greater detail.

Future recommendations:

- Retune and test
  - Reduce proportional gain
  - Scale up thrust allocation output limits
- Manual control
  - Control by force vector
  - Control with Xbox or PS3 controller
- System identification
  - Actuator Dynamics
  - Vessel Dynamics
- Control law

5.3.1 Retune and Test

Based on the experimental test results, a proposition for a new test has been suggested, with a rescaling of the thrust allocation output, as well as a re-tuning of the proportional gain being implemented. This is needed to both overcome the problem with saturated thrust allocation with large environemental forces perturbing the system, as well as reduce the proportional gain oscillations, respectively. The DP and LSM control objectives should then be re-tested in both light and moderate sea states.

5.3.2 Manual Control

The DP application lets the operator manually control the actuators directly, which is ok for a vessel with a small amount of actuators. However, it would also practical for the operator to control by setting the thrust vector, $\boldsymbol{\tau}$, directly, such that the thrust allocation algorithm takes care of the actuator set points instead. This makes controlling the vessel alot more intuitive, as well as making it possible to scale up with more actuators, without changing how the operator manually controls the vessel. Another control mode could also be to have the control algorithm control the yaw motion of the vessel, while the operator can control the surge and sway motion of the vessel, allowing for more complex motion planning strategies.

Now, the manual controller is controlled by moving sliders on a separate window in the GUI, as shown in figure 3.8 in chapter 3. A much more intuitive and user friendly controller would be to use a commercially available *Xbox 360* or *Playstation 3* controller. The reason for this is that the controllers are designed to be ergonomical and intuitive. They also have a incredibly large user base that know how to use the controllers, which also means the controllers have been tested and verified by millions of people. This again makes training easier, as there is a large chance that an operator knows how to use the controller from before, by playing video console games.

The DP application would then have to be augmented with another layer for communicating with the controller. For windows operating systems, it is common to use *Device Input* libraries. For linux, this is achieved by accessing the device input addresses, and by using one of the many open source libraries already available for PS3 and Xbox360 controllers.

5.3.3 System identification

This thesis has not focused on finding any system identification of the Telemetron vessel, but instead using an approximated model based on augmenting a similar vessel model, as well for the actuator dynamics.

Even though the tests gave reasonable results based on these approximations, the results could have been even better by having done some simple tests to parametrize the vessel model and actuators to some extent.

The approximated model could also have been improved by including the coriolis forces in the model. These forces can be calculated purely based on the mass and inertia of the the vessel [4], and is possible to implement without the need for more parameters.

5.3.4 Control Law

The control law derived in this thesis is a simple nonlinear PID controller. Improving upon this controller would mean having to implement a more complex controller, like an LQR, MPC, or by using methods such as integral backstepping or nonlinear feedback, most of which are derived in [4]. Some of these controllers also rely on having a more accurate model of the system, and some even require us to perform state estimation of the vessel states that we can't measure directly. This would most likely improve the controllers performance, but at the same time create a much more complex system.

The nonlinear PID is simple and robust, and since this thesis focus was to achieve DP control objectives by means of simple control laws, it will not be recommended that the control law is changed to a more complex one.

However, had the results of this thesis proved negative, and a simple control law was not able to achieve the DP control objectives, then there wuold have been need to investigate other control laws and techniques, as those mentioned above.

Appendix A

Parameters

A.1 Beaufaurt Scale

The Beaufaurt scale goes from 0 to 12, but only the first 7 states will be presented here, as anything outside of that will render a DP system useless on a small vessel, such as the Telemetron vessel.

Table A.1: Beaufaurt Scale - Weather conditions

Beaufaurt Scale	Description	Wind Speed	Wave Height
0	Calm	< 1 km/h	0 m
1	Light air	1 - 5 km/h	0 - 0.2 m
2	Light breeze	6 - 11 km/h	0.2 - 0.5 m
3	Gentle breeze	12 - 19 km/h	0.5 - 1 m
4	Moderate breeze	20 - 28 km/h	1 - 2 m
5	Fresh breeze	29 - 38 km/h	2 -3 m
6	Strong breeze	38 - 49 km/h	3 - 4 m

A.2 Parameter Values

Code A.1: constants.m

```
% PROPULSION SYSTEM
% Unit 1: Bow tunnel thruster
D1X     = 2.36; % x−coordinate tunnel thruster, positive forward (m)
U1MAX = 600.0;  % Max RPM (%)
FORCECOEFF1= 0.00670;   % F1 = force_coeff_1 ∗ u1 ∗ abs(u1)
T1_ACT = 0.5;   % Actuator time constant (s)
% Unit 2: Stern outboard engine
D2X = −3.92; % x−coordinate propeller, positive forward (m)
D2Y = 0.0; % y−coordinate propeller, positive starboard (m)
U2MAX = 1000.0; % Max RPM (%)
FORCECOEFF2 = 0.00672; % F2 = force_coeff_2 ∗ u2 ∗ abs(u2)
T2_ACT = 0.5; % Actuator time constant (s)
% Additional parameters for outboard engine
DELTA_MAX = 40.0∗D2R; % Max azimuth angle (rad)
T_ANGLE = 0.1; % turn rate time constant (s)

% VESSEL DATA
LPP = 8.45; % Length between perpendiculars (m)
B = 2.71; % Beam (m)
T = 0.76; % Draft (m)
MASS = 2400.0;  % Mass (kg)
GRAV = 9.81;  % Acceleration of gravity (m/s^2)
R66 = 0.25∗LPP;
XG = 0.8;  % Forward center of gravity (m)
YG = 0.0;  % Starboard center of gravity (m)
```

```
ZG = -0.5; % Downwards center of gravity (m)
XB = 0.8;  % Forward center of buoyance (m)
YB = 0.0;  % Starboard center of buoyance (m)
ZB = T/3.0;  % Downwards center of buoyance (m)
A11 = 0.1*MASS; % Added mass in the center of buoyancy
A22 = 0.4*MASS;
A66 = 0.5*MASS*R66*R66;
% Mass included added mass in surge, sway and yaw
M_SURGE = MASS + A11;   % kg
M_SWAY = MASS + A22;   % kg
M_YAW = MASS + A66;   % kg m^2
% Time constants
T_SURGE = 10.0;
T_SWAY = 10.0;
T_YAW = 10.0;

% DP CONTROL SYSTEM
% Natural frequencies in surge, sway and yaw
WN_SURGE = 0.5; % rad/s, -> 5/T_SURGE
WN_SWAY = 0.5;  % rad/s, -> 5/T_SWAY
WN_YAW = 1;     % rad/s, -> 10/T_YAW, retuned from WN_YAW = 0.5
% Relative damping ratios in surge, sway and yaw
ZETA_SURGE = 1;
ZETA_SWAY = 1;
ZETA_YAW = 1;
% Integral wind-up limits: max thrust in surge, sway and yaw
TAU_MAX_SURGE = 10000;  % N   (10 kN)
```

```
TAU_MAX_SWAY = 10000; % N   (10 kN)
TAU_MAX_YAW = 10000*LPP*2/3; % Nm

% ENVIRONMENTAL FORCES
V_CURRENT = 0.5; % Current speed (m/s)
BETA_CURRENT = pi/2 % Current direction--> going to (rad)
T_w = 3 % Wave time period (sec)
```

Appendix B

Code

B.1 Matlab

See digital file appendix named "Matlab".

This folder contains all Matlab files related to the project, along with recorded data from the experiments and figures used in the thesis.

The file named "Simulation.m" is the main file, which can be used to run and reproduce the results found in the thesis.

The file named "constants.m" containts all the parameters used in the vessel model, as well as for the DP control algorithm.

B.2 Qt

See digital file appendix named "Qt".

This contains two folders

- Qt Applications - Executables
- Qt Projects - Code

The last one contains the entire codebase of the DP application and vessel model application, which needs Qt 5.5 and the Eigen library to compile.

The first one contains two deployed executables named

- "DP Application.exe"
- "Vessel Model.exe"

The rest are *.dll* files needed to run the software. Start both executables on the same computer for local testing, or start each on a separate computer and connect via ethernet or wifi. The port number for the vessel model is the default value presented in the client application.

Bibliography

[1] Araki, M. (2010). Pid control.

[2] Chen, C.-T. (1999). *Linear System Theory and Design*. Oxfor University Press, State University, New York.

[3] Engines, C. Cummins engines - propulsion systems. https://cumminsengines.com/propulsion-systems.

[4] Fossen, T. I. (2014). *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley, Hoboken, NJ.

[5] Halvorsen, H. (2008). Dynamic positioning for unmanned surface vehicles.

[6] Hasselmann, e. a. (1973). Measurements of wind-wave growth and swell decay during the joint north sea wave project (jonswap).

[7] Hofmann-Wellenhof, t. (1994). *Global Positioning System: Theory and Practice*. Springer, New York, NY.

[8] IPS, V. P. Volvo penta evc system - dynamic positioning. http:

//www.volvopenta.com/SiteCollectionDocuments/Penta/EVC/Eng/Dynamic%20Positioning.pdf.

[9] Johansen, T.-A. (2015). Low-cost dp system.

[10] Kåre Bjørvik, P. H. (2010). *Reguleringsteknikk*. Kybernetes Forlag, Sør-Trøndelag, Trondheim.

[11] Mark W. Spong, Seth Hutchinson, M. V. (2006). *Robot Modeling and Control*. Wiley.

[12] Olav Egeland, T. G. (2003). *Modeling and Simulation for Automatic Contrll*. Marine Cybernetics, Trondheim, Norway.

[13] Pierson, e. a. (1973). Proposed spectral form for fully developed wind seas based on the similarity theory of s. a. kitaigorodskii.