NTNU
Norwegian University of
Science and Technology

Master's thesis
2019
Håvard Snefjellå Løvås

Master's thesis

NTNU
Norwegian University of
Science and Technology
Faculty of Engineering
Department of Marine Technology

Håvard Snefjellå Løvås

DP Autotuning by use of Derivative-free Optimization

June 2019

NTNU
Norwegian University of
Science and Technology

DP Autotuning by use of Derivative-free Optimization

Håvard Snefjellå Løvås

Marine Technology
Submission date: June 2019
Supervisor: Roger Skjetne
Co-supervisor: Jon Bjørnø and Øyvind Kjerstad

Norwegian University of Science and Technology
Department of Marine Technology

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

MSC THESIS DESCRIPTION SHEET

Name of the candidate:	Løvås, Håvard
Field of study:	Marine control engineering
Thesis title (Norwegian):	DP autotuning ved bruk av gradientfri optimalisering
Thesis title (English):	DP autotuning by use of derivative-free optimization

Background
Tuning of multi-input multi-output (MIMO) industrial control systems is always a difficult task that normally are solved by experienced engineers before system startup. Dynamic Positioning (DP) control systems is an example of this, where the DP controller for a ship is tuned on sea trials during many lengthy tests. With new methods available from optimization and artificial intelligence, such as genetic algorithms, particle swarm optimization, etc., it is today likely that a computer by use of an appropriate AI algorithm will be able to do a better job than the human operator for the tuning task. This leads further to more autonomous DP control systems, where possibly the DP gains can be better updated by an artificial intelligence for given (and changing) prevailing conditions.

The objective of this thesis is to study socalled derivative-free optimization (DFO) methods for DP tuning, and compare and contrast these with respect to complexity of parameterization, computational efficiency, and challenges for practical use on a real vessel, where a chosen DFO method shall be implemented for the C/S Inocean Cat I Drillship (CSAD) in MC-Lab.

Work description
1) Perform a background and literature review to provide information and relevant references on:
   - DP control system, control law, observer, etc.
   - Derivative-free optimization methods.
   - PID-tuning.
   - Cybership test platform in MC-Lab.

   Write a list with abbreviations and definitions of terms, explaining relevant concepts related to the literature study and project assignment.

2) Present your case study and formulate the DP autotuning problem:
   a) Vessel, DP system setup, simulation model, experimental setup.
   b) DP control law for tuning.
   c) DP observer – assumption on full-state feedback or using an observer?
   d) Inputs and outputs for the DFO tuning module.
   e) Provide a flow-chart that illustrates how the optimization loop is run and results evaluated.

3) Using your favorite performance function for optimization, run, compare, and evaluate different derivative-free optimization methods on your simulation model. One such method shall be `fminsearch` in Matlab. In addition, you should select at least two more methods DFO methods:
   a) Start with tuning manually by trial and error and present the resulting DP gains as your “manual baseline”.
   b) For each optimization method, run, present, and evaluate the resulting gains and achieved performance.
   c) For each method, evaluate the complexity of parameterization, computational efficiency, global vs. local minima, ability to handle constraints, ability to pre-train on historic data, challenges for practical use on a real vessel, and conclude on an optimization method.

4) Propose a set of optimization performance functions and overall performance evaluation. Using your favorite optimization method and DP tuning on the simulation model, perform autotuning with the different performance functions, evaluate and discuss the results, and illustrate how these performance functions provide different results. Explain why.
   a) Conclude on performance function to use for comparing different optimization methods later.

5) Select the method you feel is best for practical testing and implement on CSAD in MC-Lab as a proof of concept. Present the experimental setup and any issues regarding initialization of the optimization, pre-training, etc. Run the experiments, both in calm conditions and at least one wave condition. Analyze, present, and discuss the results.

Specifications

The scope of work may prove to be larger than initially anticipated. By the approval from the supervisor, described topics may be deleted or reduced in extent without consequences with regard to grading.

The candidate shall present personal contribution to the resolution of problems within the scope of work. Theories and conclusions should be based on mathematical derivations and logic reasoning identifying the various steps in the deduction.

The report shall be organized in a logical structure to give a clear exposition of background, results, assessments, and conclusions. The text should be brief and to the point, with a clear language. Rigorous mathematical deductions and illustrating figures are preferred over lengthy textual descriptions. The report shall have font size 11 pts., and it is not expected to be longer than 70 A4-pages, 100 B5-pages, from introduction to conclusion, unless otherwise agreed upon. It shall be written in English (preferably US) and contain the following elements: Title page, abstract, acknowledgements, thesis specification, list of symbols and acronyms, table of contents, introduction with objective, background, and scope and delimitations, main body with problem formulations, derivations/developments and results, conclusions with recommendations for further work, references, and optional appendices. All figures, tables, and equations shall be numerated. The original contribution of the candidate and material taken from other sources shall be clearly identified. Work from other sources shall be properly acknowledged using quotations and a Harvard citation style (e.g. *natbib* Latex package). The work is expected to be conducted in an honest and ethical manner, without any sort of plagiarism and misconduct. Such practice is taken very seriously by the university and will have consequences. NTNU can use the results freely in research and teaching by proper referencing, unless otherwise agreed upon.

The thesis shall be submitted with an electronic copy to the main supervisor and department according to NTNU administrative procedures. The final revised version of this thesis description shall be included after the title page. Computer code, pictures, videos, dataseries, etc., shall be included electronically with the report.

Start date: January, 2019 **Due date:** As specified by the administration.

Supervisor: Roger Skjetne
Co-advisor(s): Jon Bjørnø (on CSAD issues), Øivind K. Kjerstad (KM).

Trondheim, 22.06.2019

____
Roger Skjetne
Supervisor

Preface

This master thesis is written as part of the study program Marine Technology at the Norwegian University of Science and Technology (NTNU). The work done in the thesis has been done using their testing facility, the Marine Cybernetics Laboratory (MC-Lab). The thesis presents a literature review of dynamic positioning (DP) and derivative-free optimization (DFO), derives an autotuning methodology for tuning a DP controller and implements the concept on the newest vessel in the MC-Lab fleet, the C/S Inocean Cat I drillship (CSAD).

The thesis has been fascinating to work with, and I feel fortunate to get to work with real-world experiments. It has been challenging at times, but it felt rewarding when the autotuning concept worked in practice. Through the process, I have gained knowledge about marine control systems in terms of theory and practical implementation, which has boosted my interest in the field.

The reader should preferably know hydrodynamics, marine cybernetics, and general control theory.

Acknowledgments

Throughout the project, there have been many contributors to helping in different ways.

Firstly, I would like to give a special thanks to my supervisor, Roger Skjetne, for guidance throughout the last year.

I would also like to thank for the help from my co-advisors, Øyvind K. Kjerstad at Kongsberg Maritime and Jon Bjørnø for all sorts of practical issues. Jon has spent countless hours helping me with hardware and software in Marine Cybernetics Laboratory, as well as many other problems. Jon and I were also fortunate enough to present C/S Inocean Cat I Drillship for His Majesty King Harald V during the NTNU Oceans in May.

Also, I would like to thank the other students for their collaboration on CSAD, Sondre Haug, Edvard Flaatten and Aksel K. Nordstoga. A special thanks should be given to Aksel for many long hours together in MC-lab.

Lastly, I would like to thank Torgeir Wahl for helping me with the wavemaker and other general matters.

Abstract

This thesis presents the development of an autonomous tuning methodology for a Dynamic Positioning (DP) controller. The method is implemented in the Marine Cybernetics Laboratory (MC-Lab) on the 1:90 model of a DP vessel, the C/S Inocean Cat I Drillship (CSAD).

To derive a DP system for the CSAD, a 3 Degrees of Freedom (DOF) model of the vessel and a motion control system for trajectory tracking is established. The motion control system includes the proportional-integral-derivative (PID) controller that is responsible for calculating the necessary forces and moments for following the trajectory. It is this component of the DP system that is the subject of tuning, and the control parameters are established as the tuning variables.

The 3 DOF model and the motion control system are used to establish the simulation model. The simulation model is used to investigate different types of performance indicators to evaluate the control parameters of the PID controller. With the control parameters as optimization variables and a performance indicator (PI) as an objective function, derivative-free optimization (DFO) algorithms are compared and evaluated. The integral of absolute error (IAE) is selected as the PI and the particle swarm optimization (PSO) as the DFO algorithm for the practical implementation.

A 2-step, 3 DOF transient maneuver is defined as the trajectory tracking test in MC-Lab. This test is used to evaluate the performance in the autotuning. The autotuning functionality is implemented in Simulink and is modified to function in real-time. The implemented system is able to do autonomous tuning of the control parameters without human intervention for multiple hours. Moreover, autotuning of the PID controller is done in two environmental conditions, one in calm water and one in moderate/rough waves. The result is a trajectory tracking with maximal positional errors below 1 cm and maximal heading errors below 0.5 degrees.

Sammendrag

Denne avhandlingen presenterer utviklingen av en autonom tuningmetode for en dynamisk posisjoneringskontroller. Metoden er implementert i Marine Cybernetics Laboratory (MC-Lab) på 1:90-modellen av et dynamisk posisjoneringsfartøy, C/S Inocean Cat I Drillship (CSAD).

For å utlede et dynamisk posisjoneringssystem for CSAD, etableres en simuleringsverifiseringsmodell med 3 frihetsgrader av fartøyet og et bevegelseskontrollsystem for banefølging. Bevegelseskontrollsystemet inkluderer proposjonal-integral-derivatkontrolleren (PID-kontroller) som er ansvarlig for å beregne nødvendige krefter og momenter for å følge banen. Det er denne komponenten av dynamisk posisjoneringssystemet som skal tunes, og kontrollparametrene er etablert som tuningsvariablene i autotuningen.

Simuleringsverifiseringsmodellen og bevegelseskontrollsystemet brukes til å etablere utgjør til sammen simulerings- modellen. Den brukes til å undersøke ulike typer ytelsesindikatorer for å evaluere kontrollparametrene til PID-kontrolleren. Med kontrollparametrene som optimaliseringsvariabler og ytelsen som en objektiv funksjon, blir gradientfrie optimaliseringsalgoritmer sammenlignet og evaluert. Integralet av absolutt feil er valgt som ytelsesindikator og partikkelsvermoptimalisering som gradientfri optimaliseringsalgoritme for den praktiske implementeringen.

En 2-trinns, transient manøver med bevegelser i 3 frihetsgrader er definert som banesporingstesten i MC-Lab. Denne testen brukes til å evaluere ytelsen i autotuningen. Autotuning- funksjonaliteten er implementert i Simulink og er modifisert for å fungere i sanntid. Det implementerte systemet er i stand til å gjøre autonom tuning av kontrollparametrene uten menneskelig innblanding gitt at det ikke er noen eksterne feil. Videre gjøres autotuning av PID-kontrolleren i to sjøtilstander, en i rolig vann og en i moderate bølger. Resultatet er en banesporing med maksimale posisjonsfeil under 1 cm og maksimale kursfeil under 0.5 grader.

Table of Contents

List of Tables

List of Figures

Abbreviations

NED	=	North, East, Down, Reference frame
GNSS	=	Global Navigation Satellite System
GA	=	Genetic Algorithm
PID	=	Proportional-Integral-Derivative
DFO	=	Derivative-Free Optimization
DOF	=	Degrees Of Freedom
AI	=	Artificial Intelligence
IMU	=	Inertial Measurement Unit
WF	=	Wave Frequency
LF	=	Low Frequency
DP	=	Dynamic Positioning
MC-Lab	=	Marine Cybernetics Laboratory
CSAD	=	C/S Arctic Drillship
PI	=	Performance indicator
ROV	=	Remotely Operated Vehicle
SISO	=	Single-Input-Single-Output
BBO	=	Black Box Optimization
PSO	=	Particle Swarm Optimization
RPM	=	Rotations per Minute
PS3	=	Playstation 3
MCS	=	Motion Control System
GNC	=	Guidance, Navigation and Control
QTM	=	Qualisys Tracking Manager
I/O	=	Input/Output
MRS	=	Magnitude-rate Saturation
LLAC	=	Low-Level Actuator Control
IAE	=	Integral of Absolute Error
IAEC	=	Integral of Absolute Error and Control
IAEW	=	Integral of Absolute Error times Work
IADC	=	Integral of Absolute Derivative of Control
ISE	=	Integral of Squared Error
ITAE	=	Integral of Absolute Error multiplied by Time
RBF	=	Radial Basis Functions
MSS	=	Marine Systems Simulator

Nomenclature

α	=	Orientations of thrusters
ψ	=	Heading of vessel
χ	=	Course of vessel
β	=	Crab angle of vessel
ζ	=	Relative damping
ω_b	=	Bandwidth
ξ	=	Integral state in PID
η	=	Carriage-fixed pose
η	=	Carriage-fixed pose
η_d	=	Carriage-fixed pose, Desired
η_m	=	Carriage-fixed pose, Measured
$\hat{\eta}$	=	Carriage-fixed pose, Estimated
η_e	=	Carriage-fixed error, Estimated
$\bar{\eta_e}$	=	Normalized carriage-fixed error
ν_d	=	Body-fixed velocities, Desired
$\dot{\nu}_d$	=	Body-fixed accelerations, Desired
$\hat{\nu}$	=	Body-fixed velocities, Estimated
ν_e	=	Body-fixed error in velocity, Estimated
ν_c	=	Body-fixed current velocity, Estimated
$\bar{\hat{\nu}}$	=	Normalized Body-fixed velocities, Estimated
τ	=	Control input
$\bar{\tau}$	=	Control input, Normalized
τ_{sat}	=	Control input, saturated
τ_{PID}	=	Feed-back control from PID
τ_w	=	Wave loads
τ_{FF}	=	Feed-forward control
$\bar{\tau}$	=	Normalized Control input
U	=	Vessel Speed
M	=	Mass Matrix
$D(\nu)$	=	Damping Matrix
$C(\nu)$	=	Coriolis and Centripetal Matrix
$R(\psi)$	=	Rotation Matrix
$T(\alpha)$	=	Thruster configuration matrix
$K_{p,i,d}$	=	Proporional, derivative and integral gains, body-fixed

Chapter 1

Introduction

1.1 Motivation

The tuning of a Dynamic Positioning (DP) control system is both time consuming and complicated. Today, the tuning is done at sea trials manually. The operators evaluate the response during a set of predefined tests and make adjustments to the control parameters based on expertise. However, the high dimensionality of the control parameters and the uncertainties during sea trials make the tuning an almost impossible task. The manual tuning is also affected by human errors and varying weather conditions. Meanwhile, derivative-free optimization (DFO) methods are getting more intelligent and it is believed that such methods, applied by a computer, can outperform a human operator for the tuning task.

This motivates an investigation of the potential of autonomous tuning of a real-world DP control system. This investigation is facilitated by the advanced marine cybernetics laboratory (MC-Lab) at NTNU and it's new DP vessel, the C/S Inocean Cat I Drillship (CSAD).

1.2 Problem formulation and objectives

The problem that the thesis tries to answer is:

- Given the CSAD vessel, how can a DFO loop be designed to make the vessel capable of efficient and autonomous tuning of the DP control system in the MC-Lab?

The problem formulation above includes assessing topics like DFO, performance indicators (PI), DP systems as well as combining it all using software and hardware in the MC-Lab.

To answer the question, the main objectives have been formulated as

- Implement an autotuning DFO loop for a simulation model of the vessel. Compare and select the PIs and DFO algorithms that are most suited for the MC-lab setup.
- Implement an autotuning DFO loop for the CSAD in MC-lab able of doing autonomous tuning (without operator intervention).

For accomplishing these main objectives, the sub-objectives at the top of the thesis have been formulated in cooperation with the supervisor, Roger Skjetne.

1.3 Scope and delimitations

This thesis starts by doing a literature review on relevant methods and theory. Then the methods are investigated through simulations and experiments.

The thesis is centred around the experimental setup in MC-lab and the vessel CSAD. A large share of the time has gone into understanding this system to establish a DP system that is adequate and a DFO-tuning implementation that is practical for the setup. It is therefore attempted to explain a mathematical model of the system, the CSAD DP system, and the autotuning setup before establishing the DFO as relevant to this system. The goal of the thesis is to conceptually test autotuning, as well as contributing the motion control system of the CSAD, so that it becomes easier to use for future candidates.

Some of the limitations of this thesis are:

- The performed tests were executed in a controlled environment in MC-lab with a 1:90 scale model.

1. The positioning system in MC-lab is very precise, and has little noise
2. Current and wind was not present during the tests
3. The suggested test duration would have been approximately 10 times longer for the for the full-scale vessel

- The tuning was performed with a fixed thrust allocation. It is possible that a more optimal thrust allocation, would affect the results of the tuning.
- The observer struggled with estimating velocities due to the transient movements in the test. It is possible that a better velocity estimation would affect the result of the tuning.

1.4 Contributions of the Thesis

The main contributions of the thesis are:

- It has been demonstrated that the implemented autotuning method is able of tuning the DP controller without any human intervention for several hours, only limited by battery failure or other serious failures. The resulting DP controller has maximal errors below 1 cm/ 0.5°deg errors for a 3 DOF reference tracking in both in calm water and in waves.
- A modular online autotuning block in Simulink has been created and can easily be employed for tuning of other vessels. The methodology could also be utilized for tuning of the DP observer.
- A Matlab program has been set up such that any of Matlab's global optimization tools can be used for autotuning of the DP simulation model. By adjusting the input and output parameters, it is possible to apply optimization to the observer as well.
- A high-precision, highly modular motion control system for reference tracking has been established for CSAD. Especially, a guidance system with feasible velocity constraints, a tuned nonlinear proportional-integral-derivative (PID) controller with feed-forward terms and a predictable fixed thrust allocation with magnitude and rate constraints (MRS).

1.5 Outline of the Thesis

This thesis is built upon the developement of an autotuning methodology for the CSAD. It is organized in:

Chapter 1 introduces the thesis to the reader. It explains the motivation, problem formulation, objectives, scope, delimitations and contributions of the thesis.

Chapter 2 provides relevant background to DP, DFO, PID tuning, CSAD and MC-lab.

Chapter 3 presents the mathematical modeling for the for the simulation verification model (SVM), as well as the control design model (CDM) used for model-based observer and control design.

Chapter 4 presents the DP system as a motion control system. It includes a trajectory generator, an observer, the tracking controller and the thrust allocation.

Chapter 5 presents the autotuning setup. A test maneuver is defined and the autotuning setup for the simulation model and for CSAD is presented.

Chapter 6 details how DFO is used for autotuning. It defines the bounds of the optimization, different PIs, and DFO algorithms. The PIs are tested and compared and a favored PI is chosen. Then the DFO algorithms are tested and compared and a favorite algorithm is chosen. Then the implementation of the PI and DFO in the autotuning is presented.

Chapter 7 presents the results from the laboratory including calm water autotuning and autotuning in waves.

Chapter 8 gives presents the conclusions and further work

Chapter 2

Background

The main focus of the thesis is to establish a DFO-loop for optimizing the PID controller in a DP system using the test platform, CSAD in MC-Lab. It therefore seems intentional to provide some background information about these topics to the reader.

2.1 Dynamic Positioning

According to (Sørensen, 2019), DNV GL's definition of a DP system is

Definition 2.1.1. *A DP vessel is by the class societies e.g. DNVGL (2018) defned as a vessel that maintains its position and heading (fixed location or pre-determined track) exclusively by means of active thrusters. This is obtained either by installing tunnel thrusters in addition to the main screw(s), or by using azimuthing thrusters, which can produce thrust in different directions.*

According to (Sørensen, 2019), there are more than 2000 DP vessels operating worldwide in offshore oil and gas, shipping, cruise ships and fisheries to mention some applications. Most DP-systems are low-speed applications, meaning that the vessel either keeps a fixed position and heading, or slowly moves from one point to another. However, some tracking functions have been developed to enable DP applications like cable laying and ROV operations. According to (Sørensen, 2019) a trend is that high-speed operation functionality is merging with low-speed (DP) functionality to have one system for all speeds and all types of operations.

The most common DP systems control the 3 Degrees of Freedom (DOF) surge, sway and yaw through the following modes of control.

- Manual Control: The operator can generate force/moment setpoints the 3 DOFs
- Damping Control: It is used in DP for obtaining a smooth transition between transit speed and fixed position operations.
- Set-point Control: Feedback from positional error and from low frequency velocities. Often referred to as station keeping.
- Tracking Control: The vessel tracks a reference trajectory from one set-point to another.

The DP vessel can be considered as a motion control system consisting of a guidance, navigation and control system as illustrated in Figure 2.1.

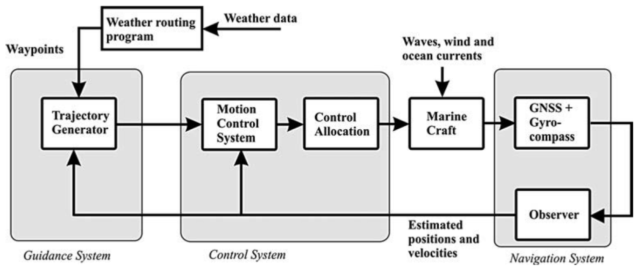


Figure 2.1: Guidance, Navigation and Control

The guidance system calculates the reference (desired) position, velocity and acceleration that is used by the DP controller. For DP purposes, an open-loop guidance system is commonly used for tracking control. This guidance system takes in a constant setpoint and calculates a smooth trajectory for the position, velocity and acceleration.

Navigation is the science of determining the vessels position, attitude and course. According to (Fossen, 2011), it is usually done using global navigation satelite system (GNSS) and gyro compass and accelerometers. To use these for estimation of the vessels position and velocity, a DP observer is commonly employed. A commonly used DP observer is the nonlinear passive observer.

According to (Fossen, 2011) control is the the system responsible for calculating the necessary forces and moments for the vessel to fulfil the control objective. For a DP vessel the objective can for instance be trajectory tracking. A DP control algorithm is often a combination of a feed-forward and feed-back control law. The feed-forward control law uses signals from the guidance system or other sensors, while the feed-back control law uses the estimated values from the observer and the reference from the guidance to apply control. For feed-back control, a nonlinear PID controller can be used. For the PID to function properly, it has to be properly tuned.

2.2 PID tuning

The Proportional-Integral-Derivative (PID) controller has been used as a method of feedback control in many industrial applications because of it's robustness and simplicity in structure (Sahib, 2015). For a simple single-input single-output (SISO) system it is expressed as:

$$e(t) = y(t) - r(t) \tag{2.1}$$

$$u(t) = \underbrace{-K_p e(t)}_{\text{P}} \underbrace{-K_i \int_0^t e(t)dt}_{\text{I}} \underbrace{-K_d \frac{d}{dt} e(t)}_{\text{D}} \tag{2.2}$$

Where e(t) is the error between the actual value of y and the desired reference r(t) is the reference or set-point. It can be considered as the desired value ofthe output variable, y(t). u(t) is the control action. $K_p,\ K_i,\ K_d$, are the proportional, integral and derivative control gains, respectively. Theproportional term, P, is proportional to the current error. The derivative term,D, is proportional to the change in error. The integral term, I, is proportional to the integral of error.

Tuning of a PID controller, is the adjustment of the control gains, $K_p,\ K_i,\ K_d$, to meet some requirements or performance. For example, doing manual tuning by trial and error

1. Select a set of control gains, $K_p,\ K_i,\ K_d$, and run a relevant test.
2. Evaluate the performance from the test and adjust the control gains based on experience. If the previous set of gains were better, you go back to these and make a different adjustment. If the new gains are better, make som adjustment to them.
3. Repeat step 1. and 2. until desired performance is achieved.

Note that a tuning can be complex for a SISO system where there are only 3 control gains. However, for a multiple-input multiple-output (MIMO) system, the control gains become matrices, so that for instance a 3 input, 3 output system has 27 control gains.

2.3 Derivative-Free Optimization Methods

According to (Audet, 2016) DFO are methods within mathematical optimization that does not use use information about the actual derivative to find optimal solutions. The key concept with DFO is that it uses only function values of the objective function for optimization. The objective function is thought of as being wrapped in a black box, and the only information that is available is the objective function, $f(x)$. These methods have attracted attention from researchers in the last decade, with interest still increasing. According to (Audet, 2016), black-box optimization is often the most feasible alternative when doing simulation-based design. However, it should be noted that modern gradient-based methods almost always outperforms DFO algorithms if the gradient is available and can be calculated at a reasonable cost. According to (Audet, 2016), black box optimization (BBO) using DFO is especially suitable when conducting experiments where there are no explicit mathematical expressions and thus no gradient

In 1965, John Melder and Roger Mead introduced the Nelder-Mead Simplex Method that has since become a popular optimization that is effective and intuitive. Other popular methods for optimization include evolutionary strategies like genetic algorithms, dating back to at least 1971 (Audet, 2016). Other DFO methods include swarm algorithms, particularly the popular Particle Swarm Optimization (PSO) by (Kennedy, 1995). PSO has shown success in BBO and has become popular in recent years. Another popular DFO method is the surrogate model optimization (SGO). Typically, these are methods employing regression or interpolation to approximate the objective function, thereby having a cheap evaluation of the objective function.

2.4 C/S Arctic Drillship

In 2013, Inocean designed an arctic drillship for Statoil. The ship was called the Cat I Arctic Drillship was a conceptual design of a DP and turret moored mobile offshore drilling unit (MODU). In 2016, the CSAD was built and instrumented by co-advisor Jon Bjørnø for research on Thruster-Assisted Positioning Mooring as a part of his master thesis. According to (Bjørnø, 2016) the vessel is a 1:90 model of Statoil's Cat I Arctic Drillship shown in Figure 2.2 with model scale dimensions of CSAD in Table 2.1

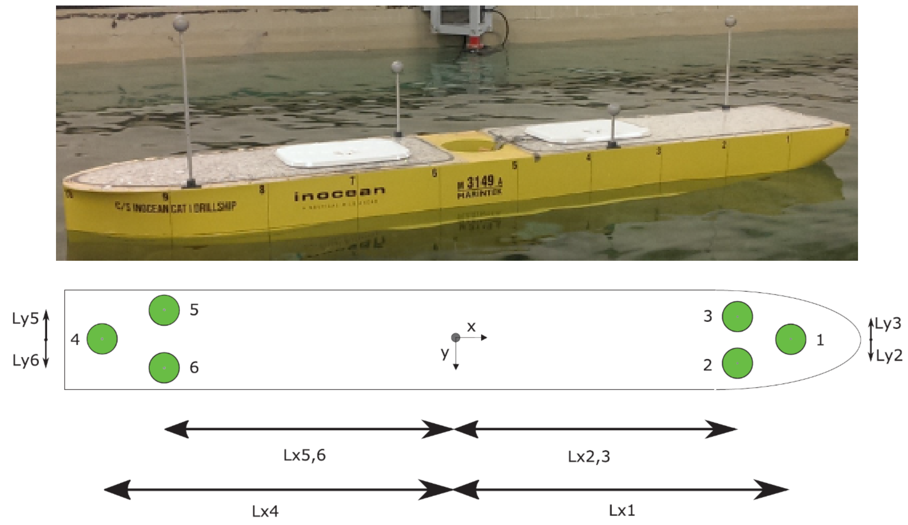


Figure 2.2: CSAD vessel and thruster configuration [Courtesy: Frederich (2016)]

Table 2.1: Model ship Dimensions

L_{oa}	2.578[m]
B	0.440[m]
D	0.211[m]
T	0.133[m]
Δ	127.92[kg]

As Figure 2.1 illustrates, there are 6 thrusters on the vessel. The low-level thruster control is done using a speed controller for the RPM and a servo motor for the control of the thruster angles. All the thrusters are azimuth thrusters called the "Aero-naut Precision Schottel".

CSAD is controlled using a PlayStation 3 (PS3) controller or using a guidance, navigation and control (GNC) system in Simulink. The PS3 controller can either control the generalised forces (surge, sway and yaw), or it can control the individual actuators (front and aft thrusters). Furthermore, the custom GNC system outputs the thruster angles and speed of all 6 azimuth thrusters. The real-time controller used onboard the vessel is the CompactRIO (cRIO), which compiles the custom simulink block in real-time.

For a complete understanding of the setup of the CSAD, see Bjørnø (2016).

2.5 Marine Cybernetics Laboratory

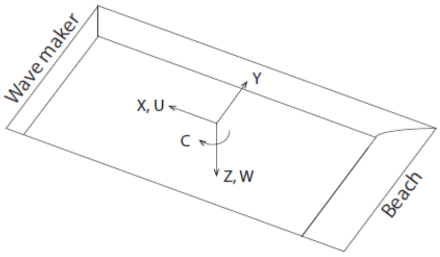
The Marine Cybernetics Laboratory consists of a wave basin and is mainly used for testing motion control systems for marine vessels. The instrumentation of the laboratory includes a towing carriage which can be used for specialised hydrodynamic tests.

The MC lab has a fleet of vessels, with the newest vessel being the CSAD. The laboratory has a real-time positioning system for both underwater vehicles and surface vehicles. This system emulates a full scale global navigation satelite system. The Qualisys motion capture system is composed of 3 Oqus cameras that detects the distance to silver spheres in the frame and finds the positions of the spheres by using triangulation. These silver spheres are called the reflectors and they are attached to the ship at known locations. Using geometry, the 6 DOFs can be derived from the position of 3 reflectors. However, for precision and redundancy purposes, 4 reflectors are used for the CSAD.

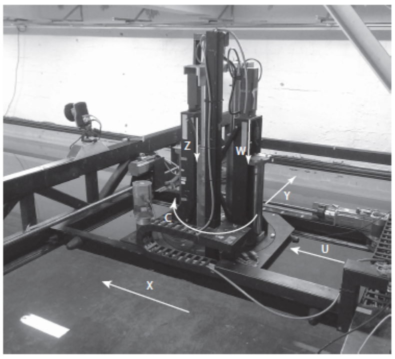
The wave maker is a single paddle wave making machine with a width equal to that of the basin, 6 m. Furthermore, it has the following capacities:

- Regular waves: $H < 0.25$m, $T = 0.3 - 3s$
- Irregulalar waves: $H < 0.15$m, $T = 0.3 - 3s$
- Available Spectrum: JONSWAP
- Wave controller update rate: 10 Hz
- No. wave gauge on paddle: 4
- Stroke length on actuator: 590 mm
- Speed limit: 1.2 m/s

Furthermore, the towing carriage can be controlled from the computer or manually. The computer mode is operated through, while the manual mode is operated from the console at the towing carriage. Figure 2.3 shows the setup of the basin and the towing carriage. Note that the measured position of the vessel is relative to the towing carriage.



(a) Basin



(b) Towing carriage

Figure 2.3: Illustration of the setup of the basin [Courtesy of (NTNU, 2015b)].

Chapter 3

Mathematical Modelling

The mathematical modelling is used to design the 3 Degree of Freedom (DOF) simulation verification model (SVM). It is also used to derive the control design model (CDM) which is used to design the controller and observer.

At first, a 6 DOF model by (Bjørnø, 2016) was developed. Then, in (Lyngstadaas, 2018), an updated 3 DOF model with updated parameters was established, and it is this model and these parameters that are used in this thesis.

3.1 Kinematics

Firstly, to establish a model of the vessel, a definition of the vessel motion is necessary. This is key because the different sensors give measurements that are relative to different reference frames. The reference frames used in MC-Lab are the body-fixed reference frame and the carriage-fixed reference frame. The relation between the two is seen in Figure 3.1 in the xy-plane.

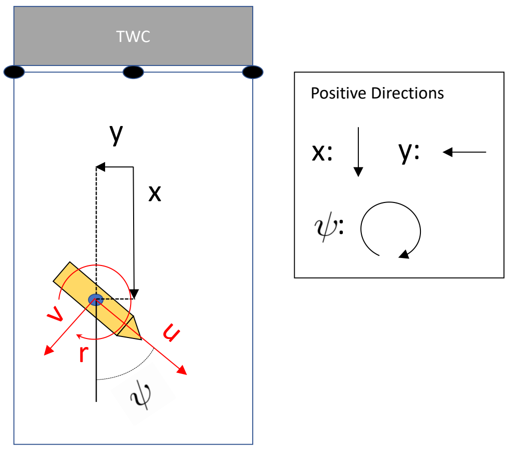


Figure 3.1: TWC is the Towing Carriage where the QualiSys Cameras are attached

Carriage-fixed reference frame, with positions $\eta_1 = [x, y, z]^\top$ and Euler angles $\eta_2 = [\theta, \phi, \psi]^\top$: This coordinate system is used like the North-East-Down (NED) reference frame is used for real vessels. This coordinate system assumes a flat earth and does not take earth curvature into consideration. The positions and orientations of the vessel are measured in this reference frame by the QualiSys Tracking Managers (QTM).

The body-fixed reference is a moving reference frame with translational and rotational velocities $\nu_1 = [u, v, w]^\top$ and $\nu_2 = [p, q, r]^\top$. The origin of the frame is in o_b, mid-ship in the waterline with the axis oriented to coincide with the principal axis of inertia. x_b is directed from aft to fore, y_b is directed towards starboard and z_b directed down. It is in this coordinate system that the Inertial Measurement Unit (IMU) measures translational and rotational accelerations. Furthermore, it is in this coordinate system that the forces and moments are defined.

By assuming small roll angle, pitch angle, and heave motion ($\theta \approx 0$, $\phi \approx 0$, $w \approx 0$), the motion can be described in the xy-plane in 3 DOF with $\eta = [x, y, z]^\top$ and $\nu = [u, v, w]^\top$:

$$\dot{\eta} = R(\psi)\nu \tag{3.1}$$

$$R(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

3.2 3 DOF Simulation Verification Model

The 3 DOF SVM should be a high-fidelity model that aims to describe a vessel's motions as precise as possible. According to (Fossen, 2011), the model should include the vessel's dynamics, the propulsion system, environmental loads and the measurement system should also be modelled. This mathematical model is implemented in Simulink for simulation of the vessel. Like the actual setup in MC-Lab, the SVM takes in the commanded angles, α, and control input, u and outputs the measured pose, η_m like in Figure 3.2

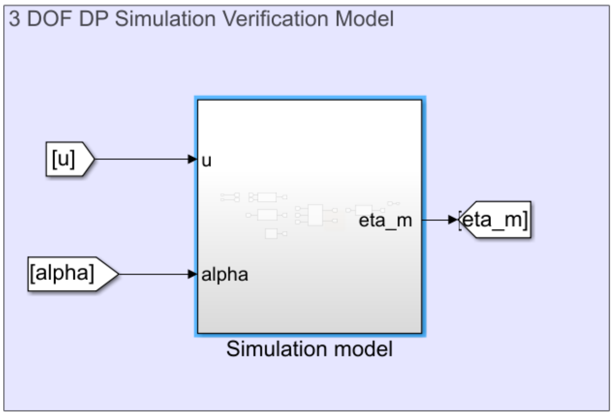


Figure 3.2: Simulink block for the SVM

3.2.1 Vessel Dynamics

The vessel model was adopted from (Lyngstadaas, 2018) and is the nonlinear 3 DOF ship model in Equation (3.4) assuming $\nu_c = 0$ as no current was used for the simulations nor experiments.

$$\dot{\eta} = R(\psi)\nu \tag{3.3}$$

$$M\dot{\nu} + C(\nu)\nu + D(\nu)\nu = \tau_{wave} + \tau \tag{3.4}$$

The vessel dynamics take in the wave loadss and control forces, τ_w and τ, respectively. Then the model calculates the pose vector, η. Moreover, M is the inertia matrix, C(ν) is the Coriolis and centripetal matrix, and D(ν) is the damping matrix. The inertia matrix, M, in 3 DOF is given as:

$$M = M_{RB} + M_A \tag{3.5}$$

Where the rigid body mass, M_{RB} and Added mass M_A are

$$M_{RB} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & mx_g \\ 0 & mx_g & \mathrm{I}_z \end{bmatrix} \tag{3.6}$$

$$M_A = \begin{bmatrix} -X_{\dot{u}} & 0 & 0 \\ 0 & -Y_{\dot{v}} & -Y_{\dot{r}} \\ 0 & -N_{\dot{v}} & -N_{\dot{r}} \end{bmatrix} \tag{3.7}$$

Where the mass of CSAD, $m = 127.92$ kg and the distance from the body fixed origin to the center of mass, $x_g = 0.00375$ m. The moment of inertia about the z-axis, $I_z = 61.967$ kg/m^2. The Coriolis and centripetal matrix is:

$$C = C_{RB}(\nu) + C_A(\nu) \tag{3.8}$$

Where the rigid body and added Coriolis and centripetal matrices are:

$$C_{RB}(\nu) = \begin{bmatrix} 0 & 0 & -m(x_g\, r + v) \\ 0 & 0 & m\mathrm{u} \\ (x_g\, r + v) & -m\mathrm{u} & 0 \end{bmatrix} \tag{3.9}$$

$$C_A(\nu) = \begin{bmatrix} 0 & 0 & -c_{A,13}(\nu) \\ 0 & 0 & c_{A,23}(\nu) \\ c_{A,13}(\nu) & -c_{A,23}(\nu) & 0 \end{bmatrix} \tag{3.10}$$

Where

$$D_L(\nu) = \begin{bmatrix} -X_u & 0 & 0 \\ 0 & -Y_v & -Y_r \\ 0 & -N_v & -N_r \end{bmatrix} \tag{3.11}$$

$$D_{NL}(\nu) = \begin{bmatrix} d_{NL,11}(\nu) & 0 & 0 \\ 0 & d_{NL,22}(\nu) & d_{NL,23}(\nu) \\ 0 & d_{NL,32}(\nu) & d_{NL,33}(\nu) \end{bmatrix} \tag{3.12}$$

Where

$$d_{NL,11}(\nu) = -X_{|u|u}|u| - X_{uuu}u^2 \tag{3.13}$$
$$d_{NL,22}(\nu) = -Y_{|v|v}|v| - Y_{|r|v}|r| - Y_{vvv}v^2 \tag{3.14}$$
$$d_{NL,23}(\nu) = -Y_{|r|r}|r| - Y_{|v|r}|r| - Y_{rrr}r^2 - Y_{ur}u \tag{3.15}$$
$$d_{NL,32}(\nu) = -N_{|v|v}|v| - N_{|r|v}|v| - N_{vvv}v^2 - N_{uv}u \tag{3.16}$$
$$d_{NL,33}(\nu) = -N_{|r|r}|r| - N_{|v|r}|v| - N_{rrr}r^2 - N_{ur}u \tag{3.17}$$
$$\tag{3.18}$$

Where the terms

$$Y_{ur} = X_{\dot{u}} \tag{3.19}$$
$$N_{uv} = -(Y_{\dot{v}} - X_{\dot{u}}) \tag{3.20}$$
$$N_{ur} = Y_{\dot{r}} \tag{3.21}$$

include the Munk moment.

3.2.2 Thruster Dynamics

The thruster model takes in the control signals (u,α) from the thrust allocation. The thruster model calculates the force through:

$$\tau(\alpha, u) = T(\alpha)K_T u \tag{3.22}$$

Where the thrust configuration matrix T is given in equation 3.23

$$T(\alpha) = \begin{bmatrix} c(\alpha_1) & c(\alpha_2) & c(\alpha_3) & c(\alpha_4) & c(\alpha_5) & c(\alpha_6) \\ s(\alpha_1) & s(\alpha_2) & s(\alpha_3) & s(\alpha_4) & s(\alpha_5) & s(\alpha_6) \\ \phi(\alpha_1) & \phi(\alpha_2) & \phi(\alpha_3) & \phi(\alpha_4) & \phi(\alpha_5) & \phi(\alpha_6) \end{bmatrix} \tag{3.23}$$

Where c and s are the trigonometric functions cosine and sine, while $\phi(\alpha_i) = L_i \cos(\beta_i) \sin(\alpha_i)$ where $L_i = \sqrt{L_{i,x}^2 + L_{i,y}^2}$ and $\beta_i = tan(L_{i,x}/L_{i,y})$. The thrusters were fixed at $\alpha = [\pi, \pi/4, -\pi/4, 0, 5\pi/4, 3\pi/4]^\top$. Figure 3.3 and Table 3.1 show the locations of the thrusters. Note that the orientations of thrusters (α_i) are defined relative to the body-fixed x axis with positive rotation clockwise.

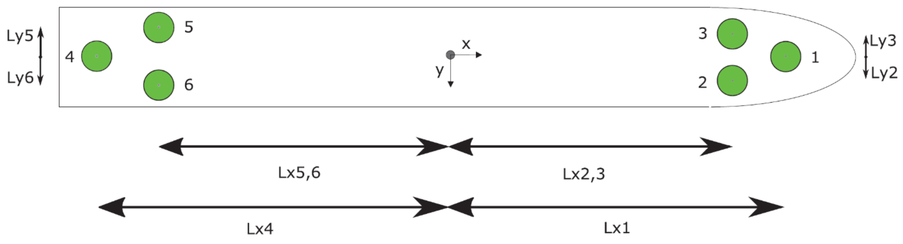


Figure 3.3: Thruster configuration [Courtesy: Frederich (2016)]

Table 3.1: Thruster model scaled position [Courtesy: Frederich (2016)]

Thruster	Position X[m]	Position Y[m]
1	1.0678	0.0
2	0.9344	0.11
3	0.9344	-0.11
4	-1.1644	0.0
5	-0.9911	-0.1644
6	-0.9911	0.1644

The thrust coefficient matrix is K_T = diag([1.49,1.49,1.49,1.49,1.49,1.49]). Lastly, u_i is the actuator input in Volt. According to Bjørnø et al. (2017) $u_i \in [-0.5, 0.5]$ [V]. u is therefore saturated at these limits. The rate constraints of the thrusters are taken into consideration in the thrust allocation and was therefore not modelled here.

3.2.3 Wave Loads

The practical implementation of this concept in Simulink is done using a block called ”Waves” in conjunction with an response amplitude operator (RAO) block from the marine systems simulator (MSS) toolbox with parameters found by (Bjørnø, 2016). Because it was computationally heavy to use many waves components, only 4 out of 200 were used. 4 components is far from enough for representing the total energy. However, as this thesis focuses on the tuning, it is prioritized with computional efficiency over accurate waves. Furthermore, the waves gave a some disturbance significant disturbance.

3.2.4 Measurement model

A simplified measurement modelling is done in this thesis by adding some white noise to the actual positions

$$\eta_m = \eta + v \tag{3.24}$$

According to (Fossen, 2011) the measurement noise can be assumed to be a zero-mean Gaussian white noise process with covariance matrix, R. The measurements from QTM are precise and the error is believed to be in the magnitude of 1 mm. Therefore, white noise with power [0.001 , 0.001 m, 0.001 rad]$^\top$ was added.

3.3 3 DOF Control Design Model

The control design model will be used for model-based controller and observer design. It is desired to create a low-frequency (LF) CDM of the vessel to be used for controller design, as well as a wave-frequency (WF) model, a bias model and a measurement model to use for observer design. It is desired that this model describes the physical characteristics of the system. The CDM is adopted from (Sørensen, 2019).

3.3.1 Low-frequency Control Design Model

To simplify the SVM from Equation 3.4, low-speed maneuvers are assumed. The LF vessel dynamics can then be described as:

$$\dot{\eta} = R(\psi)\nu \tag{3.25}$$

$$M\dot{\nu} = -D_L\nu + R(\psi)b + \tau \tag{3.26}$$

3.3.2 Wave Frequency Control Design Model

The WF wave motion is described using a state-space representation of a linear wave spectra.

$$\dot{\xi}_w = A_w \xi_w + E_w w_w \tag{3.27}$$

$$\eta_w = C_w \xi \tag{3.28}$$

Where η_w is the wave motion, and orientation vector, w_w is a zero-mean Gaussian white noise vector. The system matrix A_w, the disturbance matrix E_w and the measurement matrix, C_w can be expressed as:

$$A_w = \begin{bmatrix} 0_{3x3} & \mathrm{I}_{3x3} \\ -\Omega^2 & -2\Lambda \end{bmatrix} \tag{3.29}$$

$$C_w = \begin{bmatrix} 0_{3x3} & \mathrm{I}_{3x3} \end{bmatrix}, E_w = \begin{bmatrix} 0_{3x3} \\ \mathrm{K}_w \end{bmatrix} \tag{3.30}$$

Where $\Omega = diag([\omega_1, \omega_2, \omega_3])$, $\Lambda = diag([\zeta_1, \zeta_2, \zeta_3])$ and $K_w = diag([K_{w1}, K_{w2}, K_{w3}])$ and the model is equivalent to the decoupled second-order transfer functions:

$$\frac{\eta_{w_i}}{w_{w_i}} = \frac{K_{w_i} s}{s^2 + 2\zeta_i \omega_i s + \omega_i} \tag{3.31}$$

Where ζ_i is a damping coefficient, ω_i is the dominating frequency and $K_{w_i} = 2\lambda_i \omega_i \sigma_i$ where sigma is a constant describing the wave intensity.

3.3.3 Bias Model

The bias model describes the slowly varying forces and moments due to 2. order wave and wind loads and current. It will also account for for errors in the modelling. The recommended model by (Sørensen, 2019) is the 1-order Markov model

$$\dot{b} = -T_b^{-1} b + E_b w_b \tag{3.32}$$

Where T_b are the time constants for these slowly varying forces in surge sway and yaw. w_b is a zero-mean Gaussian white noise vector and E_b is a scaling matrix.

3.3.4 Measurement Modelling

The Measurements are modelled as

$$y = \eta + C_w \xi + v \tag{3.33}$$

Where v$\in \mathbf{R}^3$ is the zero-mean Gaussian measurement noise vector.

3.3.5 Total Control Design Model

The control design model can be written in state space form by combining the results from Section 3.3.1 - 3.3.4.

$$\dot{\xi}_w = A_w \xi_w + E_w w_w \tag{3.34a}$$

$$\dot{\eta} = R(\psi)\nu \tag{3.34b}$$

$$\dot{b} = -T_b^{-1} b + E_b w_b \tag{3.34c}$$

$$M\dot{\nu} = -D_L \nu + R(\psi) b + \tau \tag{3.34d}$$

$$y = \eta + C_\omega \xi \tag{3.34e}$$

The CDM is a state-space model of the system that will be used for control design in Section 4.3 and for observer design in Section 4.2.

Chapter 4

Motion Control

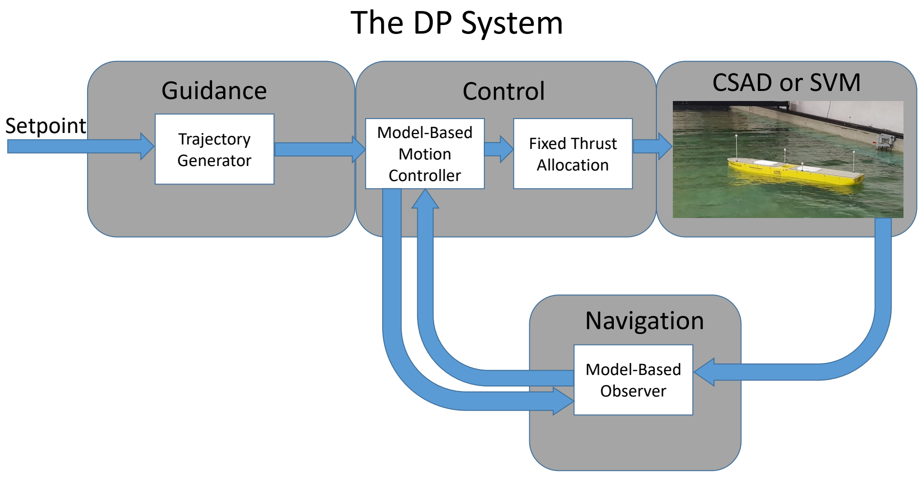


Figure 4.1: The DP system as a motion control system

Figure 4.1 describes the DP as a the motion control system as relevant for this thesis. The motion control system is divided into a guidance system, a navigation system and a control system.

The DP system designed in this thesis has trajectory tracking as the control objective. Section 4.1 describes the trajectory generator. Then, Section 4.2 describes the model-based DP observer. Section 4.3 describes the model-based tracking controller subject to tuning and Section 4.4 presents the fixed thrust allocation for allocating the low-level control input, u and α.

4.1 Guidance System

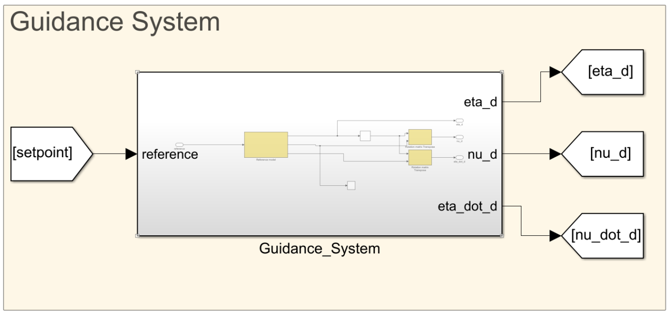


Figure 4.2: The Guidance block in Simulink

Simply explained, the guidance system used in Figure 4.2 is a trajectory generator that generates a smooth trajectory to a setpoint, η_r. It takes in the set point and generates a smooth trajectory, along a line from one set point to another. It also generates the desired body-fixed accelerations and velocities.

As described by (Sørensen, 2019) a reference model can be used to calculate feasible trajectories for the desired vessel motion. It is suggested by (Fossen, 2011) that the the reference model should be of the third degree and should be designed as a low-pass filter cascaded with a mass-damper-spring system.

$$a_d + \Omega v_d + \Gamma x_d = \Gamma x_{ref} \tag{4.1}$$

$$\dot{x}_{ref} = -A_f x_{ref} + A_f \eta_r \tag{4.2}$$

Where a_d, v_d and x_d correspond to the desired carriage-fixed acceleration, velocity and position such that

$$\dot{\nu}_d = R^\top(\psi) a_d = R^\top(\psi)\ddot{\eta}_d \tag{4.3}$$

$$\nu_d = R^\top(\psi) v_d = R^\top(\psi)\dot{\eta}_d \tag{4.4}$$

$$\eta_d = x_d \tag{4.5}$$

Note that in Equation 4.3, low yaw rate is assumed such that $\dot{R}(\psi)\dot{\eta}_d \approx 0$. Moreover,

- Ω is the diagonal matrix diag($[2\zeta_1\omega_1 \; 2\zeta_2\omega_2 \; 2\zeta_3\omega_3]$)
- Γ is the diagonal matrix diag($[\omega_1^2 \; \omega_2^2 \; \omega_3^2]$)
- A_f is the diagonal matrix diag($[\frac{1}{t_1} \; \frac{1}{t_2} \; \frac{1}{t_3}]$) = diag($[\omega_1 \; \omega_2 \; \omega_3]$)

By taking the Laplace transform and combining the Equation 4.1 and 4.2 in each decoupled degree of freedom, the following is achieved:

$$\frac{x_{di}}{\eta_{ri}} = \frac{\omega_i^2}{s^2 + 2\zeta_i\omega_i s + \omega_i^2} \cdot \frac{\omega_i}{s + \omega_i} \tag{4.6}$$

For i = 1,2,3. Where the parameters are given as

- $\omega_i = 0.1 \; rad/s$ is the cutoff frequency of the reference model. By setting $\omega_1 = \omega_2$, the trajectory follows a straight line.
- $\zeta_i = 1$ is the relative damping ratio of the reference model.

The resulting guidance system follows a line from one setpoint to the next. The DP problem can then be defined as in Figure 4.3.

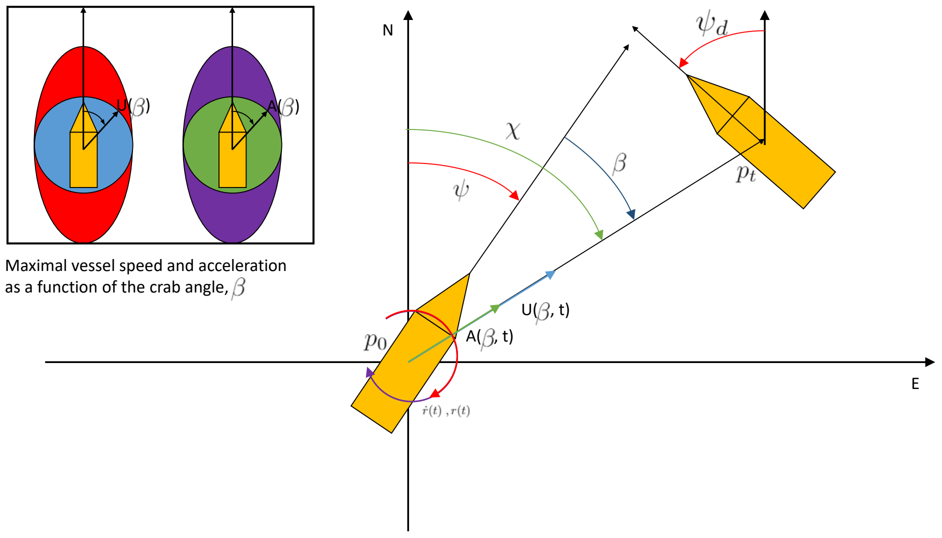


Figure 4.3: The guidance objective

Assume that the vessel follows a straight line from one point to another with a speed $U(\beta, t)$ and a rotational speed r(t) like in Figure 4.3. The crab angle, β is the angle between the the course (χ) and the heading (ψ). In this thesis the speed is to be equal for all crab angles so that $U(\beta, t) = U(t)$. Moreover, U is saturated by a maximal velocity U_{max} and r is saturated by a maximal yaw rate r_{max} so that:

$$U_{sat} = sat(\sqrt{u^2 + v^2}) \tag{4.7}$$
$$u_{sat} = U_{sat} \cos \beta \tag{4.8}$$
$$v_{sat} = U_{sat} \sin \beta \tag{4.9}$$
$$r_{sat} = sat(r) \tag{4.10}$$

In (Nørgaard Sørensen et al., 2018) so-called feasible sets of the velocities were investigated for the CSAD. In principle, they found the limits for coupled motion like in Figure 4.4.

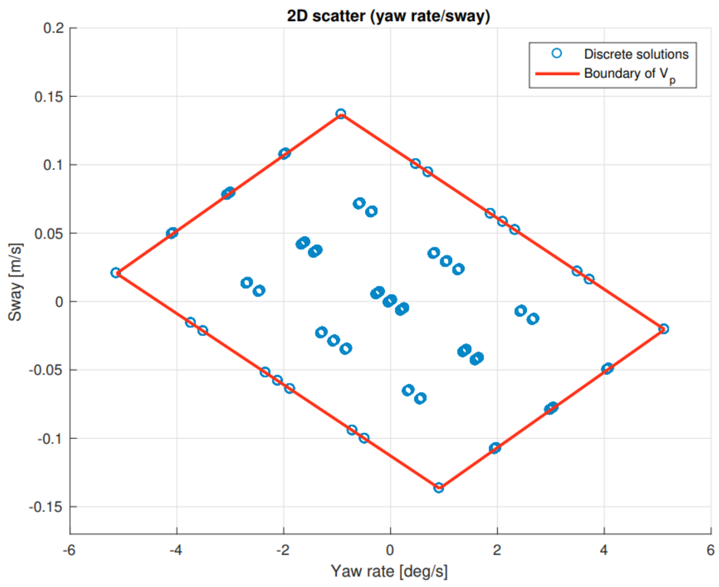


Figure 4.4: Feasible velocities for combined surge/sway: [Courtesy (Nørgaard Sørensen et al., 2018)]

The speed is limited to $U_{max} = 0.075$ m/s and the yaw rate limited by r_{max} = 3 deg/s This is just a simplification and can lead to unfeasible velocities for rare cases. In total, the parameters were chosen to be:

Table 4.1: Reference trajectory parameters

Parameter	Value
U_{max}	0.075 m/s
r_{max}	3 [deg/s]
ζ_i	1
ω_i	0.1

4.2 Model-based observer

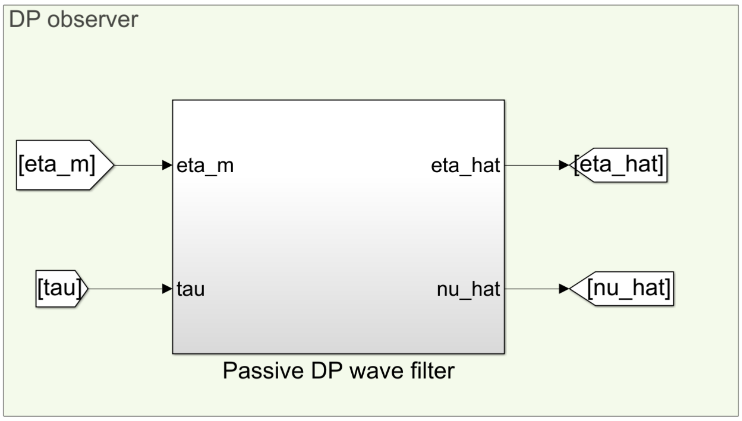


Figure 4.5: The Simulink block of the observer

Figure 4.5 illustrates the inputs and outputs of the observer. It takes in the measured pose, η_m and commanded forces, τ and calculates the LF position and velocity estimates, $\hat{\eta}$ and $\hat{\nu}$. It is required that the observer has the following functionality:

- *Wave filtering.* The motion of CSAD is classified into LF and WF motion. For DP purposes, the WF motions are not controlled. This is often because the vessel does not have the power nor thrust capacity to do a appreciable difference. The observer filters out the WF motion and sends a LF motion signal to the controller.
- *Reconstruction of non-measured data.* For CSAD, the measured states are towing carriage-fixed pose measurements, $\eta_m = [x_m,\ y_m,\ \psi_m]^\top$. However, the controller needs the velocities in the body-fixed coordinate system. The observer reconstructs the LF velocities for the controller.

In this thesis an observer called the *nonlinear passive observer (NLP) is used.* The NLP observer is advantageous in its tunability and that it meets the requirements of global exponential stability (GES).

The nonlinear observer design is based on the 3 DOF CDM found in Section 3.3.5 and is represented in state-space form as

$$\dot{\hat{\xi}} = A_\omega \hat{\xi} + K_1 \bar{y} \tag{4.11a}$$
$$\dot{\hat{\eta}} = R(\psi)\hat{\nu} + K_2 \bar{y} \tag{4.11b}$$
$$\dot{\hat{b}} = -T_b^{-1} + K_3 \bar{y} \tag{4.11c}$$
$$M\dot{\hat{\nu}} = -D\hat{\nu} + R^T(\psi)\hat{b} + \tau + R^T(\psi)K_4 \bar{y} \tag{4.11d}$$
$$\hat{y} = \hat{\eta} + C_\omega \xi \tag{4.11e}$$

Where the observer gains in Equation 4.11 are given as

$$K_1 = \begin{bmatrix} k_1 & 0 & 0 \\ 0 & k_2 & 0 \\ 0 & 0 & k_3 \\ k_4 & 0 & 0 \\ 0 & k_5 & 0 \\ 0 & 0 & k_6 \end{bmatrix}, K_2 = \begin{bmatrix} k_7 & 0 & 0 \\ 0 & k_8 & 0 \\ 0 & 0 & k_9 \end{bmatrix}$$
$$K_3 = \begin{bmatrix} k_{10} & 0 & 0 \\ 0 & k_{11} & 0 \\ 0 & 0 & k_{12} \end{bmatrix}, K_4 = \begin{bmatrix} k_{13} & 0 & 0 \\ 0 & k_{14} & 0 \\ 0 & 0 & k_{15} \end{bmatrix}$$

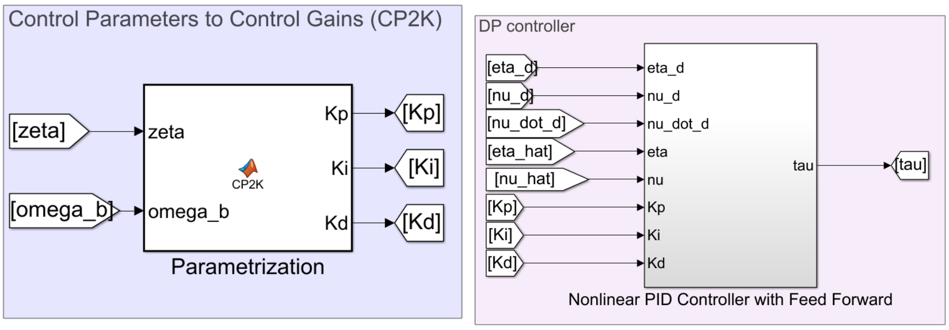
These matrices are tuned according to the tuning rules of (Fossen, 2011).

$$k_i = -2(\zeta_{ni} - \lambda_i)\frac{\omega_{ci}}{\omega_{oi}} \tag{4.12a}$$
$$k_{3+i} = 2(\zeta_{ni} - \lambda_i)\omega_{oi} \tag{4.12b}$$
$$k_{6+i} = \omega_{ci} \tag{4.12c}$$
$$k_{9+i} >> \frac{k_{12+i}}{T_{b,i}} \tag{4.12d}$$

Where

- $\zeta_n = 1 > \lambda$ is a damping parameter (Typically 1)
- $\lambda = 0.1$ is the relative damping of the wave spectrum (Typically 0.1)
- ω_o is the peak frequency of the wave spectrum
- $\omega_c > \omega_o$ is the filter cut-off frequency ($\omega_c = 1.2255\omega_o$ is used)
- $T_{b,i} >> 1$ (i = 1,...3) are the bias time constants (1000 s commonly used in full scale). But for the CSAD doing transient motion the bias can change rapidly so it is set to $T_{b,i} = 100/\sqrt{90} = 10.54$ s because of time scaling.

4.3 Model-Based DP controller



(a) Parametrization **(b)** DP controller

Figure 4.6: The parametrized model-based DP controller

The DP controller in this thesis is concerned with the control objective called trajectory tracking. Moreover, it is categorised as low-speed tracking according to (Sørensen, 2019). Trajectory tracking means that the desired vessel motion has temporal and spatial constrains, meaning that $\eta_d = \eta_d(t)$, for example.

The proposed controller shown in Figure 4.6 (b) is a nonlinear PID with feedforward adopted from (Fossen, 2003). It takes in the desired pose (η_d), velocities (ν_d), accelerations ($\dot{\nu}_d$) as well as the estimated pose ($\hat{\eta}$) and velocity ($\hat{\eta}$). It also has the PID controller gain matrices Kp, Ki and Kd as inputs to enable autotuning. These are further parametrized by the relative damping (ζ) and bandwidth (ω_b) of the controller like seen in Figure 4.6 (a). Moreover, the outputs of the controller are the desired forces in the body-fixed reference frame, τ.

The nonlinear PID with feed-forward from Figure 4.6 (b) is explained in Section 4.3.1, and the parametrization in Figure 4.6 (a) is described in Section 4.3.2.

4.3.1 Nonlinear PID controller with Feed-forward

Recall that the guidance system in Section 4.1 computes the desired positions, velocities and accelerations, η_d, ν_d and $\dot{\nu}_d$. Furthermore, recall the LF CDM from Section 3.3.1. Assume no bias. Then while following the trajectory, the model-based feed forward terms can be simplified to Equation 4.13

$$M\dot{\nu}_d + D\nu_d = \tau_{FF} \tag{4.13}$$

Given that this model was perfect, feed-back control would not be necessary. Imperfections in the model and disturbences does, however, make a feed-forward controller insufficient. Therefore, to create a robust controller, feed-back control is added. For feed-back control, the PID controller is used.

$$\eta_e = \eta - \eta_d \tag{4.14}$$

$$\dot{\xi} = \eta_e \tag{4.15}$$

$$\nu_e = R(\psi)^\top \dot{\eta_e} \tag{4.16}$$

$$\tau_{PID} = -\underbrace{MK_p}_{K_p}\underbrace{R(\psi)^\top\eta_e}_{e(t)} - \underbrace{MK_i}_{K_i}\underbrace{R(\psi)^\top\xi}_{\int_0^t e(t)dt} - \underbrace{MK_d}_{K_d}\underbrace{\nu_e}_{\frac{d}{dt}e(t)} \tag{4.17}$$

Combining the feed-forward control from Equation (4.13) and the feed-back control from the PID controller in Equation 4.17, the nonlinear PID with feed-forward from (Fossen, 2003)

$$\dot{\xi} = \eta_e \tag{4.18}$$

$$\tau = \underbrace{-M(K_iR^\top(\psi)\xi + K_pR^\top(\psi)\eta_e + K_d\nu_e)}_{\tau_{PID}} + \underbrace{M\dot{\nu}_d + D\nu_d}_{\tau_{FF}} \tag{4.19}$$

Where $\eta_e = \eta - \eta_d$ and $\nu_e = \nu - \nu_d$. Thus, the controller is tuned by adjusting the control gain matrices, K_p, K_i and K_d. Furthermore, the two last terms in (4.19) are feed-forward terms. In addition, all the control gains, K_p, K_i, K_d, are body-fixed gains.

4.3.2 The parametrization of the control gains

The tracking error from the nonlinear PID controller in Equation 4.17 is defined as $x = [\xi^\top \ \eta_e^\top \ \nu_e^\top]^\top$. The error dynamics of the vessel can be expressed in state-space form.

$$\dot{x} = T^\top(\psi) A_c T(\psi) x \tag{4.20}$$

Where

$$A_c = \begin{bmatrix} 0 & I & 0 \\ 0 & 0 & I \\ -K_i & -K_p & -(M^{-1}D + K_d) \end{bmatrix} \tag{4.21}$$

and

$$T(\psi) = \begin{bmatrix} R(\psi) & 0 & 0 \\ 0 & R(\psi) & 0 \\ 0 & 0 & I \end{bmatrix} \tag{4.22}$$

In order to do pole placement, Equation (4.20) is solved in the Laplace domain.

$$\det(Is - T^\top(\psi) A_c T(\psi)) = 0 \tag{4.23}$$

Which gives

$$\Lambda^3 + (M^{-1}D + K_d)\Lambda^2 + K_p\Lambda + K_i = 0 \tag{4.24}$$

Where $\Lambda = Diag([s, s, s])$. Compare to the characteristic polynomial for a third-order response (mass-damper cascaded with low-pass filter):

$$(\Lambda + \Omega_n)(\Lambda^2 + 2\Gamma\Omega_n\Lambda + \Omega_n^2) = \tag{4.25}$$

$$\Lambda^3 + \underbrace{\Omega_n(1 + 2\Gamma)}_{M^{-1}D+K_d}\Lambda^2 + \underbrace{\Omega_n^2(1 + 2\Gamma)}_{K_p}\Lambda + \underbrace{\Omega_n^3}_{K_i} = 0 \tag{4.26}$$

Where $\Omega = Diag([\omega_{n1} \ \omega_{n2} \ \omega_{n3}])$ and $\Gamma = Diag([\zeta_1 \ \zeta_2 \ \zeta_3])$ are the diagonal matrices containing the natural frequencies and relative damping in surge, sway and yaw. Comparing (4.24) with (4.26) yields

$$K_d = \Omega_n(1 + 2\Gamma) - M^{-1}D \tag{4.27}$$

$$K_p = \Omega_n^2(1 + 2\Gamma) \tag{4.28}$$

$$K_i = \Omega_n^3 \tag{4.29}$$

Furthermore, (Fossen, 2011) states that adequate tracking performance and stability requires the bandwidth of the motion control system to be higher than that of the reference model. The bandwidth in one dimension is given as

$$\omega_b = \omega_n \sqrt{1 - 2\zeta^2 + \sqrt{4\zeta^4 - 4\zeta^2 + 2}} \tag{4.30}$$

By this definition the bandwidth of the guidance system is 0.064 rad/s for all DOFs. Moreover, a parametrized nonlinear PID with feed-forward control has been established. The control gains, K_p, K_i and $_d$ are parametrized by 6 parameters, namely the bandwidths (ω_{bi}) and relative dampings (ζ_i) of the controller in all 3 DOFs. From hereon out, the control parameters refers to the the bandwidths and relative dampings of the controller.

The CSAD does however require both thruster orientation α and control input u for all 6 azimuth-thrusters. To transform the high-level control forces τ to the low-level commands α and u, a thrust allocation is applied in the Section 4.4.

4.4 Thruster Allocation

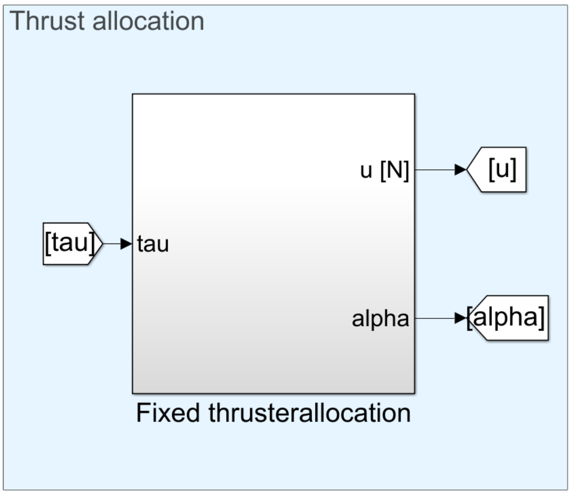


Figure 4.7: The Simulink block of the fixed thrust-allocation

The following thruster allocation was adopted from (Lyngstadaas, 2018). Recall the thrust-configuration matrix established in Section 3.2.2. For simplified and predictable thrust allocation, the orientations of the thrusters are fixed. Hence, there is a constant relationship between the commanded forces and the control input u

$$u = K_T^{-1} T^{\dagger}(\alpha)\tau_{sat} \tag{4.31}$$

Where the K_T is diagonal thrust coefficient matrix. The psuedo inverse of T(α), $T^{\dagger}(\alpha)$ was computed using the Matlab function *pinv(T)*. Moreover, τ_{sat} is the magnitude and rate saturated (MRS) forces are calculated computed as

$$\dot{\delta} = sat_r(\dot{\tau} + K(\tau - \delta) \tag{4.32}$$

$$\tau_{sat} = sat_m(\delta) \tag{4.33}$$

Where K > 0 is a diagonal tuning matrix used to decide the speed in the inner loop of the MRS model like in a low-pass filter. Setting $K_{ii} < 1$ gives a slower convergence towards the commanded forces than the rate saturation, while $K_{ii} > 1$ gives an accurate tracking that enforces the rate saturation. The parameters for the MRS model are given in Table 4.2.

Table 4.2: Parameters used for magnitude and rate saturation in the thrust allocation

Parameters	Values
τ^*_{max}	$[3\ N,\ 3\ N,\ 3\ Nm]^\top$
$\dot{\tau}_{max}$	$[2.88\ N/s,\ 1.6\ N/s,\ 1.36\ Nm/s]^\top$
K	$diag([5,\ 2.78,\ 2.36])$ [1/s]

Note that these parameters are adapted to the thrust configuration. Furthermore, the magnitude saturation is not correct according to (Lyngstadaas, 2018), and should be $\tau_{max} = [3.6\ N,\ 2\ N,\ 1.7\ Nm]^\top$. The effect of this error is that the thrust allocation can command infeasible forces for sway and yaw.

Chapter 5

Experiment setup for autotuning

In Chapter 3 the simulation verification model of CSAD was explained and in Chapter 4, the DP system was established as a motion control system. However, the motion control system needs the following inputs

1. Setpoints defining some maneuver. A maneuver needs to be defined

2. Control gains, K_p, K_i and K_d. which are calculated from the control parameters, namely control bandwidths (ω_b) and relative dampings (ζ). The parameters are the tuning variables that needs to be defined by an autotuning component.

In this chapter, the experimental setup for autotuning in simulation and laboratory are elaborated. It includes a description of the test maneuver in Section 5.1 and an explanation of the simulation and laboratory autotuning schemes in Section 5.2 and 5.3, respectivly.

5.1 Test maneuver

Initially, the plan was to do the 4-corner DP-test like in (Værnø et al., 2019). It is a useful test that includes coupled and decoupled performance in surge, sway and yaw. However, to reduce the time of each maneuver, a simple 2-setpoint test was created as shown in Figure 5.1.

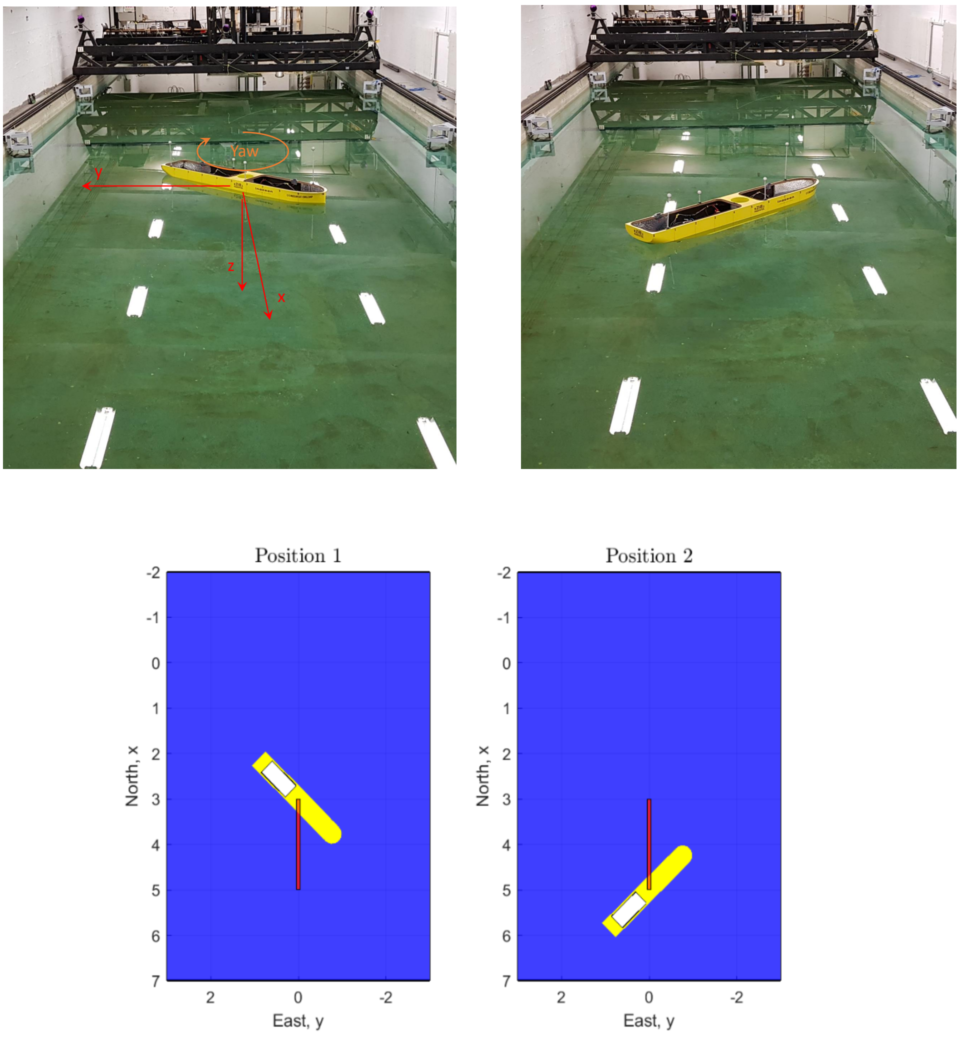
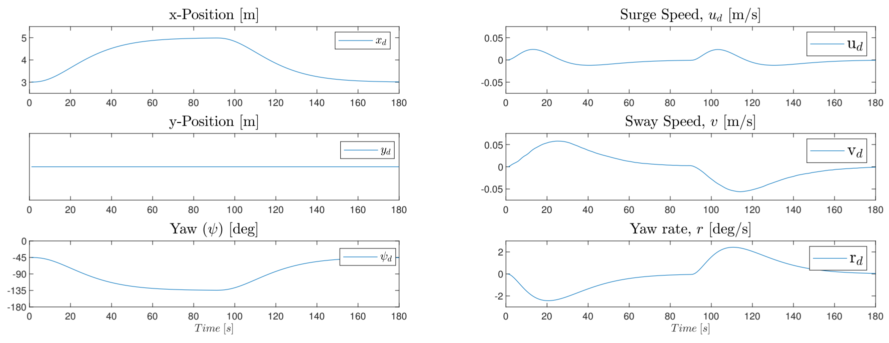


Figure 5.1: The 2-setpoints in the autotuning, $\eta_{d1} = [3, 0, -45°]$ and $\eta_{d2} = [5, 0, -135°]$. The upper figures are from the MC-Lab. The lower figures from a home-made animation tool of MC-lab. The red line defines the positional trajectory that the vessel should follow

One test in Figure 5.1 consists in two transients, namely line tracking from Position 2 to position 1, and line tracking from Position 1 to position 2. The corresponding trajectory is displayed in Figure 5.2



(a) $Pose, \eta_d$ **(b)** $Velocity, \nu_d$

Figure 5.2: Desired position and desired body-fixed velocities, η_d and ν_d.

5.2 Simulation

The simulation setup is inspired by (Værnø et al., 2019) who did DP observer tuning using DFO.

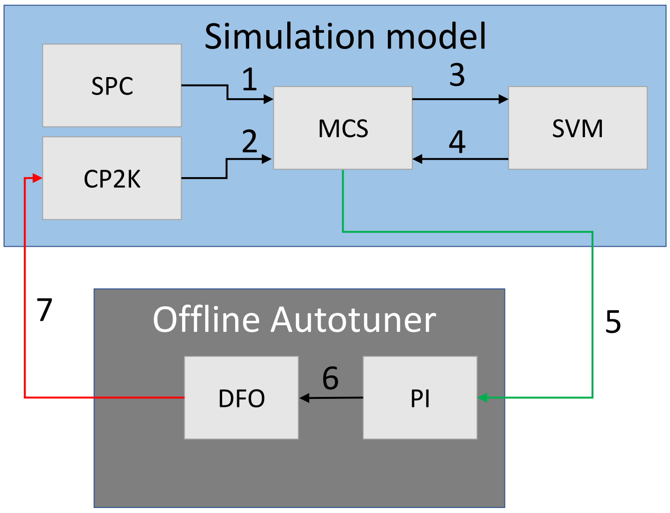


Figure 5.3: The simulation setup for autotuning. SPC is short for setpoint control and defines the maneuver, CP2K transforms control parameters to control gains, MCS is the motion control system, SVM is the simulation verification model, PI is the performance indicator function and DFO is the derivative-free optimization.

Figure 5.3 simply explains the way the offline autotuning in simulation works. It is like a feed-back loop. The numbering of the signals mean

1. Setpoint: The setpoint control (SPC) sets the setpoint to η_{d2} to the guidance in the motion control system (MCS) when the a test starts. Half-way into the simulation, it swaps setpoint to η_{d1}.

2. Control gains: The control parameters to gains (CP2G) calculates control gains K_p, K_i, K_d which are constant throughout the test. They are calculated from the control parameters.

3. Low-level actuator control signals: The MCS sends low-level actuator control signals α and u into the simulation verification model (SVM) at each timestep.

4. Measured position: The MCS recieves the measured position, η_m at each timestep.

5. Timeseries of performance data. When the simulation model is finished simulating, timeseries of performance data are sent to the offline autotuner in Matlab. The time-series of data are then used to calculate the PI. These performance time series are sent after each simulation.

6. Performance indicator: A scalar PI is sent to the DFO. The PI acts as an evaluation of the objective function that the DFO tries to optimize.

7. New control parameters. The optimization sends out new control parameters, $[\zeta_1\ \ \zeta_2\ \ \zeta_3\ \ \omega_{b1}\ \ \omega_{b2}\ \ \omega_{b3}]$. These parameters are sent in before each simulation.

To complete the system architecture, the offline autotuner in Matlab needs to be derived. This will be further detailed in Chapter 6.

5.3 MC-lab Test

The real world system utilizes a real-time embedded industrial controller called the NI CompactRIO (cRIO). This system runs real-time control systems that are programmed in LabView or Simulink code. The topology of the cRIO can be seen in Figure 5.4

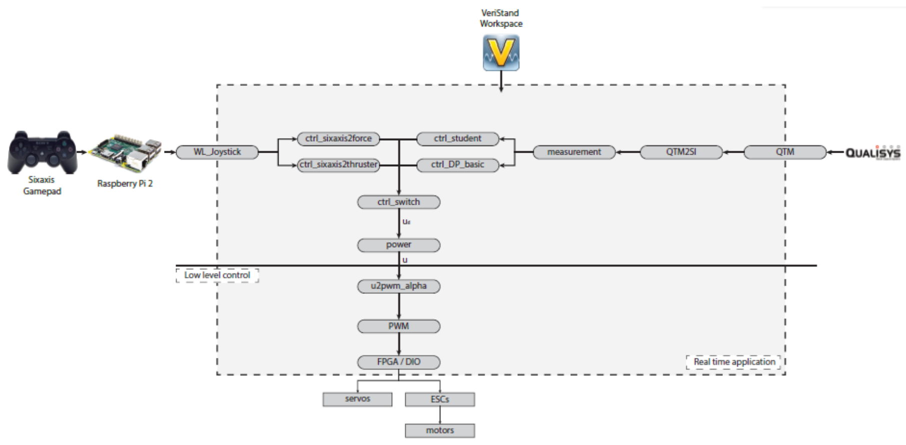


Figure 5.4: Software topology for the CompactRIO. Note that the block called ctrlstudent is called ctrlcustom in the current setup [Courtesy:(Bjørnø, 2016)]

The inputs to the cRio are

- Veristand Workspace. The Veristand workspace allows simple interaction with the system. It is possible to adjust values of so-called Veristand inputs as well as monitoring of different variables within the system and logging.
- Qualisys Tracking Managers. The camera/reflector measurs the pose. It is then sent as inputs to the cRIO.
- The sixaxxis gamepad (Playstation 3 controller) can be used for choosing control mode and can also be used for manual control.

The sixaxxis gamepad can select between 4 modes of control, including manual control of the thrusters and a custom control mode where the user can implement their controller. Hence, it is necessary to establish a custom control block that takes in only position measurements from the QTM and returns low-level control commands (α,u). The custom control consists of the motion control system as well as an online autotuning component as illustrated in Figure 5.5

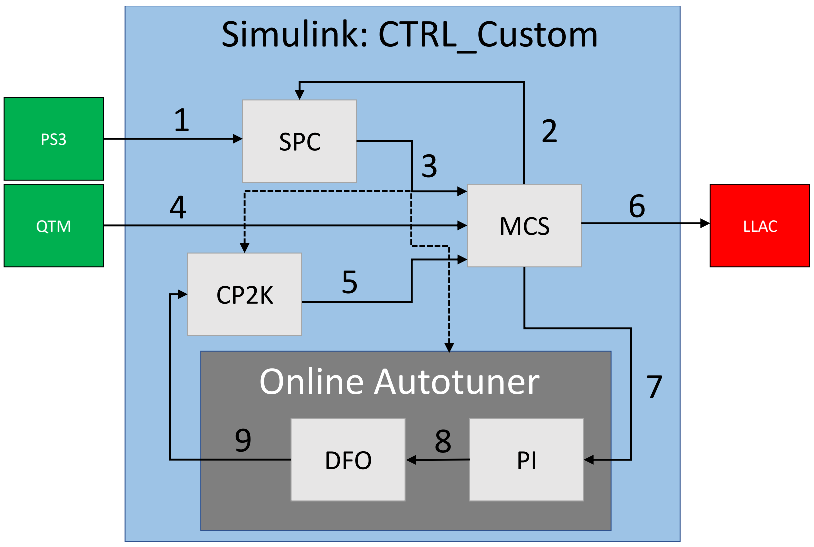


Figure 5.5: The laboratory setup for autotuning. SPC is short for setpoint control and defines the maneuver, CP2K transforms control parameters to control gains, MCS is the motion control system, SVM is the simulation verification model, PI is the performance indicator function and DFO is the derivative-free optimization. QTM stands for Qualisys Tracking Manager. LLAC stands for low-level actuator control and PS3 refers to the Playstation 3 controller/sixaxxis gamepad

Figure 5.5 simply explains the information flow between the different components of the system. The numbering of the signals mean

1. (X)-button press: Is 1 when the (X) button is pressed on the PS3 controller
2. Estimated pose. The estimated pose $\hat{\eta}$ is sent to the SPC. SPC uses it to decide whether the vessel has stabilized. If so the guidance is reset from this point.
3. Setpoint and reset signals: The setpoint control (SPC) is responsible for setting the setpoint for the MCS and task handling. When signal 1. is activated, the setpoint is set to η_{d1}. Simultaneously integral and guidance reset signals are sent out to MCS. The setpoint is set to η_{d2} when a test starts. Simultaneously integral and guidance reset signals are sent out to MCS and the PI so that it starts integrating. Halfway in the simulation, the setpoint is set to η_{d1}. When the simulation time is over, the SPC resets integrals, guidance and switches to a safe controller in CP2K, and sends a signal so that DFO applies optimization and PI stops integrating. When the vessel has stabilized, SPC changes setpoint, resets integrals, guidance and switches to the control

new control gains.

4. Measured position: The MCS recieves the measured position, η_m, from the Qualisys Tracking Manager (QTM).

5. Control gains: The control parameters to gains (CP2K) calculates control gains K_p, K_i, K_d which are constant throughout the test. They are calculated from the control parameters. However note that it has a safe controller that is activated when the test is over.

6. Low-level actuator control signals: The MCS sends low-level actuator control signals α and u into the low-level actuator control (LLAC) for the thrusters.

7. Live performance data. During the test, performance data are sent to calculate the performance indicator (PI).

8. Performance indicator: A scalar PI is sent to the DFO when the test is over. The PI then acts as an evaluation of the objective function that the DFO tries to optimize.

9. New control parameters. The optimization sends out new control parameters when the DFO has been applied, $[\zeta_1\ \zeta_2\ \zeta_3\ \omega_{b1}\ \omega_{b2}\ \omega_{b3}]$. These parameters are sent to CP2K, but are not used until the vessel has stabilized and a new test begins.

Moreover, the functionality of the online autotuner is best explained through example. Assume that Figure 5.6 describes one dimension of the tracking problem.

5.6.

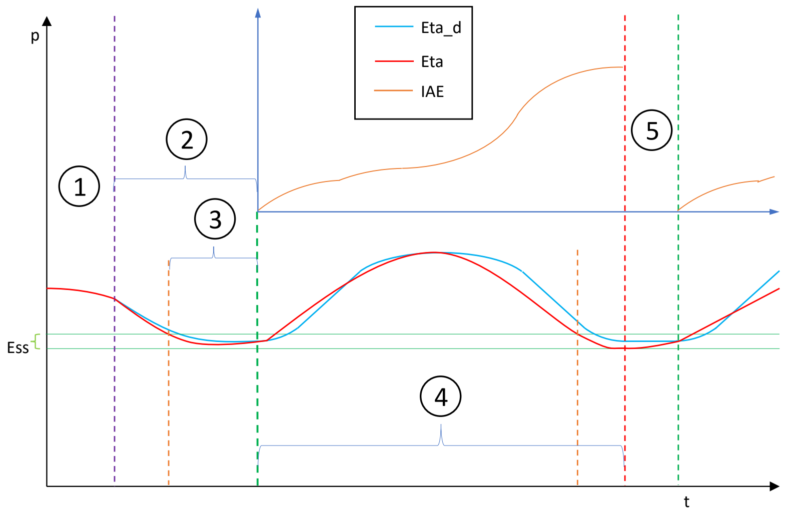


Figure 5.6: Methodology of the tuning. p is the position. Eta_d can be considered x$_d$(t), Eta can be considered as x(t) and IAE is the integral of absolute error and is an example of a performance indicator, PI

The process in 5.6 can be divided in five phases

1. This phase is the phase before the ship is put into autotuning mode. When the user presses the (X)-button on the PS3 controller, a signal is sent to the SPC. The SPC then sends reset signals to the MCS so that all integrals are reset. The position in the guidance system is reset to the current position and the setpoint is set to the origin of the autotuning, namely, $\eta_{d1} = [3, 0, -45°]^{\top}$. This way, the guidance system creates a line trajectory from where it is when (X) is pressed, to η_{d1}.

2. This phase is used for a "safe" trajectory tracking to η_{d1}. The control parameters used for this movement are predefined and are known to give a ”safe behaviour”.

3. This is the stabilising phase that is used to give the ship a similar starting basis for each performance test. The requirement is that the ship stabilises within the error band, E_{ss} for 20 seconds. The error band is $E_{ss} = [0.02, 0.02, 2°]^{\top}$.

4. This phase is the testing phase and it lasts for 3 min (180 s). When the stabilising phase is concluded, the DFO changes the control parameters and this phase initiates. The SPC initiates the setpoint to η_{d2} = [5,0,-135°]$^{\top}$.

Note that the trajectory should be created from where the vessel is at initiation. This is done to minize the effect of biased initial conditions. Moreover Halfway into the test ($t = 90$ s), the setpoint is set to the origin (η_{d1}) for the remaining 90 s. During the test, the PI (IAE) is calculated as seen in Figure 5.6.

5. When phase 4. is over, the total PI is the objective function used for the DFO to calculate new and more optimal control parameters. Meanwhile, the "safe" controller is used to stabilize the vessel at the test origin. This phase is used to stabilise the vessel further. After phase 5., phase 4. and 5. are repeated until satisfactory results are obtained.

As mentioned in Section 5.2, the PI and DFO need to be established to complete the autotuning architecture and will be elaborated in Chapter 6.

Chapter 6

Autotuning of control gains using performance indicators and derivative-free optimization

This chapter covers how to create the actual autotuning component introduced in Section 5.2 and 5.3, with special focus on the DFOs and the PIs. First, the optimization problem is formulated in Section 6.1 and the bounds defined in Section 6.2. 3 candidate PIs are established and tested on the simulation model in Section 6.3 and one is selected. Thereafter, 3 candidate DFO algorithms are established and tested on the simulation model in Section 6.4, and the preferred algorithm is selected. And finally, the implementation of the system in MC-Lab is explained in Section 6.5.

6.1 Formulation of optimization problem

The goal is simply stated to find the control parameters, x, that yield the best performance, f(x), subjected to the some constraints $g_i(x)$. Mathematically,

$$\min \; f(x) \tag{6.1}$$

$$\text{subject to } g_i(x) \tag{6.2}$$

For $i = 1, 2, .., 6$. The control parameters, $x = [\zeta_1\ \zeta_2\ \zeta_3\ \omega_{b1}\ \omega_{b2}\ \omega_{b3}]^T$. The objective function, $f(x)$ is the PI, and the constraints, $g_i(x)$ represent the linear upper and lower bounds of the control parameters:

$$g_i(x) : x_{i,min} \leq x_i \leq x_{i,max} \tag{6.3}$$

In simulation testing, it was implemented as in Figure 6.1

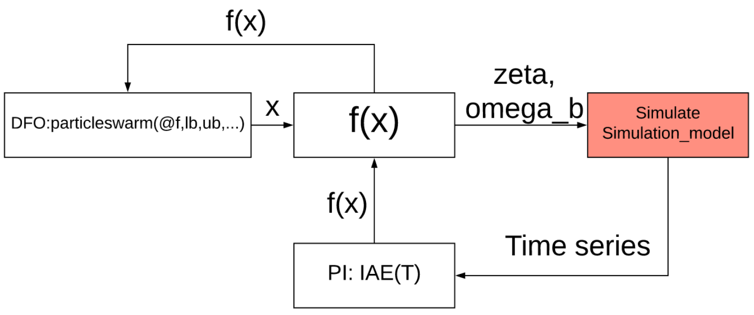


Figure 6.1: Example of implementation of offline autotuning with PSO as DFO algorithm and IAE as PI.

In the simulation setup described by Figure 6.1, there are three Matlab functions, namely an inbuilt DFO function (e.g., particleswarm), an objective function, f(x), that assigns the control parameters, x, to the Simulink model and runs the simulation. The simulation model contains toworkspace-blocks, that sends time series back to Matlab after the simulation is over, making it possible to calculate the PI (e.g., IAE). The PI is sent back to the DFO as objective function, f(x).

6.2 The bounds of the optimization

Recall the parametrization of the control gains derived in Section 4.3.2

$$\begin{aligned}
\mathbf{K_d} &= \Omega_n(1 + 2\Gamma) - M^{-1}D \\
\mathbf{K_p} &= \Omega_n^2(1 + 2\Gamma) \\
\mathbf{K_i} &= \Omega_n^3 \\
\omega_{ni} &= \frac{\omega_{bi}}{\sqrt{1 - 2\zeta_i^2 + \sqrt{4\zeta_i^4 - 4\zeta_i^2 + 2}}}
\end{aligned}$$

for $i = 1, 2, 3$ where ω_{bi} is the bandwidth and $\Omega_n = Diag([\omega_{n1}\ \omega_{n2}\ \omega_{n3}])$ and $\Gamma = Diag([\zeta_1\ \zeta_2\ \zeta_3])$ are the natural frequency matrix and relative damping matrix, respectively. The parametrization has the following advantages :

1. Reduced the number of parameters of the optimization.
2. Enabled linear bounds. The elements K_p, K_i and K_d are interrelated and it is not practical to use linear bounds for these as this will often lead to failure. On the other hand, the bandwidth and relative damping can easily use linear bounds.
3. The parametrisation is intuitive and the parameters can be understood as physical entities.

It is stated in (Fossen, 2011) that the relative damping can be chosen between 0.8-1.0 and that the bandwidth is between 0.01 rad/s for large oil tankers to 0.1 rad/s for small ships and underwater vehicles. Froude scaled with $\lambda = 90$, this equates to model scale bandwidths between 0.1 rad/s for large oil tankers to 1 rad/s for small ships and underwater vehicles.

It is also stated that adequate tracking performance requires the bandwidth of the control system to be higher than the bandwidth of the guidance system. Recall from Section 4.3 that the bandwidth of the guidance system is 0.064 rad/s in all DOFs.

CSAD is a relatively large vessel in full-scale ($L_{oa} = 232$), and it was therefore assumed that a bandwidth domain of 0.1-0.5 rad/s would suffice. Moreover, the search domain of the relative damping was set to 0.7-1.5 to create a larger search domain. This gave the bounds in Table 6.1.

Table 6.1: The upper and lower bounds of the optimization

Parameter	Surge	Sway	Yaw
$\omega_{b,min}$	0.1 $[rad/s]$	0.1 $[rad/s]$	0.1 $[rad/s]$
$\omega_{b,max}$	0.5 $[rad/s]$	0.5 $[rad/s]$	0.5 $[rad/s]$
ζ_{min}	0.7 $[-]$	0.7 $[-]$	0.7 $[-]$
ζ_{max}	1.5 $[-]$	1.5 $[-]$	1.5 $[-]$

Which is equivalent to the relatively large search-space described by the minimal and maximal control gains:

$$
\begin{array}{cccc}
 & \overbrace{\hphantom{\begin{bmatrix}0.024 & 0 & 0\end{bmatrix}}}^{K_p} & \overbrace{\hphantom{\begin{bmatrix}0.001 & 0 & 0\end{bmatrix}}}^{K_i} & \overbrace{\hphantom{\begin{bmatrix}0.199 & 0 & 0\end{bmatrix}}}^{K_d} \\
\text{Min:} & \begin{bmatrix}0.024 & 0 & 0\\ 0 & 0.024 & 0\\ 0 & 0 & 0.024\end{bmatrix} & \begin{bmatrix}0.001 & 0 & 0\\ 0 & 0.001 & 0\\ 0 & 0 & 0.001\end{bmatrix} & \begin{bmatrix}0.199 & 0 & 0\\ 0 & 0.194 & -0.026\\ 0 & 0.003 & 0.017\end{bmatrix} \\
\text{Max:} & \begin{bmatrix}7.140 & 0 & 0\\ 0 & 7.140 & 0\\ 0 & 0 & 7.140\end{bmatrix} & \begin{bmatrix}2.385 & 0 & 0\\ 0 & 2.385 & 0\\ 0 & 0 & 2.385\end{bmatrix} & \begin{bmatrix}5.305 & 0 & 0\\ 0 & 5.301 & -0.026\\ 0 & 0.003 & 5.124\end{bmatrix}
\end{array}
$$

6.3 Performance indicator functions

The PI relevant for this thesis is a scalar value that defines the performance of the trajectory tracking. More specifically, the trajectory tracking test defined in Section 5.1.

The performance functions are evaluated on the following criterions

1. Tracking Accuracy
2. Avoiding excessive control action (wear and tear)
3. Quality
4. Tuning

To evaluate performance, the following time series are utilized

- The time series of the pose error, $\eta_e(t) = \hat{\eta}(t) - \eta_d(t)$. This time series can be used to evaluate the accuracy of the tracking.
- The time series of the commanded generalised forces, $\tau(t)$. It can be used as a counterweight to the accuracy.
- The time series of the velocity, $\hat{\nu}(t)$. The velocity can be used to for example calculate the total energy consumption.

The time series have different magnitudes and units. Therefore, inspired by (Lyngstadaas, 2018), it is chosen to normalise the time series. The division in 6.4 is element-wise.

$$\bar{\eta}_e(t) = \frac{\eta_e(t)}{\eta_{e,max}}\, , \bar{\nu}(t) = \frac{\nu(t)}{\nu_{max}}\, , \bar{\tau}(t) = \frac{\tau(t)}{\tau_{max}} \tag{6.4}$$

Where the absolute limits are:

Table 6.2: Max error, velocity and control effort

Parameter	Value
$\eta_{e,max}$	[2 m, 2 m, 90° deg]$^\top$
ν_{max}	[0.4142 m/s, 0.109 m/s, 6.327 deg/s]$^\top$
τ_{max}	[3.6 N, 2.0 N, 1.7 Nm]$^\top$

In (Sørensen and Breivik, 2015), controllers are compared using the integral of absolute error (IAE), integral of squared error (ISE), integral of absolute error multiplied by time (ITAE) and introduces the integral of absolute error multiplied by work (IAEW). Moreover, in (Eriksen and Breivik, 2017) the measure integral of absolute differentiated control, IADC. These PIs are described below.

- Integral of absolute error, IAE:

$$IAE(t) = \int_0^t |\bar{\eta}_e(\sigma)| d\sigma$$

  IAE describes the overall accuracy. It acts as a simple accuracy measure, but it is intuitive in that the perfect IAE is zero. To the authors knowledge, it is the most common performance measure

- Integral of squared error (ISE):

$$ISE(t) = \int_0^t \bar{\eta}_e(\sigma)^\top \bar{\eta}_e(\sigma) d\sigma$$

  ISE is similar to IAE in that it emphasizes the accuracy of the tracking. It does however lay more weight on large errors.

- Integral of absolute error multiplied by time, ITAE:

$$ITAE(t) = \int_0^t t|\bar{\eta}_e(\sigma)| d\sigma$$

  The ITAE focus more on errors later in the simulation. Thus, transient response is not weighed much, while stationary errors are important. This measure is not so relevant for this thesis because the error has the same importance throughout the tracking test, independent on time.

- Integral of absolute error times work, IAEW:

$$IAEW(t) = \int_0^t |\bar{\eta}_e(\sigma)| d\sigma \int_0^t |\bar{\nu}(\sigma)^\top \bar{\tau}(\sigma)| d\sigma$$

The IAEW is a measure of error vs work. It weighs error, velocity and control input equally. Theoretically, the integral of error can be reduced to zero, while the integral of work can only be reduced to the minimal work for the maneuver.

- Integral of absolute derivative control, IADC:

$$IADC(t) = \int_0^t |\bar{\dot{\tau}}(\sigma)| d\sigma$$

The IADC is a pure wear and tear measure. It emphazises changes in control input, and gives no weight to the accuracy, and needs to be in combination with an accuracy measure to make sense. Since it is a derivative, it is highly susceptible to noise.

- The integral of absolute error and control, IAEC

$$IAEC(t) = \int_0^t |\bar{\eta}_e(\sigma)| + \rho|\bar{\tau}(\sigma)| d\sigma$$

Where $\rho = 0.15$ was set based on some tests. The measure weighs both control input and error. Theoretically, the integral of error can be reduced to zero, while the integral of control input can only be reduced to some minimum. This makes the tuning of ρ difficult.

The three chosen candidates were IAE, IAEW and IAEC. Inspired by (Værnø et al., 2019) who did observer tuning, it is chosen to use the 1. norm of the error, velocity and control input instead of the 2. norm. In summary, the 3 chosen candidate performance indicator functions are:

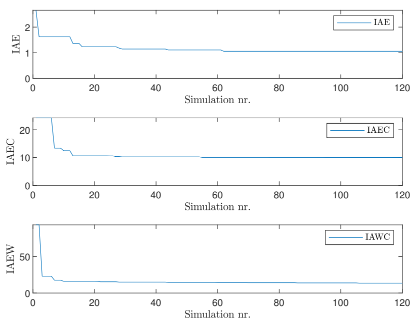
$$IAE(t) = \int_0^t |\bar{\eta}_e(\sigma)|_1 d\sigma \tag{6.5}$$

$$IAEW(t) = \int_0^t |\bar{\eta}_e(\sigma)|_1 d\sigma \int_0^t |\bar{\nu}(\sigma)^\top \bar{\tau}(\sigma)|_1 d\sigma \tag{6.6}$$

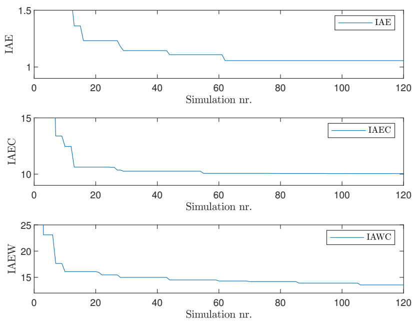
$$IAEC(t) = \int_0^t |\bar{\eta}_e(\sigma)|_1 + \rho|\bar{\tau}(\sigma)|_1 d\sigma \tag{6.7}$$

In terms of optimization, they are objective functions. To test these objective function, they were tested on the simulation model described in Chapter 3. and 4. PSO was used for the optimization with swarm size, N = 6, self-adjustment weight $b_1 = 0.1$, social adjustment weight $b_2 = 1.49$ and the algorithm used an adaptive inertia weight, $a \in [0.1,\ 1.1]$.

Figure 6.2, Table 6.3 and Table 6.4 indicates the results from optimization with 120 simulations with concern to these three performance integrals.



(a) Zoomed out



(b) Zoomed out

Figure 6.2: Convergence of PI as a Function of simulation number

Table 6.3: The performance of the optimization methods with respect to 1-IAE, 2-IAEC, and 3-IAEW. Hence, the diagonal elements should be the highest in each column. However note that this is not the case for the first column

Method	IAE	$IAEC$	$IAEW$
1-IAE	1.02	14.4	11.6
2-IAEC	1.35	13.5	11.1
3-IAEW	1.00	17.5	10.0

Table 6.4: Control parameters obtained when optimizing with respect to 1-IAE, 2-IAEC, and 3-IAEW

Method	ζ_{surge}	ζ_{sway}	ζ_{yaw}	$\omega_{b,surge}$	$\omega_{b,sway}$	$\omega_{b,yaw}$
1-IAE	1.05	0.70	0.87	0.49	0.44	0.27
2-IAEC	0.79	0.70	0.76	0.50	0.20	0.27
3-IAEW	1.05	0.70	0.70	0.49	0.35	0.34

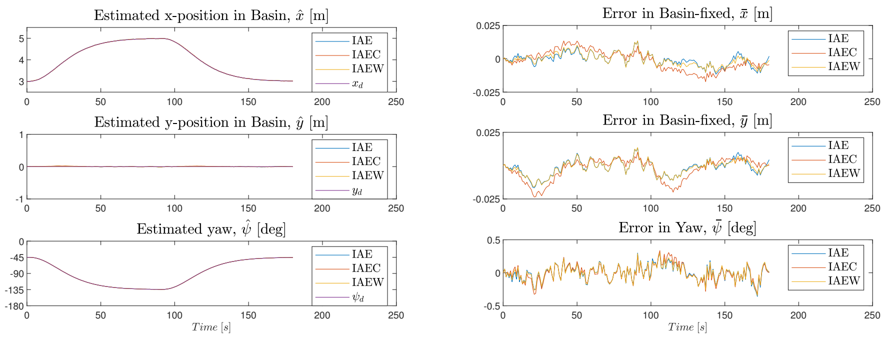
Table 6.3 illustrates that optimizing with respect to IAEW yields a better IAE than optimization with respect to IAE itself. This likely occurs because IAE is a factor in IAEW. Thus, it appears that IAE and IAEW yield similar results. Moreover, control parameters in Table 6.4 are illustrated below as control gains:

$$\text{IAE:}\ \overbrace{\begin{bmatrix} 2.0 & 0 & 0 \\ 0 & 0.46 & 0 \\ 0 & 0 & 0.32 \end{bmatrix}}^{K_p} \overbrace{\begin{bmatrix} 0.53 & 0 & 0 \\ 0 & 0.085 & 0 \\ 0 & 0 & 0.039 \end{bmatrix}}^{K_i} \overbrace{\begin{bmatrix} 2.5 & 0 & 0 \\ 0 & 1.01 & -0.026 \\ 0 & 0.003 & 0.71 \end{bmatrix}}^{K_d} \tag{6.8}$$

$$\text{IAEC:}\ \begin{bmatrix} 0.8 & 0 & 0 \\ 0 & 0.10 & 0 \\ 0 & 0 & 0.22 \end{bmatrix} \begin{bmatrix} 0.18 & 0 & 0 \\ 0 & 0.008 & 0 \\ 0 & 0 & 0.025 \end{bmatrix} \begin{bmatrix} 1.4 & 0 & 0 \\ 0 & 0.44 & -0.026 \\ 0 & 0.003 & 0.52 \end{bmatrix} \tag{6.9}$$

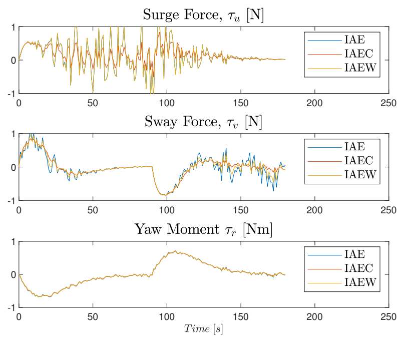
$$\text{IAEW:}\ \begin{bmatrix} 2.1 & 0 & 0 \\ 0 & 0.29 & 0 \\ 0 & 0 & 0.27 \end{bmatrix} \begin{bmatrix} 0.54 & 0 & 0 \\ 0 & 0.041 & 0 \\ 0 & 0 & 0.037 \end{bmatrix} \begin{bmatrix} 2.5 & 0 & 0 \\ 0 & 0.79 & -0.026 \\ 0 & 0.003 & 0.58 \end{bmatrix} \tag{6.10}$$

The results in Equation 6.8 and 6.10 illustrate that the surge and yaw parameters are very similar for IAE and IAEW. Moreover, the optimized gains are not the maximum or minimum control gains of the bound for any of the methods. The simulation results in terms of pose and errors are given in Figure 6.3 and control inputs, τ and velocities are given in Figure 6.4.

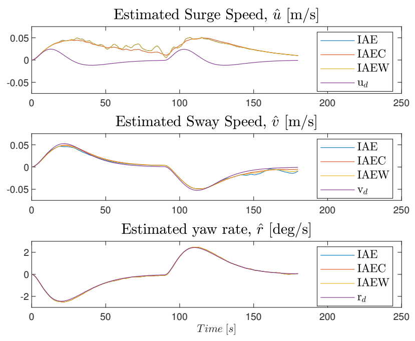


(a) Pose ($\hat{\eta}$) vs Reference (η_d) **(b)** Errors, η_e

Figure 6.3: Pose and error plots for the resulting control parameters



(a) Control input, τ



(b) Velocities ($\hat{\nu}$) vs Reference (ν_d)

Figure 6.4: Control input and velocities

Figure 6.3 Illustrates tracking accuracy. In Figure 6.3(a) it can be seen that all the solutions give decent tracking capabilities in the simulation. In Figure 6.3(b) the tracking accuracy is seen more clearly. It appears that the observer struggles with filtering the measurement noise. More importantly, it can be seen that tracking accuracy is lower (higher error) for the IAEC-gains. However, for the control forces and moments in Figure 6.4, IAEC-gains has lower oscillation in the surge and sway forces. Moreover, the surge force is oscillatory, which is likely because of the poor velocity estimate in surge in Figure 6.4(b). On the other hand, the velocity estimates are relatively smooth and precise for the sway speed and yaw rate.

In total, the IAE performance metric was chosen based on the arguments stated in Table

Table 6.5: Selection matrix for performance indicators

	IAE	IAEC	IAEW
Accuracy	Good	Ok	Good
Wear and Tear	Poor	Ok	Poor
Quality	Good	Ok	Poor
Tuning	No	Yes	No

Simply stated, IAEC was discarded because it requires a tuning parameter. Moreover, IAEW was discarded because it uses the estimated velocity, $\hat{\nu}$, which makes the quality of the performance measure poor. Besides the tuning parameter, IAEC is the favorite performance metric. However, from testing it is proposed to use only the feedback control, $\tau_{PID} = \tau - \tau_{FF}$, since this component ideally oscillates about zero.

6.4 DFO algorithms

The DFO algorithms use the objective function f(x) defined as the IAE. Their goal is to find the control parameters, x, that minimize the objective function, f(x). The objective function, IAE, will be based on data from real-world experiments for MC-Lab testing. Thus, the tuning as an optimization problem has the following characteristics:

- Non-smooth: Objective functions are contaminated with stochastic, random noise (waves, measurement noise etc.). This can make the objective function non-smooth.
- Nonlinear: The objective function is nonlinear because gain parametrization is nonlinear, controller is nonlinear, etc.

For nonsmooth, nonlinear problems, Matlab especially recommends the following solvers for optimization (Matlab, 2015):

1. *particleswarm*: The PSO has little supporting theory, but is often an efficient algorithm. According to (Sahib, 2015) and (Kaliappan and Thathan, 2014) PSO yielded the best performance in optimizing the control parameters of a PID controller when comparing some global optimization algorithms. It was also referred to as both efficient and accurate.
2. *surrogateopt*: The SGO provably converges to global optimum for bounded problems. The algorithm is also recommended for time consuming objective functions which is the case when there is a model test or a high-fidelity simulation in the objective function, IAE.
3. *fminsearch*: The Nelder-Mead simplex algorithm is known to work well for low-dimensional unbounded problems. Simple to use because of few tuning options. Furthermore, the Nelder-Mead simplex algorithm (fminsearch) was used in (Værnø et al., 2019) for DFO of a DP observer and therefore seems highly relevant.

Therefore, these three algorithms are investigated further in Section 6.4.1, 6.4.2, 6.4.3 and compared in Section 6.4.4. Moreover, the psuedo-code for the algorithms are in Appendix A.1, A.2 and A.3.

6.4.1 PSO

The particle swarm optimization is a global optimization method inspired by how swarms of birds and schools of fish collectively searches an area. The key is that the individuals in the swarm use their own best known position ($\vec{p}_1$), the globally best known position ($\vec{p}_2$) and its velocity ($\vec{v}_k$) to calculate how to move around in space.

The algorithm can be described as following

1. Initialization: The swarm is initialized randomly or at selected points in the search space with random velocities. The swarm , S = $\{x_1,\ x_2,,,x_N\}$ contains N candidate solutions, the so called particles. In the context of DP-tuning, a particle is a set of control parameters x = $[\zeta_1,\ \zeta_2\ \zeta_3\ \omega_{b1},\ \omega_{b2}\ \omega_{b3}\ ]$
2. Evaluation and updating: The fitness (IAE for DP tuning) of particle i in the swarm is then calculated, f(x_i).Then the locally best known and globally best known position ($\vec{p}_1$ and $\vec{p}_2$) and fitness of each particle is updated. The following updating law is used for each particle.

$$\vec{v}_{k+1} = \vec{a} \otimes \vec{v}_k + \vec{b}_1 \otimes \vec{r}_1 \otimes (\vec{p}_1 - \vec{x}_k) + \vec{b}_2 \otimes \vec{r}_2 \otimes (\vec{p}_2 - \vec{x}_k) \tag{6.11}$$
$$x_{k+1} = x_k + v_{k+1} \tag{6.12}$$

   Where

   - $\vec{r}_{1,2} \in [0, 1]$ are uniformly distributed random vectors.
   - $\vec{a}$ are the inertia weights.
   - $\vec{b}_{1,2}$ are the social and self adjustment weight. If the social weight dominates, the algorithm gravitates towards the best known position and the algorithm predictably converges quickly. However, using a higher self adjustment weight prevents local convergence and allows for a global search.
   - $\vec{p}_{1,2}$ are the best known local and global positions, respectively.
   - $\vec{v}_k$ is the previous velocity of the particle.
   - $\vec{p}_k$ is the previous position of the particle.
3. Step 2 is then repeated until satisfactory convergence is achieved.

The algorithm has a 4 tuning parameters, N, $\vec{a}$, $\vec{b}_1$ and $\vec{b}_{1,2}$. According to (Trelea, 2002) the swarm size, N should be equal to the dimension of the problem, namely, N = 6. Moreover, $\vec{a}$, $\vec{b}_1$ and $\vec{b}_2$ can be set to scalars, a, b_1 and b_2.

6.4.2 SGO

Surrogate model optimization is a global optimization method used when the real model is too complex making simulations and experiments too time-consuming. In the case of DP-tuning the surrogate model is an approximation of the objective function, to approximate the relationship between control parameters and performance (e.g., IAE). It uses function evaluations of the objective function (simulations or real-world tests) to to approximate the objective function through regression or interpolation. The function approximation in Matlab is done with Radial Basis Function (RBF) interpolation introduced by (Powell, 1990). *surrogateopt* uses a cubic RBF interpolation with a linear tail. According to (Holmström, 2008) it can be expressed as:

$$s_n(x) = \sum_{i}^{n} \lambda_i \phi(||x - x_i||_2) + b^\top x + a \tag{6.13}$$

Where

- $s_n(x)$ is the surrogate model which tries to approximate the performance function, f(x). Where f might be the integral of absolute error.
- x are the control parameters, $x = [\zeta_1\ \zeta_2\ \zeta_3\ \omega_{b1}\ \omega_{b2}\ \omega_{b3}]^\top$
- $\phi(r) = r^3$ Is the cubic RBF where the radial distance r = $||x - x_i||$
- λ_i are the weigths weighing the radial bias functions
- $b^\top x + a$ is a linear function which approximates linear relations

The unknown parameters $\lambda,\ b$ and a. Are found from the linear equations:

$$\begin{pmatrix} \Phi & P \\ P^\top & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ c \end{pmatrix} = \begin{pmatrix} F \\ 0 \end{pmatrix} \tag{6.14}$$

Where Φ is the nxn matrix with $\Phi_{ij} = \phi(||x_i - x_j||_2)$ and

$$P = \begin{pmatrix} x_1^\top & 1 \\ x_2^\top & 1 \\ \vdots & \vdots \\ x_n^\top & 1 \end{pmatrix}, \lambda = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{pmatrix}, c = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_d \\ a \end{pmatrix}, F = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{pmatrix} \tag{6.15}$$

When the parameters are found using Equation 6.14, one has a model of the surrogate function. The more points (higher n), the more precise the surrogate model theoretically becomes.

Simply, the algorithm works as following:

1. Initialize points $n = d + 1$ points within the bounds X = $\{x_1,\ x_2, , , x_{d+1}\}$ where d is the dimension of the optimization. Then evaluate the objective function f(x) in these points and update the surrogate parameters. The best point is called the incumbent point.

2. Evaluate the surrogate, s(x), at many points around the incumbent point and select the best point based on $g(x) = wS(x) + (1 - w)D(x)$. The function weighs both the normalized surrogate function $S(x)$ and the normalized distance function, $D(x)$. If the weight w is high, the algorithm converges quickly and if it is low, the algorithm focus more on the global search.

3. Evaluate the objective function, f(x), at the x that gave the lowest g(x). If this function value is sufficiently lower than the incumbent value, x becomes the new incumbent point. Anyway, the surrogate is updated using this value.

4. Repeat step 2.-3. are repeated until satisfactory convergence is achieved.

6.4.3 Nelder-Mead Simplex Method

According to (Lagarias et al., 1998), the Nelder-Mead Simplex algorithm has become one of the most widely used methods of unconstrained optimization since it's publication in 1965. The method is a local search method that attempts to minimize the nonlinear objective function, f(x), without any information about the gradient. For a problem of dimension d, the algorithms creates a simplex of n+1 points. The figure below visualizes the how the algorithm works in 2 dimensions:

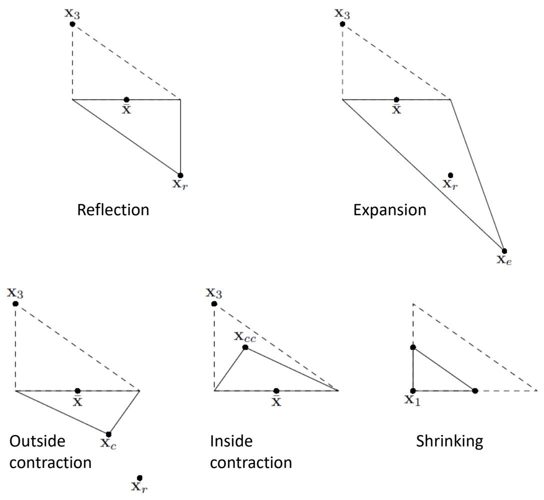


Figure 6.5: Nelder-Mead Simplex algorithm

As Figure 6.5 illustrates, the algorithm creates a simplex with 3 points. The dashed triangle illustrates a simplex where X_3 is the worst point. The algorithm is simply explained in the following steps detailing what happens in Figure 6.5:

1. Initialize points $n = d + 1$ points around x_0 by adding 5 $\%$ to each component $\{x_1, x_2, \ldots, x_n\} = \{x_0, x_0 + 0.05x_{0,1}, \ldots, x_0 + 0.05x_{0,d}\}$ where d is the dimension of the optimization. Then evaluate the objective function f(x) in these points.

2. Sort points from lowest ($f(x_1)$) to highest ($f(x_n)$) corresponding function values. Generate the reflected point, r, and evaluate, f(r). If f(r) is between the best and second worst function value, replace the worst value (x_n) with the reflected point, r, and go to step 2.

3. If the reflected value, f(r), is better, compute the expansion point, s. If f(s) is better than f(r) replace x_n with s and if not replace it by r. Then return to step 2.

4. If the reflected point is better than the worst point, but worse than the second worst point, an outside contraction is performed to calculate the point c. If f(c) is better than the reflected point, x_n is replaced with c. If that is not the case, the simplex is shrunk to half size as shown in figure 6.5. Then all the new points need to be evaluated. Then the algorithm goes to step 2.

5. If the reflected point is worse than the worst point, an inside contraction is

performed to calculate the point cc. If f(cc) is better than the worst point, x_n is replaced with cc. If that is not the case, the simplex is shrunk to half size as shown in figure 6.5 and all the new points need to be evaluated. Then the algorithm goes to step 2.

6. This goes on until satisfactory convergence is achieved.

6.4.4 Comparison of DFO algorithms

The quality of the DFO algorithms are measured by their:

1. Convergence rate
2. Convergence
3. Generality
4. Global vs. Local optima
5. Tuning parameters
6. Complexity of implementation
7. Bounds
8. Ability to use historic data

To evaluate points, 1.-3., testing is necessary, while for points 4. through 8., arguments are made based on the descriptions of the algorithms in Section 6.4.1, 6.4.2 and 6.4.3.

Three tests were made, including two tests with known optimas as well as well as the autotuning task for the simulation model. All 3 tests are done for 6-dimensional problems.

Tests with known global optimum The Rosenbrock function and Rastrigin function are difficult functions to optimize. Rosenbrock because of it's valley-like shape and Rastrigin for its multi-modal (multiple local optima) shape.

$$f_{rb}(x) = \sum_{i=1}^{n} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right] \tag{6.16}$$

$$f_{rs}(x) = 10n + \sum_{i=0}^{n}(x_i - 10\cos(2\pi x_i)) \tag{6.17}$$

Where $x_i \in [-5, 10]$ and the global minimum $f(x) = 0$ is in $x_i = 1$ for Rosenbrock and $x_i = 0$ for Rastrigin. The results from 10 optimization tests is given in Table 6.6.

Table 6.6: Best results from 10 random optimizations

Method	$fminsearch$	$particleswarm^*$	$surrogateopt$
Rosenbrock: F_{100}	4400	0.18	46
Rosenbrock: F_{1000}	0.078	0.034	5.3
Rastrigin: F_{100}	92	2.2	18
Rastrigin: F_{1000}	75	0	8

where F_{100} is the lowest value after 100 function evaluations and is used as a measure for convergence rate while F_{1000} is the lowest value after 1000 function evaluations and is used as a measure for convergence.

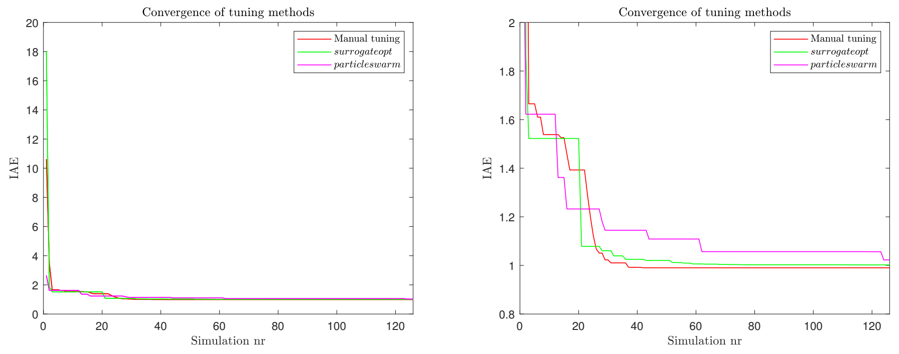
From these results, it appears that the *particleswarm* is superior, however that is not necessarily the case. For the Rosenbrock *particleswarm* had the best convergence rate for swarm size N$\geq$ 6, but *fminsearch* generally had the best convergence during the 10 tests. For the Rastrigin function, *particleswarm* and *surrogateopt* performed similarily for swarm size, N = 6, while *particleswarm* outperformed a lot for swarm size, N = 60. *fminsearch* generally performed very poorly.

Autotuning on Simulation model The autotuning was done using approximately 120 test runs per algorithm (with 5 different start points for *fminsearch*), and it was compared with a manual tuning by trial and error as a baseline. The resulting control parameters are seen in Table 6.7

Table 6.7: Results from autoutuning with different DFOs

Method	ζ_{surge}	ζ_{sway}	ζ_{yaw}	$\omega_{b,surge}$	$\omega_{b,sway}$	$\omega_{b,yaw}$	IAE
Manual	1.10	0.7	0.7	0.50	0.34	0.37	0.99
$fminsearch^*$	0.879	0.243	0.829	0.700	0.718	0.344	0.86
$particleswarm$	1.046	0.700	0.869	0.490	0.444	0.266	1.02
$surrogateopt$	1.056	0.700	1.093	0.500	0.361	0.176	1.00

These results are illustrative of the differences between the different methods. $fminsearch$ works in a way that is quite similar to manual manual tuning. That is, it starts at some start point and iterativly tries to find better and better control gains. However, it does not take bounds into consideration, and for 5 different starting points it converged to 5 vastly different end points, all outside the defined bounds. The best control parameters found by $fminsearch$ are displayed in Table 6.7. Because its inability of using constraints, it is inconvenient for use in a practical case where there should be bounds. Also, it converges to very different solutions, making it unpredictable in terms of general performance. It is therefore not considered further in the thesis. The convergence of the other methods are shown in Figure 6.6.



(a) Convergence of IAE **(b)** Zoomed in

Figure 6.6: Convergence of IAE as a Function of simulation number

It should be noted that other comparisons were made between these three methods and that it varies which of the methods achieve the best convergence, and that the difference is of little significance. The important point is that all the methods converge quite good within 30 simulations. Also, considering the size of the optimization domain (e.g., $0.024 \leq K_{p,1,1} \leq 7.14$), the resulting control gains below are very similar:

$$
\begin{array}{cccc}
 & \overbrace{\hphantom{\begin{bmatrix}2.035 & 0.464 & 0.316\end{bmatrix}}}^{K_p} & \overbrace{\hphantom{\begin{bmatrix}0.534 & 0.085 & 0.039\end{bmatrix}}}^{K_i} & \overbrace{\hphantom{\begin{bmatrix}2.470 & 1.011 & -0.026\end{bmatrix}}}^{K_d} \\
\text{PSO:} & \begin{bmatrix}2.035 & 0 & 0\\ 0 & 0.464 & 0\\ 0 & 0 & 0.316\end{bmatrix} & \begin{bmatrix}0.534 & 0 & 0\\ 0 & 0.085 & 0\\ 0 & 0 & 0.039\end{bmatrix} & \begin{bmatrix}2.470 & 0 & 0\\ 0 & 1.011 & -0.026\\ 0 & 0.003 & 0.710\end{bmatrix} \\
\text{SGO:} & \begin{bmatrix}2.191 & 0 & 0\\ 0 & 0.307 & 0\\ 0 & 0 & 0.306\end{bmatrix} & \begin{bmatrix}0.591 & 0 & 0\\ 0 & 0.046 & 0\\ 0 & 0 & 0.030\end{bmatrix} & \begin{bmatrix}2.572 & 0 & 0\\ 0 & 0.814 & -0.026\\ 0 & 0.003 & 0.767\end{bmatrix} \\
\text{Man:} & \begin{bmatrix}2.525 & 0 & 0\\ 0 & 0.272 & 0\\ 0 & 0 & 0.322\end{bmatrix} & \begin{bmatrix}0.701 & 0 & 0\\ 0 & 0.038 & 0\\ 0 & 0 & 0.049\end{bmatrix} & \begin{bmatrix}2.804 & 0 & 0\\ 0 & 0.764 & -0.026\\ 0 & 0.003 & 0.659\end{bmatrix}
\end{array}
$$

Selection matrix To summarize the algorithms, consider Table 6.8

Table 6.8: Selection matrix for chosing optimization method

	$particleswarm$	$surrogateopt$
Convergence rate	Good/very good	Good
Convergence	Good/very good	Good
Generality	Good	Good
Global vs. Local	Global/Local	Global
Complexity of Implementation	Low	Medium
Complexity of parametrization	Medium	Low
Bounds	Yes	Yes
Ability to use historic data	Suitable	Very suitable

- Convergence rate: For the test functions, PSO had a better performance, while they performed similarily for tuning of the simulation model.
- Convergence: For the test functions, PSO had a better performance, while they performed similarily for tuning of the simulation model.
- Generality: Both PSO and SGO converge to very similar results as the manual tuning, when considering the size of the domain. They also generally yield a consistent results.
- Global vs local: They are both global optimization methods. SGO can be proved to find global optimum, while PSO is an effective heuristic that can focus on convergence or global search, depending on the parameters.
- Complexity of Implementation: The PSO as presented in Section 6.4.1 and is considered to be simpler than SGO to implement as an online autotuning algoritm.
- Both algorithms respect bounds which is important for the practical setup.
- Ability to use historic data. The more historic data available, the better, the surrogate is able to approximate the objective function. particle swarm can easily also use historic data, simply by initializing the swarm using the historic data. SGO is perfect for this type of initialization.

Based on these criterions, PSO was selected as the algorithm to use in MC-Lab for the autotuning.

6.5 Implementation of Autotuning component

In the lab setup, the algorithm needs to work in real-time and was programmed in manually. The DFO is only run once after each test is finished.

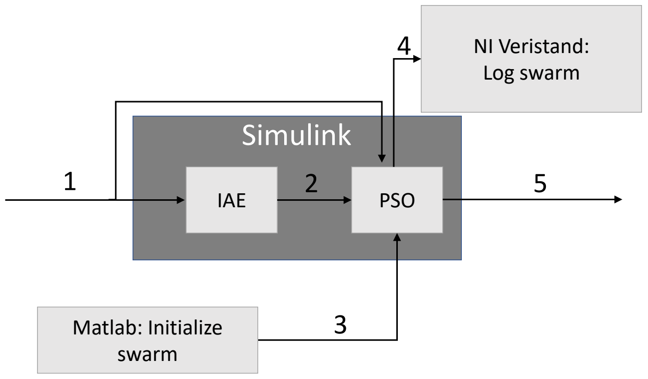


Figure 6.7

Figure 6.7 shows the signal flow of the PSO where 1 is the error used for calculation of IAE and boolean variables signaling which phase of the test the CSAD is in. 2 is the IAE which is sent to the PSO when a test is over for optimization. 3 is the initial swarm which can be based on historic data, swarm from previous tuning or a random initiation. 4 is the logging of optimization parameters so that it is possible to resume optimization by inserting these as the initial swarm. 5 is the control parameters.

A more detailed flow is described below

1. When Simulink starts running, initialize $[x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6]$ within the lower and upper boundary either randomly or using historic data as input. This is referred to as signal 3 in Figure 6.7. Initialize the best local positions to the initial positions and the global best position to [0,0,0,0,0,0]$^\top$. Initialize the global best fitness to infinity and the local best fitnesses to infinity. Also initialize the velocities of the particles randomely as $\{v_1, v_2, \dots, v_6\} \sim U(-|ub - lb|, |ub - lb|)$ where ub and lb are the upper and lower bounds of the optimization. During the first test, the vessel uses a known untuned controller with $[\zeta_1\ \zeta_2\ \zeta_3\ \omega_{b1}\ \omega_{b2}\ \omega_{b2}]^\top = [1,\ 1,\ 1,\ 0.1,\ 0.1,\ 0.1]^\top$

2. After a test. When a test is over and the IAE is finished integrating the error (signal 1) it is sent to PSO (signal 2) which updates the local and global

best function values and positions. Then it updates the position and velocity as well as handling bounds as the complete algorithm in Appendix A.1. Note further that the self-adjustment weight, $b_1 = 0$, the social adjustment weight, $b_2 = 1.7$ and the inertia $a = 0.6$ are used for updating positions and velocities. The test number parameter is increased by one, and the next particle in the swarm will be used for the next test (signal 5)

3. Before/during a test: When the vessel has settled sufficiently using the safe controller, the next set of control parameters are sent to the controller and IAE starts to integrate. The system continously logs all the swarm data (signal 4) needed to resume the autotuning, should something fail.

Chapter 7

Model Scale Testing Results

In this chapter the results of the autotuning in the MC-lab is presented. At first, the results of the autotuning in calm water is presented. Then the results of the autotuning in waves is presented. A video demonstrating the conceptual autotuning has also been made to show the autotuning.

7.1 Video illustrating concept

A video for demostrating the autotuning was made and can either be found through the link:
`https://www.youtube.com/watch?v=m8iEJI-xaQ8`
or the QR-code:

Figure 7.1: The QR-code for the autotuning demonstration

Note that the video was made before successful autotuning in waves was done.

7.2 Results from autotuning in calm water

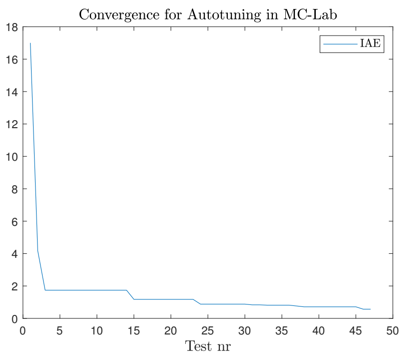
By using PSO as DFO method and the IAE as a PI, it was attempted to optimize the control gains for the PID in the DP tracking controller through testing in the MC-Lab in calm water. It resulted in the parameters in Table 7.1 and gains in Equation 7.1

Table 7.1: Results

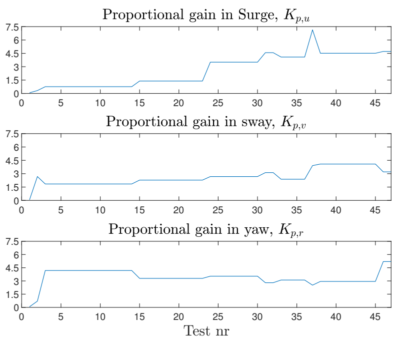
#Tests	ζ_{surge}	ζ_{sway}	ζ_{yaw}	$\omega_{b,surge}$	$\omega_{b,sway}$	$\omega_{b,yaw}$	IAE
47	1.321	1.345	1.394	0.500	0.400	0.480	0.568

$$\overbrace{\begin{bmatrix} 4.71 & 0 & 0 \\ 0 & 3.20 & 0 \\ 0 & 0 & 5.19 \end{bmatrix}}^{K_p} \overbrace{\begin{bmatrix} 1.47 & 0 & 0 \\ 0 & 0.81 & 0 \\ 0 & 0 & 1.60 \end{bmatrix}}^{K_i} \overbrace{\begin{bmatrix} 4.10 & 0 & 0 \\ 0 & 3.39 & -0.026 \\ 0 & 0.003 & 4.21 \end{bmatrix}}^{K_d} \tag{7.1}$$

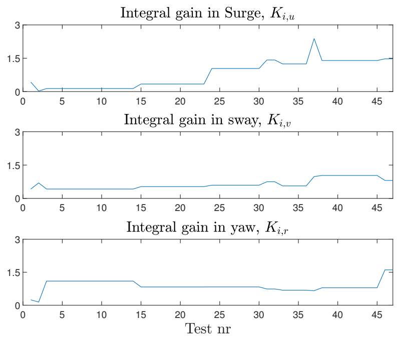
The convergence of the IAE and the developement of the best known proportional, integral and derivative gains is are illustrated in Figure 7.2



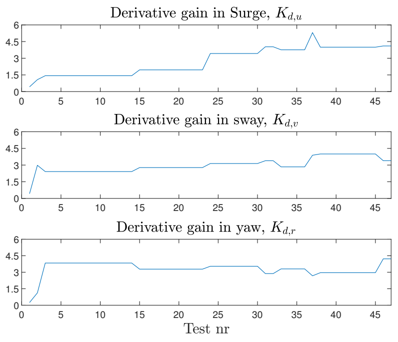
(a) Convergence of IAE



(b) Developement of proportional terms



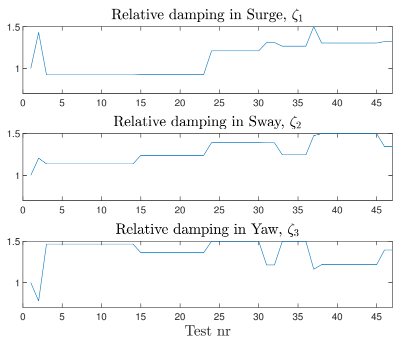
(c) Developement of integral terms



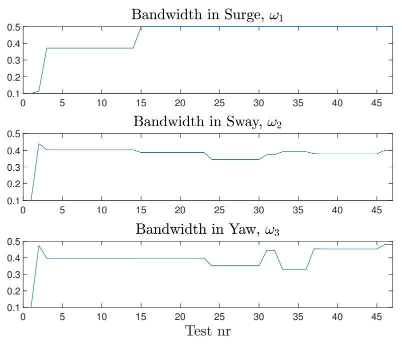
(d) Developement of derivative terms

Figure 7.2: (a) is the convergence of IAE (b)-(d) development of the best known control gains as a function of test number.

The convergence of IAE in Figure 7.2 (a) is satisfactory. The algorithm gradually finds better gains and converges to even lower IAE than in simulation. For the tuning plots in Figures 7.2 (b), 7.2 (c), and 7.2 (d), it is important to note that that the gains in Figure 7.2 are parametrized so that $K_{p,max}/K_{p,min} \approx 300$ and $K_{i,max}/K_{i,min} \approx 2400$ and $30 \leq K_{d,max}/K_{d,min} \leq 300$. The convergence of the control parameters are shown in Figure 7.3.



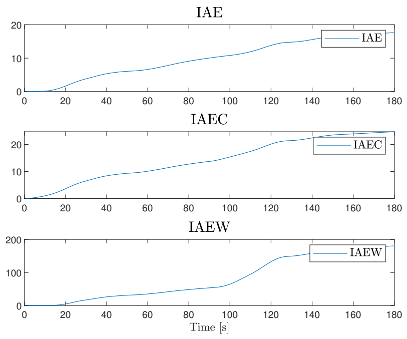
(a) Relative dampings



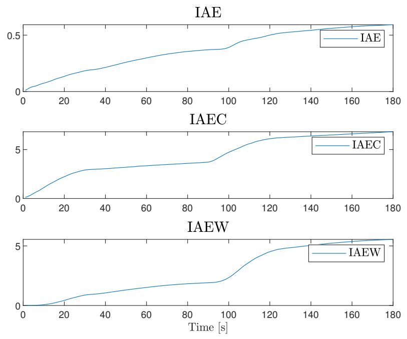
(b) Bandwidths

Figure 7.3: Development of best relative damping parameters (ζ) and bandwidths (ω_b) as a function of test number

The performance of the best control parameters were compared with the performance of a baseline controller with control bandwidth $\omega_b = 0.1$ and relative damping $\zeta = 1$ for all DOFs. To illustrate the difference in the performance, consider Figure 7.4.



(a) Before tuning

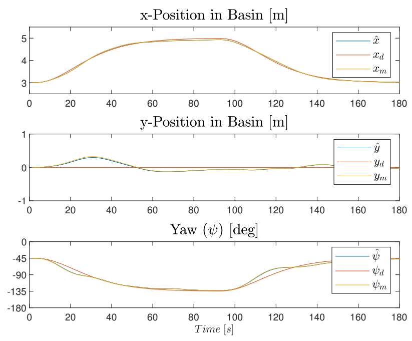


(b) After tuning

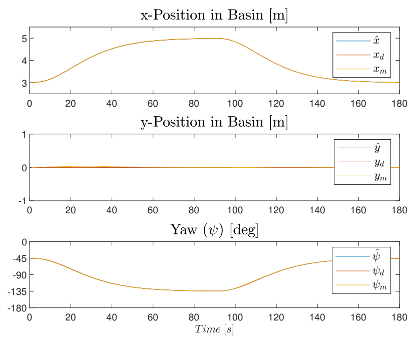
Figure 7.4: PIs before and after tuning in waves. The figures give the performance functions IAE, IAEC and IAEW during a run. Note the difference in the scales for (a) and (b)

In Figure 7.4, it is observed that all performance indices are much better after tuning than before. Moreover, the performance indices are much lower than the ones obtained in the simulation study in Chapter 6.

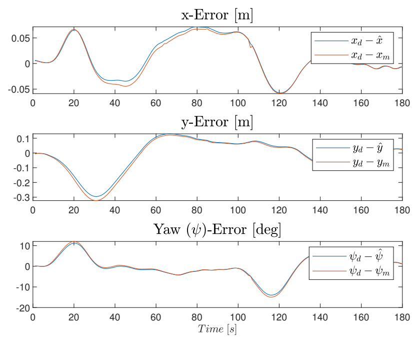
The performance of the tracking control can also be evaluated from the time series of the pose and errors. The reference tracking and errors are given in Figure 7.5.



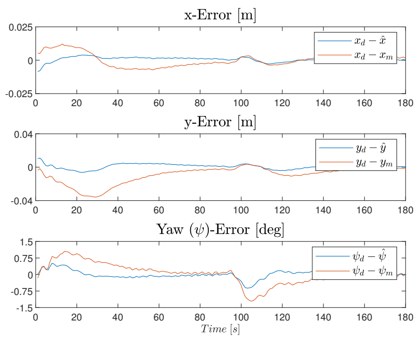
(a) Before tuning



(b) After tuning



(c) Before tuning



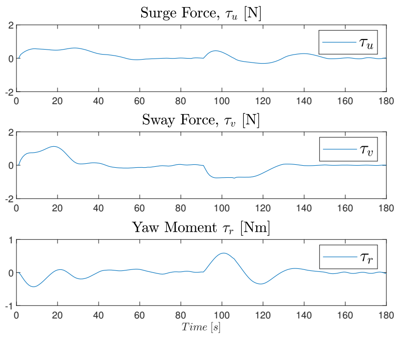
(d) After tuning

Figure 7.5: (a) and (b) Show the estimated, desired and measured positions and headings, While (c) and (d) shows the estimated and actual errors.

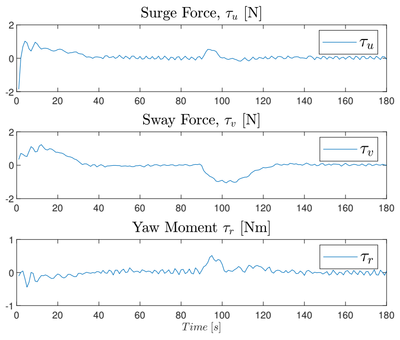
In Figure 7.5 (a) and 7.5 (b) It is clearly seen that the untuned controller struggles with following the trajectory, especially in sway and yaw as seen close up in Figure 7.5 (c). The actual errors are included in 7.5 (c) and 7.5 (d) to illustrate the inaccuracies in the position estimates from the observer. For evaluating the performance of the tracking controller, the estimated tracking error $(\eta_d - \hat{\eta})$ is used. And Figure 7.5 (d) shows that this error has a max value of around 1 cm in surge and sway and 0.5° deg in yaw.

It is observed in 7.5 (d) that the error in surge and sway start with some initial error. This is because the trajectory starts the same place (η_{d1}) independent of where the vessel is at the time of test initiation. This is no major problem since the error is so low (must be lower than 2 cm/2 deg). To fix this, the reference trajectory is initialized to the estimated position, thereby giving an initial tracking error that is zero. This was used for the wave test.

The control signals are important for evaluating the wear and tear of the control system. The control signals of the tuned and untuned system is given in Figure 7.6 (a) and Figure 7.6 (b).



(a) Before tuning

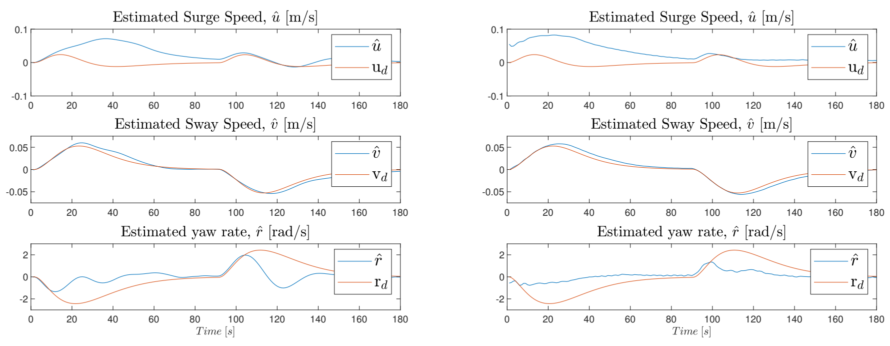


(b) After tuning

Figure 7.6: Control input, τ

The untuned controller has slowly varying control signals, while the tuned controller in Figure 7.6 (b) has more oscillation. However, these oscillations are small in magnitude and in frequency and none of the forces or moments are close to saturation.

The next plots to evaluate are the velocity plots in Figure 7.7.



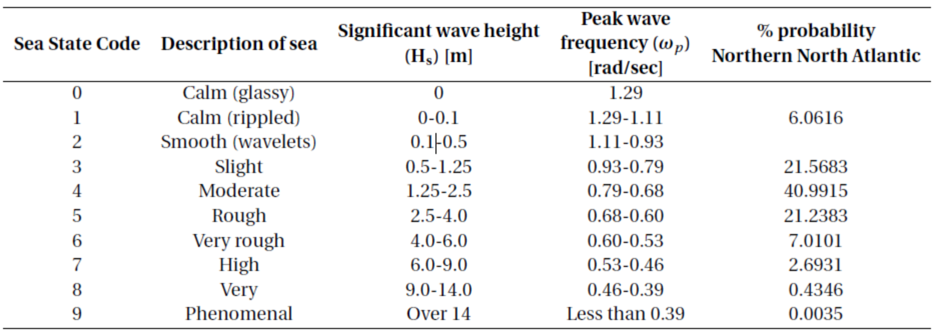
(a) Before Tuning **(b)** After Tuning

Figure 7.7: Estimated and Actual Velocities. Note that the yaw rate is supposed to be in deg/s and not rad/s

Figure 7.7 show that the estimates for the surge speed and yaw rate are poor, while the sway estimate is accurate.

7.3 Results for Rough/Moderate Sea State

Recall the wave-making machine in MC-Lab as described in Section 2.5. In this setup it creates waves according to the the JONSWAP spectrum. To chose wave condition, it was chosen to use the definitions of sea states from (Price, 1974). They are shown in Figure 7.8.



Sea State Code	Description of sea	Significant wave height (H_s) [m]	Peak wave frequency (ω_p) [rad/sec]	% probability Northern North Atlantic
0	Calm (glassy)	0	1.29	
1	Calm (rippled)	0-0.1	1.29-1.11	6.0616
2	Smooth (wavelets)	0.1-0.5	1.11-0.93	
3	Slight	0.5-1.25	0.93-0.79	21.5683
4	Moderate	1.25-2.5	0.79-0.68	40.9915
5	Rough	2.5-4.0	0.68-0.60	21.2383
6	Very rough	4.0-6.0	0.60-0.53	7.0101
7	High	6.0-9.0	0.53-0.46	2.6931
8	Very	9.0-14.0	0.46-0.39	0.4346
9	Phenomenal	Over 14	Less than 0.39	0.0035

Figure 7.8: Table showing the definitions of sea states [Courtesy: (Price, 1974)]

The sea-state that was utilized was the Moderate/Rough sea-state with $H_s = 2.5\ m$

and $\omega_p = 0.68\ rad/sm$. However, these are given in full-scale. By Froude scaling with a scale parameter, $\lambda = 90$, it becomes $H_s = 0.0278\ m$ and $\omega_p = 6.45\ rad/s$.

After testing the autotuning concept with waves it was found to be time consuming to stabilise with a 45 ° angle to the waves. Also, the positioning system functioned poorly near $\eta_2 = [5\ m, 0\ m, -135°\ deg]^\top$ Therefore, the wave test was adapted so that the vessel is directed towards the waves during the stabilization and the maneuver happens closer to the cameras. The modified test is illustrated in Figure 7.9.

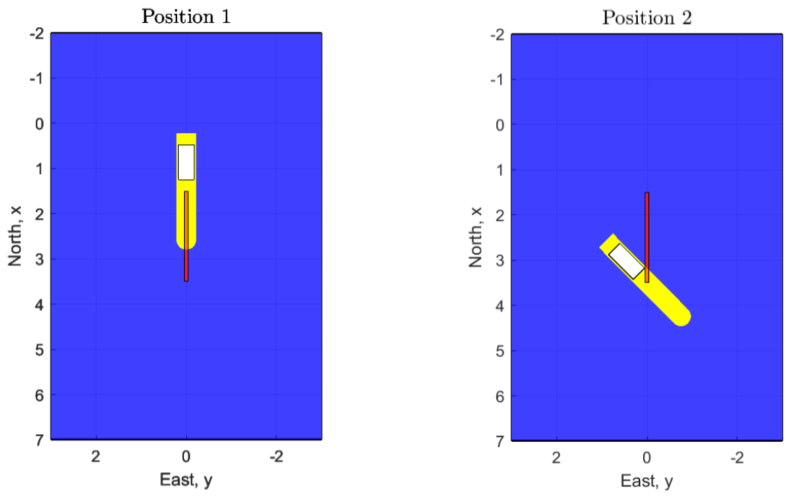


Figure 7.9: The 2-setpoint autotuning with waves, $\eta_{d1} = [1.5, 0, 0°]$ and $\eta_{d2} = [3.5, 0, -45°]$. The waves propagate in the negative x-direction. The red line indicates the line that the vessel tracks

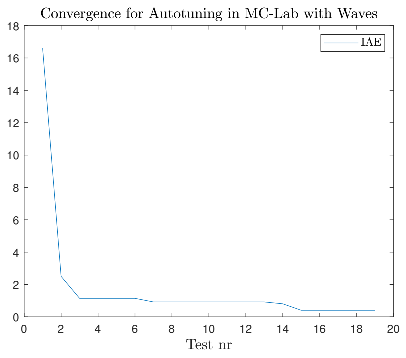
The resulting control parameters are given in Table 7.2 and gains Equation 7.2.

Table 7.2: Results of tuning in waves

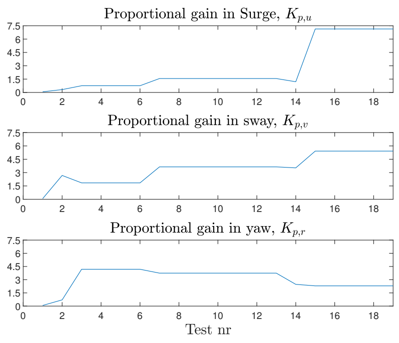
#Tests	ζ_{surge}	ζ_{sway}	ζ_{yaw}	$\omega_{b,surge}$	$\omega_{b,sway}$	$\omega_{b,yaw}$	IAE
19	1.50	1.38	1.07	0.50	0.50	0.50	0.41

$$\overbrace{\begin{bmatrix}7.14 & 0 & 0\\ 0 & 5.42 & 0\\ 0 & 0 & 2.30\end{bmatrix}}^{K_p}\overbrace{\begin{bmatrix}2.39 & 0 & 0\\ 0 & 1.73 & 0\\ 0 & 0 & 0.63\end{bmatrix}}^{K_i}\overbrace{\begin{bmatrix}5.31 & 0 & 0\\ 0 & 4.47 & -0.026\\ 0 & 0.003 & 2.47\end{bmatrix}}^{K_d} \tag{7.2}$$

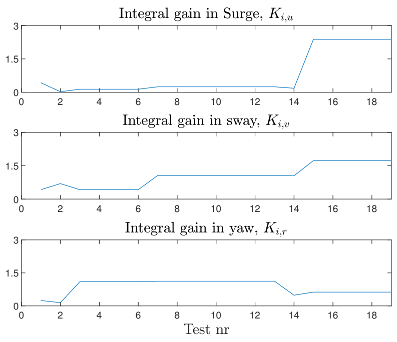
The convergence of IAE and the proportional, integral and derivative gains are showwn in Figure 7.10.



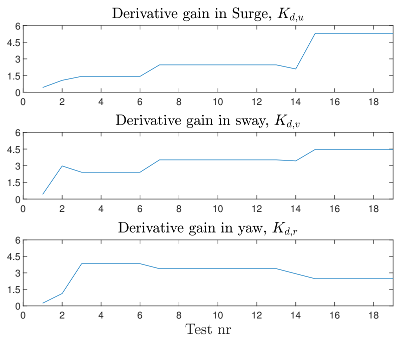
(a) Convergence of IAE



(b) Developement of proportional terms



(c) Developement of integral terms

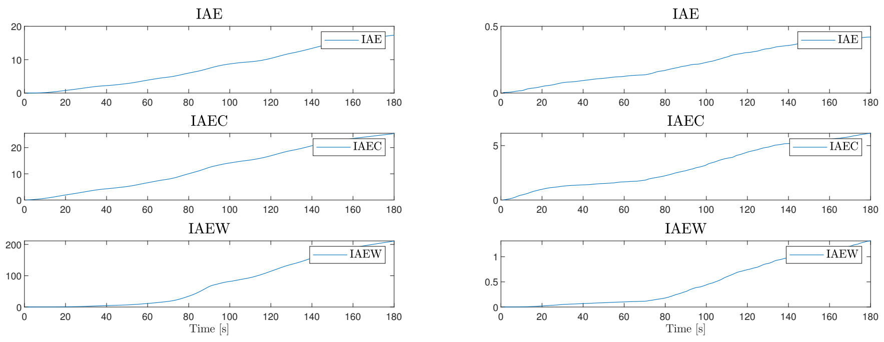


(d) Developement of derivative terms

Figure 7.10: Convergence of IAE and the developement of the best control gains as a function of test number.

A reduced number of tests were used for the autotuning in waves, more precisely 19. This was caused by a slow stabilization phase in waves, because the stabilizing controller was slow. It is believed that this could have been completely avoided by using the best controller from calm water. Secondly, at this point, the algorithm had found a good solution that gave a low IAE. However, the resulting tracking controller gave a great tracking performance.

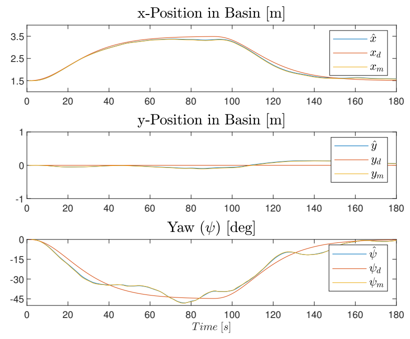
Ther performance indices achieved are given in Figure 7.11



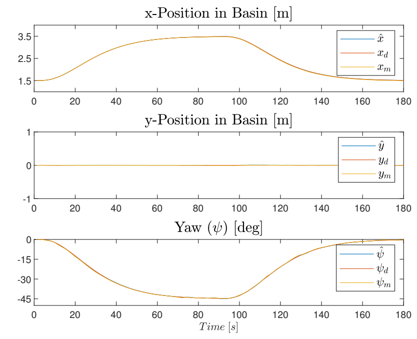
(a) Before Tuning **(b)** After Tuning

Figure 7.11: Performance indices before and after tuning in waves

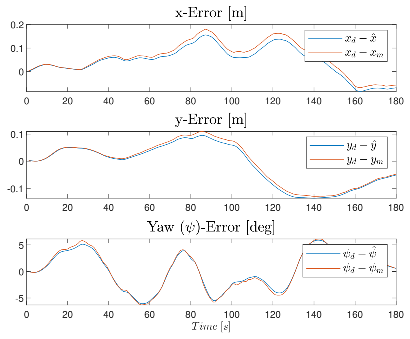
The results in Figure 7.11 (a) and Figure 7.11 (b) illustrate an enormous difference in performance between the poorly tuned control parameters and the tuned. The main cause is that the error of the tracking is so high for the untuned controller as illustrated in Figure 7.12.



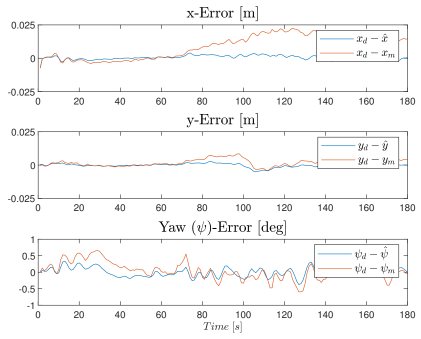
(a) Before Tuning (note scale)



(b) After Tuning



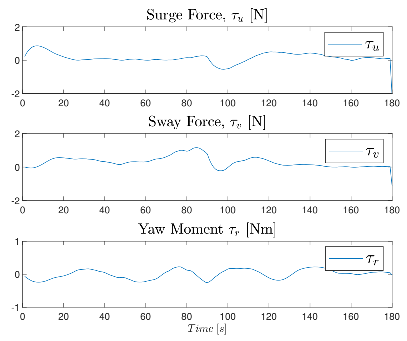
(c) Before Tuning



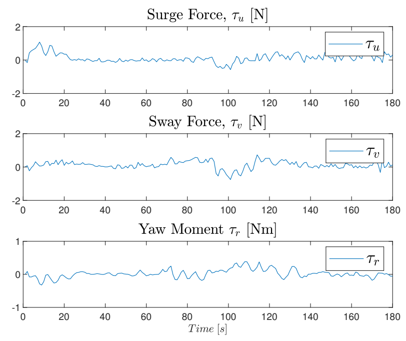
(d) After Tuning

Figure 7.12: (a) and (b) Show the estimated, desired and measured positions and headings, while (c) and (d) shows the estimated and actual errors.

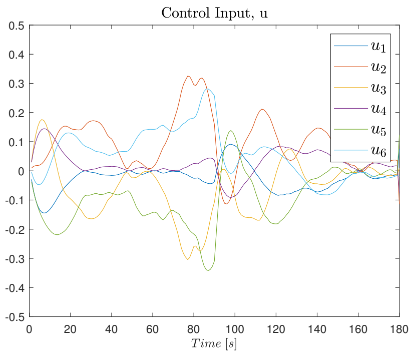
Recall that the experiment in calm waters had initial errors which is not the case in Figure 7.12 (d). The maximal tracking errors are around 0.5 cm/0.5 deg in 7.12 (d) (blue lines). Compared to the untuned error in 7.12 (c) which is up towards 15 cm in surge and sway and 5 deg in yaw, this is obviously much better. The corresponding forces, τ and low-level control input, u are given in Figure 7.13



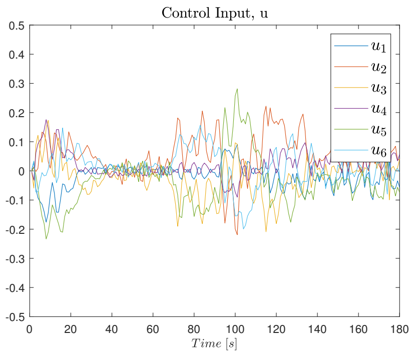
(a) Before Tuning



(b) After Tuning



(c) Before Tuning

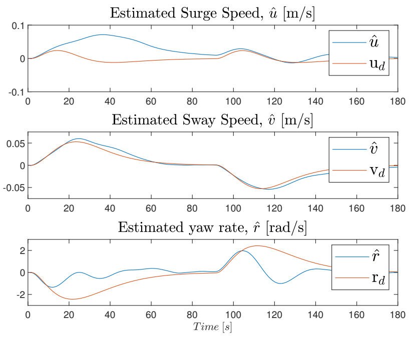


(d) After Tuning

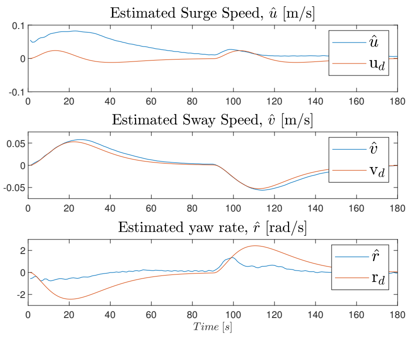
Figure 7.13: (a) and (b) Illustrates the commanded generalized forces from the Controller, and (c) and (d) are the low-level control signals, u, sent to each thruster in Volt

As expected, the tuned control effort in Figure 7.13 (b) is more high-frequent. This is because the optimization only focuses on tracking accuracy. The resulting thruster input u is given in Figure 7.13 (c) and 7.13 (d). Note that u$\in [-0.5, 0.5]$, so neither the untuned or tuned controller ever gets saturation of the thrusters. Moreover, note that the thruster input, in Figure 7.13 (d) is quite low from 40 s to 70 s before it increases. From 70 to 90 s, the vessel basically stands still. However Figure 7.13 (d) illustrates that the thruster inputs rises during this period. However, if zoomed in, it can be seen that signal $u_1 \approx u_4$ and $u_2 \approx u_3$ and $u_5 \approx u_6$. The reader is referred to Section 3.2.2 for a detailed description of the thruster configuration.

Lastly, the velocity plots are given in the Figure 7.14 to illustrate the performance in terms of tracking the velocities.



(a) Before Tuning



(b) After Tuning

Figure 7.14: Estimated and desired Velocities. Note that the yaw rate should be in deg/s

The observer is able to estimate the sway speed quite well in Figure 7.14. The surge speed and yaw rate estimates are more problematic, however.

7.4 Summary of results

To briefly summarize:

- The performance metric, IAE, converges gradually for autotuning in both calm water and waves.
- The resulting control gains for calm water perform well at the trajectory tracking control objective with maximal tracking errors of 1 cm/ 0.5° deg which had been lower had it not been for the initial error.
- The resulting control gains for waves perform well at the trajectory tracking control objective with maximal tracking errors of 0.5 cm/ 0.5° deg. The problem with the initial error was fixed for this test.
- The control gains for tuning in calm water converged nicely considering the size of the search space. The control gains in waves converges less, probably is because of the fewer test iterations.
- It is believed that the optimization can get stuck at bounds. This is improved by setting the self adjustment weight, b_1 to a non-zero value.
- The control input, τ is high frequent, but is never near actuator saturation for the tuned control parameters.

- The velocity estimation is good for sway speed, but poor for surge and yaw. This is likely caused by a non-optimal observer tuning and the transient nature of the maneuver which makes bias estimation hard. Moreover, the observer estimates the pose wrongly for all 3 DOFs with a couple of cm.
- The increased stabilization time in waves, make the tuning in more time-consuming and less efficient.

Chapter 8

Conclusions and Further work

8.1 Conclusions

The potential of autonomous DP controller tuning has been successfully tested on the 1:90 model of a DP vessel, the CSAD. It has involved establishing an improved motion control system, including an improved guidance system, an enhanced and tuned nonlinear PID controller with feed-forward and a predictable (not optimal) thrust allocation. When the motion control system was established, a methodology for autotuning with DFO was created for the established simulation model. Different autotuning methods were tested on the model, before concluding on using PSO for the objective of minimizing the IAE. Further, this methodology was adapted to the experimental setup in MC-lab and implemented as a modular autotuning block in Simulink. The autotuning was then tested for calm water and a moderate/rough sea state.

DFO proved to be able to iteratively tune the control parameters of the nonlinear PID controller to achieve a satisfactory tracking performance. The PI, IAE, converges gradually, both in calm water and waves. The tuned control parameters gave a tracking performance with maximal errors lower than 1 cm/ 0.5° deg for the 3 DOF tracking objective in both calm water and waves. The control input, τ, was somewhat oscillatory but was never near actuator saturation. Hence, autotuning seemed to be feasible in both calm conditions and waves.

The autotuning of a PID in a DP system was found to be affected by the other parts of the system, in particular, the observer. The nonlinear passive observer gave poor estimates for surge and yaw, which gives unnecessary derivative control, that could affect the tuning. Moreover, it is known from the simulation study that the tuning is effected by choice of PSO parameters. The test focused on fast convergence by setting the self-adjustment weight (b_1) to zero. The last major issue that was found was that waves increase the stabilizing time, decreasing the efficiency of the tuning.

To summarize, the implemented online DFO autotuner in a loop with the established motion control system is able to tune the PID controller without human intervention iteratively. The tuned system is also able to do accurate trajectory tracking. Thus, the system can perform the main objective of the thesis.

8.2 Further work

This thesis focused on implementing a functioning autotuning loop in the laboratory and did the necessary study of the DFO methods, PIs and dynamic positioning to create a functioning autotuning setup. I hope that I have opened some doors and my thesis can help others who research within similar fields.

A couple of suggestions for further work are

- Improvements can be made on the motion control system of the CSAD.
    1. Improve the observer: This could be done by including more states, for example using the Inertial Measurement Unit (IMU). Another interesting project would be to do online observer tuning of the CSAD.
    2. Improve the thrust allocation: The current thrust allocation on the CSAD is predictable, but very inefficient in terms of energy usage.
- Further work on autonomous DP-tuning: A better PI should be established, one that weighs both accuracy and wear and tear. Furthermore, the potential of surrogate model optimization should be studied further.
- Applying autotuning to less intuitive controllers (e.g., backstepping controller). For a PID controller, the manual operator has an understanding of the tuning variables that the DFO lacks. However, given a non-intuitive controller it would be interesting to see the performance of the DFO.

Bibliography

Audet, C., 2016. Blackbox and derivative-free optimization: theory, algorithms and applications. Springer Science+Business Media New York.

Bjørnø, J., 2016. Thruster-assisted position mooring of c/s inocean cat i drillship. Norwegian University of Science and Technology, Trondheim, Norway.

Bjørnø, J., Skjetne, R., Frederich, A. R., 2017. "modeling, parameter identification and thruster-assisted position mooring of c/s inocean cat i drill- ship. Proceedings of the 36th ASME International Conference on Ocean, Offshore and Arctic Engineering.

Eriksen, B.-O. H., Breivik, M., 2017. Modeling, identification and control of high-speed asvs: Theory and experiments.
URL `http://hdl.handle.net/11250/2483676`

Fossen, T., 2011. Handbook of Marine Craft Hydrodynamics and Motion Control.

Fossen, T. I., 2003. Fuel-efficient rudder and propeller control allocation for marine craft: Experiments with a model ship. IEEE Transactions on Control Systems Technology.

Frederich, P., 2016. Constrained optimal thrust allocation for c/s inocean cat i drillship. Norwegian University of Science and Technology, Trondheim, Norway.

Holmström, K., 2008. An adaptive radial basis algorithm (ARBF) for expensive black-box global optimization.

Kaliappan, V., Thathan, M., 2014. Design of optimum pid controller for nonlinear process using evolutionary algorithms. Journal of Theoretical and Applied Information Technology 69 (3), 522–529.

Kennedy, J., 1995. Particle swarm optimization.

Lagarias, J. C., Reeds, J. A., Wright, M. H., Wright, P. E., 1998. Convergence properties of the nelder–mead simplex method in low dimensions. SIAM Journal on Optimization 9 (1), 112–147.

Lyngstadaas, O. N., 2018. Ship motion control concepts considering actuator constraints. NTNU.

Matlab, 2015. Global optimization toolbox solver characteristics. Retrieved 30. April, from https://se.mathworks.com/help/gads/improving-optimization-by-choosing-another-solver.html.

NTNU(2015b), 2015. Marine cybernetics laboratory (mc-lab). Retrieved 10th of February 2019, from http://www.ntnu.edu/imt/lab/cybernetics.

Nørgaard Sørensen, M. E., Lyngstadaas, O. N., Eriksen, B.-O. H., Breivik, M., 2018. A dynamic window-based controller for dynamic positioning satisfying actuator magnitude constraints. IFAC PapersOnLine 51 (29), 140–146.

Powell, M. J. D., 1990. The Theory of Radial Basis Function Approximation in 1990. Clarendon Press.

Price, W. G., 1974. Probabilistic theory of ship dynamics.

Sahib, M. A., 2015. A new multiobjective performance criterion used in pid tuning optimization algorithms. Cairo University, Journal of Advanced Research.

Sørensen, A., 2019. Marine Cybernetics, lecture notes.

Sørensen, M. E. N., Breivik, M., 2015. Comparing nonlinear adaptive motion controllers for marine surface vessels. IFAC PapersOnLine 48 (16), 291–298.

Trelea, I. C., 2002. The particle swarm optimization algorithm: convergence analysis and parameter selection.

Værnø, S. A., Skjetne, R., Kjerstad, K., Calabrò, V., 2019. Comparison of control design models and observers for dynamic positioning of surface vessels. Control Engineering Practice 85, 235–245.

Appendix

Appendix A

A.1 Psuedo-code for PSO

Algorithm 1: Pseudo-code for Particle Swarm Optimization

```
Input  : lowerB, upperB, N, a, $b_1$, $b_2$
Output: $x_g$
Initiate $n = d$ random points to be evaluated between lowerB and upperB and
  initialize f:
X = $\{x_1, x_2, \ldots, x_n\} \sim U(lowerB, upperB)$
F = $\{inf, inf, \ldots, inf)\}$
Initialize velocities uniformly:
V = $\{v_1, v_2, \ldots, v_n\} \sim U(-|upperB - lowerB|, |upperB - lowerB|)$
Set initial local best and global best:
$F_l$ = F, $X_l$ = X
$f_g$ = $min\ f(x_i)$, $x_g = \min_{x_i}\ f(x_i)$
iterationNr = 1
while terminationCriteria = false do
    for $i \leftarrow 1$ to $n$ do
        Evaluate objective function, $f(x_i)$
        if $f(x_i)$ < $F_l(i)$ then
            $F_l(i) = f(x_i)$
            $X_l(i) = x_i$
        end
        if $f(x_i)$ < $f_g$ then
            $f_g = f(x_i)$
            $x_g = x_i$
        end
        Create two random vectors , $r_1,\ r_2 \sim U(0, 1)$
        Update velocity vector and position:
        $v_i = av_i + b_1 r_1 \cdot (x_i - X_l(i)) + b_2 r_2 \cdot (x_i - x_g)$
        $x_i = x_i + v_i$
        Consider boundaries for all dimensions:
        for $j \leftarrow 1$ to $d$ do
            if $x_i(j)$ > upperB(j) then
                $x_i(j)$ = upperB(j)
                $v_i(j)$ = 0
            else if $x_i(j)$ < lowerB(j) then
                $x_i(j)$ = lowerB(j)
                $v_i(j)$ = 0
            end
        end
    end
end
```

A.2 Psuedo-code for SGO

Algorithm 2: Pseudo-code for Surrogate Optimization

```
Input  : lowerB, UpperB, w
Output: xBest
Initiate $n \geq d + 1$ random points to be evaluated between lowerB and upperB:
X = $\{x_1, x_2, \ldots, x_n\}$, F = $\{f(x_1), f(x_2), \ldots, f(x_n)\}$
$xIncumbent = \min_{x_i} \; f(x_i)$
while terminationCriteria = false do
    Create 100-1000 sample around xIncumbent, within bounds, $x_j$
    Calculate the min and max value of the surrogate among the sample points:
    $s(x)$ from Eq. 6.13
    $s_{min} = min \; s(x_j)$
    $s_{max} = max \; s(x_j)$
    Calculate scaled surrogate for sample points
    $S(x) = (s(x) - s_{min})/(s_{max} - s_{min})$
    Calculate the minimal and maximal distance between sampled and evaluated
      points:
    $d_{ij} = ||x_i - x_j||,$
    $d(x) = min \; ||x_i - x||,$
    $d_{min} = min \; d_{ij},$
    $d_{max} = max \; d_{ij}$
    Calculate the scaled distance:
    $D(x) = (d_{max} - d(x))/(d_{max} - d_{min})$
    Evaluate the merit function at every sample point:
    $g(x) = wS(x) + (1 - w)D(x)$
    Select the sample that minimizes the merit function:
    $xAdaptive = \min_{x_j} \; g(x_j)$
    Evaluate the objective function using the adaptive point and compare to
      objective function in the incumbent point:
    if $f(xAdaptive)$ is sufficiently lower than $f(xIncumbent)$ then
        xIncumbent = xAdaptive
    end
    if n < $n_{max}$ then
        n = n + 1
        Include $x_{n+1}$ in X
        Include f($x_{n+1}$) in F
        Calculate updated surrogate parameters using Equation 6.14
    end
end
$xBest = \min_{x_i} \; f(x_i)$
```

A.3 Psuedo-code for fminsearch

Algorithm 3: Pseudo-code for Nelder-Mead simplex method

```
Input  : $x_0$
Output: xBest
Initiate $n = d + 1$ points by adding 5% of each component of $x_0$ and evaluate:
X = $\{x_1, x_2, \ldots, x_n\} = \{x_0, x_0 + 0.05x_{0,1}, \ldots, x_0 + 0.05x_{0,d}\}$
F = $\{f(x_1), f(x_2), \ldots, f(x_n)\}$
while terminationCriteria = false do
    Sort all points of X from lowest to highest
    f($x_n$) = max  $f(x_i)$
    Generate the reflected point:
    $m = \sum_{i=1}^{n-1} x_i/(n-1)$ $r = 2m - x_n$
    Evaluate the objective function in the reflected point, f(r)
    if $f(x_1) \leq f(r) < f(x_{n-1})$ then
        Replace $x_n$ with r and terminate this iteration
    else if $f(r) < f(x_1)$ then
        Calculate the expansion point:
        s = m - 2(m-$x_n$)
        Evaluate f(s)
        if $f(s) < f(r)$ then
            Replace $x_n$ with s and terminate this iteration
        else
            Replace $x_n$ with r and terminate this iteration
        end
    else
        if $f(r) < f(x_{n+1})$ then
            Calculate an "outside contraction", c = m + (r-m)/2
            if $f(c) < f(r)$ then
                Replace $x_n$ with c and terminate this iteration
            else
                Shrink: $x_i = x_1 + (x_i - x_1)/2$ and evaluate f for $i = 2, 3, ...n$
            end
        else
            Calculate an "inside contraction", cc = m + ($x_n$-m)/2
            if $f(cc) < f(x_n)$ then
                Replace $x_n$ with cc and terminate this iteration
            else
                Shrink: $x_i = x_1 + (x_i - x_1)/2$ and evaluate f for $i = 2, 3, ...n$
            end
        end
    end
    $xBest = x_n$
end
```

Appendix B

B.1 Content in attached ZIP-file

The attached zip-file contains

- All the data from the MC-Lab autotuning in calm water and with waves, stored in two separate MAT-files. For both these files there is a 3 D array, called "$run\_specific\_data$" that contains 180 by N by 15. This array can be used to look at data for any of the tests within the autotuning. The configuration of this array can be understood from the function "$calc\_run\_specific\_data.m$".
- "$ctrl\_custom.slx$" and initiation file. Using this block as the custom control for CSAD, it will automatically start autotuning. The corresponding Veristand files are also included.
- A simulation folder that starts autotuning if the main function is run. The main function also has some examples of how the different performance indicator functions are called and how the optimizations are run