Michael Hoyer

System Identification and Machine Learning

Master's thesis in Engineering and ICT
Supervisor: Martin Ludvigsen
Co-supervisor: Svein Ivar Sagatun, Øystein Barth Utbjoe
June 2021

Master's thesis

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

NTNU
Norwegian University of
Science and Technology

equinor

Michael Hoyer

System Identification and Machine Learning

Master's thesis in Engineering and ICT
Supervisor: Martin Ludvigsen
Co-supervisor: Svein Ivar Sagatun, Øystein Barth Utbjoe
June 2021

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

Project description

Problem Description and Background

Equinor is planning full-scale testing of crawler robots in its operation in 2021. The robots will perform inspection and light maintenance autonomously. The goal is to limit the exposure of workers to harmful environments and to enable NNM (Not Normally Manned) and UN (UnManned) operational models for new fields.

Autonomous systems rely on simulations for much of their verification process. Models describing robots and its surrounding (i.e. the installation) are at the core of these simulations. Establishing good models for a robot can be challenging, as it may require extensive domain knowledge full scale testing. Methods that can aid in this process is therefore of significant value for robot operators.

System Identification (SI) is in many ways similar to regression analysis and machine learning (ML). Recent years have brought many advances in software tooling and hardware that aid in machine learning applications. Due to the similarities between machine learning and system identification, it is likely that these advances also will benefit the latter.

Objective

This thesis aims to test parameter estimation with SI for mechanical systems, focusing on UUVs that have known model-structures. It shall answer if methods and tools from ML can be used to improve the performance of SI.

SI shall be done with simulations and real measurements of an UUV moving in a basin. All results will be discussed with regards to accuracy, computational performance, and in-accuracies in the model structure. Literature studies will be conducted on the following topics a) SI and parameter estimation applied to mechanical systems with a focus on UUV and b) the use of ML for SI – included is a study of relevant ML software and hardware for SI.

Scope of Work

1. The candidate shall perform a literature study on system identification (SI) and parameter estimation applied to mechanical systems with a focus on UUV
2. The candidate shall perform a literature study on the use of machine learning (ML) for SI and in particular applied to mechanical systems – included is a study of relevant ML software tooling for SI.
3. Methods and analysis
   (a) The candidate shall implement a mathematical model of the UUV for testing and simulation of the SI methods described below.
   (b) The candidate shall develop and test (based on experimental and simulated data) at least one selected ML based SI method for parameter estimation
   (c) The candidate shall estimate parameters of an UUV experimentally, using experimentally acquired data.
   (d) Results from (b), (c) and (d) shall be evaluated and compared using selected metrics.
4. Experimental work
   (a) The experiment shall be conducted at the NTNU facility MC-lab.
   (b) The experiment shall consist of the following: the vehicle moves unconstrained in the basin and receives input from its thrusters. The vehicles degrees of freedom shall be exited both in isolation and in combination, such that cross-coupling terms become active. Its pose shall be measured with an Qualisys camera measurement system. Angular velocity and linear acceleration shall be measured through an onboard IMU.
   (c) Analysis and discussion of the suitability of the resulting data from the experiment

Format and Formalities

The report shall be written in English and edited as a research report including

- A project plan defining the objective and scope of work
- Background theory on
    - Mathematical models of interest
    - Parameter estimation methods
    - Relevant ML software tooling
    - Selected SI algorithms
- Description of method including
    - Detailed plan and documentation of the experiment
    - Data processing and analysis
    - Implementation of algorithms
- Results
- Discussion
- Conclusion including a proposal for further work.

Source code and datasets from the experiment shall be provided on GitHub. It is supposed that the Department of Marine Technology, NTNU, and Equinor can use the results freely in its research work, unless otherwise agreed upon, by referring to the student's work. The thesis should be submitted within June 10.

Supervisors: Martin Ludvigsen (NTNU) Svein Ivar Sagatun (Equinor) Øystein Barth Utbjoe (Equinor)

Summary

Simulation is becoming more important for the development of robots due to cost savings and its key role in validation of autonomous systems. This sparks the need for cheap and accurate models of the robots that are to be simulated. Many classes of moving robots have well established equations that model their movements accurately, but the system specific parameters they depend on are often based on educated guesses or expensive experiments. Unmanned underwater vehicles (UUVs) is one of those classes. Their importance in sub-sea operations only keep growing, as breakthroughs in price and usability make them suitable to more and more applications. The research of this thesis is motivated by the need for better methods to estimate parameters of moving robots models, focusing on UUVs as example platform.

In this thesis techniques from system identification and machine learning are used to estimate UUV model parameters based on simulations and measurements of the Beluga UUV. The measurements are gathered using internal and external sensors of high quality, covering pose and velocity in all six degrees of freedom. Background theory on UUV modelling and system identification is presented, as well as a technical overview of Beluga UUV. The chosen method is able to identify the UUV parameters from simulations in reduced dimensionality cases.

Sammendrag

Simulering har blitt viktigere for utviklingen av roboter på grunn av kostnadsbesparelser og dens avgjørende rolle i valideringen av autonome systemer. Dette øker behovet for billige og presise modeller av robotene som skal simuleres. Mange typer roboter har godt etablerte ligninger som beskriver bevegelsene deres, men de systemavhengige parameterne som disse ligningene er avhengige av velges ofte ved gjetting eller dyre eksperimenter. Ubemannede undervannsroboter (UUV) er en av robottypene dette gjelder for. Disse blir stadig vekk viktigere for undervassoperasjoner, siden gjennombrudd i pris og brukervennlighet har gjort dem aktuelle for flere og flere oppgaver. Denne masteroppgaven er motivert av behovet for bedre metoder til estimering av parametere til bevegelige roboters modeller, og fokuserer på UUVer som testplattform.

I denne mastergraden brukes metoder fra systemidentifikasjon og maskinlæring til å estimere UUV modellparametere basert på simuleringer og målinger av UUVen Beluga. Målingene er samlet ved bruk av interne og eksterne sensorer av høy kvalitet, som sammen dekker alle tilstander i Belugas seks frihetsgrader. Bakgrunnsteori om matematisk modellering av UUVer og systemidentifikasjon presenteres, samt en teknisk gjennomgang av Beluga. Den valgte systemidentifikasjonsmetoden er i stand til å identifisere UUV parametere fra simuleringer i tilfeller med redusert antall frihetsgrader.

Preface

This Master's thesis is the final part of my M.Sc. in Engineering and ICT, with a specialization in Marine Cybernetics. It is written at the Department of Marine Technology at the Norwegian University of Science and Technology during the spring of 2021.

I want to thank my supervisors Martin Ludvigsen, Svein Ivar Sagatun and Øystein Utbjoe for their guidance, insights and helpful feedback along the way. This thesis would not have been possible without them. I also want to thank my friends from Vortex for their companionship and comradely support in getting Beluga ready for diving.

Contents

Summary **iv**

Sammendrag **v**

1 Introduction **2**

1.1 Motivation 2

1.2 Related Work 3

1.3 Contributions 4

1.4 Thesis Structure 4

2 Modeling of Unmanned Underwater Vehicles **5**

2.1 Kinematics 5

2.1.1 Reference Frames 6

2.1.2 Notation 6

2.1.3 Quaternions 7

2.1.4 Kinematic Equation 7

2.2 Kinetics 8

2.2.1 Hydrostatics 8

2.2.2 Rigid-body Kinetics 9

2.2.3 Hydrodynamics 10

2.3 Equations of Motion 12

2.3.1 Full Model 12

2.3.2 Diagonal Model 13

2.3.3 Linear Diagonal Model 14

2.3.4 Non-linear Surge Model 14

2.3.5 Linear Surge Model 15

3 System Identification **16**

3.1 Machine Learning and Regression 16

3.2 Data Recording and Preprocessing 17

3.3 Model Structure 18

3.4 Overfit and underfit 18

3.5 Parameter Estimation 19

3.5.1 Minimizing Prediction Errors 20

3.5.2 Evaluation Metrics 20

3.5.3 Optimization Methods 21

3.5.4 Multi-Objective Optimization 21

3.5.5 Population methods 22

3.5.6 Running Performance Metric 23

3.5.7 NSGA-II 24

3.6 Model Validation 26

3.7 Other Parameter Estimation Methods 27

4 Beluga UUV **28**

4.1 Overview 28

4.2 Propulsion System 29

4.3 Sensors 31

4.4 Hardware issues 31

5 The Experiment **33**

5.1 The Experiments Objective in a SI Context 33

5.2 Experiment Facilities 34

5.2.1 Marine Cybernetics Laboratory 34

5.3 Experiment Design 35

5.3.1 Measurements 35

5.3.2 Movement patterns 36

5.4 Measurement Evaluation and Preprocessing 38

5.4.1 Error Sources and Measurement Noise 38

5.4.2 Measurement Synchronization 38

5.4.3 Quaternions 39

5.4.4 Measurement Quality Checks 39

6 System Identification of UUV Models **40**

6.1 Model Predictions 40

6.2 Multi-Objective Optimization 41

6.2.1 Objective Function 41

6.2.2 Optimization with NSGA-II 41

6.2.3 Decision Making 42

6.3 Identification using Simulated Measurements 42

6.4 Software Tooling and Implementation 43

7 Results **45**

7.1 Beluga Time Series 45

7.2 Measurement Quality Checks 53

7.3 Identification on Surge Models using Simulated Measurements 58

7.4 Identification on Surge Models using Real Measurements 64

7.5 Visualization of the Objective Function 67

7.6 Computation Time Analysis 69

8 Discussion **70**

8.1 Beluga Measurements 70

8.2 SI Using Simulated Measurements 71

8.3 SI Using Real Measurements 72

8.4 Computational Performance 73

9 Conclusion and Further Work **75**

9.1 Conclusion 75

9.2 Proposal for Further Work 75

A Statistical Describtions **77**

References **89**

List of Tables

2.1 Vectors describing motion of marine vehicles in 6DOF 6
2.2 Number of parameters required in the full model 13
2.3 Number of parameters required in the diagonal model 13
2.4 Number of parameters required in the low-velocity linear diagonal model 14
2.5 Number of parameters required in the non-linear surge model 14
2.6 Number of parameters required in the linear surge model 15

5.1 Measured and estimated states 36
5.2 Predefined tests 37

6.1 Used parameters for NSGA-II. m is the number of parameters in $\boldsymbol{\theta}$ . . 42

7.1 Metadata about recordings 46
7.2 Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in surge-1 50
7.3 Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in sway-1 51
7.4 Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in heave-1 52
7.5 Chosen and estimated parameters for different models and $\boldsymbol{\tau}$ 58
7.6 Estimated parameters from Beluga measurements using linear and non-linear surge models. The true parameters are unknown. 64

A.1 Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in yaw-1 78

A.2 Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in heave_surge-1 79

A.3 Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in surge_sway-1 80

A.4 Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in random-1 81

A.5 Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in random-2 82

A.6 Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in random-3 83

A.7 Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in random-4 84

A.8 Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in random-5 85

A.9 Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in random-6 86

A.10 Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in random-7 87

A.11 Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in random-8 88

List of Figures

3.1 Overfit and underfit. Source: towardsdatascience.com 19

3.2 NSGA-II procedure 24

3.3 Illustration of rolling forecasting origin split. Blue marks the training set, and red the validation set. The iteration number is marked by the y-axis and the samples are along the x-axis. 26

4.1 Beluga 29

4.2 Illustration of Belugas propulsion system 29

4.3 T200 thrust ranges at different voltages. The direction of thrust changes at 1500 PWM. 30

4.4 Illustration of Beluga propulsion software stack 31

4.5 Beluga as it was during the experiment 32

5.1 Picture of MC-lab 34

5.2 Beluga during recording of random movements 36

5.3 Sketch of mission FSM 37

6.1 Different $\boldsymbol{\tau}$ used for simulation 43

7.1 `https://github.com/michoy/system_identification` 45

7.2 Overview of movements in surge test 47

7.3 Overview of movements in sway test 48

7.4 Overview of movements in heave tests 49

7.5 Integration check for surge position in surge test 54

7.6 Derivation check for surge velocity in surge test 54

7.7 Integration check for yaw angle in surge test 55

7.8 Integration check for sway position in sway test 55

7.9 Derivation check for sway velocity in sway test 56

7.10 Integration check for yaw angle in sway test 56

7.11 Integration check for heave position in heave tests 57

7.12 Derivation check for heave velocity in heave tests 57

7.13 Comparison between simulated $\mathbf{y}$ and estimated $\hat{\mathbf{y}}$ using the linear surge model and $\boldsymbol{\tau} = X_{triple\_ramp}$ 59

7.14 Comparison between simulated $\mathbf{y}$ and estimated $\hat{\mathbf{y}}$ using the linear surge model and $\boldsymbol{\tau} = X_{many\_sine}$ 60

7.15 Comparison from between simulated $\mathbf{y}$ and estimated $\hat{\mathbf{y}}$ using the non-linear surge model and $\boldsymbol{\tau} = X_{triple\_ramp}$ 61

7.16 Comparison from between simulated $\mathbf{y}$ and estimated $\hat{\mathbf{y}}$ using the non-linear surge model and $\boldsymbol{\tau} = X_{many\_sine}$, run 1 62

7.17 Comparison from between simulated $\mathbf{y}$ and estimated $\hat{\mathbf{y}}$ using the non-linear surge model and $\boldsymbol{\tau} = X_{many\_sine}$, run 2 63

7.18 Fit between measured position and velocity in surge together with model predictions with linear surge model 65

7.19 Fit between measured position and velocity in surge together with model predictions with non-linear surge model 66

7.20 Topology of $\mathbf{f}_{EoM}$ around $\boldsymbol{\theta}^*$ for different $\boldsymbol{\tau}$. $\boldsymbol{\theta}^*$ is marked with a black star 68

7.21 Computation time of n calls of `predict` 69

Nomenclature

ASE	Analytical and Semi-Empirical method
AUV	Autonomous Underwater Vehicle
CB	Center of Buoyancy
CG	Center of Gravity
CO	Center of Origin
DOF	Degree(s) Of Freedom
DP	Dynamic Positioning
DVL	Doppler Velocity Los
EKF	Extended Kalman Filter
ESC	Electronic Speed Controller
FSM	Finite State Machine
IMU	Inertial Measurement Unit
MAE	Mean Absolute Error
MCU	Micro Controller Unit
ML	Machine Learning
MSE	Mean Squared Error
NED	North-East-Down coordinate frame
NSGA	Non-dominated Sorting Genetic Algorithm
PWM	Pulse Width Modulation
ROS	Robot Operating System

ROV	Remotely Operated Vehicle
SBC	Single Board Computer
SDK	Software Development Kit
SI	System Identification
SLAM	Simultaneous Localization And Mapping
SMAPE	Symmetric Mean Absolute Percentage Error
UUV	Unmanned Underwater Vehicle

Chapter 1

Introduction

1.1 Motivation

Realistic simulations of robots have become increasingly important in the robot development process. Factors driving this trend are 1) the ability to do concept verification without (often expensive) physical hardware and 2) the increased options for solving problems, such as answering "what-if" questions and doing verification of autonomous systems (Žlajpah (2008)). Simulated copies of real robots are referred to as digital twins. The usefulness of a digital twin is directly dependent on how well it achieves the application depended requirements to realism. Therefore, digital twins need good models of the robots they describe. So far in this project, no robot suppliers that provide satisfactory simulation models have been identified. This is a challenge for digital twins. While work is currently being done by DNV-GL (2020) attempting to solve the issue, it is still in its early days, and its success is yet to be determined. However, there are existing methods that estimate these models directly today.

System identification is the process of estimating models of a system based on input and output. It is not a new field (Psichogios and Ungar (1992)), but it has seen an increased interest in recent years (Fan et al. (2019)). The field is closely related to *regression analysis*, which is an important part of machine learning. Many of the methods that are today referred to as the classical methods of machine learning have been used in system identification for several decades. Their overall similarity becomes apparent when considering that both fields share a common goal: creating good models of systems based on data collected from them. Machine learning has enjoyed much attention this decade, and there is a rich ecosystem of software and literature available on the subject. The field of system identification does not share the same benefits. However, since the two fields are so closely related, it should be possible to apply recent machine learning advancements to system identification.

The robot-inspired models from Fossen (2011) are well-established for describing unmanned underwater vehicles (UUVs) and ships. Both simulation and control of seagoing vehicles is usually done using them, since the models are intuitive and easy to implement. They do however depend on many hydrodynamic parameters that are specific to each vehicle, and these have proven very difficult to come by - especially for UUVs. Analytical derivation of them is almost impossible in realistic cases, and experimental methods are complicated and expensive (Conte et al. (2004)). The estimation error in using experimental methods is up to 50% for some parameters (Caccia et al. (2000)). Much of the estimation difficulties stem from the fact that UUVs operate in non-linear environments and often have highly coupled degrees of freedom (DOFs). System identification has shown potential as a method for estimating UUVs hydrodynamic parameters (Holven (2018)) and is investigated further in this thesis.

1.2 Related Work

Eidsvik and Schjølberg (2016) and Holven (2018) both looked into hydrodynamic parameter estimation for UUVs in their thesis. Eidsvik covers much ground on experimental work, while Holven also looks into system identification. Cardenas and de Barros (2019) combine an analytical and semi-empirical estimation method (ASE) with a parameter estimator based on the extended Kalman filter (EKF) and applies the method to a non-linear model of an autonomous underwater vehicle (AUV).

There is a richer literature available on parameter estimation for ship models. Fossen et al. (1996) use an off-line EKF utilizing two measurement series in parallel to estimate the parameters of a dynamic positioning (DP) ship model based on full-scale sea trials. Support vector machines, a method popularized by the advent of machine learning this decade, is used by Zhang and Zou (2013) to estimate hydrodynamic coefficients from captive model tests. Dai et al. (2019) uses evolutionary algorithms to estimate coupled hydrodynamic terms of ships heave and pitch motion.

The more general problem of parameter estimation for state-space models has been investigated by Kantas et al. (2015), who survey applications of particle methods for parameter estimation of non-linear non-Gaussian state-space models. Pillonetto et al. (2014) identify mathematical tools and concepts that have been developed in the machine learning community, that they want to make available to the to the control community. They focus kernel based regulation and its connections to Bayesian estimation of Gaussian processes.

1.3 Contributions

In this thesis system identification is applied to UUV models, using techniques from the machine learning community. The focus is on parameter estimation of UUVs mass and damping terms for movements in surge direction. Identification is tested on both simulated and real measurements. The main contributions of this thesis can be summarized as

- acquisition of high quality measurements of an UUVs position, orientation, linear velocity and angular velocity during various maneuvers
- applied system identification to parameter estimation for UUV models using the prediction-error minimizing approach and the multi-objective genetic algorithm NSGA-II for optimization.

1.4 Thesis Structure

Background theory on modeling of UUVs and system identification is presented in chapter 2 and 3. Chapter 4 gives a technical overview of Beluga UUV, with a focus on its thrust delivery system. Explanations on the how measurements of Beluga are gathered experimentally are given in chapter 5. The chosen system identification method is described in chapter 6. Results and their discussion are presented in chapter 7 and 8. Chapter 9 contains the conclusion and proposals for further work.

Chapter 2

Modeling of Unmanned Underwater Vehicles

Unmanned Underwater Vehicles (UUV) are robotic vehicles that can operate under water without a human occupant. The class can be divided into two subcategories: Autonomous Underwater Vehicles (AUV) and Remotely Operated Vehicles (ROV). AUVs carry out missions without human interference. They are mostly used for bathymetric surveys (Paull et al. (2014)), but companies are working on expanding their use cases to inspection of subsea installations and light intervention tasks (Liljeback and Mills (2017), Midthassel (2021)). ROVs carry out missions while being controlled by a human operator. They are usually connected to a surface ship, which follows them for the entire mission. ROVs are typically used for inspections and interventions. The line between AUV and ROVs is becomming blurred, as some AUVs are starting to support operations in semi-autonomous and ROV modes (Stinger (2021)).

A common approach to modelling UUVs is to split the task into kinematics and kinetics.

2.1 Kinematics

The study of *kinematics* treats the geometrical aspects of motion. It focuses solely on motions themselves, ignoring the source that caused them. The following subsections present an overview of kinematics relevant to UUVs based upon the work in Fossen (2011).

Parameter	**Combined**	**Linear**	**Angular**
NED position	$\boldsymbol{\eta} = [\boldsymbol{p}^n_{nb}, \boldsymbol{q}^n_b]^T$	$\boldsymbol{p}^n_{nb} = [x, y, z]^T$	$\boldsymbol{q}^n_b = [\eta, \epsilon 1, \epsilon 2, \epsilon 3]^T$
BODY velocity	$\boldsymbol{\nu} = [\mathbf{v}^b_{nb}, \boldsymbol{\omega}^b_{nb}]^T$	$\mathbf{v}^b_{nb} = [u, v, w]^T$	$\boldsymbol{\omega}^b_{nb} = [p, q, r]^T$
BODY force	$\boldsymbol{\tau} = [\boldsymbol{f}^b_b, \boldsymbol{m}^b_b]^T$	$\boldsymbol{f}^b_b = [X, Y, Z]^T$	$\boldsymbol{m}^b_b = [K, M, N]^T$

Table 2.1: Vectors describing motion of marine vehicles in 6DOF

2.1.1 Reference Frames

Reference frames are an important element to kinematics. These are coordinate systems used to describe motion relative to various geometrical points. Movement only makes sense when it is described relative to something else. There are two reference frames commonly used when modelling UUVs: NED and BODY.

The North-East-Down (NED) frame is a inertial reference frame that is defined tangential to earths surface. Its x-axis points towards north, the y-axis points towards east and the z-axis points towards earths center. The coordinate origin is usually defined by where the UUVs system is initiated. NED is used as a global reference frame for slow moving vessels that do not move very far along the frame, where the curvature of the earth would start to have an impact.

BODY frame is fixed to the body of the vehicle of interest. Its x-axis is defined along the bodies longitudinal axis, the y-axis is defined along the lateral axis and the z-axis is normal to the others, directed from bottom to top. The BODY frames motion is defined relative to the inertial reference frame. Its coordinate origin is defined by the Center of Origin (CO). BODY is used to describe a vehicles pose relative to the inertial frame.

2.1.2 Notation

Vectors describing a UUVs motion in SNAME notation are conveniently presented in table 2.1. Its contents can be explained as

- $\boldsymbol{p}^n_{nb}$ - position of the point CO in BODY with respect to NED expressed in NED
- $\boldsymbol{q}^n_b$ - orientation of BODY with respect to NED expressed in NED
- $\boldsymbol{v}^n_{nb}$ - linear velocity of BODY with respect to NED expressed in BODY
- $\boldsymbol{\omega}^n_{nb}$ angular velocity of BODY with respect to NED expressed in BODY
- $\boldsymbol{f}^b_b$ force with line of action through the point CO in BODY expressed in NED
- $\boldsymbol{m}^b_b$ moment about the point CO in BODY expressed in NED

2.1.3 Quaternions

There are two different representations used to describe a vehicles orientation: *Euler angles* and *Unit quaternions*. Euler angles are more intuitive, but unit quaternions are more robust and computationally faster. A common issue that arises when using Euler angles, is that rotation matrices contain representation singularities for certain angels. This is often referred to as *gimbal lock*. Unit quaternions avoid this issue by using a fourth parameter to describe orientation.

Quaternions are defined as

$$\mathbf{q} = [\eta, \epsilon_1, \epsilon_2, \epsilon_3]^T \tag{2.1}$$

where ϵ_1, ϵ_2 and ϵ_3 influence an axis of rotation and η influences the amount of rotation around that axis.

When talking about quaternions in the context of representing orientation, one usually refers to *unit* quaternions. These are quaternions which satisfy the constraint $\mathbf{q}^T\mathbf{q} = 1$, meaning their length must be equal to one. It is important to ensure that the unit constraint is satisfied during numerical integration involving quaternions. This can be done by performing simple normalization after each integration step.

2.1.4 Kinematic Equation

The kinematic equation of an UUV model describes the transformation between BODY and NED and can be expressed differently depending on whether Euler angles or unit quaternions are used to represent orientation. The kinematic equation using quaternions is expressed as

$$\dot{\boldsymbol{\eta}} = \mathbf{J}_q(\boldsymbol{\eta})(\boldsymbol{\nu}_r + \boldsymbol{\nu}_c) \tag{2.2}$$

where $\boldsymbol{\nu}_r$ is the vehicles velocity relative to ocean current, $\boldsymbol{\nu}_c$ is the ocean currents velocity and $\mathbf{J}_q(\boldsymbol{\eta})$ defines the transformation from BODY to NED given by

$$\mathbf{J}_q(\boldsymbol{\eta}) = \begin{bmatrix} \mathbf{R}_b^n(\mathbf{q}) & \mathbf{0}_{3x3} \\ \mathbf{0}_{4x3} & \mathbf{T}_q(\mathbf{q}) \end{bmatrix} \tag{2.3}$$

where $\mathbf{R}_b^n(\mathbf{q})$ is the linear velocity transform and $\mathbf{T}_q(\mathbf{q})$ is the angular velocity transform.

2.2 Kinetics

The study of kinetics treats forces and how they cause motions. For modeling UUVs one has to study their *hydrostatics*, *rigid-body kinetics* and *hydrodynamics*. The following is based upon the work of Fossen (2011) and Faltinsen (1993).

2.2.1 Hydrostatics

Hydrostatic forces in the context of UUVs are referred to as *restoring forces*. The name stems from the face that the forces tend to restore vehicles to their steady-state orientation. The UUVs restoring forces stem from buoyancy due to displaced water $\mathbf{f}_b^n$ and gravity working on the rigid-body mass $\mathbf{f}_g^n$. The forces can simply be expressed in NED as

$$\begin{aligned} \mathbf{f}_b^n &= -[0, 0, B]^T \\ \mathbf{f}_g^n &= [0, 0, W]^T \end{aligned} \tag{2.4}$$

Since we desire to express the equations of motion in BODY, we need to rotate the vectors in equation 2.4 from NED to BODY

$$\begin{aligned} \mathbf{f}_b^b &= -\mathbf{R}^T(\mathbf{q}_b^n)\mathbf{f}_b^n \\ \mathbf{f}_g^b &= \mathbf{R}^T(\mathbf{q}_b^n)\mathbf{f}_g^n \end{aligned} \tag{2.5}$$

There are two important geometrical points needed to describe the restoring forces: Center of Buoyancy (CB) and Center of Gravity (CG). CB is the point through which $\mathbf{f}_b^b$ acts and CG is the point through which $\mathbf{f}_g^b$ acts. The vectors defining these to points with respect to CO are $\mathbf{r}_i^b = [x_i, y_i, z_i]^T, i \in \{b, g\}$. The moments caused by buoyancy and gravity can then be expressed as

$$\begin{aligned} \mathbf{m}_b^b &= \mathbf{r}_b^b \times \mathbf{f}_b^b \\ \mathbf{m}_g^b &= \mathbf{r}_g^b \times \mathbf{f}_g^b \end{aligned} \tag{2.6}$$

Combining all of the above the restoring forces in BODY can be expressed as

$$g(\boldsymbol{\eta}) = -\begin{bmatrix} \mathbf{f}_g^b + \mathbf{f}_b^b \\ \mathbf{m}_b^b + \mathbf{m}_g^b \end{bmatrix} = -\begin{bmatrix} \mathbf{R}^T(\mathbf{q}_b^n)(\mathbf{f}_b^n + \mathbf{f}_g^n) \\ \mathbf{r}_b^b \times \mathbf{R}^T(\mathbf{q}_b^n)\mathbf{f}_b^n + \mathbf{r}_g^b \times \mathbf{R}^T(\mathbf{q}_b^n)\mathbf{f}_g^n \end{bmatrix} \tag{2.7}$$

2.2.2 Rigid-body Kinetics

A UUVs rigid-body kinetics are found by applying Newton's second law and Euler's first and second axioms.

The rigid-body kinetics can be expressed in vectorial form as

$$\mathbf{M}_{RB}\dot{\boldsymbol{\nu}} + \mathbf{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}_{RB} \tag{2.8}$$

where $\mathbf{M}_{RB}$ is the rigid-body system inertia matrix, $\mathbf{C}_{RB}$ is the coriolis-centripetal matrix and $\boldsymbol{\tau}_{RB} = [X, Y, Z, K, M, N]$ is a generalized vector of external forces and moments.

The rigid-body system inertia matrix $\mathbf{M}_{RB}$ has a unique parameterization given by

$$\begin{aligned}\mathbf{M}_{RB} &= \begin{bmatrix} m\mathbf{I}_{3x3} & -m\mathbf{S}(\mathbf{r}_g^b) \\ m\mathbf{S}(\mathbf{r}_g^b) & \mathbf{I}_b^b \end{bmatrix} \\ &= \begin{bmatrix} m & 0 & 0 & 0 & mz_g & -my_g \\ 0 & m & 0 & -mz_g & 0 & mx_g \\ 0 & 0 & m & my_g & -mx_g & 0 \\ 0 & -mz_g & my_g & I_x & -I_{xy} & -I_{xz} \\ mz_g & 0 & -mx_g & -I_{yx} & I_y & -I_{yz} \\ -my_g & mx_g & 0 & -I_{zx} & -I_{zy} & I_z \end{bmatrix}\end{aligned} \tag{2.9}$$

where $\mathbf{I}_{3x3}$ is the identity matrix, m is the vehicles mass, S is the cross-product operator and $\mathbf{I}_b^b$ is the inertia dyadic.

The coriolis and centripetal matrix $\mathbf{C}_{RB}$ does not have a unique parameterization. According to Theorem 3.2 in Fossen (2011) it can always be represented such that $\mathbf{C}_{RB}$ is skew-symmetric, which has advantages when designing non-linear controllers. One such parameterization is the *linear velocity-independent parameterization*, which is advantageous in environments where currents can be assumed to be irrotational.

$$\mathbf{C}_{RB}(\boldsymbol{\nu}) = \begin{bmatrix} m\mathbf{S}(\boldsymbol{\omega}_{nb}^b) & -m\mathbf{S}(\boldsymbol{\omega}_{nb}^b)\mathbf{S}(\mathbf{r}_g^b) \\ m\mathbf{S}(\mathbf{r}_g^b)\mathbf{S}(\boldsymbol{\omega}_{nb}^b) & -\mathbf{S}(\mathbf{I}_b^b\boldsymbol{\omega}_{nb}^b) \end{bmatrix} \tag{2.10}$$

2.2.3 Hydrodynamics

2.2.3.1 Hydrodynamic Inertia and Coriolis effect

The hydrodynamic system inertia matrix $\mathbf{M}_A$, also referred to as the *added mass* matrix, describes the effects caused by water that is forced to move by the UUV. It is expressed as

$$\mathbf{M}_A = -\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} = -\begin{bmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{w}} & X_{\dot{p}} & X_{\dot{q}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{w}} & Y_{\dot{p}} & Y_{\dot{q}} & Y_{\dot{r}} \\ Z_{\dot{u}} & Z_{\dot{v}} & Z_{\dot{w}} & Z_{\dot{p}} & Z_{\dot{q}} & Z_{\dot{r}} \\ K_{\dot{u}} & K_{\dot{v}} & K_{\dot{w}} & K_{\dot{p}} & K_{\dot{q}} & K_{\dot{r}} \\ M_{\dot{u}} & M_{\dot{v}} & M_{\dot{w}} & M_{\dot{p}} & M_{\dot{q}} & M_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{w}} & N_{\dot{p}} & N_{\dot{q}} & N_{\dot{r}} \end{bmatrix} \tag{2.11}$$

where $X_{\dot{v}}$ is the added mass force X along the x-axis due to an acceleration $\dot{v}$ in the y direction. The other elements of the latter matrix are to be interpreted in the same way.

The hydrodynamic coriolis–centripetal matrix $\mathbf{C}_A(\nu)$, also referred to as the *added mass coriolis* matrix, can be expressed as

$$\mathbf{C}_A(\nu) = \begin{bmatrix} \mathbf{0}_{3x3} & -\mathrm{S}(\mathbf{A}_{11}\mathbf{v}^b_{nb} + \mathbf{A}_{12}\boldsymbol{\omega}^b_{nb} \\ -\mathrm{S}(\mathbf{A}_{11}\mathbf{v}^b_{nb} + \mathbf{A}_{12}\boldsymbol{\omega}^b_{nb} & -\mathrm{S}(\mathbf{A}_{21}\mathbf{v}^b_{nb} + \mathbf{A}_{22}\boldsymbol{\omega}^b_{nb} \end{bmatrix} \tag{2.12}$$

and is thus only dependent on ν and $\mathbf{M}_A$.

$\mathbf{M}_A$ can be simplified by assuming port-starboard symmetry of the UUV. This assumption is often reasonable. The resulting matrix is

$$\mathbf{M}_A = -\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} = -\begin{bmatrix} X_{\dot{u}} & 0 & X_{\dot{w}} & 0 & X_{\dot{q}} & 0 \\ 0 & Y_{\dot{v}} & 0 & Y_{\dot{p}} & 0 & Y_{\dot{r}} \\ Z_{\dot{u}} & 0 & Z_{\dot{w}} & 0 & Z_{\dot{q}} & 0 \\ 0 & K_{\dot{v}} & 0 & K_{\dot{p}} & 0 & K_{\dot{r}} \\ M_{\dot{u}} & 0 & M_{\dot{w}} & 0 & M_{\dot{q}} & 0 \\ 0 & N_{\dot{v}} & 0 & N_{\dot{p}} & 0 & N_{\dot{r}} \end{bmatrix} \tag{2.13}$$

Assuming symmetry in three planes (port-starboard, fore-aft, top-bottom) allows simplifying $\mathbf{M}_A$ to a diagonal matrix

$$\mathbf{M}_A = -diag\{X_{\dot{u}}, Y_{\dot{v}}, Z_{\dot{w}}, K_{\dot{p}}, M_{\dot{q}}, N_{\dot{r}}\} \tag{2.14}$$

2.2.3.2 Hydrodynamic Damping

Damping of marine vehicles is caused by several effects. These are *potential damping*, *skin friction*, *wave drift damping*, *damping due to vortex shedding* and *lifting forces*.

It is difficult to separate these effects in practice. Therefore, it is convenient to write total hydrodynamic damping as the sum of a linear and a non-linear part

$$\mathbf{D}(\boldsymbol{\nu}_r) = \mathbf{D}_L + \mathbf{D}_{NL} \tag{2.15}$$

where $\mathbf{D}_L$ describes the linear damping effects and $\mathbf{D}_{NL}$ describes those that are non-linear damping effects. It is assumed that these two parts may be superimposed.

The linear damping matrix $\mathbf{D}_L$ includes effects from potential damping and skin friction. It is a full $6x6$ matrix written as

$$\mathbf{D}_L = \begin{bmatrix} X_u & X_v & X_w & X_p & X_q & X_r \\ Y_u & Y_v & Y_w & Y_p & Y_q & Y_r \\ Z_u & Z_v & Z_w & Z_p & Z_q & Z_r \\ K_u & K_v & K_w & K_p & K_q & K_r \\ M_u & M_v & M_w & M_p & M_q & M_r \\ N_u & N_v & N_w & N_p & N_q & N_r \end{bmatrix} \tag{2.16}$$

Non-linear damping $\mathbf{D}_{NL}$ includes effects from wave drift damping, vortex shedding and lifting forces. One way model it is by using only a quadratic term, neglecting all terms with an higher order than two. This approximation is described as rough by Fossen (2011), but was found to be very accurate by Morrison and Yoerger (1993).

Quadratic damping can be written as

$$\mathbf{D}_{NL}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r = \begin{bmatrix} |\boldsymbol{\nu}_r|^T \mathbf{D}_{n1}\boldsymbol{\nu}_r \\ |\boldsymbol{\nu}_r|^T \mathbf{D}_{n2}\boldsymbol{\nu}_r \\ |\boldsymbol{\nu}_r|^T \mathbf{D}_{n3}\boldsymbol{\nu}_r \\ |\boldsymbol{\nu}_r|^T \mathbf{D}_{n4}\boldsymbol{\nu}_r \\ |\boldsymbol{\nu}_r|^T \mathbf{D}_{n5}\boldsymbol{\nu}_r \\ |\boldsymbol{\nu}_r|^T \mathbf{D}_{n6}\boldsymbol{\nu}_r \end{bmatrix} \tag{2.17}$$

where D_i are diagonal $6x6$ matrices.

If a vehicle performs non-coupled motion, one may assume a diagonal damping structure. This simplifies the damping matrices to

$$\begin{aligned} \mathbf{D}_L &= -diag\{X_u, Y_v, Z_w, K_p, M_q, N_r\} \\ \mathbf{D}_{NL} &= -diag\{X_{|u|u}|u_r|, Y_{|v|v}|v_r|, Z_{|w|w}|w_r|, K_{|p|p}|p|, M_{|q|q}|q|, N_{|r|r}|r|\} \end{aligned} \tag{2.18}$$

At very low velocities the damping matrix can be modeled with only linear effects.

$$\mathbf{D}(\boldsymbol{\nu}_r) = \mathbf{D}_L \tag{2.19}$$

2.3 Equations of Motion

The theory from this chapter can be used to make equations of motion, or models, that describe the movement of UUVs. Different equations of motions can be made based on what assumptions one chooses to make. This section lists a few convenient possibilities. For each model the assumptions, equations and the number of required parameters are given.

2.3.1 Full Model

The most complete model without any simplification is given by

$$\begin{gathered} \dot{\boldsymbol{\eta}} = J_q(\boldsymbol{\eta})(\boldsymbol{\nu}_r + \boldsymbol{\nu}_c) \\ (\mathbf{M}_{RB} + \mathbf{M}_A)\dot{\boldsymbol{\nu}}_r + (\mathbf{C}_{RB}(\boldsymbol{\nu}_r) + \mathbf{C}_A(\boldsymbol{\nu}_r))\boldsymbol{\nu}_r + (\mathbf{D}_L + \mathbf{D}_{NL}(\boldsymbol{\nu}_r))\boldsymbol{\nu}_r + \mathbf{g}(\boldsymbol{\eta}) = \boldsymbol{\tau} \end{gathered} \tag{2.20}$$

where $\boldsymbol{\theta} = [\mathbf{M}_A, \mathbf{D}_L, \mathbf{D}_{NL}]^T$ has to be estimated. Assuming no symmetries and normal velocity the number of required parameters is shown in Table 2.2.

Equation 2.20 can be simplified to a more convenient representation if one assumes no current in the vehicles environment. Since this simplification does not affect the offline model estimation, but improves notational brevity, it will be used from here on out. The simplification allows expressing the equations of motion as

$$\begin{gathered} \dot{\boldsymbol{\eta}} = J_q(\boldsymbol{\eta})\boldsymbol{\nu} \\ (\mathbf{M}_{RB} + \mathbf{M}_A)\dot{\boldsymbol{\nu}} + (\mathbf{C}_{RB}(\boldsymbol{\nu}) + \mathbf{C}_A(\boldsymbol{\nu}))\boldsymbol{\nu} + (\mathbf{D}_L + \mathbf{D}_{NL}(\boldsymbol{\nu}))\boldsymbol{\nu} + \mathbf{g}(\boldsymbol{\eta}) = \boldsymbol{\tau} \end{gathered} \tag{2.21}$$

$\mathbf{M}_A$	$\mathbf{D}_L$	$\mathbf{D}_{NL}$	**total**
36	36	36	108

Table 2.2: Number of parameters required in the full model

2.3.2 Diagonal Model

Assumptions

- Port-starboard symmetry
- Fore-aft symmetry
- bottom-top symmetry
- decoupled motion

The model keeps the structure of equation 2.21. $\mathbf{M}_A$ is given by equation 2.14, $\mathbf{D}_L$ and $\mathbf{D}_{NL}$ by equation 2.18. The resulting parameters are given in table 2.3.

$$\begin{gathered} \dot{\boldsymbol{\eta}} = J_q(\boldsymbol{\eta})\boldsymbol{\nu} \\ (\mathbf{M}_{RB} + \mathbf{M}_A)\dot{\boldsymbol{\nu}} + (\mathbf{C}_{RB}(\boldsymbol{\nu}) + \mathbf{C}_A(\boldsymbol{\nu}))\boldsymbol{\nu} + (\mathbf{D}_L + \mathbf{D}_{NL}(\boldsymbol{\nu}))\boldsymbol{\nu} + \mathbf{g}(\boldsymbol{\eta}) = \boldsymbol{\tau} \end{gathered}$$

$\mathbf{M}_A$	$\mathbf{D}_L$	$\mathbf{D}_{NL}$	**total**
6	6	6	18

Table 2.3: Number of parameters required in the diagonal model

2.3.3 Linear Diagonal Model

Assumptions

- UUV moves with low velocity
- Port-starboard symmetry
- Fore-aft symmetry
- Top-bottom symmetry
- Decoupled motion

$$\dot{\boldsymbol{\eta}} = J_q(\boldsymbol{\eta})\boldsymbol{\nu}$$
$$(\mathbf{M}_{RB} + \mathbf{M}_A)\dot{\boldsymbol{\nu}} + \mathbf{D}_L\boldsymbol{\nu} + \mathbf{g}(\boldsymbol{\eta}) = \boldsymbol{\tau}$$

$\mathbf{M}_A$	$\mathbf{D}_L$	$\mathbf{D}_{NL}$	**total**
6	6	0	12

Table 2.4: Number of parameters required in the low-velocity linear diagonal model

2.3.4 Non-linear Surge Model

Assumptions

- Only surge movement
- Neutral orientation

$$\dot{x} = u$$
$$(m_{RB} + m_A)\dot{u} + d_L u + d_{NL}u^2 = X$$

$\mathbf{M}_A$	$\mathbf{D}_L$	$\mathbf{D}_{NL}$	**total**
1	1	1	3

Table 2.5: Number of parameters required in the non-linear surge model

2.3.5 Linear Surge Model

Assumptions

- Only surge movement
- Neutral orientation
- UUV moves with low velocity

$$\dot{x} = u$$
$$(m_{RB} + m_A)\dot{u} + d_L u = X$$

M$_A$	**D**$_L$	**D**$_{NL}$	**total**
1	1	0	2

Table 2.6: Number of parameters required in the linear surge model

Chapter 3

System Identification

Ljung (1987) defined system identification (SI) as dealing with the problem of building mathematical models of dynamical systems based on observed data from the system. The data referred to usually consists of given input to the system and the measured output.

The process of system identification can be split into three parts: 1) gathering the required data, 2) finding a suitable model structure and 3) estimating its parameters. After these steps are completed, the result should be validated. If the results are satisfactory, the task is completed. If not, the three steps should be revisited until the desired performance is achieved. The whole procedure is called *the system identification loop*.

3.1 Machine Learning and Regression

Machine learning is concerned with algorithms that can learn from data. It has been around for several decades, but it has seen an increase in interest over the last few years. Recent advances in big data and cloud computing have made it a viable solution to many problems, and it is today used across many different industries, including the maritime sector. (Dobrkovic et al. (2015))

Machine learning can be divided into *supervised learning*, *unsupervised learning* and *reinforcement learning*. Reinforcement learning is about making an agent learn how to best behave in an environment through trial and error. Unsupervised learning is learning where the data is not labeled. Its goal is to identify patterns in the data.

Supervised learning is learning where data is available, and associated outcomes are known. It includes the subcategory *regression*.

Regression analysis is concerned with finding a model that best describes the relationship between a *target variable* and one or more *predictor variables*. It can be applied to numerous problems, ranging from predicting future stock prices to forecasting city districts' demand for taxis in a given hour. Regression analysis can generally be split into three parts: 1) preprocessing of data, 2) selection of model structure, and 3) optimization - just like system identification.

Regression analysis is the part of machine learning that in many ways is similar to system identification. Both fields have the goal of discovering the model that best describes a system based on measurements of the system. In both procedures, one first selects a suitable model-structure and then finds optimal parameters for it, such that the model optimally aligns with the system's measurements.

3.2 Data Recording and Preprocessing

When doing regression, it is essential to gather *informative* data about the system of interest. Informative data can be defined as data that allows discrimination between any two different models in the set of candidate models (Ljung (1987)). The concept of informative data is closely tied to *persistent excitation*. Boyd and Sastry (1986) proved that a signal is persistently exciting if its spectrum is concentrated on $k \geq M$ lines, where M is the number of unknown parameters.

It is crucial to SI that the measurements $\mathbf{Z}$ must excite all DOFs sufficiently, i.e. that the experiment is persistently exciting. This includes measurements of each DOF in isolation, as well as coupled DOFs together. For UUVs common potentially coupled DOFs are

- surge, pitch and heave
- surge, sway and yaw
- sway, roll and heave

Each of these may be predominantly coupled in only two of the three DOFs.

Gathering required data is commonly done through a specifically designed identification experiment. This experiment allows the user to choose exactly what inputs are given to the system, and to carefully record the systems response to those inputs. The collected data set is referred to as **Z**. The goal of the experiment is to extract as much information about the system as possible. Good system identification is only possible if the experiment is informative enough. The task of designing good experiments that provide the required information about a system is referred to as *experiment design*.

The next step after the data is recorded is getting familiar with the data at hand. It is clear from the definition of machine learning that independent of what method is used, the result will depend heavily on the quality of the available data. Most data sets suffer from imperfections, such as missing data, incorrect data, or wrong formats. Each of these issues have to be discovered before they can be addressed. The second step is *preprocessing*. In this step, the data is prepared to be used in machine learning. Issues discovered in data exploration get corrected here to the extent possible.

3.3 Model Structure

Model structures define a parameterized structure, that given a set of parameters and input produces an output. A suitable model structure has the potential to accurately describe the system of interest. Finding it is the most difficult part of system identification according to Ljung (1987).

Many industries have well-established model structures for their use cases, such as the equations of motion for ships and underwater vehicles presented in Fossen (2011) are in the maritime sector. The availability of model structures can be a significant asset when doing system identification.

Model structures have varying degrees of physical interpretability. Those that cannot be directly linked to the physics of the system, are referred to as *black box* models. Models that have clear links to the physics are called *white box*, and those that fall somewhere in between are referred to as *grey box*. (Ljung (1987))

3.4 Overfit and underfit

When doing regression there are two phenomena one often encounters: *overfit* and *underfit*. Figure 3.1 shows a visual representation of both. Overfit occurs when models become too specialized for the training data. They learn small patterns that only exist in that particular data set and thus perform badly when applied to new ones. Underfit is the opposite of overfit. In this case, the model fails to learn enough about the training data and produces bad predictions. When doing regression, it is important to strike

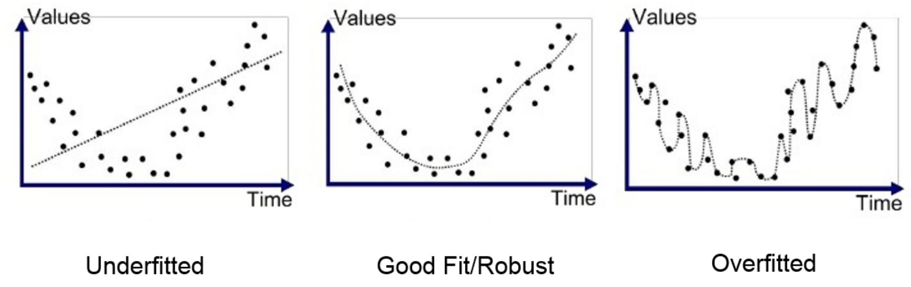


Figure 3.1: Overfit and underfit. Source: towardsdatascience.com

the right balance between overfitting and underfitting. This is especially important when the sample size is small. (Chapelle et al. (2002))

3.5 Parameter Estimation

Once a candidate model structure is found, its parameters have to be estimated. This is done through optimization, which Brunton and Kutz (2019) call the corner stone of regression and machine learning.

The process of optimization can be split into two steps. The first step is creating a *objective function* that, given a set of parameters, quantifies the model error. The second step is then to find those parameters that minimize the objective function. Mathematically it can be expressed as

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta} \in \mathbf{D}_M}{\arg\min} \, f(\boldsymbol{\theta}, \mathbf{Z}) \tag{3.1}$$

where D_M is the *design space* of possible parameters for the chosen model structure, f is the objective function, $\mathbf{Z}$ is the supplied data and $\boldsymbol{\theta}^*$ is a *solution* or a *minimizer.* A vector of parameters is referred to as a *design point* and can be written

$$\boldsymbol{\theta} = [\theta_1, \theta_2, .., \theta_M]^T \tag{3.2}$$

where each θ_i is a *design variable.* (Kochenderfer and Wheeler (2019))

A common challenge for optimization methods is the existence of *local extrema.* These are extrema in a local space containing them, but they are not the extrema in the entire design space. The latter point es referred to as a *global extrema.* Most optimization methods cannot know if they have found a local or global extremum. This is a problem since the latter is usually of most interest. A common approach to alleviating this issue is to ensure good *initial guesses* of the parameters. (Ljung (1987))

3.5.1 Minimizing Prediction Errors

Objective functions should quantify how well a selected model describes the observed data. One way to achieve this is by looking at prediction errors between the observed data and the models predictions. This can be expressed as

$$\epsilon(t, \boldsymbol{\theta}) = y(t) - \hat{y}(t|\boldsymbol{\theta}) \tag{3.3}$$

where $y(t)$ is observed data from Z and $\hat{y}(t|\theta)$ is a model prediction. The objective function can then be defined in terms of the prediction error $\epsilon(t, \theta)$ and an *evaluation metric* λ

$$f(\boldsymbol{\theta}, \mathbf{Z}) = \lambda(\epsilon(t, \boldsymbol{\theta})) \tag{3.4}$$

3.5.2 Evaluation Metrics

There are many different evaluation metrics available that have their own advantages and issues. Examples of evaluation metrics are *mean squared error (MSE), mean absolute error (MAE)* and *symmetric mean absolute percentage error (SMAPE).* Mean squared error is defined as

$$\lambda_{MSE} = \frac{1}{N}\sum_{t=1}^{N} \epsilon(t, \boldsymbol{\theta})^2 \tag{3.5}$$

It heavily penalizes predictions that are far off the true value. This incentives close fits to the true values, but can cause problems if the measurements are noisy or contain outliers. Mean absolute error is defined as

$$\lambda_{MAE} = \frac{1}{N}\sum_{t=1}^{N} |\epsilon(t, \boldsymbol{\theta})| \tag{3.6}$$

It is not as sensitive to noise and outliers, but it is more prone to underfitting. It also tends to induce sparsity into design points, which can result in models with fewer terms that are easier to interpret (Brunton and Kutz (2019). SMAPE is defined as

$$\lambda_{SMAPE} = \frac{100\%}{N}\sum_{t=1}^{N} \frac{|\epsilon(t, \boldsymbol{\theta})|}{\frac{1}{2}(|y(t)| + |\hat{y}(t, \boldsymbol{\theta})|)} \tag{3.7}$$

This metric is independent of scale and can thus compare model performances across different data sets. Its drawback is that its errors will blow up if y approaches zero, as there will be a division by zero. (Hyndman and Koehler (2006))

3.5.3 Optimization Methods

Optimization methods find the maxima or minima of functions. Many different such methods exist, and just as with the evaluation metrics, they have different advantages and drawbacks. Optimization methods can roughly be categorized as either *gradient-based*, *direct* or *stochastic*. They can also be distinguished on whether they work for *single-objective* or *multi-objective* objective functions.

Gradient-based methods rely on derivatives of the objective function to traverse along its surface until a local minimum is reached. The always use the first order derivative, and sometimes also the second order derivative. Direct methods rely on non-derivative information about the objective function to reach local minima. There are many different direct methods using various criteria about the objective function for their descent. Stochastic methods use randomness to strategically explore larger portions of $\mathbf{D}_M$. This increases the chances of finding the global minium, or at least better local minima.

3.5.4 Multi-Objective Optimization

Single-objective optimization concerns itself with objective functions compute only a single objective, while multi-objective optimization deals with objective functions with multiple objectives. Multi-objective optimization introduces additional challenges compared to single-objective. Having multiple objectives often requires making trade-off between them during optimization, meaning that the design point that minimizes one objective might perform poorly in other objectives. A common example of this can be in engineering, where a tradeoff between cost, performance and time-to-market has to be made. Deciding on good tradeoffs between objectives can be challenging and it is often easier to use *Pareto optimality*.

A solution is *Pareto optimal* if no objective can be improved upon without worsening another one. In most problems there is not one single Pareto optimal point, but rather a set of them. This set is called the *Pareto frontier*. A common approach to multi-objective optimization is to first compute the Pareto frontier and then selecting a design point from there based on some additional criteria. One method for performing such selection is through *hypervolume based knee point detection*, which is characterized by that a small improvement in one objective will cause a large deteriation in antoher. The method in described in more detail in Dai et al. (2019).

Pareto optimality can also be described in terms of *dominace*. One design point $\boldsymbol{\theta}_1$ is

```
function genetic_algorithm(f, R, n_gen_max)
    while termination criterion not satisfied
        P = select(f, R)
        Q = empty_set()
        for p1, p2 in P
            add(Q, crossover(p1, p2))
        R = mutate(Q)
    return pareto_frontier(f, R)
```

Listing 3.1: Genetic algorithm

said to *dominate* another $\boldsymbol{\theta}_2$, iff

$$\begin{aligned} f_i(\boldsymbol{\theta}_1) &\leq f_i(\boldsymbol{\theta}_2), \forall i \in [0, 1, .., m] \\ f_i(\boldsymbol{\theta}_1) &< f_i(\boldsymbol{\theta}_2), \exists i \in [0, 1, .., m] \end{aligned} \tag{3.8}$$

for a objective function f with m objectives. The Pareto frontier is thus defined as the set of design points that are not dominated by any other points in the design space, i.e. the non-dominated (ND) design points.

3.5.5 Population methods

Population methods simultaneously optimize a set of design points, instead of only focusing on one point. Each design point is referred to as an *individual*, and the set of all individuals is called the *population*. Information can be shared between individuals to improve the global optimization. Most population methods are stochastic and it is often easy to parallelize their computation. A common form of population methods are *genetic algorithms*. These take inspiration from natures natural selection. Each individuals design point is represented as its *chromosome*. The algorithm then iterates through *generations*, where only the *fittest*, i.e. best performing, individuals are *selected* as *parents* to pass on their genes to the *offspring*. The next next generation is then created through the genetic operations *crossover* and *mutation*.

Listing 3.1 shows a basic implementation of a genetic algorithm, where R is the population, P is the set of parent individuals and Q is the set of generated offspring. Different options are available for the population methods subroutines selection, crossover and mutation. A short description on a few options is given below.

Selection is the task deciding on which individuals from P should be chosen to generate offspring. Tournament selection consists of choosing k random individuals from P and selecting the fittest of them. The process is repeated until enough offspring is generated. Tournament selection with $k = 2$ is also known as binary tournament selection.

Crossover is about deciding how chromosomes, i.e. design points, of two parents should be combined in offspring. Simulated binary crossover uses probability distributions that are centered around the parents design points to choose the offspring's design points. A parameter η_c influences the distributions spread. A small η_c (around 1-5) will give a wide spread, allowing the algorithm to explore more of the design space around the parents. A larger η_c will reduce the spread, which leads to offspring's that contain design points close to one of their parents. A more detailed explanation of of simulated binary crossover is given by its creators in Deb et al. (1995).

Mutation allows genetic algorithms to explore more of the state space by creating random alterations in the design points of offspring. The probability of a mutation occurring is commonly set to $1/m$, where m is the number of elements in the design points. Polynomial mutation uses the same probability distribution as simulated binary crossover to generate altered design points. Its spread is also controlled by a parameter η.

A termination criterion is required to decide when the optimization should end. The number of generations is commonly used as criteria, but this has the drawback of not utilizing any information on the current state of the optimization.

3.5.6 Running Performance Metric

Blank and Deb (2020b) recently proposed a running performance metric that can be used to quantify a genetic algorithms performance over time. The metric is useful both for visualization of performance as well as as a termination criterion. It is based on the realization that most genetic algorithms have two distinct phases: the convergence phase $\mathbb{C}^E$ and the diversity creation phase $\mathbb{C}^D$. During $\mathbb{C}^E$ the algorithm converges towards extrema, i.e. its best estimates improve rapidly. During $\mathbb{C}^D$ the best estimates no longer improve significantly, but the diversity of the ND set increases. Ideally, the optimization should terminate when both $\mathbb{C}^E$ and $\mathbb{C}^D$ have finished and the algorithm has stopped improving its result.

Performance during $\mathbb{C}^E$ is described using the normalized change in the objective space of two points z^* and z^{nad}. The ideal point z^* is defined as the minimal objective value for all design variables in ND. Opposite to the ideal point is the nadir point z^{nad}, which consists of the maximum objective values of the design variables in ND. A performance metric is found by investigating the maximum normalized change of

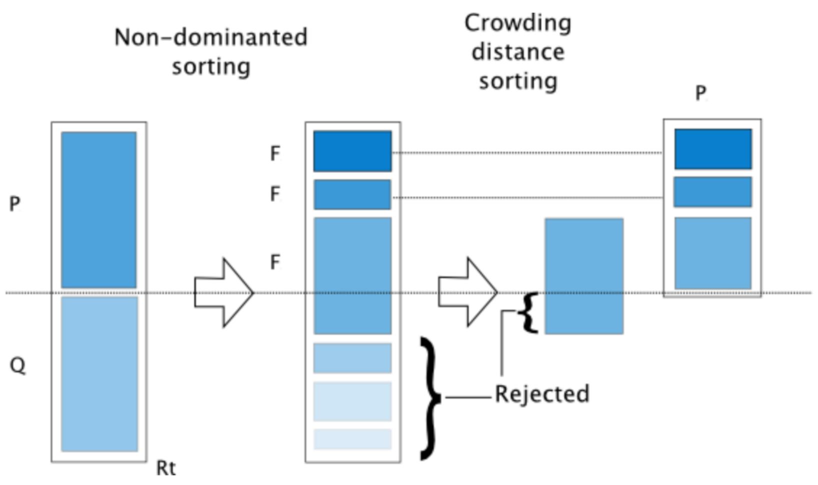


Figure 3.2: NSGA-II procedure

these points, defined as

$$
\begin{aligned}
\Delta_{t-1,t}\mathbf{z}^* &= \max_i^m \frac{z_i^*(t-1) - z_i^*(t)}{z_i^{nad}(t) - z_i^*(t)} \\
\Delta_{t-1,t}\mathbf{z}^{nad} &= \max_i^m \frac{z_i^{nad}(t-1) - z_i^{nad}(t)}{z_i^{nad}(t) - z_i^*(t)}
\end{aligned}
\tag{3.9}
$$

where m is the number of design variables and t is the generation.

The performance during $\mathbb{C}^D$ is quantified using the smallest distance $\varnothing_t$ from a design point in P_{t-1} to P_t. Normalization is done using z^* and z^{nad} at time t. This must be considered when visualizing the running metric, since every previously calculated $\varnothing$ needs to be recalculated to fit the new normalization.

A termination criterion using $\Delta_{t-1,t}\mathbf{z}^*$, $\Delta_{t-1,t}\mathbf{z}^{nad}$ and $\varnothing_t$ can now be defined as

$$
\begin{aligned}
\max(\Delta_{t-1,t}\mathbf{z}^*, \Delta_{t-2,t-1}\mathbf{z}^*, .., \Delta_{t-w,t-w+1}\mathbf{z}^*) &< \epsilon \\
\max(\Delta_{t-1,t}\mathbf{z}^{nad}, \Delta_{t-2,t-1}\mathbf{z}^{nad}, .., \Delta_{t-w,t-w+1}\mathbf{z}^{nad}) &< \epsilon \\
\max(\varnothing_t, \varnothing_{t-1}, .., \varnothing_{t-w}) &< \epsilon
\end{aligned}
\tag{3.10}
$$

where w is a lookback distance, ϵ is a termination threshold and t is the generation.

3.5.7 NSGA-II

Deb et al. (2002) presents a modified version of an genetic algorithm called the Non-dominated Sorting Genetic Algorithm II (NSGA-II). It improves upon the classic genetic

```
function select(f, R)
    max_n_parents = size(R) / 2
    F = non_dominated_sort(f, Rt)
    P = empty_set()
    for Fi in F
        if size(P) + size(Fi) <= max_n_parents
            add(P, Fi)
        else
            crowding_distance_sort(Fi)
            add(P, Fi[1:(max_n_parents - size(P)]
            break
    return P

function nsga2(f, R, n_gen_max)
    for generation in 1 : n_gen_max
        P = select(f, R)
        Q = empty_set()
        for p1, p2 in P
            add(Q, crossover(p1, p2))
        R = union(P, mutate(Q))
    return pareto_frontier(f, R)
```

Listing 3.2: NSGA-II

algorithm by offering 1) better performance through the use of *elitism*, 2) time complexity of $O(MN^2)$ compared to the other genetic algorithms $O(MN^3)$ (where M is the number of objectives and N is population size) and 3) not requiring tuning of the sharing parameter due to the use of a *crowded-comparison* approach.

NSGA-II distinguishes itself from other genetic algorithms on how it 1) selects individuals from a population and 2) creates new populations. The selection is done by first sorting individuals from the population R_t into fronts F_i based on how many other individuals dominate them. The least dominated fronts are then put into the next parent population P_{t+1}, until a front F_x does not fully fit within the size limit of P_{t+1}. Crowding distance sorting is then performed on F_x and the individuals with the most distance to neighbors in the objective space are selected to fill the remaining places in P_{t+1}. The remaining individuals in F_x as well as the remaining fronts $F_{[x+1:]}$ are rejected. P_{t+1} is then used to generate an offspring generation Q_{t+1} through ordinary crossover and mutation. Note that Q_{t+1} is only half the size of the original population R_t. The next population R_{t+1} is created as the union of P_{t+1} and Q_{t+1}. This introduces elitism by ensuring that the best individuals are kept throughout the generations. Listing 3.2 and figure 3.2 show the selection process of NSGA-II.

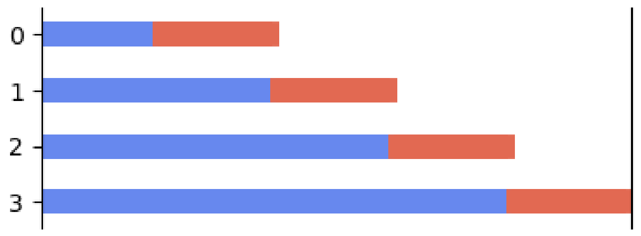


Figure 3.3: Illustration of rolling forecasting origin split. Blue marks the training set, and red the validation set. The iteration number is marked by the y-axis and the samples are along the x-axis.

3.6 Model Validation

After a model is created, it has to be *validated*. This process ensures that a model is not overfitted to patterns in the data that do not generalize into deployment. Validation allows higher certainty about a models performance when faced with new data during deployment. The most common validation technique is *cross-validation*. It works by first splitting the available data into a training and a test set. Training data is used for training the model, while test data is used to evaluate the models performance. It is important to never use data from the test set when adjusting or optimizing the model, as it would invalidate the evaluation. The test set is further split into subsets, but different methods split it in different ways. The names of cross-validation methods are used inconsistently, but the ideas remain the same. The following two methods are based on definitions by Brunton and Kutz (2019) and Hyndman and Koehler (2006).

K-fold cross-validation splits the training set into a validation set and a new, smaller training set. The training set is then further split into k subsets, from which k models are created. The performance of these k models is evaluated on the validation set. One can then choose to use the model with the best validation performance, or take the average across all models.

Evaluation on a rolling forecasting origin is a method well suited for time dependent data. The split is illustrated in figure 3.3. It works by initially selecting the earliest samples as a training set and using adjacent samples for validation. In the following iteration additional samples are added to the training set, and a new validation set is created from adjacent samples. The process is repeated until all available training data is in use. The methods name stems from the forecasting origin of the validation set that "rolls" forward for each iteration. An important property of the rolling forecasting origin is that it does not break time dependencies as the validation set always trails the training set.

3.7 Other Parameter Estimation Methods

System identification is only one of the methods used for parameter estimation of UUVs. Other methods can broadly by categorized as *experimental*, *numerical* and *empirical*.

Experimental parameter estimation methods are the most similar to system identification. They revolve around using carefully designed experiments to record measurements of the vehicle (or a scale model of it) in controlled environments. Examples are constant velocity or acceleration based tests (Newman (2018)) and free decay tests (Faltinsen (1993)).

Numerical methods use computers to approximate hydrodynamic forces acting around a CAD model. Since Navier-Stokes equation cannot be numerically computed for real life applications yet, numerical methods rely on simplifications to arrive at results. A common numerical method today is *strip theory* (Fossen and Øyvind N. Smogeli (2004)).

Empirical methods find hydrodynamic parameters by approximating the vehicle by geometrical shapes with known hydrodynamic properties. A list containing added mass and damping coefficients for such known shapes can be found in DVL-GL (2011).

Chapter 4

Beluga UUV

4.1 Overview

Beluga is a UUV developed by students at Vortex NTNU. It was designed and build between autumn 2020 and spring 2021, originally meant to compete at the annual RoboSub competition is San Diego. It is build around an aluminum frame, which gives it great flexibility in terms of equipment change, but also leads to Belugas quite complicated shape. Through Vortex's contact with the Norwegian maritime industry the organization has been able to equip Beluga with industry-level sensors of high quality. The UUV weights 25.36 kg and at the time of testing had a slightly negative netto buoyancy. Figure 4.1 shows Beluga seen from above.

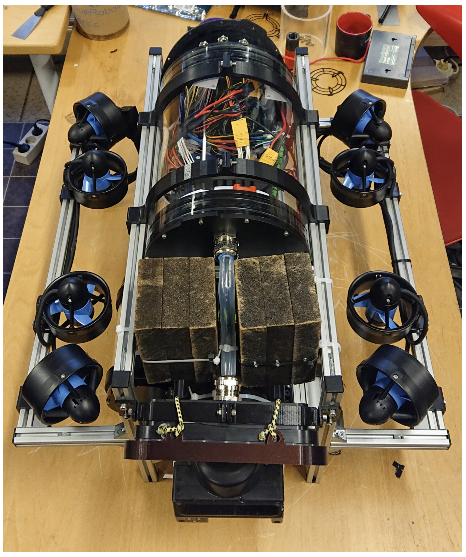


Figure 4.1: Beluga

4.2 Propulsion System

Beluga has eight thrusters placed along the its wings, as can be seen in figure 4.1. Four thrusters operate in the vertical plane and four in the horizontal plane. Those in the vertical plane are placed further in front and in the back than the vertical ones, and they are oriented in 45 deg angles towards the x-axis.

Belugas thrusters are controlled by two 4-in-1 KISS electronic speed controllers (ESC), which receive commands from a Adafruit PCA9685 PWM board, that in turn is controlled from a Nvidia Xavier AGX over an I2C connection. The speed controllers are not centered and therefore require an offset to be applied their commanded PWM signal. Each ESC has an input range of 1100 to 1900, where 1500 means no thrust.

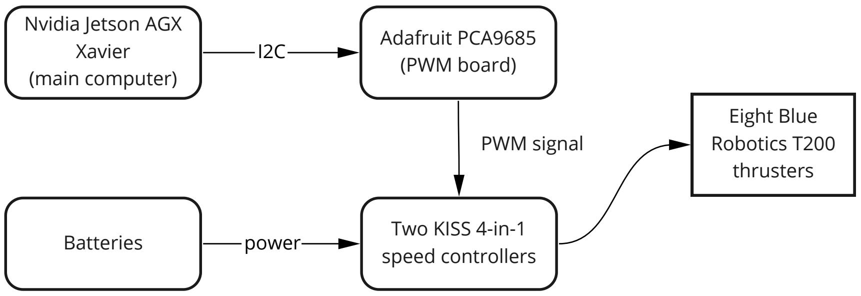

Nvidia Jetson AGX Xavier (main computer)
I2C
Adafruit PCA9685 (PWM board)
PWM signal
Eight Blue Robotics T200 thrusters
Batteries
power
Two KISS 4-in-1 speed controllers

Figure 4.2: Illustration of Belugas propulsion system

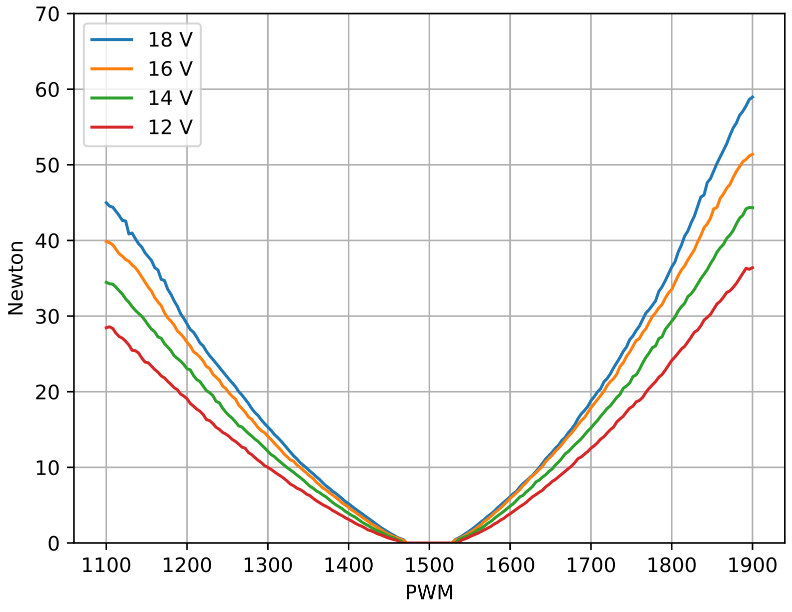


Figure 4.3: T200 thrust ranges at different voltages. The direction of thrust changes at 1500 PWM.

All thrusters are powered by two batteries that supply between 16.7 and 14 Volts depending on charge state. Peak current supply is 160 Ampere. Delivered thrust by each thruster varies depending on the direction thrust is given in. Delivered thrust also depends on the supplied voltage level, as shown in figure 4.3. Maximum thrust at 16V is 51.4N forwards and 39.2N backwards. At 14V this reduces to 44.1N forwards and 34.5N backwards. The difference in maximum thrust given by a voltage change of 2V is thus 7N forwards and 5N backwards. These changes occur while the batteries are well within their operational range and are significant when delivering strong thrusts. Note that figure 4.3 does not show a precise measurement for a voltage supply of 16.7V, which the batteries supply when fully charged.

The software stack running the propulsion system is illustrated in figure 4.4. The stack is made up of four nodes: Joystick Guidance, Thrust Allocator, Thruster Interface and PCA9685 Driver. Joystick Guidance receives inputs from a joystick and mapps them to a 6DOF desired thurst $\boldsymbol{\tau}^{ENU} = [X, Y, Z, K, M, N]^T$. Due to legacy reasons the software operates with a East North Up (ENU) frame instead of the more common NED frame. Thrust Allocator receives $\boldsymbol{\tau}^{ENU}$ and calculates what thrust will be required by each thruster to achieve it. This is done using a thruster allocation matrix such as described in Fossen (2011). Measurements of each thrusters position and orientation in relation

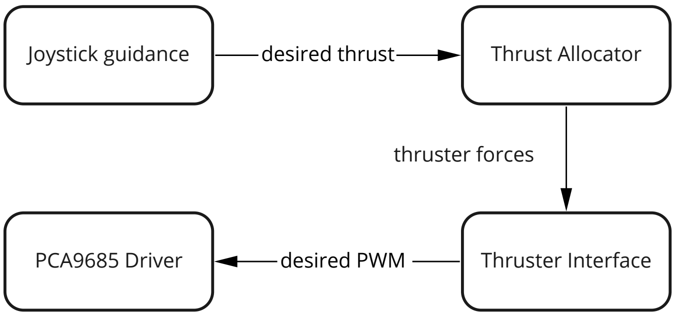


Figure 4.4: Illustration of Beluga propulsion software stack

to CO are needed to create the thruster allocation matrix. The resulting thruster forces are then sent to the thruster interface, which is responsible for mapping desired forces to desired PWM signals. This mapping is illustrated in figure 4.3, and is as mentioned earlier depended on the voltage level of the batteries. Finally, the desired PWM is recieved by the PCA9685 Driver. This node adds offsets to each signal before sending them to the PCA9685 by I2C communication. These offsets are manually tuned.

4.3 Sensors

Beluga is equipped with an IMU, a DVL and a voltage measurement sensor. Both IMU and DVL are industry-level sensor, which both provide reliable measurements of high quality. The IMU is a stim300 IMU by Sensonor and the DVL is a DVL1000 by Nortek. Belugas system voltage is measured with a power sense module from Blue Robotics.

4.4 Hardware issues

Belugas construction is not completed yet and its shell is not attached. This results in hydrodynamic properties that will be hard to model using analytical, empirical or numerical methods. Figure 4.5 shows Beluga UUV as it was used during the experiment described in chapter 5.

Belugas buoyancy installation is not in place yet and consists of many individual buoyancy elements that are attached to the main frame using zip-ties. The elements can move slightly back and forth around their mounting position. One of the 3d printed mounts for Belugas main electronics housing broke during initial testing. Duct tape

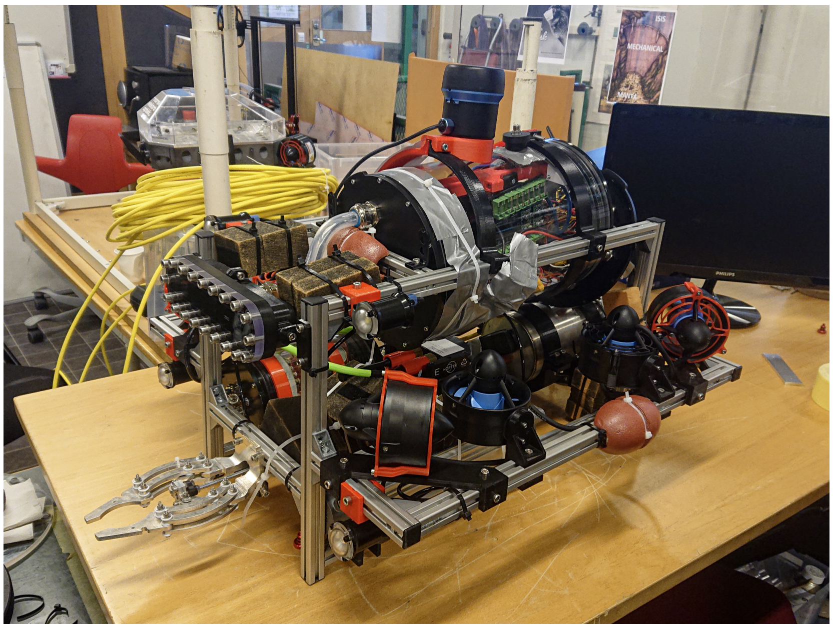


Figure 4.5: Beluga as it was during the experiment

and zip-ties where instead used to keep the housing in place. This allowed for some movement of the housing as well as the sonar mounted on top during operation.

Chapter 5

The Experiment

This chapter covers an experiment for model estimation of an UUV using SI. Beluga UUV is used as testing platform. The experiment is conducted at NTNUs Marine Cybernetics laboratory (MC-lab).

5.1 The Experiments Objective in a SI Context

The goal of this experiment is to estimate parameters $\boldsymbol{\theta}$ of a UUV model through system identification methods, where $\boldsymbol{\theta}$ is given by

$$\boldsymbol{\theta} = \begin{bmatrix} \mathbf{M}_A \\ \mathbf{D}_L \\ \mathbf{D}_{NL} \end{bmatrix} \tag{5.1}$$

To gather the necessary measurements of the system, the UUV is excited in different ways while its response to various thrusts $\boldsymbol{\tau}$ is measured. The measured states are

$$\mathrm{y} = \begin{bmatrix} \boldsymbol{\eta} \\ \mathrm{v}_{nb}^{b} \\ \boldsymbol{\omega}_{nb}^{b} \end{bmatrix} \tag{5.2}$$

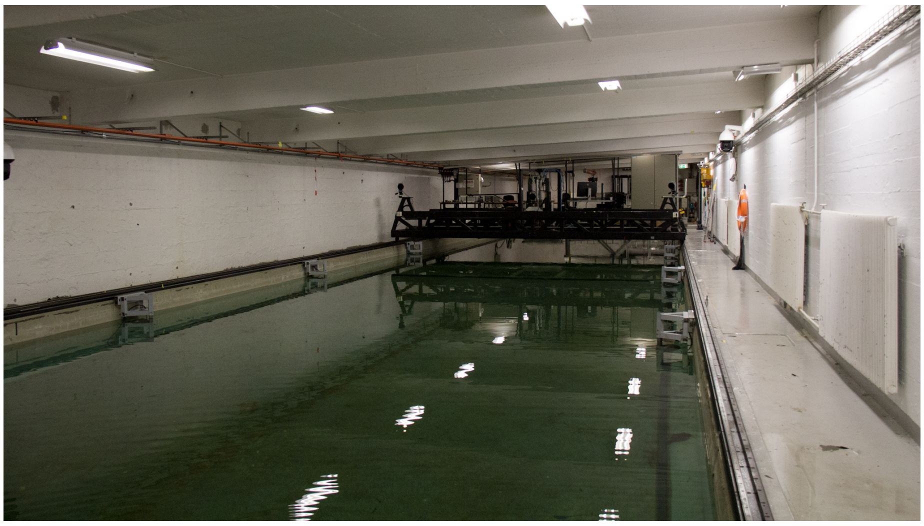


Figure 5.1: Picture of MC-lab

5.2 Experiment Facilities

5.2.1 Marine Cybernetics Laboratory

NTNUs Marine Cybernetics laboratory (MC-lab) is a 40m x 6,45m x 1,5m basin located at Tyholt in Trondheim. It is operated by the Department of Marine Technology and mainly used by master students and phd-candidates. The laboratory is equipped with a 6DOF real-time positioning system capable of detecting submerged objects. Figure 5.1 shows the basing photographed from the entry corner.

The 6DOF real-time positioning system is provided by Qualisys. It consists of Oqus Qualisys infra-red cameras, of which 3 cameras are above surface and 6 are below surface. These are used to detect small ball-shaped infra-red reflectors are attached to the object of interest during operation. Qualisys Track Manager (QTM) software is used to find accurate position measurements by comparing the reflectors positions across the cameras. The update frequency of pose measurements can be set between 100 Hz and 500 Hz. Mounting points for the cameras can be seen as grey structures along the basin side and on the carriage in figure 5.1. Ultraviolet is emitted below the surface to enhance measurements subsea measurements. Outside light should be reduced as much as possible to reduce interference. The six underwater cameras cover a limited volume that about 4 meters long, 2 meters wide and as deep as the basin. The space available for testing is limited by that volume.

Pose measurements from QTM can achieve millimeter precision if the system is set up well and calibrated. A calibration is valid for 6 hours, but good results can be expected for the duration of a day. The position of the reflectors on the object of interest is important for QTMs estimate stability. Bad placement of the reflectors can cause QTM to lose track of the objects pose frequently and sometimes reinitialize the objects orientation incorrectly. For example, QTM might reinitialize an object upside down. For optimal stability the reflectors should be placed in several planes and in an asymmetrical layout. There have to be at least three reflectors attached to the object for QTM to create measurements, but using more reflectors increases QTMs stability.

An objects CO can be defined in relation to the reflectors in several ways. One method is to define CO position in relation to one of the reflectors. CO orientation can defined by aligning the x-axis and y-axis with lines between two and two reflectors.

QTMs measurements are published in real-time to a ROS-network by a driver written by the folks at KTH (KTH (2020)). The driver includes a Kalman filter for calculating linear and angular velocity. More information on QTM its set up at MC-lab can be found on the labs official website of Marine Technology (2021).

5.3 Experiment Design

5.3.1 Measurements

Good measurements are essential to successful system identification. In this experiment, pose, velocity and thrust are the main states of interest. Through various sensors the UUVs position and velocity are measured. Three sensors are used to measure position and velocity. These are the MC-lab QTM system, Belugas IMU and Belugas DVL. QTM is used for position and orientation, IMU for angular velocity and DVL for linear velocity. Belugas thrust output is estimated based on known characteristics of the thruster and supplied voltage in any moment. The complete list of measured and estimated states is given in Table 5.1. The measurements are stored in compressed bag files using bzip2 for compression.

Custom 3d printed mounts are used to attach eight infra-red reflectors to Belugas frame in an asymmetrical layout. Three reflectors are attached to the drone in an L-shape, where the orthogonal lines are perpendicular to the x- and y-axis. These reflectors are used to set up CO correctly in QTM. The ceiling lights of MC-lab are turned off during measurements, such that interference from surface lights is kept to a minimum. The ultraviolet lighting of the pool during testing can be seen in figure 5.2. The system is calibrated once a day. Properties of QTMs frame of reference are

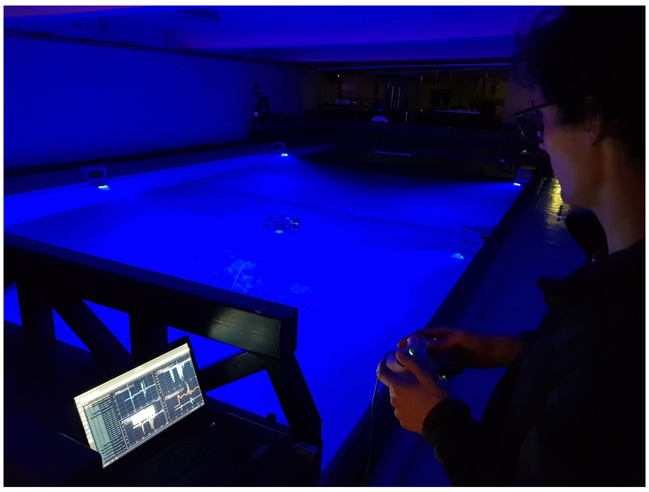


Figure 5.2: Beluga during recording of random movements

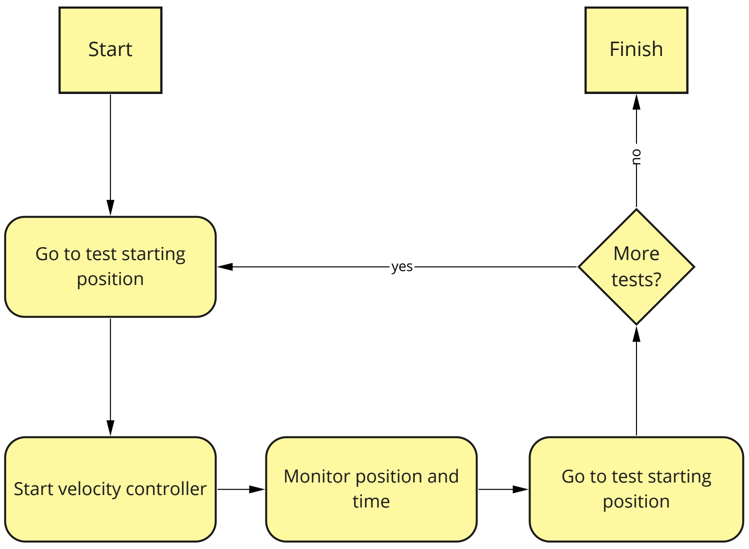
defined during calibration. It is set up using ENU as orientation and origin placed on the basin floor.

5.3.2 Movement patterns

There are two categories of movement patterns: random movements by human operator control and predefined velocity tests using FSMs and Belugas control system. Predefined velocity tests are used to isolate single DOFs and coupled terms. Belugas control system is used to achieve desired motions and velocities, while FSMs are used to control the order in which tasks are executed. During random movements Beluga is controlled by joystick while its control system is set to either depth hold or open loop. The primary goal during random tests is to excite all DOFs and combinations of DOFs sufficiently.

State	**Source**
Pose $\boldsymbol{\eta}$	Measured with QTM
Linear velocity $\boldsymbol{v}_{nb}^{n}$	Measured with DVL
Angular velocity $\boldsymbol{\omega}_{nb}^{n}$	Measured with IMU
Delivered thrust $\boldsymbol{\tau}$	Estimated with procedure described in Chapter 4
Desired thrust	Output from software system
System voltage	Measured with voltage sensor
Thruster PWM signals	Output from software system

Table 5.1: Measured and estimated states



Start
Finish
no
Go to test starting position
yes
More tests?
Start velocity controller
Monitor position and time
Go to test starting position

Figure 5.3: Sketch of mission FSM

The FSM used for predefined tests is illustrated in figure 5.3. It consist of a series of tests, where each test is a series of four states: 1) go to pose, 2) set velocity, 3) monitor and 4) go to pose. The first and fourth state uses the LOS-guided controller to reach the proximity of a desired pose. Once it is close enough, it switches to the DP controller for accurate positioning. The second state simply gives the velocity controller a target velocity. The third state monitors if Beluga is within the boundaries the operational volume and check if a timeout has occurred. The operational volume is defined as $x \in [-2, 2], y \in [-0.85, 0.85], z \in [0.4, 1.05]$, where the z-axis is positive upwards and zero is on the basin floor. Timeouts occur after 15 or 20 seconds depending on the test. A different series of tests is used depending on what DOF or DOFs is excited. Table 5.2 shows all predefined tests.

Due to problems with the controller working against Belugas strong restoring forces,

DOF(s)	**Velocities**
surge	± 0.05, 0.1, 0.15, 0.2, 0.25 and 0.3 m/s
sway	± 0.05, 0.1, 0.15, 0.2, 0.25 and 0.3 m/s
heave	± 0.05, 0.1 and 0.15 m/s
yaw	± 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5 and 0.6 rad/s
surge and sway	both at ± 0.05, 0.1, 0.15, 0.2, 0.25 and 0.3 m/s
surge and heave	heave ± 0.05, 0.1 and 0.15 m/s and surge at ± 0.1 and 0.2 m/s

Table 5.2: Predefined tests

roll and pitch are not tested in predefined tests and have to be sufficiently excited in by operator control.

5.4 Measurement Evaluation and Preprocessing

5.4.1 Error Sources and Measurement Noise

Effects that pollute the measurements are

- water currents caused by moving through it
- ground and surface effects
- voltage spikes caused by quick changes in thrust
- thrust estimate skew caused by tuning of ESCs
- variations in physical properties caused by slack in mounting of physical parts
- buoyancy elements soaking up water
- measurement noise and bias

5.4.2 Measurement Synchronization

Belugas measurements are not recorded at synchronized timesteps due to the distributed architecture of Belugas software system. Therefore all measurements must be synchronized before further usage. This can be done through linear interpolation. First, a function is created for each feature that best fits its values. That function is then used to get values for the feature at timesteps it has not been measured at. All features are recreated with the same synchronized sampling frequency. For this project, all features were recreated with sampling frequencies of 10 Hz. These recreated features are approximations, but since the system at hand changes slowly and the sampling frequencies are relatively high, it is assumed that the introduced error is negligible. Different start and end times of measurement nodes are addressed by deleting all measurements from times where not all nodes are available.

5.4.3 Quaternions

QTM changes the signs of quaternions $\mathbf{q}_b^n$ when Beluga crosses orientations of 180°, eg. when yaw transitions from 179° to $-179°$. This causes problems both when derivating $\boldsymbol{\eta}$ and when comparing $\boldsymbol{\eta}$ to an integrated $\boldsymbol{\nu}$. The change of sings causes large spikes during derivation and mismatches between integrated values, as they do not contain this change of signs. These issues are prevented by removing the sign changes introduced by QTM.

Quaternions are great for robust and computationally inexpensive rotations, but they are not as intuitive to understand as Euler angles. Therefore, orientations $\mathbf{q}_b^n$ are mapped to Euler angles and added to the dataset. The *zyx* rotation sequence is used for the mapping, following the convention used in Fossen (2011).

5.4.4 Measurement Quality Checks

Measurement quality is checked with integration and derivation tests. $\boldsymbol{\nu}$ is integrated and compared to $\boldsymbol{\eta}$. $\boldsymbol{\eta}$ is derivated and compared to $\boldsymbol{\nu}$. Errors between calculated and measured values are defined as $error = measured - calculated$. Measurements that show clear deficiencies in these tests are cut such that the bad parts are removed. The resulting datasets are stored as *csv* files.

Descriptive statistics are calculated for each measurement of each test. These contain the number of datapoints N, mean value μ, standard deviation σ, minimum value and maximum value.

Chapter 6

System Identification of UUV Models

The system identification method used in this thesis is based on prediction error minimization from section 3.5.1, but extended to multiple dimensions. It consists of two steps:

Step 1 Define an objective function $\mathbf{f}_{EoM}(\boldsymbol{\theta}, \boldsymbol{\tau}_z, \mathrm{y}_z)$ based on the error between measured states y_z and predicted states $\mathrm{y}_{EoM}(\boldsymbol{\theta}, \boldsymbol{\tau}_z)$.

Step 2 Minimize $\mathbf{f}_{\mathbf{EoM}}$ with the goal of finding the parameters $\hat{\boldsymbol{\theta}}$ that best explain the measurements y_z given the input $\boldsymbol{\tau}_z$.

6.1 Model Predictions

The predictions, i.e. simulations, $\hat{\boldsymbol{y}}_{EoM}$ are generated through numerical integration of the equations of motion from section 2.3. $\hat{\boldsymbol{y}}_{EoM}$ is a time series taking the form

$$\hat{\boldsymbol{y}}_{EoM} = \begin{bmatrix} \hat{\boldsymbol{\eta}}(1), \ldots, \hat{\boldsymbol{\eta}}(N) \\ \hat{\boldsymbol{\nu}}(1), \ldots, \hat{\boldsymbol{\nu}}(N) \end{bmatrix} \tag{6.1}$$

where $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$ are the states defines in table 2.1. The numerical integration is done using the *forward Euler method* given by

$$\hat{\boldsymbol{y}}_{EoM}(t) = \hat{\boldsymbol{y}}_{EoM}(t-1) + h * \dot{\boldsymbol{y}}_{EoM}(t-1) \tag{6.2}$$

where h is a timestep set to 0.1. If $\boldsymbol{y}_{EoM}$ contains quaternions, these are normalized at each timestep (see section 2.1.3). The function $\dot{\boldsymbol{y}}_{EoM}$ is selected based on which UUV model is used for parameter estimation and takes the shape

$$\dot{\boldsymbol{y}}_{EoM} = \begin{bmatrix} \dot{\boldsymbol{\eta}} \\ \dot{\boldsymbol{\nu}} \end{bmatrix} \tag{6.3}$$

where the corresponding $\dot{\boldsymbol{\eta}}$ and $\dot{\boldsymbol{\nu}}$ can be found in section 2.3. They take parameters $\boldsymbol{\theta}$ and a vector of inputs $\boldsymbol{\tau}_z(t)$ as arguments.

6.2 Multi-Objective Optimization

6.2.1 Objective Function

The objective function $\mathbf{f}_{EoM}$ is defined as

$$\begin{aligned} \mathrm{f}_{EoM}(\boldsymbol{\theta}, \boldsymbol{\tau}_z, \mathrm{y}_z) &= \boldsymbol{\lambda}_{MAE}(\mathrm{y}_z - \hat{\mathrm{y}}_{EoM}) \\ &= \frac{1}{N}\sum_{t=1}^{N} abs[\mathrm{y}_z(t) - \hat{\mathrm{y}}_{EoM}(t, \boldsymbol{\theta}, \boldsymbol{\tau}_z, \mathrm{y}_z(0))] \end{aligned} \tag{6.4}$$

where y_z are measurements of the UUVs states from an experiment dataset $\mathbf{Z}$, $\boldsymbol{\tau}_z$ is the series of inputs given during the same experiment and N is the length of $\mathbf{Z}$. $\hat{\mathrm{y}}_{EoM}$ is a state prediction scheme presented in section 6.1, which has parameters $\boldsymbol{\theta}$. abs is absolute value applied element-wise.

The function uses mean absolute error as evaluation metric due to its sparsity-inducing properties described in Brunton and Kutz (2019).

6.2.2 Optimization with NSGA-II

$\mathbf{f}_{EoM}$ is optimized using NSGA-II, which is explained in section 3.5.7. Initial sampling is done randomly. Selection from the parent population P is done with binary tournament selection. Crossover is done using simulated binary crossover. Mutation is implemented as a poynomial mutation, where the probability of mutation is inversely proportional with the number of design variables. The algorithms is terminated based on the running performance metric. This metric is saved during execution and used for visualization of the algorithms performance post-optimization. The population size is set to increase with the number of parameters. All used parameters can be found in table 6.1.

Selection pressure	2
Crossover eta	15
Mutation eta	15
Mutation probability	$1/m$
Population size	10^m

Table 6.1: Used parameters for NSGA-II. m is the number of parameters in $\boldsymbol{\theta}$

6.2.3 Decision Making

After optimization is done and a Pareto front is attained, a decision has to be made about which design point one chooses to trust the most. Knee points simplify the task by reducing the selection from the entire Pareto front down to one or more knee points that are more likely to be correct. The procedure for finding knee points is taken from Blank and Deb (2020a).

6.3 Identification using Simulated Measurements

The method for creating predictions $\hat{\mathbf{y}}_{EoM}$ is also used to create simulated, i.e. synthetic, measurements based on a chosen design point $\boldsymbol{\theta}^*$. This is useful for testing different system identification schemes, as the best estimate $\hat{\boldsymbol{\theta}}$ can be compared to the chosen design point. The simulation can be closely linked to identification, making it easy to swiftly test system identification for different models and parameters.

The identification scheme described so far remains largely the same, but with an additional **step 0** added at the beginning. The step consists of creating simulated measurements $\mathbf{y}_{sim}$ using the procedure described in section 6.1, a chosen series of inputs $\boldsymbol{\tau}_{sim}$ and vector of parameters $\boldsymbol{\theta}^*$. The length of $\boldsymbol{\tau}_{sim}$ determines the length of the measurement series $\mathbf{y}_{sim}$.

The objective function in **step 1** is now given by

$$\mathbf{f}_{EoM}(\boldsymbol{\theta}, \boldsymbol{\tau}_{sim}, \mathbf{y}_{sim}) = \boldsymbol{\lambda}_{MAE}(\mathbf{y}_{sim} - \hat{\mathbf{y}}_{EoM}(t, \boldsymbol{\theta}, \boldsymbol{\tau}_{sim})) \tag{6.5}$$

Two different ramp functions and sinusoidal functions are used as $\boldsymbol{\tau}_{sim}$. They can be seen in figure 6.1. Each $\boldsymbol{\tau}_{sim}$ is chosen to represent realistic maneuvers. The inputs are used in simulations restricted to surge, and therefore follow the naming convention X from table 2.1. X_{single_ramp} ramps up to a forward thrust of 30 newton, lead by and followed by a 10 second period of zero thrust. X_{triple_ramp} contains three ramps of

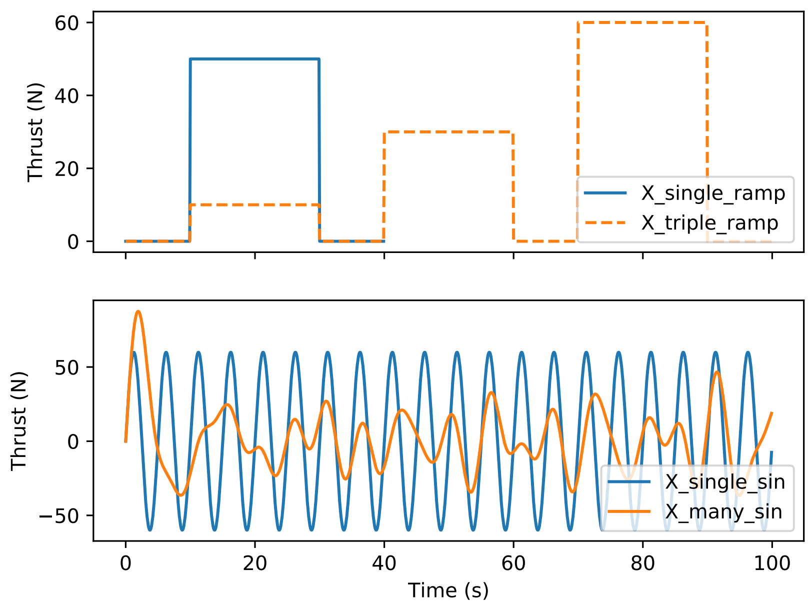


Figure 6.1: Different τ used for simulation

increasing force; first $10N$, then $30N$ and finally $60N$. Periods of zero thrust surround all ramps. The sinusoidal functions are chosen as

$$X_{sin} = \sum_{i=1}^{e} A_i sin(\frac{2\pi}{P_i} t) \tag{6.6}$$

where e is the number of contributing sinusoidal functions. X_{single_sin} has only a single contributing function, with $A = 60$ and $P = 5$. X_{many_sin} has contributions from 10 functions. There share the amplitude $A = 10$, but have different periods $P = [1, 11, 21, 31, 41, 51, 61, 71, 81, 91]$. Both X_{single_sin} and X_{many_sin} are 100 seconds long.

6.4 Software Tooling and Implementation

Python is used as implementation language for parameter estimation. Its rich ecosystem of packages make it easy to a popular choice for tasks where work on data is important. The implementation of NSGA-II from Pymoo (Blank and Deb (2020a)) is used for optimization.

Pythons slow performance is an issue when doing computationally expensive tasks, such as optimization. Just-in-time (JIT) compilation is used to alleviate this issue and improve computational performance. This works by compiling python code at runtime. This introduces a time penalty each time a function is first called, as it has to be compiled. Subsequent calls however execute significantly faster.

JIT is implemented through the package Numba, which is presented in Lam et al. (2015). Numba uses LLVM to compile specially decorated python function. The package only supports a subset of pythons functionality, which includes most of the python standard library and Numpy. Polymorphism is not supported.

The implementation is set up s.t. all design points from a generation are available at the same time. This enables user-defined parallelization of the computation. Numba has straightforward parallelization support through a decorator argument and a special range function for parallelization of `for` loops.

If the $\boldsymbol{\theta}$ used for the prediction $\hat{\boldsymbol{y}}_{EoM}$ contains unrealistic values, it can cause the prediction to become unstable. A limit is imposed on $\dot{\boldsymbol{y}}_{EoM}$ during numerical integration to prevent computational issues with number overflows. The limit set to $10m/s$ and $10rad/s$ for Beluga, as any values approaching this magnitude are clearly impossible during ordinary operation.

Chapter 7

Results

7.1 Beluga Time Series

The measurements of Beluga are split into 14 tests. Table 7.1 shows a list of all tests along with their duration, bag file size and descriptions of their content. There are in total 3 hours, 14 minutes and 32 seconds of measurements. Approximately 2 hours of this is recordings of random movements, and the reaming 1 hour 15 minutes is isolated tests. All measurements are available through the Github repository at the link in figure 7.1. This includes raw data in .bag file format and preprocessed data as .csv files.

Three of the 14 tests are described in more detail in this chapter: *surge-1*, *sway-1* and *heave-1*. Figure 7.2, 7.3 and 7.4 give overviews of the main DOFs timeseries from all three tests. Tables 7.2, 7.3 and 7.4 show statistical description of all measured quantities. Statistical descriptions of the remaining tests can be found in appendix A.

Figure 7.1: `https://github.com/michoy/system_identification`

Name	Duration	Size	Description
surge-1	13 min 16 sec	35.2 MB	Surge at ± 0.05, 0.1, 0.15, 0.2, 0.25 and 0.3 m/s
sway-1	13 min 18 sec	35.6 MB	Sway at ± 0.05, 0.1, 0.15, 0.2, 0.25 and 0.3 m/s
heave-1	4 min 23 sec	11.8 MB	Heave at ± 0.05, 0.1 and 0.15 m/s
yaw-1	10 min 59 sec	29.3 MB	Yaw at ± 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5 and 0.6 rad/s
surge-sway-1	19 min 35 sec	52.9 MB	Surge and sway at ± 0.05, 0.1, 0.15, 0.2, 0.25 and 0.3 m/s
surge-heave-1	14 min 19 sec	38.7 MB	Heave at ± 0.05, 0.1 and 0.15 m/s and surge at ± 0.1 and 0.2 m/s
random-1	11 min 6 sec	30.5 MB	Various movements with depth hold assistance
random-2	9 min 58 sec	27.1 MB	Various movements with depth hold assistance
random-3	16 min 46 sec	45.6 MB	Various movements with depth hold assistance
random-4	9 min 58 sec	26.9 MB	Various movements with depth hold assistance
random-5	6 min 11 sec	16.6 MB	Various movements with depth hold assistance
random-6	19 min 19 sec	52.0 MB	Various movements with depth hold assistance
random-7	19 min 58 sec	52.4 MB	Various movements without depth hold assistance
random-8	25 min 8 sec	65.1 MB	Various movements without depth hold assistance

Table 7.1: Metadata about recordings

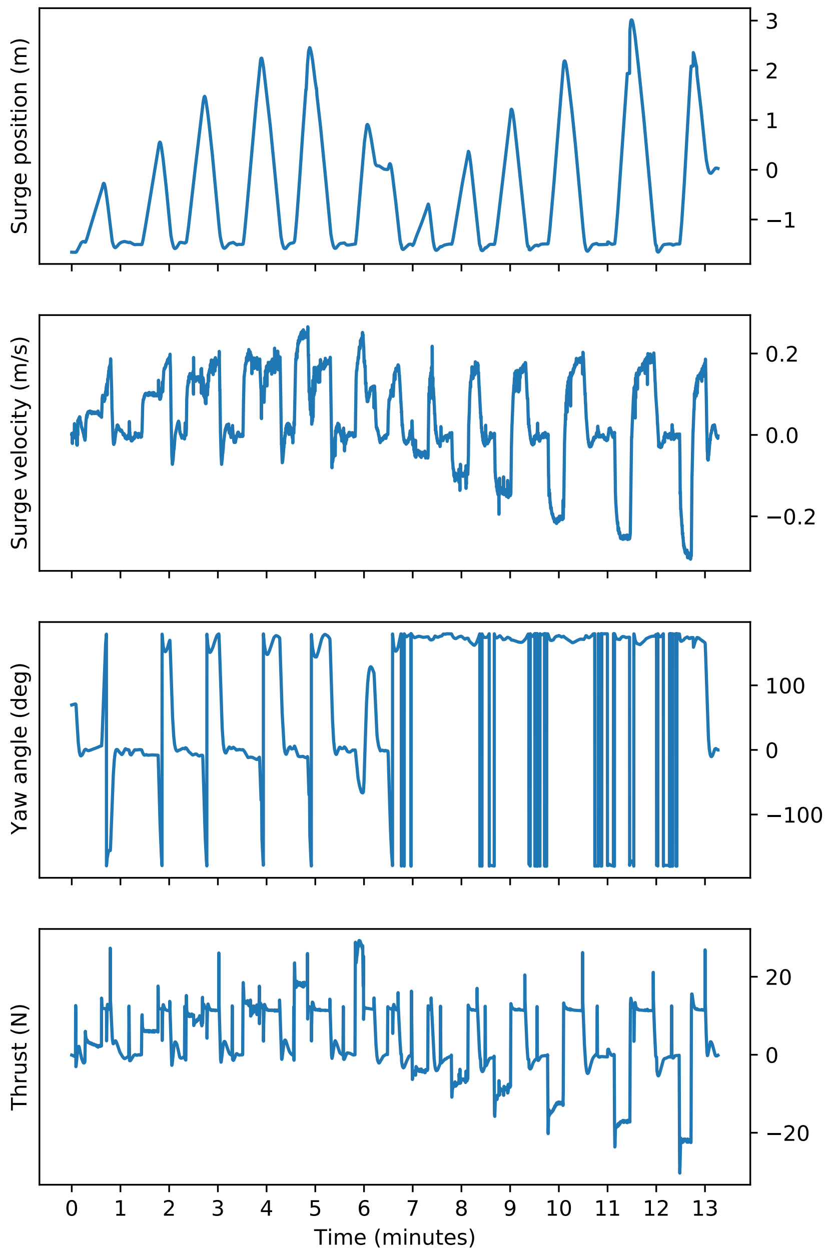


Figure 7.2: Overview of movements in surge test

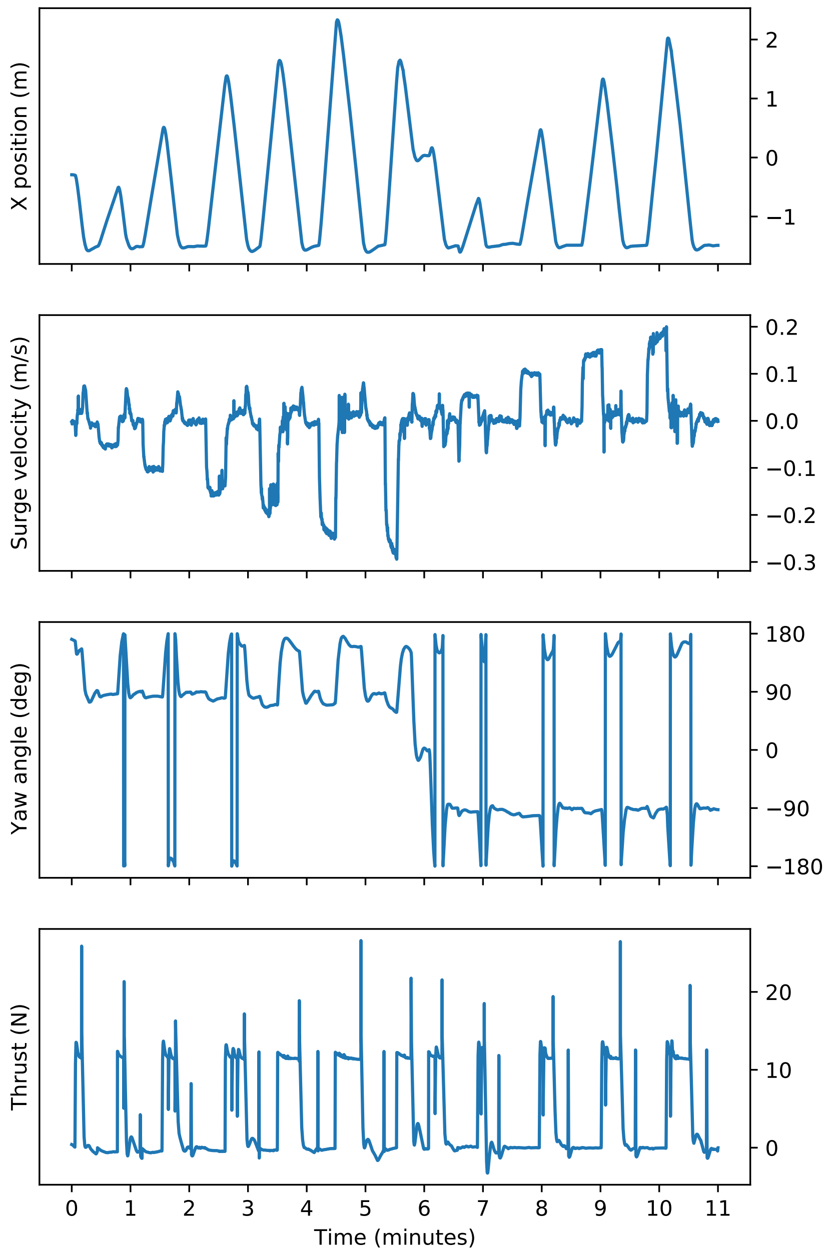


Figure 7.3: Overview of movements in sway test

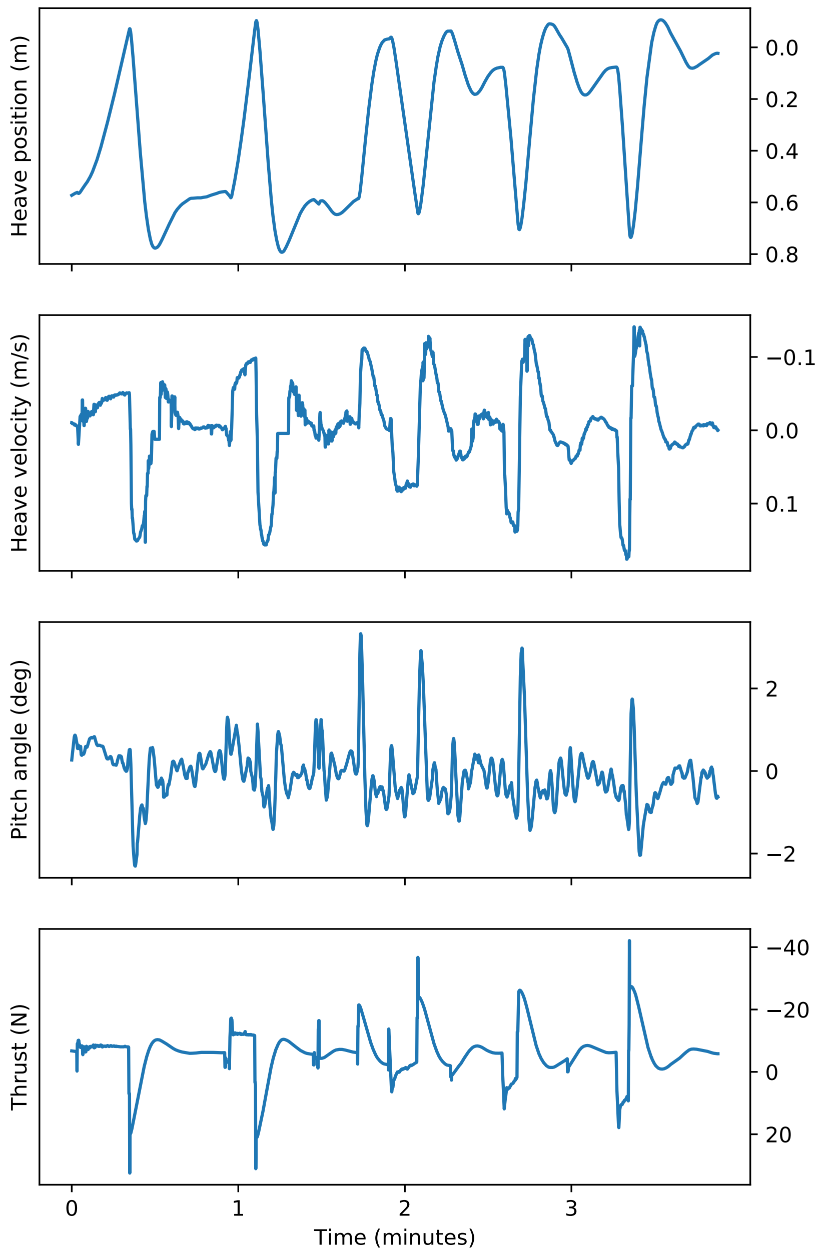


Figure 7.4: Overview of movements in heave tests

	N	mean	std	median	min	max
force_x	7961	3.37	9.08	2.16	-30.35	29.27
force_y	7961	-0.08	1.49	-0.07	-8.31	8.37
force_z	7961	-4.91	2.29	-5.20	-23.14	2.59
torque_x	7961	-0.06	0.14	0.00	-0.56	0.20
torque_y	7961	0.06	0.13	0.00	-0.24	0.37
torque_z	7961	0.16	1.49	0.30	-15.88	13.29
surge_vel	7961	0.04	0.11	0.02	-0.31	0.27
sway_vel	7961	0.00	0.02	0.00	-0.07	0.12
heave_vel	7961	0.00	0.01	-0.00	-0.08	0.06
roll_vel	7961	-0.00	0.02	-0.00	-0.08	0.08
pitch_vel	7961	0.00	0.02	-0.00	-0.09	0.08
yaw_vel	7961	-0.03	0.18	-0.00	-0.91	0.77
position_x	7961	-0.49	1.18	-0.93	-1.66	3.02
position_y	7961	-0.10	0.19	-0.02	-1.10	0.47
position_Z	7961	0.44	0.06	0.44	0.27	0.70
roll	7961	0.04	0.04	0.03	-0.04	0.13
pitch	7961	0.04	0.05	0.03	-0.09	0.15
yaw	7961	-14.26	10.13	-21.90	-25.32	6.32

Table 7.2: Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in surge-1

	N	mean	std	median	min	max
force_x	6601	3.73	5.52	0.06	-3.26	26.59
force_y	6601	-1.47	8.85	-0.32	-36.30	23.67
force_z	6601	-5.40	1.86	-5.71	-14.40	-0.04
torque_x	6601	-0.03	0.11	0.00	-0.33	0.41
torque_y	6601	0.05	0.12	0.00	-0.17	0.39
torque_z	6601	0.41	1.34	0.38	-13.74	10.86
surge_vel	6601	0.04	0.06	0.01	-0.05	0.21
sway_vel	6601	-0.01	0.08	0.00	-0.29	0.20
heave_vel	6601	0.00	0.01	-0.00	-0.04	0.05
roll_vel	6601	0.00	0.02	-0.00	-0.13	0.12
pitch_vel	6601	0.00	0.02	0.00	-0.09	0.07
yaw_vel	6601	-0.01	0.17	0.00	-0.83	0.60
position_x	6601	-0.61	1.03	-0.99	-1.61	2.33
position_y	6601	-0.17	0.24	-0.04	-0.98	0.22
position_Z	6601	0.45	0.05	0.44	0.31	0.60
roll	6601	0.04	0.05	0.02	-0.04	0.14
pitch	6601	0.03	0.04	0.03	-0.04	0.18
yaw	6601	0.06	2.10	1.20	-3.90	3.36

Table 7.3: Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in sway-1

	N	mean	std	median	min	max
force_x	2629.00	0.13	2.86	-0.22	-5.30	24.68
force_y	2629.00	-0.70	1.25	-0.30	-7.17	5.00
force_z	2629.00	-5.89	6.87	-6.39	-42.10	32.46
torque_x	2629.00	-0.25	0.14	-0.30	-0.64	0.16
torque_y	2629.00	-0.01	0.04	-0.01	-0.22	0.13
torque_z	2629.00	0.16	1.26	0.06	-6.58	10.54
surge_vel	2629.00	0.00	0.03	0.00	-0.14	0.22
sway_vel	2629.00	-0.00	0.02	-0.00	-0.14	0.14
heave_vel	2629.00	-0.00	0.06	-0.01	-0.14	0.18
roll_vel	2629.00	-0.00	0.03	-0.00	-0.14	0.15
pitch_vel	2629.00	0.00	0.01	0.00	-0.05	0.10
yaw_vel	2629.00	-0.02	0.09	-0.00	-0.73	0.38
position_x	2629.00	0.05	0.15	0.01	-0.03	0.86
position_y	2629.00	-0.02	0.14	-0.00	-0.72	0.23
position_Z	2629.00	0.34	0.27	0.33	-0.11	0.79
roll	2629.00	0.08	0.03	0.08	-0.03	0.18
pitch	2629.00	0.00	0.02	-0.00	-0.04	0.08
yaw	2629.00	-6.13	0.77	-6.30	-6.49	-2.14

Table 7.4: Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in heave-1

7.2 Measurement Quality Checks

Quality checks are performed on the main DOFs of measurements in tests *surge-1*, *sway-1* and *heave-1*. The checks are plotted such that the measured timeseries is plotted together with the integrated or deviated timeseries in the top plot, and the error between them is plotted below. In cases where angles and checked, both the normalized and the continuous angle are plotted. Normalization is done st. every angle is $[-180, 180)$. Figure 7.5, 7.6 and 7.7 show checks for surge position, surge velcity and yaw angle for the surge test. Figure 7.8, 7.9 and 7.10 show position along x-axis of the inertial frame, sway velocity and yaw angle for the sway test. Figure 7.11 and 7.12 show heave position and heave velocity for the heave test.

Overall the checks show small errors in most cases. The deviated checks are very smooth, seldom containing spikes. Noticable exceptions are in figure 7.6 at the five minute mark and after the eleventh minute. Figure 7.11 indicates that DVL measurements in heave velocity have a bias towards upwards velocity. Figure 7.12 shows unrealistic DVL measurements at the 30 seconds mark as well as the 1 minute 20 seconds mark. Figure 7.9 shows strange DVL measurements in sway at 3.5, 4.5 and 10 minutes, noticeably all occurring when the acceleration quickly changes from high to low. Figure 7.7 indicates a slight bias towards lower yaw angles.

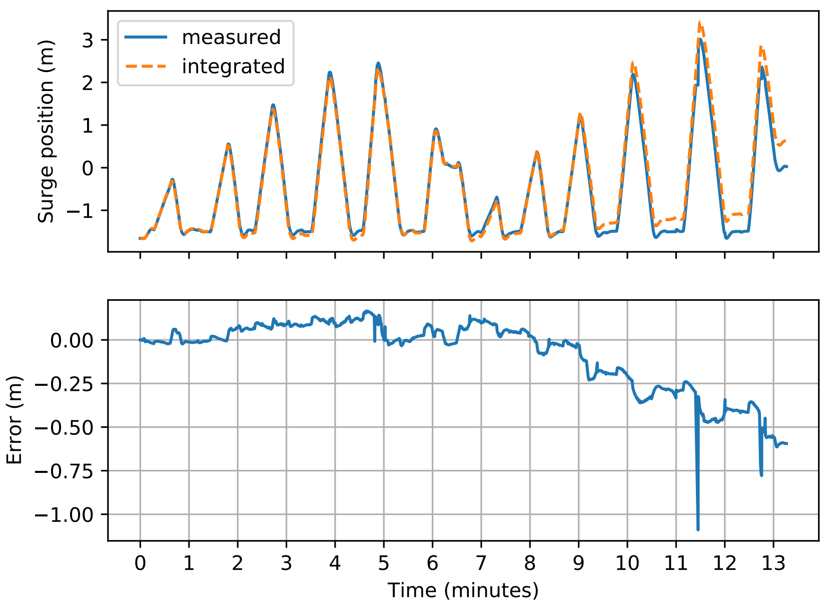


Figure 7.5: Integration check for surge position in surge test

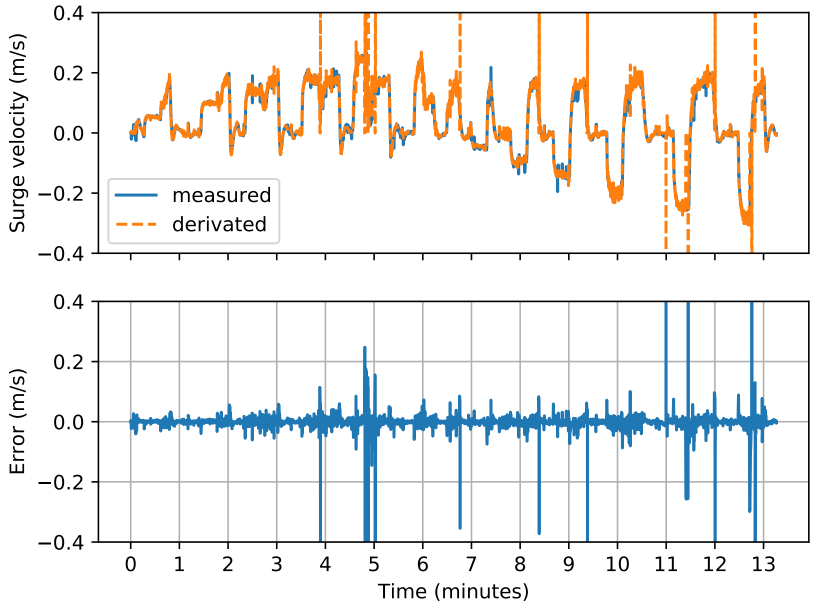


Figure 7.6: Derivation check for surge velocity in surge test

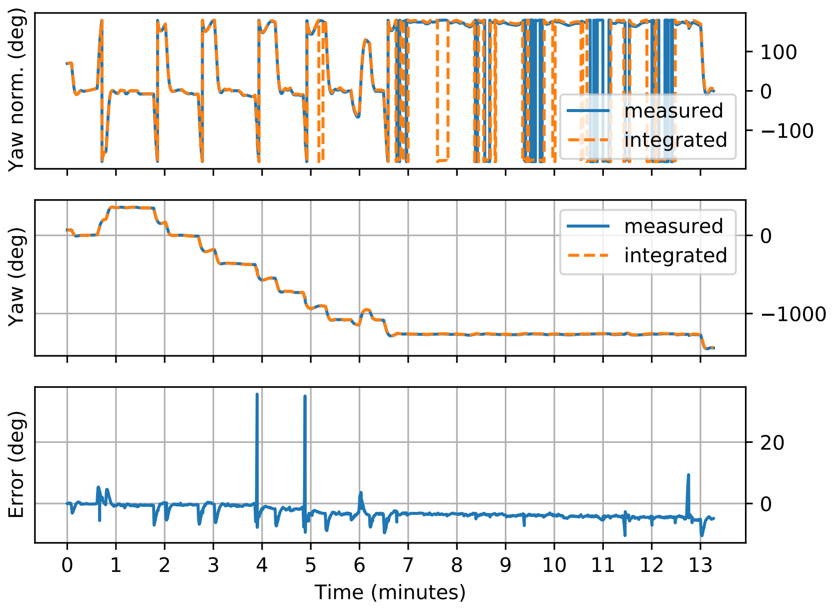


Figure 7.7: Integration check for yaw angle in surge test

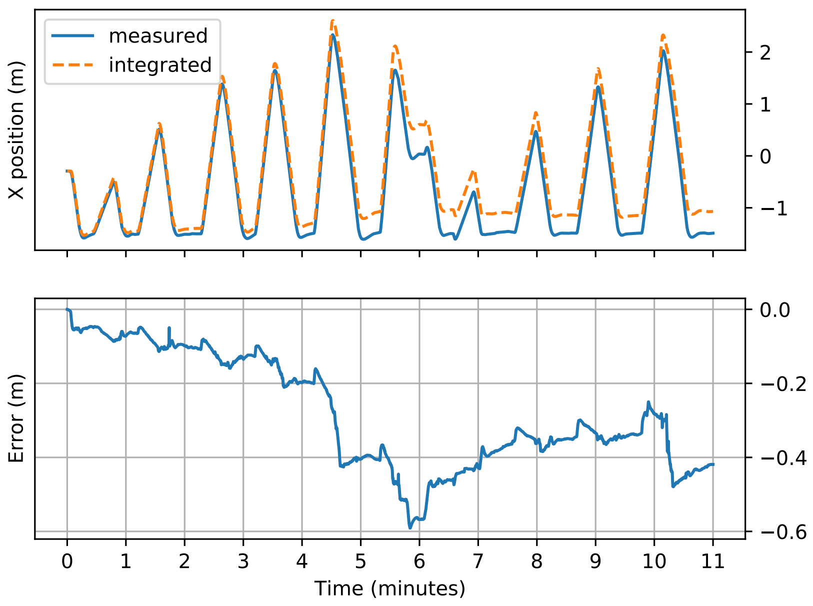


Figure 7.8: Integration check for sway position in sway test

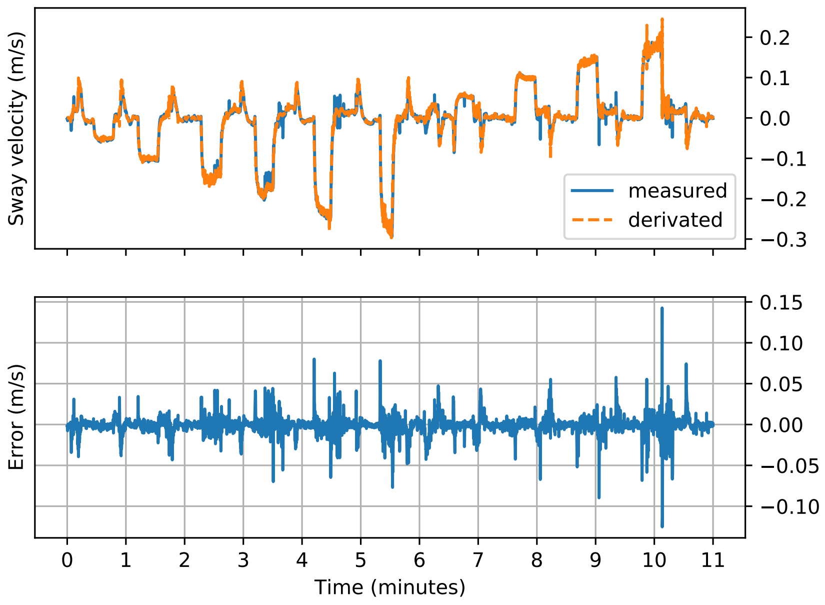


Figure 7.9: Derivation check for sway velocity in sway test

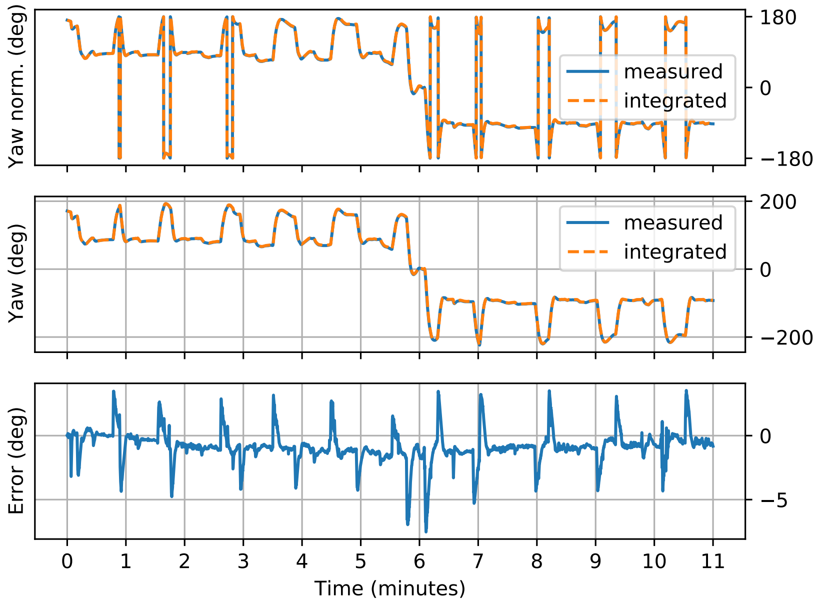


Figure 7.10: Integration check for yaw angle in sway test

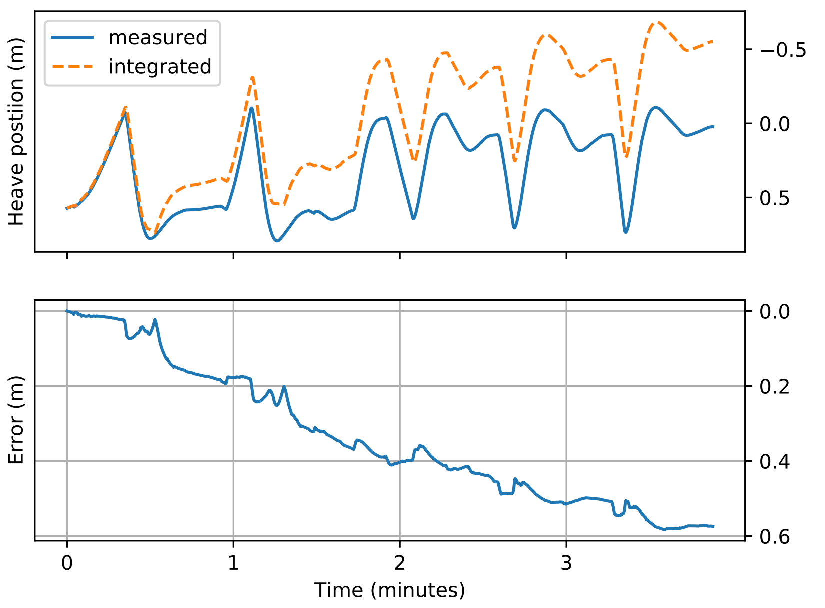


Figure 7.11: Integration check for heave position in heave tests

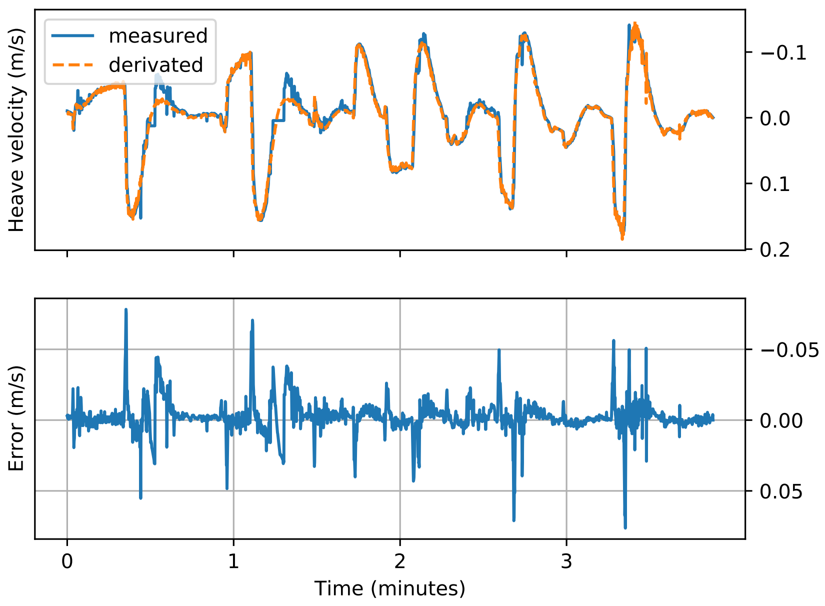


Figure 7.12: Derivation check for heave velocity in heave tests

7.3 Identification on Surge Models using Simulated Measurements

Simulated measurements using a chosen $\boldsymbol{\theta}^*$ are used to test the identification scheme (chapter 6) for the linear and non-linear surge models (section 2.3) and two different inputs $\boldsymbol{\tau}$. Table 7.6 presents the estimated parameters $\hat{\boldsymbol{\theta}}$ compared to the true parameters $\boldsymbol{\theta}^*$. Figure 7.13 and 7.14 show the fit of predictions $\hat{\mathrm{y}}$ to the simulated y for the linear surge model and inputs $\boldsymbol{\tau} = X_{triple\_ramp}$ and $\boldsymbol{\tau} = X_{many\_sine}$ (from section 6.3). Figure 7.15 shows the same for the non-linear model and input $\boldsymbol{\tau} = X_{triple\_ramp}$. Figure 7.16 and 7.17 show results from two different runs using the non-linear model and $\boldsymbol{\tau} = X_{many\_sine}$ as input. All estimates represents open-loop (no controller) response and use the knee point chosen from the final non-dominated set.

Both estimates using the linear model are very good and find the exact parameters that where used in the simulation. The non-linear model is clearly more difficult to estimate. None of the identifications where able to find $\boldsymbol{\theta}^*$. Run 1 using $\boldsymbol{\tau} = X_{many\_sine}$ performed best, finding the correct damping parameters $\boldsymbol{\theta}^*_d$ and $\boldsymbol{\theta}^*_{d\_nl}$, but missing the mass $\boldsymbol{\theta}^*_m$ by $36kg$. Even if the correct $\boldsymbol{\theta}^*$ was not found, the predictions $\hat{\mathrm{y}}$ produced by the different $\hat{\boldsymbol{\theta}}$ are visually very close to the true y. The difference in $\hat{\boldsymbol{\theta}}$ between non-linear many-sine runs 1 and 2 is worth noting.

Model and $\boldsymbol{\tau}$	$\boldsymbol{\theta}^*_m$	$\hat{\boldsymbol{\theta}}_m$	$\boldsymbol{\theta}^*_d$	$\hat{\boldsymbol{\theta}}_d$	$\boldsymbol{\theta}^*_{d\_nl}$	$\hat{\boldsymbol{\theta}}_{d\_nl}$
Linear triple ramp	60	59.999	20	20.000	-	-
Linear many sine	60	60.122	20	19.998	-	-
Non-linear triple ramp	60	94.841	20	15.872	2	4.911
Non-linear many sine, run 1	60	96.101	20	20.536	2	2.0486
Non-linear many sine, run 2	60	114.904	20	25.534	2	0.679

Table 7.5: Chosen and estimated parameters for different models and $\boldsymbol{\tau}$

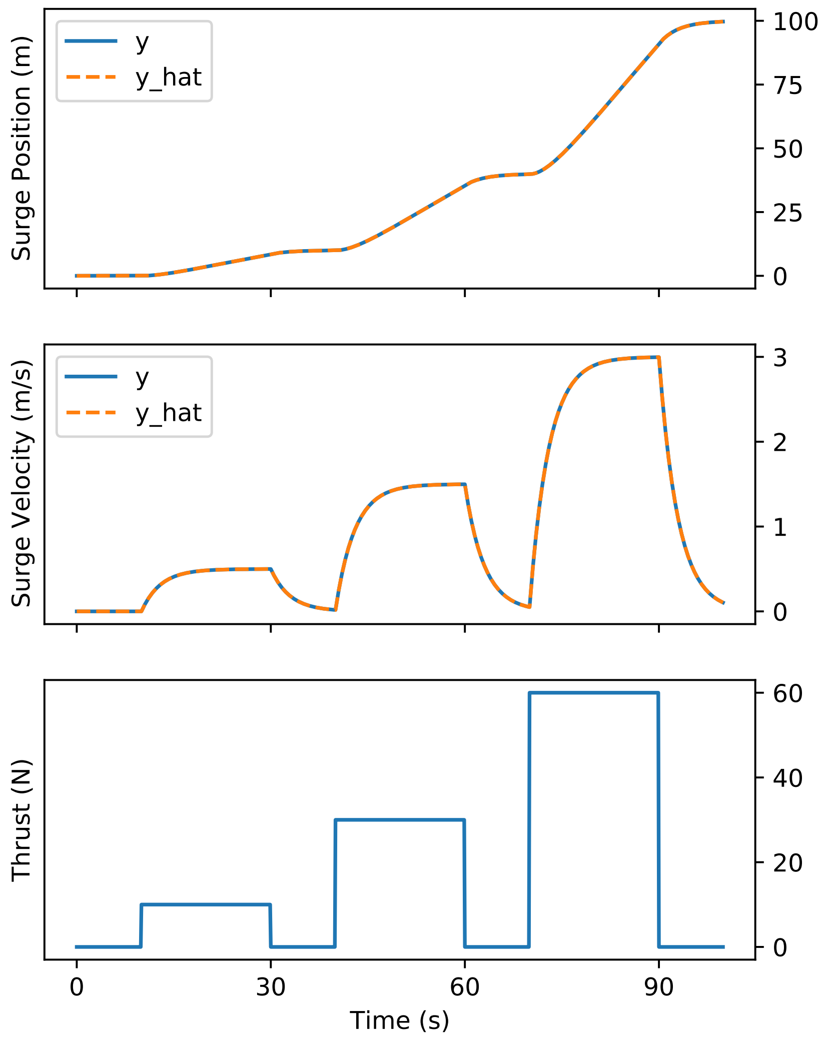


Figure 7.13: Comparison between simulated $\mathbf{y}$ and estimated $\hat{\mathbf{y}}$ using the linear surge model and $\boldsymbol{\tau} = X_{triple\_ramp}$

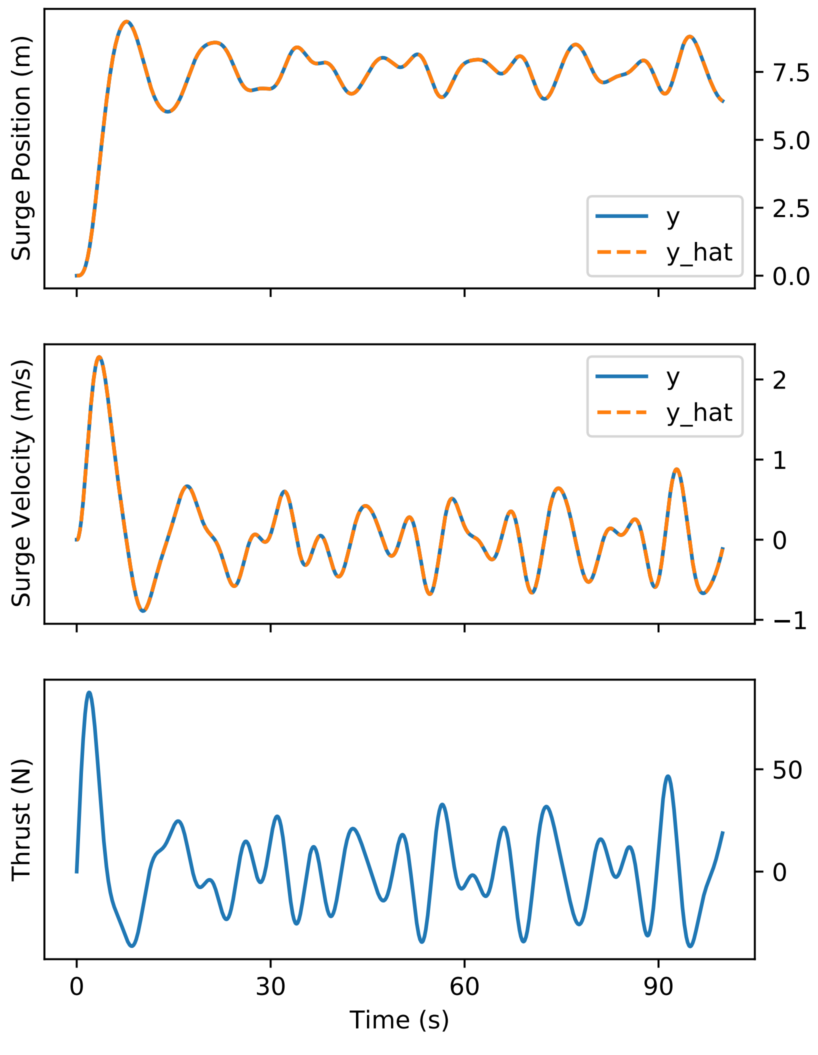


Figure 7.14: Comparison between simulated $\mathbf{y}$ and estimated $\hat{\mathbf{y}}$ using the linear surge model and $\boldsymbol{\tau} = X_{many\_sine}$

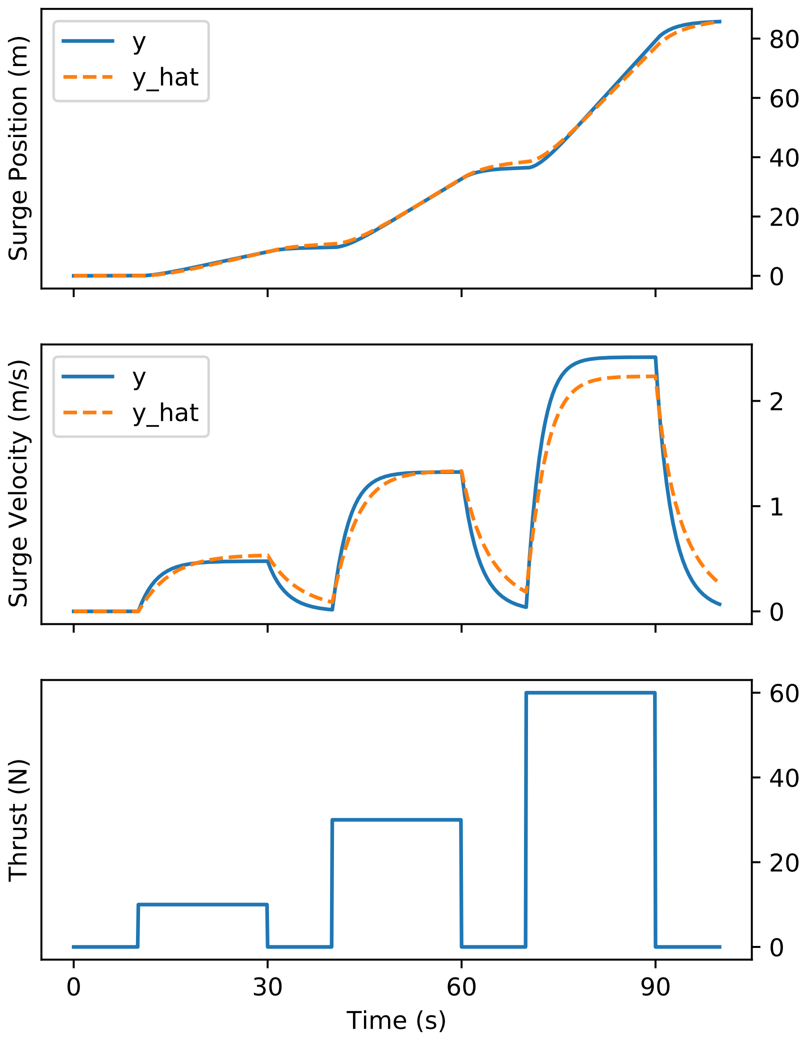


Figure 7.15: Comparison from between simulated y and estimated ŷ using the non-linear surge model and $\boldsymbol{\tau} = X_{triple\_ramp}$

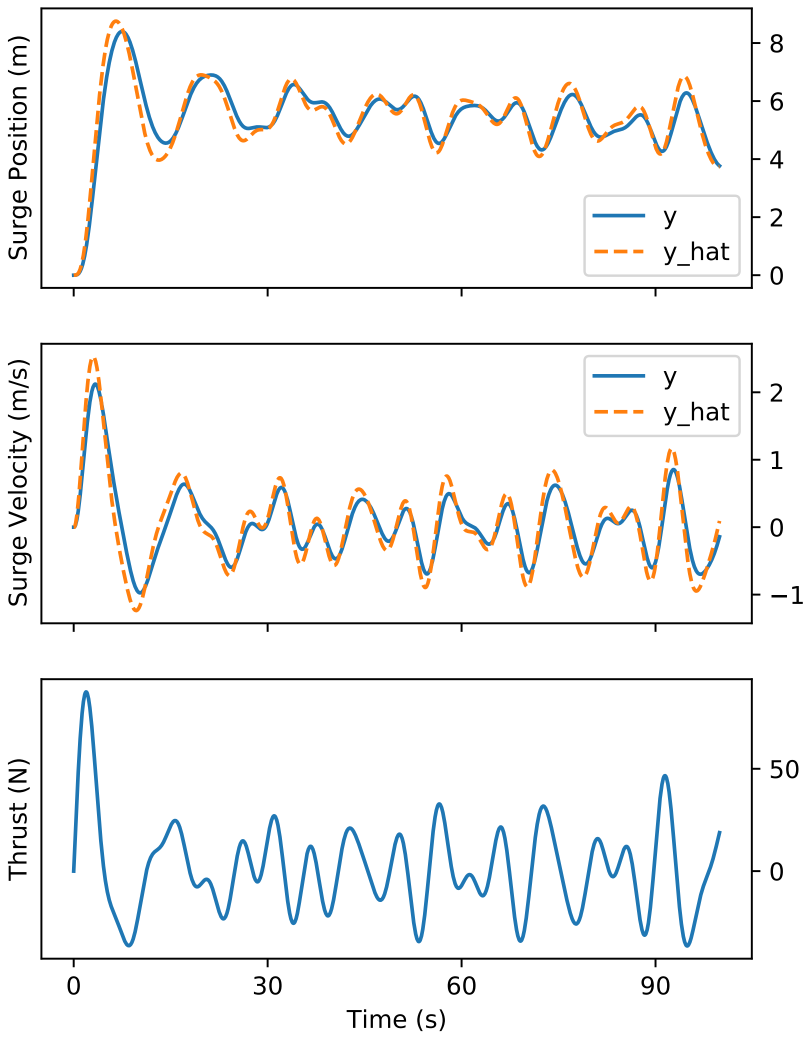


Figure 7.16: Comparison from between simulated $\mathbf{y}$ and estimated $\hat{\mathbf{y}}$ using the non-linear surge model and $\boldsymbol{\tau} = X_{many\_sine}$, run 1

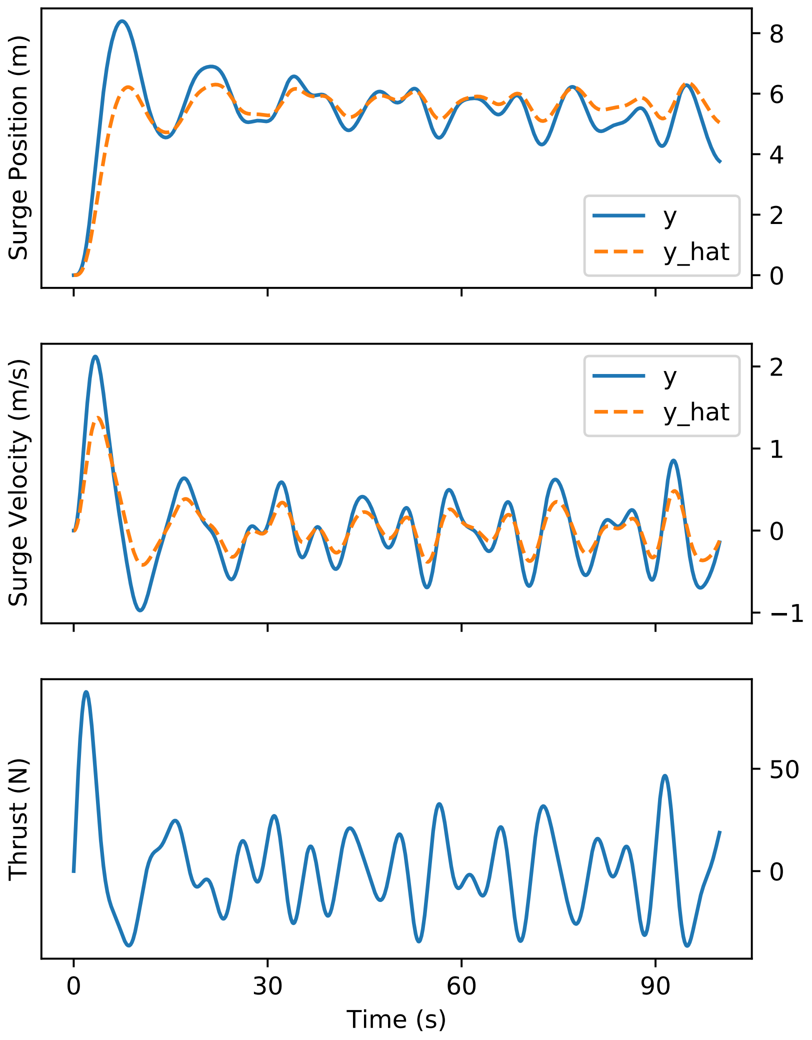


Figure 7.17: Comparison from between simulated y and estimated ŷ using the non-linear surge model and $\boldsymbol{\tau} = X_{many\_sine}$, run 2

7.4 Identification on Surge Models using Real Measurements

Measurements from the surge tests are used to estimated Belugas hydrodynamic parameters for the linear and non-linear surge models from section 2.3. Only measurements between the 7. and 11. minute are used, s.t. no rotations or bad measurements are included. Figure 7.18 and 7.19 show a comparison between measurements $\mathbf{y}$ and model predictions $\hat{\mathbf{y}}$ for the linear and non-linear models respectively. The corresponding estimated $\hat{\boldsymbol{\theta}}$ are shown in table 7.6. In each case the knee point is used to chose $\hat{\boldsymbol{\theta}}$ from the non-dominated set.

The linear and non-linear estimates gave quite different $\hat{\boldsymbol{\theta}}_m$, but rather similar $\hat{\boldsymbol{\theta}}_d$. It is clear from figure 7.18 and 7.19 that both model predictions $\hat{\mathbf{y}}$ have similar shapes that follow the topology of $\mathbf{y}$. Predictions in velocity are more box-shaped than the measurements, and quite similar the input thrust $\boldsymbol{\tau}$. Velocity estimates of the non-linear model estimate have more rounded edges, making them slightly more similar to the measurements. Both estimates have very good fit at velocities close to zero and worse fit at higher velocities.

Model	$\hat{\boldsymbol{\theta}}_m$	$\hat{\boldsymbol{\theta}}_d$	$\hat{\boldsymbol{\theta}}_{d\_nl}$
Linear surge model	26.949	115.898	-
Non-linear surge model	96.869	118.187	0.867

Table 7.6: Estimated parameters from Beluga measurements using linear and non-linear surge models. The true parameters are unknown.

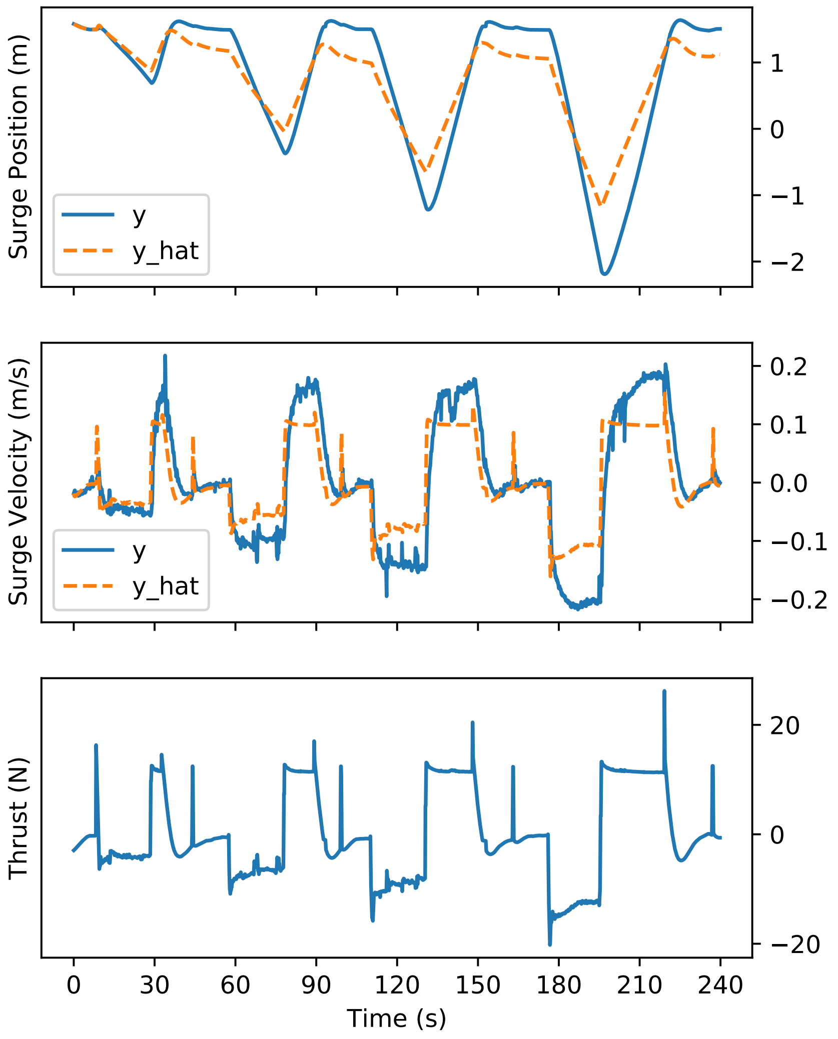


Figure 7.18: Fit between measured position and velocity in surge together with model predictions with linear surge model

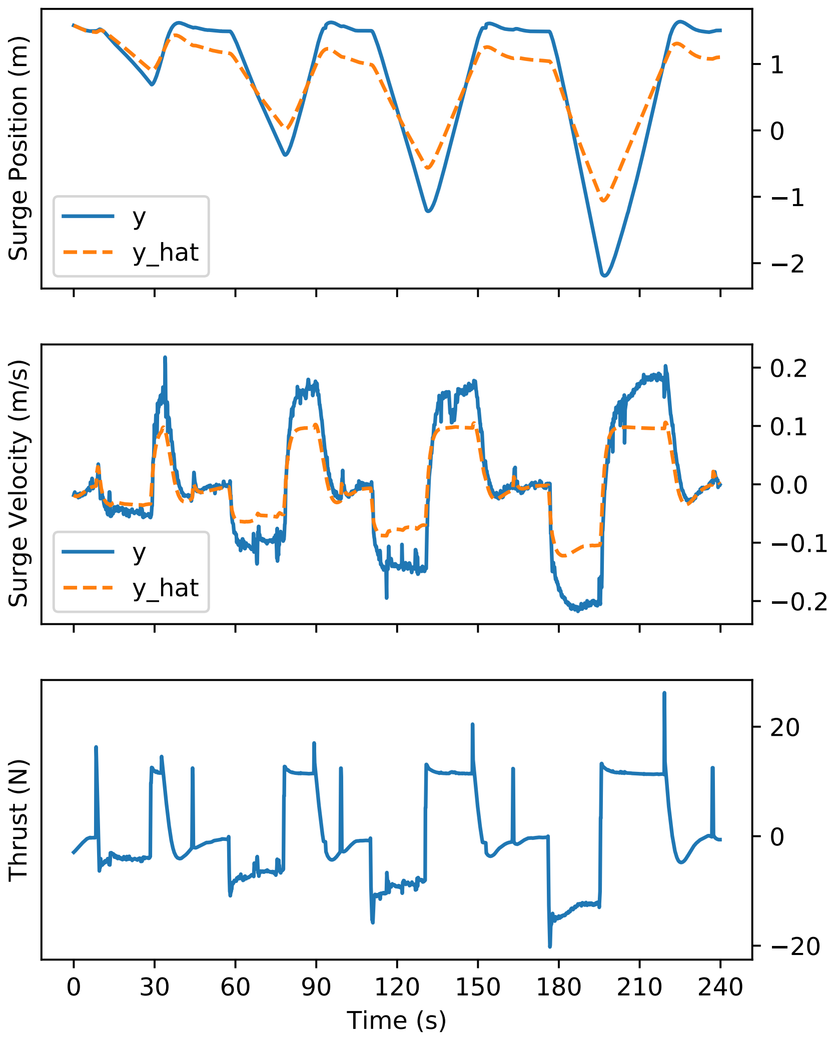


Figure 7.19: Fit between measured position and velocity in surge together with model predictions with non-linear surge model

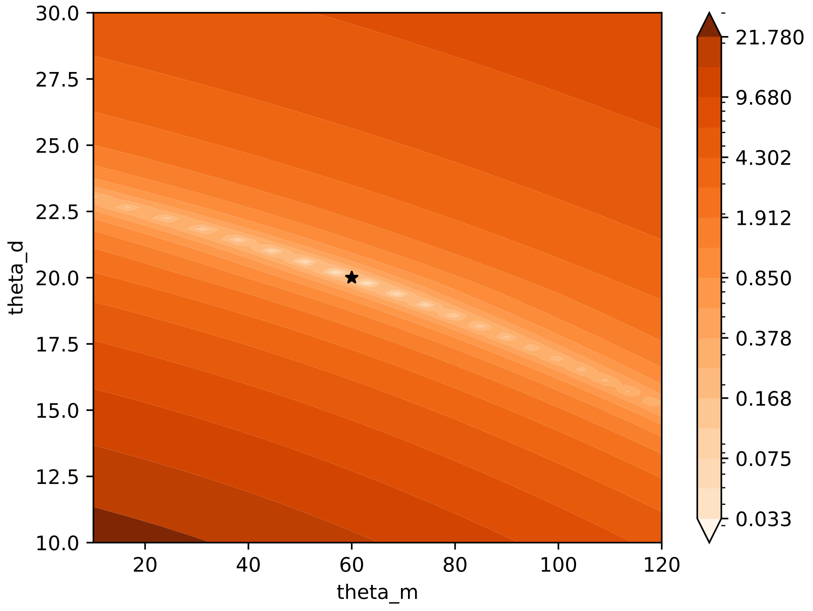
7.5 Visualization of the Objective Function

Figure 7.20 visualizes the topology of $\mathbf{f}_{EoM}$ for the two dimensional linear surge model and for two inputs X_{single_ramp} and X_{triple_ramp} (inputs explained in figure 6.1). The EoM of the linear surge model is used since it is easy to visualize thanks to only having two parameters, mass $\boldsymbol{\theta}_m$ and damping $\boldsymbol{\theta}_d$. $\boldsymbol{\theta}_m$ is plotted along the x-axis and $\boldsymbol{\theta}_d$ along the y-axis. The color of the plot is a gradient given by

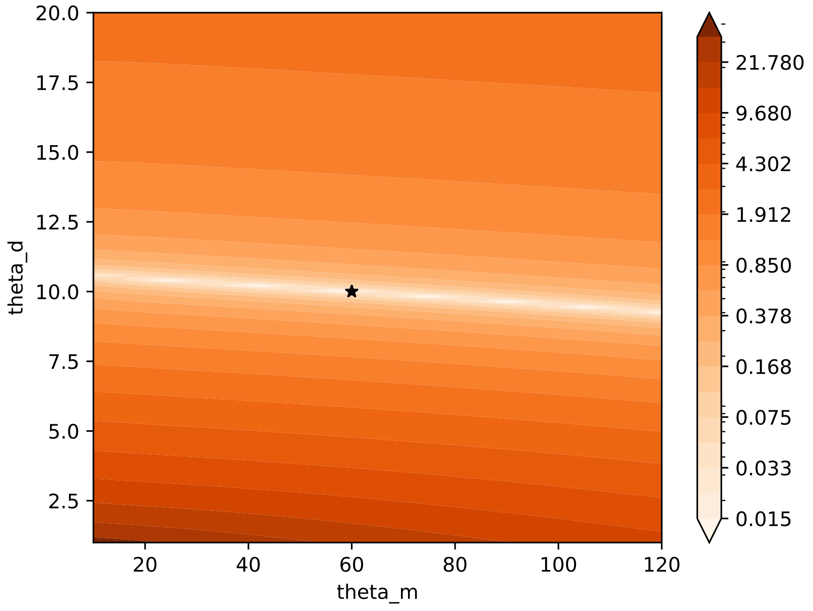
$$z = \sum_{i=1}^{m} \mathbf{f}_i \tag{7.1}$$

where m is the number of objectives in $\mathbf{f}_{EoM}$, $\mathbf{f}$ is given by equation 6.4 and the subscript EoM is removed for notational brevity. Both simulations are made using $\theta_m^* = 60$ and $\theta_d^* = 20$.

There is a clear line on which all the lowest objectve values lie. The line is however not even, but instead contains many small islands of local minima. $\boldsymbol{\tau} = X_{single\_sine}$ prodces a more horizontal line in $\mathbf{f}_{EoM}$ compared to $\boldsymbol{\tau} = X_{single\_ramp}$, indicating that the former causes $\mathbf{f}_{EoM}$ to be less sensitive to different masses $\boldsymbol{\theta}_m$.



(a) $\boldsymbol{\tau} = X_{single\_ramp}$



(b) $\boldsymbol{\tau} = X_{single\_sine}$

Figure 7.20: Topology of $\mathbf{f}_{EoM}$ around $\boldsymbol{\theta}^*$ for different $\boldsymbol{\tau}$. $\boldsymbol{\theta}^*$ is marked with a black star

7.6 Computation Time Analysis

Figure 7.21 shows how many seconds it took to compute $\hat{y}_{EoM}$ n times. EoF from *linear surge model* are used and the length of each $\hat{y}_{EoM}$ is 5 minutes, i.e. 3000 timesteps. The computation is performed without JIT, with JIT and with JIT and parallelization. Special care was taken to ensure that no JIT compiled code was cached between tests.

JIT gave substantial performance benefits, while parallelized code actually performed worse than non-parallelized code. Computation without JIT performes better for few repetitions, but is significantly slower for 100 or more repetitions.

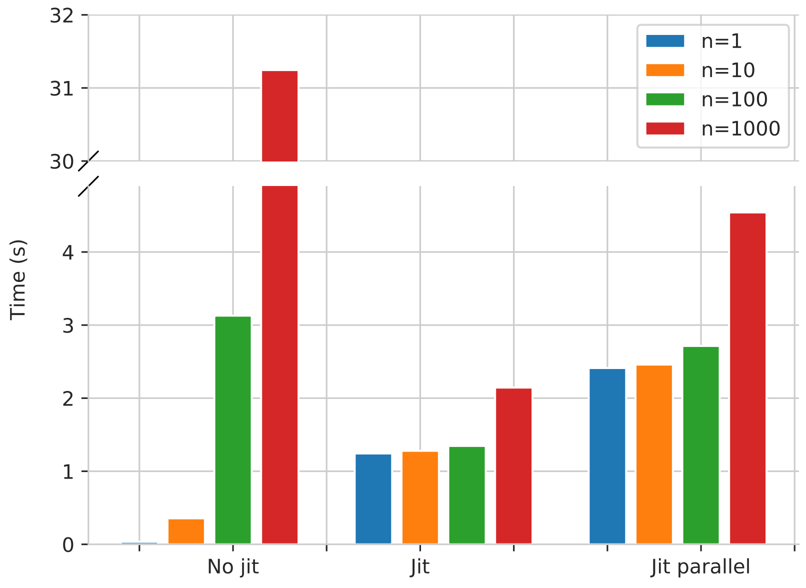


Figure 7.21: Computation time of n calls of `predict`

Chapter 8

Discussion

8.1 Beluga Measurements

The measurements cover most DOFs in isolation, as well as commonly coupled terms. The only exceptions are tests requiring roll and pitch velocity to be controlled. These are omitted due to the controllers struggle with maintaining constant velocities in these DOFs. The struggle is caused by Beluga being very stable around the x- and y-axis, forcing the controller to give large inputs which must vary greatly and even change direction depending on how Beluga is rotated.

The very smooth positional derivatives indicate high quality position measurements with almost no noise. The fact that all positional derivatives are usable, and sometimes even more reasonable than velocities directly measured with other sensors, is quite remarkable. This is well illustrated in figure 7.9 and 7.12, where the derivatives are smoother than the velocity measurements in some places.

Figure 7.11 indicates a measurement bias in heave velocity. Since there is apparent bias visible in surge and sway velocity from figures 7.5 and 7.8, this cannot be caused be a sensor misalignment and is more likely caused by a bad configuration of the DVL, such as for example a bad configuration of the gravitational constant.

The DVL measurements behave strange during sudden changes in heave and sway velocity in figures 7.12 and 7.9. At 30 seconds and 1 min 20 seconds in figure 7.12 DVL measurements freeze for a few seconds. The strange measurements might be caused by the DVL operating near the lower limit of its operating altitude.

Figure 7.7 and 7.10 both indicate a bias in measured angular yaw velocity. In the sway tests the direction of drift changes at the six minute mark. This is because Beluga starts moving in the opposite direction. Two possible causes for this error are a) a misalignment of the IMU around Belugas x-axis or y-axis or b) a poor calibration of the IMU. A possible solution is to estimate the misalignment and correct for it in post-processing by doing a static transform on all IMU measurements.

8.2 SI Using Simulated Measurements

NSGA-II performs well for linear two dimensions and estimates parameters $\hat{\boldsymbol{\theta}}$ that are almost identical to those used for simulating measurements. This is illustrated by the perfect match between $\mathbf{y}$ and $\hat{\mathbf{y}}$ for the two dimensional linear models in figure 7.14 and the closeness between $\hat{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}^*$ for the linear models in table 7.5. The algorithm is clearly able to avoid local minima and converge to the true pareto frontier.

The situation changes when a non-linear third dimension is included. This is illustrated by the imperfect fit for the three dimensional non-linear models in figures 7.13, 7.16 and 7.17, as well as the differences between $\hat{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}^*$ in table 7.5. $\hat{\boldsymbol{\theta}}$ for the non-linear models is approximately 30% off in mass for both $\boldsymbol{\tau}$ and 25% off in damping for $\boldsymbol{\tau} = X_{triple\_ramp}$.

A common denominator for all $\hat{\boldsymbol{\theta}}$ in table 7.5 is their tendency to have better estimates for damping than mass. This can be explained by the objective functions higher sensitivity to damping than mass, as pointed out in section 7.5.

Even if the estimated parameters $\hat{\boldsymbol{\theta}}$ from the non-linear models are not fully equal to the true parameters $\boldsymbol{\theta}^*$, the visual fit in figures 7.13, 7.16 and 7.17 is still quite good. Figure 7.20 shows that, for two dimensional models, there are lines in $\mathbf{f}_{EoM}$ along which both objective values are low. This implies that parameter estimates along these lines will give a good prediction fit, even if the parameter estimates are far off the true values. It seems reasonable that this phenomena may extend to higher dimensions. In that case it would be a fundamental challenge to the method presented in this thesis, as there would clearly be many design points that can produce similar predictions to those of the true parameters. In other words, there might be many local minima present in $\mathbf{f}_{EoM}$ that correspond to small objective values. A possible explanation for the struggles with higher non-linear dimensions is thus, that NSGA-II converges to non-dominated sets of local minima instead of the true pareto front.

The local minima explanation is strengthened by the different estimates between run 1 and 2 of the non-linear many-sine estimations in table 7.5 and figures 7.16 and 7.17.

Since the initial population for the genetic algorithm is selected randomly, it can find different local minima from run to run. For successful optimization it depends on at least one design point being generated in proximity of the true $\boldsymbol{\theta}$ during initialization, or having a random mutation that achieves the same in one of its generations. This condition is clearly fulfilled for the two dimensional linear model, but the introduction of more design dimensions will likely lower the chances of this happening significantly. A noteworthy observation is that if the algorithm is able to find the correct parameters, it almost always does. However, if it is not able to do so, its result tends to vary from one run to another.

Issues with fitting measurements to non-linear state space models are nothing new. Kantas et al. (2015) points out that non-linear state-space models are notoriously difficult to fit data to.

A better result might be achieved by helping the algorithm get out of, or pass, local minima. This can be achieved by using a larger population size or increasing the mutation rate. Limiting factors for the former are memory size and computation time. On the utilized computer this constrained the population size to 10 000, since it only has 8 GB of RAM. Another option is to run the optimization multiple times, and select the solution which achieved the best fit. This increases the odds of finding a good estimate, because the initial population of NSGA-II is chosen randomly.

8.3 SI Using Real Measurements

The estimation scheme is able to find parameters using the proposed method. It is clear that both estimated linear and non-linear model predictions follow the measurements topology, and that the estimated parameters are partly able to describe the true behaviour. However, neither of the estimated models are able to precisely describe the measurements. This indicates that either the model structure is unfit to describe the physical behaviour of Beluga, or the estimated parameters are not ideal.

The model structures used to create predictions $\hat{\mathbf{y}}$ assume that Beluga only moves one dimension, but Beluga has six degrees of freedom in the measurements. The overview of the measurements from the surge test (fig. 7.2) show that there is still some movement in yaw during the minutes used for estimation, even if the controller keeps it stable. The means that Beluga does not travel in a perfect straight line, which the model assumes it does. This deviation will certainly introduce some error, but since both the linear and the non-linear model are able to mimic the measurements topology, is seems unlikely that the problem is entirely caused by the faults in the model structure.

NSGA-II might face the same problems with regards to local minima as in the simulation based tests from section 8.2. There are certainly similarities between the estimates made based on Belugas measurements to the findings from estimations using simulated measurements. In both cases $\hat{\mathbf{y}}$ has the same topology as $\mathbf{y}$, but fails to get a good fit at negative and positive peaks. The similarity suggests that NSGA-II has the same problem for real measurements as for simulated measurements, where it converges to a non-dominated set with design points in local minima instead of the true pareto set.

The task of estimation of hydrodynamic coefficients for UUVs is difficult and the author has found any reference to a solution to this. Estimates involve a high degree of uncertainty no matter which method is used, and it is common that parameters have up to 50% errors even for expensive experimental methods (Eidsvik and Schjølberg (2016)). System identification has shown promise as being more efficient for solving the problem, but many of its newer methods do not yet live up to the expectations (Kantas et al. (2015)). Belugas complicated shape makes estimation using analytical, empirical and numeric methods difficult, which leaves experimental identification and by system identification as options.

The fact that added mass and damping is frequency dependent adds considerable error to the estimates, since it is neglected by the models from Fossen (2011). This inherent simplification of the physics of the vehicles environment make errors added mass estimates less impacting, as the model structure is not aiming for perfect description of the physics to begin with. The frequency dependency also has to be considered in the way inputs are given to the UUV, as excitation at different frequencies will produce different coefficients for added mass and damping.

8.4 Computational Performance

The substantial performance benefits for JIT are expected, as the algorithm involves many numerical operations and nested loops. Pythons performance in these cases is known to be poor, and Numba advertises their significant performance gains in these exact areas. The much slower performance of JIT compared to no-JIT for 1 to 10 executions is caused by the time taken to compile during the first execution. After the code has compiled, the lower computational load for each successive execution becomes apparent.

Overhead of parallelization outweighs its benefits in this case. Creating threads is computationally demanding, and in this case the performance benefits from having multiple threads calculating objective functions are not enough to make up for the time lost in thread creation. Parallelization would have performed better if a) the objective function was computationally more demanding or 2) a set pool of threads was used during optimization, removing the need to create new threads for every individual in the population.

Chapter 9

Conclusion and Further Work

9.1 Conclusion

In this thesis the parameters of an UUV where estimated using a machine learning approach. A literature review was conducted on parameter estimation methods applied to UUVs, with emphasis on methods that utilize machine learning techniques. An experiment was conducted to gather full-state measurements of an UUV during various maneuvers. The measurements quality was investigated and discussed. A system identification scheme using prediction errors and a genetic algorithm for multi-objective optimization was used to estimate UUV parameters. The scheme was applied to both simulated and real measurements of a UUV. Parameters where found in both cases and their correctness was discussed with regards to accuracy and computational performance.

9.2 Proposal for Further Work

There are several interesting directions to explore for further work. One of them is the use of different optimization methods. NSGA-III is a development of NSGA-II meant to perform better for higher dimensional optimization problems (Deb and Jain (2014)). Some papers point out that NSGA-III in many cases has issues with convergence and propose improved versions. Two examples of this are Yuan et al. (2014) and Cui et al. (2019), which both focus on making better tradeoffs between convergence and diversity.

Another direction is to further investigate the properties of the proposed method. It would be interesting to visualize the effect non-linearities have the objective function, and how much complexity it adds to compared to the addition of new design dimensions. The SI methods robustness can be investigated by adding noise and bias to the simulated measurements. Using larger population sizes should increase the chances of the optimization converging to the global mimimum. This can be tested by moving the computation to cloud-based services, where high-end CPUs and large memories remove the bottlenecks present in consumer grade PCs.

The measurements gathered from Beluga UUV can also be used as data for entirely different system identification approaches. Many traditional methods exists, such as the extended Kalman filter or the least squares method. Other interesting approaches could be kernel-based, such as the one by Zhang and Zou (2013).

Appendix A

Statistical Describtions

Table A.1: Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in yaw-1

	N	mean	std	median	min	max
force_x	6597.00	-0.25	2.34	-0.43	-6.06	26.84
force_y	6597.00	-0.58	1.12	-0.46	-6.62	6.19
force_z	6597.00	-6.52	2.37	-6.91	-14.81	7.51
torque_x	6597.00	-0.17	0.17	-0.25	-0.55	0.44
torque_y	6597.00	-0.01	0.05	-0.02	-0.23	0.32
torque_z	6597.00	0.17	2.57	0.23	-17.43	13.77
surge_vel	6597.00	0.00	0.02	-0.00	-0.08	0.20
sway_vel	6597.00	-0.00	0.01	-0.00	-0.07	0.09
heave_vel	6597.00	-0.00	0.02	-0.00	-0.04	0.07
roll_vel	6597.00	-0.00	0.02	-0.00	-0.10	0.10
pitch_vel	6597.00	0.00	0.02	0.00	-0.05	0.10
yaw_vel	6597.00	-0.01	0.27	0.00	-0.80	0.78
position_x	6597.00	0.01	0.20	0.03	-1.60	0.19
position_y	6597.00	-0.00	0.05	0.00	-0.40	0.10
position_Z	6597.00	0.46	0.08	0.46	-0.00	0.60
roll	6597.00	0.07	0.04	0.08	-0.06	0.17
pitch	6597.00	0.02	0.04	0.00	-0.08	0.15
yaw	6597.00	-4.73	6.65	-6.34	-15.69	8.02

Table A.2: Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in heave_surge-1

	N	mean	std	median	min	max
force_x	8596.00	2.31	7.78	-0.28	-20.57	28.83
force_y	8596.00	-0.13	1.58	-0.09	-8.05	10.46
force_z	8596.00	-5.54	6.99	-6.42	-41.23	20.04
torque_x	8596.00	-0.10	0.17	0.00	-0.64	0.33
torque_y	8596.00	0.06	0.15	0.00	-0.45	0.54
torque_z	8596.00	0.28	1.74	0.27	-17.03	17.77
surge_vel	8596.00	0.03	0.08	0.00	-0.21	0.26
sway_vel	8596.00	0.00	0.02	0.00	-0.10	0.15
heave_vel	8596.00	-0.00	0.05	-0.01	-0.13	0.17
roll_vel	8596.00	-0.00	0.03	-0.00	-0.16	0.24
pitch_vel	8596.00	0.00	0.02	0.00	-0.09	0.14
yaw_vel	8596.00	-0.01	0.18	0.00	-0.93	0.82
position_x	8596.00	-1.03	0.70	-1.43	-1.65	1.88
position_y	8596.00	-0.04	0.14	-0.01	-0.66	0.36
position_Z	8596.00	0.38	0.23	0.42	-0.14	0.79
roll	8596.00	0.05	0.04	0.05	-0.11	0.23
pitch	8596.00	0.04	0.05	0.04	-0.08	0.23
yaw	8596.00	-6.76	5.25	-9.47	-12.99	6.33

Table A.3: Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in surge_sway-1

	N	mean	std	median	min	max
force_x	11753.00	5.19	7.55	3.90	-25.74	32.04
force_y	11753.00	-0.96	8.44	-0.26	-36.71	36.55
force_z	11753.00	-5.24	1.78	-5.17	-16.12	-0.00
torque_x	11753.00	-0.08	0.16	0.00	-0.48	0.52
torque_y	11753.00	0.03	0.14	0.00	-0.27	0.45
torque_z	11753.00	0.27	1.56	0.16	-14.55	18.22
surge_vel	11753.00	0.06	0.09	0.05	-0.25	0.26
sway_vel	11753.00	-0.00	0.07	0.00	-0.24	0.26
heave_vel	11753.00	0.00	0.01	-0.00	-0.09	0.13
roll_vel	11753.00	0.00	0.02	-0.00	-0.16	0.12
pitch_vel	11753.00	0.00	0.02	0.00	-0.10	0.16
yaw_vel	11753.00	0.00	0.19	0.00	-0.80	0.84
position_x	11753.00	-0.42	1.23	-0.87	-1.71	2.65
position_y	11753.00	-0.02	0.26	-0.01	-1.05	0.89
position_Z	11753.00	0.45	0.06	0.45	0.33	0.92
roll	11753.00	0.05	0.04	0.06	-0.07	0.15
pitch	11753.00	0.02	0.04	0.02	-0.07	0.15
yaw	11753.00	2.98	1.75	2.84	-0.17	6.32

Table A.4: Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in random-1

	N	mean	std	median	min	max
force_x	3001.00	13.21	18.21	16.14	-39.19	47.74
force_y	3001.00	0.63	23.42	0.00	-54.12	40.27
force_z	3001.00	-4.56	7.15	-5.06	-47.66	37.82
torque_x	3001.00	0.00	1.87	0.01	-24.69	24.64
torque_y	3001.00	0.04	0.85	-0.01	-2.27	13.49
torque_z	3001.00	-0.29	4.84	0.10	-29.54	19.64
surge_vel	3001.00	0.12	0.14	0.13	-0.27	0.39
sway_vel	3001.00	0.00	0.16	0.01	-0.33	0.29
heave_vel	3001.00	-0.01	0.03	-0.01	-0.19	0.18
roll_vel	3001.00	0.00	0.19	0.00	-1.94	1.95
pitch_vel	3001.00	0.01	0.07	0.00	-0.30	0.65
yaw_vel	3001.00	-0.07	0.42	-0.12	-1.33	1.46
position_x	3001.00	-0.18	0.68	-0.12	-1.85	1.62
position_y	3001.00	-0.15	0.50	-0.13	-1.52	0.87
position_Z	3001.00	0.29	0.10	0.31	-0.34	0.47
roll	3001.00	0.05	0.19	0.06	-2.18	1.40
pitch	3001.00	0.05	0.09	0.04	-1.23	0.40
yaw	3001.00	-14.12	5.64	-14.71	-23.46	-0.00

Table A.5: Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in random-2

	N	mean	std	median	min	max
force_x	5980.00	6.68	16.51	-0.09	-41.42	62.32
force_y	5980.00	1.01	12.39	0.11	-59.75	54.20
force_z	5980.00	-4.50	7.55	-4.37	-65.62	66.46
torque_x	5980.00	0.06	2.37	0.01	-26.64	26.46
torque_y	5980.00	0.18	6.53	-0.02	-25.93	28.15
torque_z	5980.00	0.43	5.48	0.02	-30.63	28.90
surge_vel	5980.00	0.08	0.13	0.05	-0.41	0.46
sway_vel	5980.00	0.02	0.08	0.01	-0.48	0.35
heave_vel	5980.00	-0.00	0.04	-0.00	-0.31	0.33
roll_vel	5980.00	-0.01	0.28	0.00	-2.39	2.14
pitch_vel	5980.00	0.02	0.22	0.01	-0.89	1.15
yaw_vel	5980.00	0.02	0.44	0.01	-1.67	1.58
position_x	5980.00	0.11	0.57	0.10	-1.14	2.15
position_y	5980.00	0.17	0.41	0.17	-1.05	1.42
position_Z	5980.00	0.21	0.13	0.22	-0.26	0.56
roll	5980.00	-3.56	3.10	-5.70	-9.10	1.98
pitch	5980.00	-0.03	0.32	-0.00	-1.53	1.55
yaw	5980.00	2.95	4.32	3.45	-9.30	12.57

Table A.6: Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in random-3

	N	mean	std	median	min	max
force_x	10065.00	4.19	16.52	-0.61	-71.02	59.74
force_y	10065.00	-0.56	16.74	-0.64	-60.75	58.94
force_z	10065.00	-5.50	7.40	-5.25	-75.01	62.10
torque_x	10065.00	0.10	2.84	0.01	-26.28	26.28
torque_y	10065.00	1.14	6.09	-0.01	-25.80	26.79
torque_z	10065.00	0.38	5.48	0.04	-30.11	29.96
surge_vel	10065.00	0.05	0.12	0.03	-0.41	0.43
sway_vel	10065.00	0.00	0.11	0.01	-0.35	0.42
heave_vel	10065.00	-0.00	0.04	0.00	-0.45	0.33
roll_vel	10065.00	0.03	0.33	0.01	-2.07	2.21
pitch_vel	10065.00	0.03	0.24	0.01	-1.09	1.41
yaw_vel	10065.00	0.02	0.40	-0.01	-1.73	1.57
position_x	10065.00	0.11	0.65	0.06	-1.70	1.95
position_y	10065.00	0.08	0.44	0.12	-1.50	1.08
position_Z	10065.00	0.26	0.10	0.26	-0.15	0.60
roll	10065.00	4.55	6.30	0.08	-6.50	20.34
pitch	10065.00	-0.02	0.27	-0.00	-1.54	1.43
yaw	10065.00	-0.31	5.82	-1.07	-11.49	19.34

Table A.7: Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in random-4

	N	mean	std	median	min	max
force_x	5981.00	6.79	15.21	0.02	-59.46	58.09
force_y	5981.00	-1.74	17.16	-0.50	-60.99	46.57
force_z	5981.00	-5.75	8.33	-5.26	-67.78	61.56
torque_x	5981.00	-0.04	1.93	0.01	-25.05	24.34
torque_y	5981.00	1.06	6.42	-0.00	-26.49	27.36
torque_z	5981.00	0.12	5.59	0.01	-30.34	29.73
surge_vel	5981.00	0.06	0.11	0.04	-0.30	0.41
sway_vel	5981.00	-0.00	0.11	0.00	-0.34	0.30
heave_vel	5981.00	-0.01	0.04	-0.00	-0.25	0.28
roll_vel	5981.00	0.03	0.27	0.01	-2.84	1.44
pitch_vel	5981.00	0.02	0.24	0.01	-1.47	1.24
yaw_vel	5981.00	-0.01	0.41	-0.02	-1.72	1.80
position_x	5981.00	0.12	0.50	0.21	-1.17	1.34
position_y	5981.00	0.17	0.34	0.18	-0.84	1.06
position_Z	5981.00	0.17	0.10	0.16	-0.13	0.58
roll	5981.00	-0.05	0.43	-0.02	-2.75	2.02
pitch	5981.00	-0.03	0.31	-0.01	-1.52	1.27
yaw	5981.00	-14.19	7.84	-18.28	-23.90	3.26

Table A.8: Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in random-5

	N	mean	std	median	min	max
force_x	3710.00	8.02	16.21	3.22	-64.99	55.08
force_y	3710.00	-4.03	15.86	-0.61	-60.06	41.72
force_z	3710.00	-8.19	5.89	-7.51	-65.73	55.15
torque_x	3710.00	-0.53	6.06	0.01	-25.05	25.05
torque_y	3710.00	-0.92	3.07	-0.03	-22.00	17.24
torque_z	3710.00	0.40	9.04	0.04	-30.82	30.79
surge_vel	3710.00	0.08	0.11	0.07	-0.45	0.45
sway_vel	3710.00	-0.01	0.09	-0.01	-0.52	0.27
heave_vel	3710.00	-0.01	0.03	-0.01	-0.20	0.20
roll_vel	3710.00	-0.06	0.60	0.00	-3.13	3.22
pitch_vel	3710.00	0.03	0.17	0.01	-0.84	1.11
yaw_vel	3710.00	0.02	0.54	-0.01	-1.91	1.74
position_x	3710.00	-0.12	0.53	-0.21	-1.30	1.22
position_y	3710.00	0.16	0.36	0.23	-1.11	0.88
position_Z	3710.00	0.10	0.11	0.10	-0.22	0.48
roll	3710.00	-13.92	11.80	-12.55	-31.69	7.31
pitch	3710.00	0.00	0.22	0.01	-1.51	1.44
yaw	3710.00	9.20	6.92	9.90	-3.26	22.07

Table A.9: Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in random-6

	N	mean	std	median	min	max
force_x	11556.00	13.55	20.34	14.11	-61.28	60.86
force_y	11556.00	-0.71	17.13	-0.04	-60.12	61.28
force_z	11556.00	-5.53	7.10	-5.91	-68.05	66.72
torque_x	11556.00	-0.14	2.62	0.01	-27.04	26.93
torque_y	11556.00	0.31	6.49	-0.01	-28.56	28.63
torque_z	11556.00	0.31	6.87	0.00	-30.14	29.99
surge_vel	11556.00	0.11	0.15	0.11	-0.56	0.63
sway_vel	11556.00	0.01	0.09	0.01	-0.32	0.34
heave_vel	11556.00	-0.00	0.04	-0.00	-0.30	0.31
roll_vel	11556.00	0.00	0.30	0.01	-3.26	2.01
pitch_vel	11556.00	0.02	0.30	0.01	-1.43	1.95
yaw_vel	11556.00	0.02	0.51	0.01	-1.83	1.83
position_x	11556.00	-0.02	0.58	0.01	-1.58	2.17
position_y	11556.00	0.17	0.46	0.16	-1.41	1.31
position_Z	11556.00	0.17	0.12	0.15	-0.47	0.53
roll	11556.00	-16.02	8.33	-12.70	-26.66	1.21
pitch	11556.00	-0.01	0.24	-0.00	-1.37	1.46
yaw	11556.00	11.14	7.21	12.75	-5.62	22.82

Table A.10: Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in random-7

	N	mean	std	median	min	max
force_x	11981.00	6.27	15.90	0.00	-60.00	60.00
force_y	11981.00	-1.40	12.47	0.00	-60.00	60.00
force_z	11981.00	-5.92	20.45	0.00	-60.00	60.00
torque_x	11981.00	0.20	3.29	0.00	-25.58	25.65
torque_y	11981.00	1.41	6.05	0.00	-25.81	27.15
torque_z	11981.00	0.48	5.32	0.00	-30.00	30.00
surge_vel	11981.00	0.06	0.11	0.03	-0.38	0.40
sway_vel	11981.00	0.00	0.07	0.00	-0.48	0.39
heave_vel	11981.00	-0.01	0.08	0.01	-0.49	0.35
roll_vel	11981.00	0.03	0.33	0.02	-2.89	3.29
pitch_vel	11981.00	0.01	0.25	0.01	-1.46	1.16
yaw_vel	11981.00	0.04	0.39	-0.00	-1.68	1.65
position_x	11981.00	0.08	0.53	0.08	-1.42	1.57
position_y	11981.00	0.20	0.33	0.24	-1.01	1.13
position_Z	11981.00	0.19	0.16	0.19	-0.38	0.70
roll	11981.00	9.02	7.63	12.42	-1.24	19.79
pitch	11981.00	-0.06	0.33	-0.03	-1.55	1.55
yaw	11981.00	24.68	12.26	25.43	-2.28	45.04

Table A.11: Statistical description of $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ in random-8

	N	mean	std	median	min	max
force_x	15056.00	8.84	19.28	0.00	-60.00	60.00
force_y	15056.00	-1.03	13.16	-0.00	-60.00	60.00
force_z	15056.00	-5.74	17.70	-0.00	-60.00	60.00
torque_x	15056.00	0.06	2.11	0.00	-25.05	25.05
torque_y	15056.00	1.17	4.86	-0.00	-25.55	26.05
torque_z	15056.00	0.34	5.88	-0.00	-30.00	30.00
surge_vel	15056.00	0.07	0.12	0.05	-0.45	0.44
sway_vel	15056.00	0.00	0.07	0.01	-0.43	0.33
heave_vel	15056.00	0.00	0.07	0.01	-0.33	0.33
roll_vel	15056.00	0.03	0.27	0.00	-1.95	2.46
pitch_vel	15056.00	0.02	0.20	0.01	-1.12	1.54
yaw_vel	15056.00	0.01	0.44	-0.00	-2.01	1.94
position_x	15056.00	0.22	0.60	0.22	-1.68	2.14
position_y	15056.00	0.05	0.37	0.04	-1.29	1.09
position_Z	15056.00	0.21	0.16	0.21	-0.51	0.67
roll	15056.00	-16.86	3.75	-18.80	-25.50	0.23
pitch	15056.00	-0.05	0.25	-0.02	-1.49	1.40
yaw	15056.00	-10.60	6.58	-12.26	-27.88	4.22

References

Blank, J. and Deb, K. (2020a). Pymoo: Multi-objective optimization in python, *IEEE Access* **8**: 89497–89509.

Blank, J. and Deb, K. (2020b). A running performance metric and termination criterion for evaluating evolutionary multi- and many-objective optimization algorithms, *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8.

Boyd, S. and Sastry, S. S. (1986). Necessary and sufficient conditions for parameter convergence in adaptive control, *Automatica* **22**(6): 629–639.

Brunton, S. L. and Kutz, J. N. (2019). *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control.*

Caccia, M., Indiveri, G. and Veruggio, G. (2000). Modeling and identification of open-frame variable configuration unmanned underwater vehicles, *IEEE Journal of Oceanic Engineering* **25**(2): 227–240.

Cardenas, P. and de Barros, E. A. (2019). Estimation of auv hydrodynamic coefficients using analytical and system identification approaches, *IEEE Journal of Oceanic Engineering* **45**(4): 1157–1176.

Chapelle, O., Vapnik, V. and Bengio, Y. (2002). Model selection for small sample regression, *Machine Learning* **48**(1): 9–23.

Conte, G., Zanoli, S., Scaradozzi, D. and Conti, A. (2004). Evaluation of hydrodynamics parameters of a uuv. a preliminary study, *First International Symposium on Control, Communications and Signal Processing, 2004.*, IEEE, pp. 545–548.

Cui, Z., Chang, Y., Zhang, J., Cai, X. and Zhang, W. (2019). Improved nsga-iii with selection-and-elimination operator, *Swarm and Evolutionary Computation* **49**: 23–33.

Dai, Y., Cheng, R., Yao, X. and Liu, L. (2019). Hydrodynamic coefficients identification of pitch and heave using multi-objective evolutionary algorithm, *Ocean Engineering* **171**: 33–48.

Deb, K., Agrawal, R. B. et al. (1995). Simulated binary crossover for continuous search space, *Complex systems* **9**(2): 115–148.

Deb, K. and Jain, H. (2014). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints, *IEEE Transactions on Evolutionary Computation* **18**(4): 577–601.

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii, *IEEE Transactions on Evolutionary Computation* **6**(2): 182–197.

DNV-GL (2020). Open simulation platform.
URL: *https://www.dnvgl.com/feature/open-simulation-platform-osp.html*

Dobrkovic, A., Iacob, M.-E. and van Hillegersberg, J. (2015). Using machine learning for unsupervised maritime waypoint discovery from streaming ais data, *Proceedings of the 15th International Conference on Knowledge Technologies and Data-driven Business*, p. 16.

DVL-GL (2011). Dnv-rp-h103: Modelling and analysis of marine operations.
URL: *https://rules.dnv.com/docs/pdf/DNVPM/codes/docs/2011-04/RP-H103.pdf*

Eidsvik, O. A. N. and Schjølberg, I. (2016). Determination of hydrodynamic parameters for remotely operated vehicles, *ASME 2016 35th International Conference on Ocean, Offshore and Arctic Engineering*.

Faltinsen, O. (1993). *Sea loads on ships and offshore structures*, Vol. 1, Cambridge university press.

Fan, X., Li, J., Li, X., Zhong, Y. and Cao, J. (2019). Applying deep neural networks to the detection and space parameter estimation of compact binary coalescence with a network of gravitational wave detectors, *SCIENCE CHINA Physics, Mechanics & Astronomy* **62**(6): 969512.

Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control: Fossen/Handbook of Marine Craft Hydrodynamics and Motion Control*.

Fossen, T. I., Sagatun, S. I. and Sørensen, A. J. (1996). Identification of dynamically positioned ships, *Control Engineering Practice* **4**(3): 369–376.

Fossen, T. I. and Øyvind N. Smogeli (2004). Nonlinear time-domain strip theory formulation for low-speed manoeuvring and station-keeping, *Modeling Identification and Control* **25**(4): 201–221.

Holven, E. B. (2018). Control system for rov minerva 2.

Hyndman, R. J. and Koehler, A. B. (2006). Another look at measures of forecast accuracy, *International Journal of Forecasting* **22**(4): 679–688.

Kantas, N., Doucet, A., Singh, S. S., Maciejowski, J., Chopin, N. et al. (2015). On particle methods for parameter estimation in state-space models, *Statistical science* **30**(3): 328–351.

Kochenderfer, M. J. and Wheeler, T. A. (2019). *Algorithms for optimization*, Mit Press.

KTH (2020).
URL: *https://github.com/KTH-SML/motion_capture_system*

Lam, S. K., Pitrou, A. and Seibert, S. (2015). Numba: A llvm-based python jit compiler, *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, LLVM '15, Association for Computing Machinery, New York, NY, USA.
URL: *https://doi.org/10.1145/2833157.2833162*

Liljeback, P. and Mills, R. (2017). Eelume: A flexible and subsea resident imr vehicle, *OCEANS 2017 - Aberdeen*, pp. 1–4.

Ljung, L. (1987). *System Identification: Theory for the User.*

Midthassel, L. (2021). Remora.
URL: *mithal.no*

Morrison, A. and Yoerger, D. (1993). Determination of the hydrodynamic parameters of an underwater vehicle during small scale, nonuniform, 1-dimensional translation, *Proceedings of OCEANS '93.*

Newman, J. N. (2018). *Marine hydrodynamics*, The MIT press.

of Marine Technology, N. D. (2021).
URL: *https://www.ntnu.edu/imt/lab/cybernetics*

Paull, L., Saeedi, S., Seto, M. and Li, H. (2014). Auv navigation and localization: A review, *IEEE Journal of Oceanic Engineering* **39**(1): 131–149.

Pillonetto, G., Dinuzzo, F., Chen, T., De Nicolao, G. and Ljung, L. (2014). Kernel methods in system identification, machine learning and function estimation: A survey, *Automatica* **50**(3): 657–682.

Psichogios, D. C. and Ungar, L. H. (1992). A hybrid neural network-first principles approach to process modeling, *AIChE Journal* **38**(10): 1499–1511.

Stinger (2021).
URL: *stinger.no*

Yuan, Y., Xu, H. and Wang, B. (2014). An improved nsga-iii procedure for evolutionary many-objective optimization, *Proceedings of the 2014 annual conference on genetic and evolutionary computation*, pp. 661–668.

Zhang, X.-G. and Zou, Z.-J. (2013). Estimation of the hydrodynamic coefficients from captive model test results by using support vector machines, *Ocean Engineering* **73**: 25–31.

Žlajpah, L. (2008). Simulation in robotics, *Mathematics and Computers in Simulation* **79**(4): 879 – 897. 5th Vienna International Conference on Mathematical Modelling/Workshop on Scientific Computing in Electronic Engineering of the 2006 International Conference on Computational Science/Structural Dynamical Systems: Computational Aspects.
URL: *http://www.sciencedirect.com/science/article/pii/S0378475408001183*

Michael Hoyer

System Identification and Machine Learning