NTNU
Norwegian University of
Science and Technology

Constrained Optimal Thrust Allocation for C/S Inocean Cat I Drillship

Preben Frederich

Marine Technology
Submission date: July 2016
Supervisor: Roger Skjetne, IMT

Norwegian University of Science and Technology
Department of Marine Technology

NTNU
Norwegian University of
Science and Technology

Constrained Optimal Thrust Allocation for C/S Inocean Cat I Drillship

Preben Frederich
July 2016

Master Thesis
Department of Marine Technology
Norwegian University of Science and Technology

Supervisor: Roger Skjetne

Co-supervisors: Andreas Reason Dahl and Hans-Martin Heyn

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

MSC THESIS DESCRIPTION SHEET

Name of the candidate: Preben Frederich

Field of study: Marine control engineering

Thesis title (Norwegian): Optimal thrust allokering for C/S Inocean Cat I Drillship

Thesis title (English): Constrained optimal thrust allocation for C/S Inocean Cat I Drillship

Background
Stationkeeping operations for offshore vessels (drillrigs, drillships, construction and intervention vessels, PSVs, etc.) are essential for offshore field development and oil and gas production. There has been much attention in the research community on stationkeeping operations, especially by Dynamic Positioning (DP) and Thruster-Assisted Position Mooring (TAPM) of turret-anchored offshore vessels. Essential for such systems is the thruster system's ability to generate the commanded thrust from the higher level DP or TAPM control system. This includes local thruster control for each individual thruster, and a well-functioning constrained optimal thrust allocation algorithm.

In this project the focus is on the model ship "C/S Inocean Cat I Drillship", a 1:90 scaled model of an Arctic drillship design by Inocean, hereafter abbreviated CSAD. This is a vessel with both DP and TAPM functions, having a rotatable turret and 6 azimuth thrusters (3 fore and 3 aft) for DP and thruster assist. In this master thesis the focus is on developing a local thruster control system and a constrained optimal thrust allocation (TA), both for a fixed azimuth configuration and rotatable azimuth configuration.

Work description
1. Perform a background and literature review to provide information and relevant references on:
   - MC-Lab and C/S Inocean Cat I Drillship.
   - Local thruster control of offshore stationkeeping vessels.
   - Constrained optimal thrust allocation methods for DP application.

   Write a list with abbreviations and definitions of terms, explaining relevant concepts related to the literature study and project assignment.
2. Provide a drawing showing the thruster configuration of CSAD and a table giving main properties of the thrusters (locations in body frame, max speed, and max forward and reverse thrust).
3. Develop a dynamic model for the thruster system for CSAD, that is, individual models for each thruster and the overall thruster configuration, giving the combined generalized thruster loads in surge, sway, and yaw.
   - Discuss properties of the thruster system, such as saturation and rate constraints, forbidden sectors, thruster losses, etc.
   - Does the planned power plant for CSAD have a maximum power limit that becomes a power constraint for CSAD?
4. Develop local thruster control for the individual thrusters, taking a commanded thrust as input and mapping this to a speed, power, and/or torque control objective.
   - Implement local thruster controls for CSAD and test in MC-Lab.
5. Develop a practical constrained optimal TA (e.g. QP) for CSAD based on prefixed azimuth angles.
   - Simulate and visualize the efficiency of the TA-algorithm on a deterministic case.
   - Investigate and prepare practical implementation of the algorithm as a "fixed azimuth mode" for CSAD in MC-Lab. Report practical challenges for real-time implementation.
6. Develop a practical constrained optimal TA for CSAD based on rotatable azimuth angles.
   - Simulate and visualize the efficiency of the TA-algorithm on a deterministic case.
   - Discuss forbidden sectors and the performance of the TA-algorithm w.r.t. these.

- Investigate and prepare practical implementation of the algorithm as a "variable azimuth mode" for CSAD in MC-lab. Report practical challenges for real-time implementation.

Tentatively:

7. Perform implementation of the practical constrained optimal TA for CSAD and test in MC-Lab, both for fixed and rotatable azimuth modes.

Guidelines

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

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

The report shall be organized in a logical structure to give a clear exposition of background, results, assessments, and conclusions. The text should be brief and to the point, with a clear language. The report shall be written in English (preferably US) and contain the following elements: Title page, abstract, acknowledgements, thesis specification, list of symbols and acronyms, table of contents, introduction and background, problem formulations, scope, and delimitations, main body with derivations/developments and results, conclusions with recommendations for further work, references, and optional appendices. All figures, tables, and equations shall be numerated. The original contribution of the candidate and material taken from other sources shall be clearly identified. Work from other sources shall be properly acknowledged using quotations and a Harvard citation style (e.g. *natbib* Latex package). The work is expected to be conducted in an honest and ethical manner, without any sort of plagiarism and misconduct. Such practice is taken very seriously by the university and will have consequences. NTNU can use the results freely in research and teaching by proper referencing, unless otherwise agreed upon.

The thesis shall be submitted with a printed and electronic copy to the main supervisor, each copy signed by the candidate. The final revised version of this thesis description must be included. The report must be submitted according to NTNU procedures. Computer code, pictures, videos, data series, and a PDF version of the report shall be included electronically with all submitted versions.

Start date: 15 January, 2016 **Due date:** As specified by the administration.

Supervisor: Roger Skjetne
Co-advisor(s): Hans-Martin Heyn and Andreas Dahl

Trondheim,

Roger Skjetne
Supervisor

Preface

This Master's Thesis has been written during the spring of 2016 as a concluding part of a Master of Science degree in Marine Cybernetics at NTNU. Different thrust allocations and thrust controls have been studied and tested. Theoretical models has been derived from simulation and theory for both thrust control and thrust allocation before implemented on C/S Inocean Cat I Drillship. Essential parameters and variables has been found through experimental testing and derived from information about the real vessel.

I would like to start off by thanking my supervisor Prof. Roger Skjetne who introduced me to C/S Inocean Cat I Drillship and all its challenges thereafter. He has also been a great help with providing guidance and suggestions whenever it was needed.

I would also like to give a big thanks to my co-supervisor Andreas Reason Dahl who provided a detailed follow-up each step of the way. Through discussions and meetings he came up with ideas implemented on the model such as the shaft speed and current measurements. He also provided the wrapping which is used to ensure optimal thruster angle path is always found.

A thanks is also given to my other co-supervisor, Hans-Martin Heyn for providing HIL equipment when needed.

Thor I. Fossen which helped provide the fundamental understanding on how optimal thrust allocation is possible to achieve by means of quadratic regulator deserves a thanks as well.

A big thanks is also given to my classmate Jon Bjørnø who built the vessel used through this thesis. He has also patiently provided help with implementing the thruster control system and experiments conducted to find the needed parameters. The function which maps data from shaft speed measurements to RPM is also provided by him.

Senior engineer Torgeir Wahl also deserves a big thanks. He built the measurements used to measure shaft speed and current seen in this report. He has also provided help for setting up the quadratic regulator into LabVIEW and given good instructions on how to control the various computers found in MC Lab.

Preben Frederich

Preben Frederich
July 10, 2016, Trondheim

Abstract

It is essential to have a good thruster system when it comes to offshore operations and dynamic positioning. Not only can this save fuel, but also make the vessel and operations safer. A good thruster system consists essentially of two components. First is a good thruster control system which controls the motors using either speed, torque or power control. The second is an optimal thrust allocation which determines how the necessary forces are distributed to each of the thrusters.

This thesis describes how modeling and implementation of the thruster system onto NTNU's latest model C/S Inocean Cat I Drillship is done. The model is a 1:90 scaled model based on an Arctic drillship called Inocean Cat I Drillship. The goal is to create a thruster system which is optimal and based upon the parameters from the real vessel.

Creating the thruster system similar is done by using information given by Inocean with scaling laws and experimental tests to find parameters. Model experiments,included bollard pull and drag tests were conducted in NTNU's Marine Cybernetics Laboratory. The remaining parameters are provided by theory or through simulation.

The various thruster controllers are simulated and tested on the model to see the differences and similarities they have.

The thrust allocation finally implemented on the vessel consists of a pseudoinverse algorithm which is limited by forbidden zones and singularities that may arise. It also takes into account the optimal angle path and restrictions on speed. A simulation model in which a quadratic regulator algorithm is used to optimize the allocation is created. Implementation of the quadratic regulator proved to be difficult because of compatibility and is therefore not yet fitted to the model. Potential methods for resolving this problem are given later in the thesis along with one of the tried methods.

Sammendrag

Det er essensielt å ha et godt thruster system når det kommer til offshore operasjoner og posisjons holding. Ikke bare kan dette spare drivstoff, men også gjøre fartøyet og operasjoner sikrere. Et godt thruster system består hovedsakelig av to ting. Første er et godt thruster kontroll system som styrer motorene ved hjelp av enten fart, moment eller kraft kontroll. Det andre er en optimal thrust allokering som bestemmer hvordan nødvendige krefter er distribuert til hver av motorene.

Denne oppgaven dreier seg om å modellere og implementere et thruster system på NTNUs nyeste modell C/S Inocean Cat I Drillship. Modellen er en 1:90 skalert modell basert på et arktisk drillskip kalt Inocean Cat I Drillship. Målet er å lage et thruster system som er optimalt og basert på parametrene fra det ekte skipet.

For å lage thruster systemet mest mulig likt blir informasjon gitt av Inocean skalert ned og eksperimentelle tester blir gjort for å finne parametere. Modell forsøkene blir gjort i NTNUs MC Lab hvor både bollard pull og drag tester er gjennomført. De resterende parameterne som ikke kan bli funnet gjennom testing er det brukt teori eller simulering for.

De forskjellige thruster kontrollerne blir simulert og testet på modellen for å vise forskjeller og likheter de har.

Thrust allokeringen som er brukt består av en pseudoinvers algoritme som er begrenset av forbudte soner og singulariteter som kan oppstå. Den tar også hensyn til optimal vinkel vei samt restriksjoner på hastighet. Det er også laget en simuleringsmodell hvor en kvadratisk regulator algoritme er brukt for å optimalisere allokeringen. Implementering av den kvadratisk regulator viste seg å være vanskelig på grunn av kompatibilitet og er derfor ennå ikke blitt utstyrt på modellen. Potensielle metoder som kan løse dette blir også gitt.

Contents

List of Figures

List of Tables

Abbreviations

AUV	autonomous underwater vehicle
AWACS	active wave absorption control system
CoG	center of gravity
CSAD	C/S Inocean Cat I drillship
CS	cybership
DOF	degree of freedom
DP	dynamic positioning
DPS	dynamic positioning system
FP	fixed pitch
MC Lab	Marine Cybernetics Laboratory
MPC	model predictive control
NED	north-east-down
PMS	power management systems
PWM	pulse width modulation
QP	quadratic programming
QR	quadratic regulator
ROV	remotely operated vehicle
RPM	revolutions-per-minute
RPS	revolutions-per-second
TAPM	thruster-assisted position mooring

Nomenclature

Greek Letters

$\boldsymbol{\alpha}$	Thruster angle
$\boldsymbol{\alpha}_d$	Desired thruster angle
$\boldsymbol{\alpha}_0$	Initial angle
$\boldsymbol{\alpha}_{\max}$	Maximum rotation
$\boldsymbol{\alpha}_{\min}$	Minimum rotation
α_c	Switching function between torque and power control
$\Delta\boldsymbol{\alpha}_d$	Previous time step angle
$\Delta\boldsymbol{u}_d$	Previous time step thrust
ε	Constant for singularity avoidance
ε_c	Switching constant for thrust coefficients
η	Vessel position
λ	Ratio coefficient between vessel and model
λ_c	Switching function for thrust coefficients
ν	Velocity matrix of vessel
ω	Angular shaft speed
ω_{r0}	Natural frequency reference generator
$\dot{\omega}$	Angular shaft acceleration
ϕ	Roll angle of vessel
ψ	Yaw angle of vessel
ρ_F	Sea water density

ρ_M	Model tank water density
$\boldsymbol{\tau}_c$	Control vector
θ	Pitch angle of vessel
ζ_r	Damping ratio reference generator
$\boldsymbol{B}(\boldsymbol{\alpha})$	Thrust configuration matrix
$\boldsymbol{B}^\dagger(\boldsymbol{\alpha})$	Pseudo inverse configuration matrix

Roman Letters

D	Propeller diameter
e	Error from reference shaft speed and actual shaft speed
$\boldsymbol{f}$	Gain Matrix Quadratic solver
$\boldsymbol{H}$	Gain Matrix Quadratic solver
$\boldsymbol{I}$	Identity matrix
I_c	Estimated moment of inertia shaft, propeller and motor
I_s	Moment of inertia shaft, propeller and motor
$\boldsymbol{K}$	Diagonal thrust coefficient matrix
k	Constant used in torque and power control
$\boldsymbol{K}_e$	Extended diagonal thrust coefficient matrix
K_i	Integral gain for commanded motor torque
K_p	Proportional gain for commanded motor torque
K_Q	Torque coefficient
K_T	Thrust coefficient
P_N	Nominal Power
n	Rotation speed of propeller
n_d	Desired rotation speed of propeller
$\dot{n}_d$	Desired acceleration of propeller
n_r	Reference rotation speed of propeller
$\dot{n}_r$	Reference acceleration of propeller

$\dot{n}_{\text{slew}}$	Maximum acceleration of propeller
K_ω	Linear friction coefficient
p	Constant used in torque and power control
P_a	Propeller power
$\boldsymbol{Q}$	Gain matrix
Q_a	Actual propeller torque
Q_{cn}	Commanded motor torque
Q_{cc}	Controller torque: combined power and torque
Q_{cp}	Controller torque: power
Q_{cp}	Controller torque: shaft speed
Q_{cq}	Controller torque: torque
Q_f	Shaft friction
Q_{ff}	Friction compensation
Q_{if}	Inertia compensation
Q_m	Motor torque
Q_N	Nominal propeller torque
r	Constant used in torque and power control
$\boldsymbol{s}$	Slack variable
T_a	Actual propeller thrust
T_c	Commanded propeller thrust
T_i	Integral time constant
T_M	Motor time constant
T_N	Nominal propeller thrust
$\boldsymbol{u}_0$	Initial thrust
$\boldsymbol{u}_d$	Desired Thrust
$\boldsymbol{u}_{\max}$	Maximum thrust allowed
$\boldsymbol{u}_{\min}$	Minimum thrust allowed
$\boldsymbol{W}$	Gain matrix

Chapter 1

Introduction

1.1 Motivation

With increasing computer technology within marine operations, more and more marine vessels rely on control systems such as dynamic positioning (DP) and thruster-assisted position mooring (TAPM) for both stationkeeping and navigation. This can especially be seen in the offshore industry on both platforms and drillships. In order to maintain stationkeeping when environmental loads are affecting the vessel, an optimal thrust system is needed. Although the thruster system is only a small part of a DP operation, there are seemingly endless possibilities for optimizing. Even the smallest optimization for reducing fuel can have a huge impact in the long term.

With this in mind, challenges of implementing an optimal thruster control system on NTNUs newest addition to its cybership fleet, called C/S Inocean Cat I Drillship (CSAD), will be presented by combining known theories.

1.2 Background

1.2.1 Stationkeeping

Stationkeeping is essential for offshore vessels in drilling operations. Wassink and van der List (2013) found that the maximal deviation angle a drilling pipe can have is 2°. This leads to a rather narrow circle the vessel are able to move within when drilling. In order to prevent drift due to environmental loads such as waves, wind, current and ice, the vessel needs to have a control system implemented which is able to ensure the vessel will stay within the circle.

1.2.2 Thruster Assisted Position Mooring

TAPM is a very relevant subject when having stationary offshore vessels. Environmental loads will create extra forces working on the mooring system, and can result in the mooring lines breaking. To counteract extra loads affecting the lines, thrusters can be used to help relieve stresses and ensure safe stationkeeping.

By utilizing position mooring together with thrusters, the fuel consumption can also be reduced drastically. Instead of the DP system constantly changing and updating the optimal thruster position and thrust, it can let some of the forces be absorbed by mooring line. The solutions can also be "tailor made" for different applications.

Not only can the mooring system help with reduction of fuel consumption, but in case of blackout while drilling, it's essential for the vessel to maintain position and keep the drilling angle within the 2°. If the angle is exceeded there are possibilities for structural damage, or in worst case, oil spills. Mooring lines can therefore also be considered a safety measure in case all other fails.

For a complete description of TAPM implemented on CSAD, the reader is referred to Bjørnø (2016).

1.2.3 Thrust Allocation

Thrust allocation main use is to distribute generalized force, in this case: surge, sway and yaw, from a motion controller to each thruster. By having a direct relation between motion controller and generalized force, the motion controller is possible to have constant even with changes in thruster configuration. The thrust allocation problem is known as a model-based problem which is in its simplest form an unconstrained optimization problem where physical limitations can be disregarded. More complex optimization processes can further be derived from this where physical limitations also are included which makes the problem a constrained optimization problem.

Desired thrust is defined with $\boldsymbol{u}_d \in R^r$ where r is number of thrusters available. The generalized force provided by motion controller and its degree of freedom (DOF) affected by these forces can be written as n $\in R^n$. An important feature in thrust allocation is how the vessel is actuated. If r $<$ n, the vessel is underactuated which means there are not enough thrusters to produce desired thrust in all DOF needed. If r $=$ n, the vessel is fully actuated which means it is capable to produce reasonable thrust in any DOF. For most marine systems, r $>$ n which means the vessel is overactuated. This is to ensure reliability and safety in case a thruster fails. When

the system is overactuated, there are often many if not infinitely many solutions to solve thrust allocation problem. It's here optimization algorithms are needed in order to find the best thruster configuration to solve for the generalized force.

There have been many solutions to the thrust allocation optimization problem and with faster computation combined with evermore research, the algorithms become more and more powerful. A list of some optimization solutions is given below

- Unconstrained thrust allocation with singularity handling:
  This was first adressed by Sørdalen (1997) where the Moore–Penrose pseudoinverse was utilized. Sørdalen suggested to scale small singular values found in pseudoinverse decomposition and set them infinitely large s.t. the inverse would become zero.
- Linear quadratic constrained thrust allocation for fixed thrusters:
  Based upon work of Tøndel et al. (2003) on parametric quadratic programming, the application to marine crafts was done by Johansen et al. (2005). Here, the optimization process is done through weighting matrices and iteration inside a cost function.
- Constraind thrust allocation for azimuth thrusters:
  Based upon the works of Johansen et al. (2004) where thrust allocation no longer has linear properties. Because of this, a singularity avoidance term is needed to take into account with the cost function.
- Convex linearly constrained quadratic thrust allocation:
  From Ruth (2008), a convex thrust allocation problem is used. The solution provides a flexible system in on-line testing which means it's possible to reconfigure constraints while the system is running.
- Model predictive control to solve control and thrust allocation in one:
  As one of the newer solution for thrust allocation problem, Scibilia and Skjetne (2012) have taken inspiration from model predictive control (MPC). The thrust allocation considers the current environmental forces and take into account mean environmental forces for next time step. This ensures the thrust allocation are able to produce desired thrust in next time step.

Forbidden Thrust Zones

There are zones where it's undesirable to have the thrust jet interact. Especially when the vessel is equipped with azimuth thrusters. These zones are commonly known as forbidden thrust zones.

For an offshore vessel there are 3 main considerations which need to be addressed.

- Thruster-thruster interaction: occurs when one or more of the thrusters are directing their jet stream into another thruster. When thruster-thruster interaction occurs, there are two main causes for thrust degradation Van Dijk and Aalbers (2001):
  - Forces due to blockage - The jet of one thruster hits another thruster
  - Thrust efficiency loss - The thruster jet interferes with suction flow of the other thruster

Furthermore, these two causes for loss usually occur simultaneously. In order to counteract this problem, the easiest way is to eliminate thrust and make the thruster positioning themselves outside these zones.

- Thruster-moon pool interaction: If a thruster jet stream is pointed toward a moon pool, it can cause disturbances both to the drilling equipment and inside the moon pool.
- Thruster-sensor interaction: One of the most common sensors which can be affected is the hydroacoustic sensor used for measurements underwater. These measurements are used for both physical and biological monitoring and detection. If a thruster is having its jet stream towards hydroacoustic sensors, the measurements could become distorted and produce big deviation especially for current measurements.

1.2.4 Thrusters

Thrusters are the main propulsion method for most marine vessels. In offshore operations thrusters are essential for stationkeeping and DP. The most common thrusters is described by (Fossen (2012), Ch. 12.3) are

- Main propellers: Vessel's main propulsion in surge for transit located on the aft of the hull seen in Figure 1.2.1. These are normally combined with rudders to give the vessel steering control in yaw.
- Tunnel thrusters: Transverse thrusters seen in Figure 1.2.2 which are embedded into the vessel hull. Their main objective is to produce thrust in sway, but can also in combination with other thrusters help to produce yaw moment. The thrusters are only effective at low speeds which means the main use is for either docking or stationkeeping.
- Azimuth thrusters: Thruster units which are able to be rotated with an angle α around the z-axis seen in Figures 1.2.3 and 1.2.4. This type of thrusters is most commonly used in DP operation because of their ability to produce thrust in different directions. Vessels with azimuth thrusters normally are combined with main propellers or tunnel thrusters or both to ensure generalized forces are satisfied. Azimuth thruster create an overactuated optimization problem with respect to both power and failure.

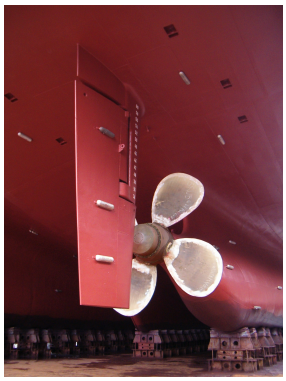

Figure 1.2.1: Main propeller with rudder

Figure 1.2.2: Tunnel thruster

Figure 1.2.3: Azimuth thruster

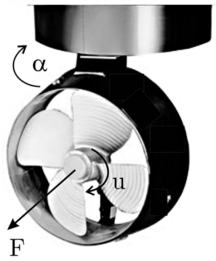

Figure 1.2.4: Azimuth thruster angle rotation

1.3 Scope and Delimitation

Overall goal of this thesis is to give an optimal thruster system through simulations and testing. The thesis begins with an introduction about how thruster systems work and the theory and literature around thruster control systems. It also provides an insight into how parameters are found through testing as well as mapping of control signals.

The thruster control is based upon known theories and created to switch between different types of controllers in real-time. The system is created individually for each thruster such that it's possible to to give each system its own thrust coefficients.

Thrust allocation mainly consists of a constrained pseudoinverse algorithm which also has a built in switch for real-time switching between fixed and azimuth thrusters. It also includes a constrained angle optimization path which ensures that thrusters always choose the correct path.

The system is created to coincide with C/S Inocean Cat I Drillships (CSAD) parameters and configuration. All simulation and testing are based upon CSAD.

Limitations in this thesis are given below as follows

- Thruster torque coefficients are only estimated and not verified. Due to problems in drag test caused by friction of thruster propellers, no advance velocity could be found.
- Only one current sensor is applied to CSAD. Because of high differences in tightening at each thruster belt from motor, the mapping given by that thruster might not be correct for the other thrusters.
- Implementation of quadratic program in real-time. Complications in the quadratic program and real-time applications made the current system not possible to implement as a solver on CSAD.

1.4 Thesis Contributions

The contributions of this thesis are focused on different solutions for the thruster system found on CSAD where the main focus is to provide and implement an optimal control system for each thruster. A Simulink model is made which provides a reliable thrust allocation and thrust control system. The system is also tested on CSAD in the Marine Cybernetics Laboratory at NTNU. Contributions is also given through

- Designing different constrained optimal thrust allocation system for CSAD
- Implementation and testing of constrained pseudoinverse thrust allocation for both fixed and rotatable thruster angles on CSAD
- Designing and implementing different thruster controls
- Implementing shaft speed measurements
- Providing a mapping between control signal and shaft speed, force and current
- Finding thruster coefficients
- Reviewing previous work done on both thruster control and thrust allocation

Chapter 2

Experimental Setup

2.1 C/S Inocean Cat I Drillship

2.1.1 Background

C/S Inocean Cat I drillship, seen in Figure 2.1, is a 1:90 scaled model of an arctic Drillship designed by Inocean for Statoil. The vessel is NTNU's newest addition in their cybership fleet. With its 6 azimuth thrusters and mooring turret connected, the model is able to undergo complex DP and TAPM operations.

The model dimensions are given as L = 2.58m, B = 0.44m and d = 0.13m and at full capacity, the model weights about 95kg. It has an onboard compactRIO which is used for real time testing and 4 reflective balls which can be used for accurate position interpretation by Qualisys in MC Lab.

Figure 2.1.1: C/S Inocean Cat I Drillship

Table 2.1: Cat I Drillship thruster parameters provided by Rolls-Royce

Main Particulars of UUC 505 FP (per unit)	
Power on the vertical input shaft	6000kW
Nominal input speed	0-600 rpm
Propeller speed at nominal input speed	0-143 rpm
Max. torque limitation on the vertical input shaft	95.5 kNm
Direction of rotation looked at the input shaft	clockwise
Gear reduction ratio	4.188:1
Propeller diameter	4200 mm
Propeller type	Fixed pitch, moderately skewed
Nozzle type	TK,tilted 5°
Nozzle inner surface material	Stainless steel
Number of blades	4
Steering speed	2 rpm

Table 2.2: Thruster and moon pools location full scale

Thruster	**Position X[m]**	**Position Y[m]**
Thruster 1	96.1	0.0
Thruster 2	84.1	9.9
Thruster 3	84.1	-9.9
Thruster 4	-104.8	0.0
Thruster 5	-89.2	-14.9
Thruster 6	-89.2	14.9
Moon pool port	57.6	-8.5
Moon pool starboard	57.6	8.5

2.1.2 Full Scale Specifications

Skorpen (2014) provides some of the key aspects about Inocean Cat I Drillship. The vessel is self supplied for up to 120 days and can handle DP operations up to 1.2m ice with a concentration of 70% ice. It's equipped with 8 mooring lines for TAPM which can withstand up to 12 MN (MegaNewton) total force.

The vessel is equipped with azimuth 6 thrusters (3 fore and 3 aft) fully rotatable with a 5° tilt downwards in pitch. Table 2.1 specifications about the thrusters produced by Rolls-Royce are given. It's worth mentioning that the thrusters were initially only able to produce 5500kW, but by request from Inocean they were upgraded to produce 6000kW. These results can be found in Appendix E.6 This is by date the highest input effect Rolls-Royce has delivered on this thruster according to a spokesperson from Inocean.

The diesel engines are of type Wärtsilä 32:40V16. A total of 6 is distributed over 3 engine rooms, and each generator set is able to produce up to 8535kW.

Table 2.2 provides a geometric position of where the thrusters are located from center of gravity (CoG). The key difference from full scale vessel to model is the moon pools. The fullscale Vessel has two moon pools of diameter XxY = 5 [m] x 5.1 [m].

Table 2.3: Thruster parameters with scaling law applied

	Directly Scaled Model	**Actual Model**	**Unit**
Max shaft speed	94.87	152	[RPS]
Power	0.8676	-	[W]
Propeller Diameter	0.0467	0.030	[m]
Steering speed	113.84	270	[Deg/s]
Max Torque	0.0015	-	[Nm]
Max Thrust	1.5034	2.6002	[N]

Figure 2.1.2: Dynamix MX 106R servo

Figure 2.1.3: O.S. Brushless motor 28mm kV950

Thruster forbidden zones

Thrustmaster (2014) conducted tests on a semi-submersible platform where pitch angle on the azimuth thrusters were altered. The test was mainly done to see how pitch angle could reduce thruster-hull and Coanda effects. By increasing the pitch angle from 90° to 97° the thruster-hull and Coanda effects became negligible whereas the thruster effect would still be efficient. They also found by increasing pitch angle thruster-thruster interaction would also become negligible since there would be little to no jet stream reaching the opposite thruster.
Inocean Cat I Drillship solution for constraints are also done in the same manner. As seen in Table 2.1, the thrusters are tilted 5° which means the forbidden zones might be possible to ignore through testing and validation of the real vessel.

2.1.3 Model Specifications

As mentioned earlier, the model is a 1:90 scale vessel. With the given parameters in full scale and applying scaling laws later discussed in Section 4.1 the model parameters can be found. Table 2.3 provides both the directly scaled model parameters and what the actual model has implemented.

The 6 azimuth thrusters are provided with steering speed from 6 Dynamixel MX-106R motors (Figure 2.1.2) and 6 OS OMA-2820-950 brushless motors (Figure 2.1.3) for thrust. The maximum steering speed is given as 45 RPM with no load applied (Robotis, nd). By assuming that the load on the model is negligible, each thruster can provide a maximum steering speed of 270 degrees per second. This means the steering speed is well over scaled speed and needs to be constrained.

Table 2.4: Thruster and mooring line location model

Thruster	**Position X[m]**	**Position Y[m]**
Thruster 1	1.0678	0.0
Thruster 2	0.9344	0.1100
Thruster 3	0.9344	-0.1100
Thruster 4	-1.1644	0.0
Thruster 5	-0.9911	-0.1644
Thruster 6	-0.9911	0.1644
Mooring hole centered	0.1065	0

Data sheet for OS OMA-2820-950 brushless motors can be be found in attachments to this thesis. Utilizing provided data for RPM/V and efficiency factor, the maximum shaft speed without any loads can be found as 152 RPS.

There are some discrepancies which are important to mention. The thrusters are able to produce both higher thrust and have higher steering speed than allowed. How these parameters are constrained will be discussed later in the thesis. Another important discrepancy is propeller diameter. Because the the propeller is not correct from full scale there are some limitations which also will be discussed later.
Table 2.4 shows the scaled positioning of thrusters. All positions are given from reference point in CoG. A mooring line hole is created on the model and has dimensions of XxY = 0.193 [m] x 0.193 [m].

Thruster Forbidden Zones

CSAD has two main considerations when it comes to forbidden zones. Firstly the thruster-thruster interaction described in Section 1.2.3. Because the model has a geometry which leads to a narrow beam, the 3 fore thrusters and 3 aft thrusters are relatively close to each other. This leads to thruster jets influencing the affected thruster hit. In Section 3.2.1 these interactions will be discussed and modeled in greater detail.

The second consideration is thruster-moon pool interaction. Due to geometric differences of moon pools from full scale to model, the constraints normally applied to avoid interaction of thrust jet is not taken into account here.

2.2 Marine Cybernetics Laboratory

2.2.1 Background

MC Lab was originally a storage tank which held vessels made from parafine wax.
The tank was later rebuild as a small wave basin which is now used for testing of different motion control systems both in over- and underwater operations. It uses three cameras above waterline to track position of surface vessels in real time, and has 6 cameras under the waterline to track remotely operated vehicles (ROVs) and autonomous underwater vehicles (AUVs).

The laboratory is operated by NTNUs department of Marine Technology and is mainly used for Master students and Phd Candidates. It's also open for student courses and external users.

2.2.2 Specifications

The dimensions of the basin are given as $40m \times 6.5m \times 1.5m$ (LxBxD). A movable towing carriage is equipped to the basin and is able to move in surge, sway, heave and yaw. Both drag tests and bollard pull were done with the carriage and are described later in the thesis in more detail.

In order to track the vessels during testing MC Lab is equipped with a real time positioning system from Qualisys. With reflective balls (seen at top of model on Figure 2.1.1) applied on the vessel, Qualisys is able to accurately give position and changes in motion.
The MC Lab is also equipped with a wave maker with active wave absorption control system (AWACS) which can both create regular waves and irregular waves through wave spectrum.

Chapter 3

Mathematical Modeling

Dynamic systems can often be divided into two. Kinematics handles the geometrical aspect of a dynamic system, while kinematics will handle forces and moments of the system. In this case, kinematics is defining reference frames used and kinetics is utilized to model both thrusters and how the thruster angles are defined.

3.1 Kinematics

Figure 3.1.1 shows a reference frame for the vessel body. The body reference frame will have a coordinate system that moves with the body. The x-axis is defined positive forwards, y-axis is defined positive to starboard side and z-axis is defined positive downwards. The origin is normally defined in the mean oscillatory position in average water plane. Equation (3.1.1) and (3.1.2) describes both position and velocity in coordinates from Figure 3.1.1

$$\boldsymbol{\eta} = \begin{bmatrix} \mathbf{P} \\ \boldsymbol{\Theta} \end{bmatrix} = [x, y, z, \phi, \theta, \psi]^T \tag{3.1.1}$$

$$\boldsymbol{\nu} = \begin{bmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{bmatrix} = [u, v, w, p, q, r]^T \tag{3.1.2}$$

The 3 first DOF in both equations relates to translational motion, while the 3 last DOF relates to rotational motion. Thruster configuration of a seagoing vessel normally has only 3 DOF possible to control unless the vessel is equipped with active fins or controllable pitch thrusters. Because CSAD does not have either of those, only surge, sway and yaw are of interest for thruster configuration. Equations (3.1.1) and (3.1.2) can therefore be rewritten to

$$\boldsymbol{\eta} = [x, y, \psi]^T \tag{3.1.3}$$

$$\boldsymbol{\nu} = [u, v, r]^T \tag{3.1.4}$$

Furthermore it's also important to note that although thrusters use body-fixed reference frame, the control system is normally using an earth-fixed body reference frame in order to control heading and position on the surface. The description between the two reference frames will not

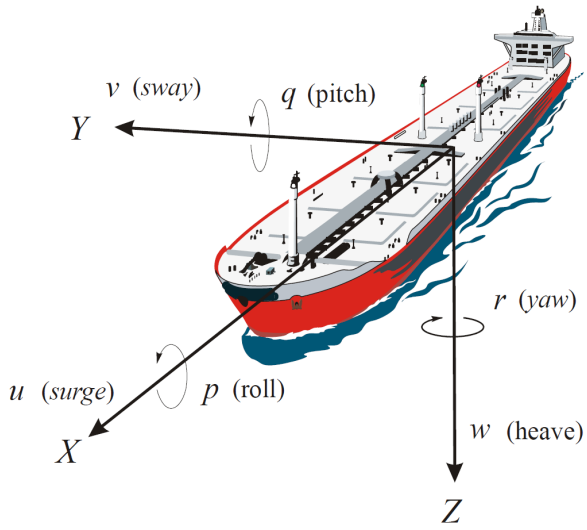

Figure 3.1.1: Definition of DOF for motion courtesy of Sørensen (2014)

be gone into details in this thesis, but below is a quick formulation on how the two reference frames can be related. Using notation found from SNAME (1950) for 6 DOF and earth fixed reference system, the equations can be written as

$$\boldsymbol{\eta}_1 = [x, y, z]^T \qquad \boldsymbol{\eta}_2 = [\phi, \theta, \psi]^T$$
$$\boldsymbol{\nu}_1 = [u, v, w]^T \qquad \boldsymbol{\nu}_2 = [p, q, r]^T$$

Here, $\boldsymbol{\eta}_1$ denotes position vector in Earth-fixed frame and $\boldsymbol{\eta}_2$ is a vector of Euler angles. $\boldsymbol{\nu}_1$ denotes linear velocity for the body-fixed frame and $\boldsymbol{\nu}_2$ denotes angular velocity for the body-fixed frame.
Relating the 6 DOF kinematic equation for body and Earth frame is done through the transformation matrix.

$$\dot{\boldsymbol{\eta}} = \begin{bmatrix} \dot{\boldsymbol{\eta}}_1 \\ \dot{\boldsymbol{\eta}}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_1(\boldsymbol{\eta_2}) & \mathbf{0}_{3x3} \\ \mathbf{0}_{3x3} & \mathbf{J_2}(\boldsymbol{\eta_2}) \end{bmatrix} \begin{bmatrix} \boldsymbol{\nu}_2 \\ \boldsymbol{\nu}_1 \end{bmatrix} = \boldsymbol{J}(\boldsymbol{\eta}_2)\boldsymbol{\nu} \tag{3.1.5}$$

where rotation matrix $\boldsymbol{J}_1(\boldsymbol{\eta}_2) \in \mathbf{SO(3)}$ and $\boldsymbol{J}_2(\boldsymbol{\eta}_2) \in R^{3x3}$ can be defined from Fossen (2012) as

$$\boldsymbol{J}_1(\boldsymbol{\eta}_2) = \begin{bmatrix} \mathrm{c}\psi\mathrm{c}\theta & -\mathrm{s}\psi\mathrm{s}\theta\mathrm{s}\phi & \mathrm{s}\psi\mathrm{s}\phi + \mathrm{c}\psi\mathrm{c}\phi\mathrm{s}\theta \\ \mathrm{s}\psi\mathrm{c}\theta & \mathrm{c}\psi\mathrm{c}\phi + \mathrm{s}\psi\mathrm{s}\theta\mathrm{s}\psi & -\mathrm{c}\psi\mathrm{s}\phi + \mathrm{s}\theta\mathrm{s}\psi\mathrm{c}\phi \\ -\mathrm{s}\theta & \mathrm{c}\theta\mathrm{s}\phi & \mathrm{c}\theta\mathrm{c}\phi \end{bmatrix} \tag{3.1.6}$$

and

$$\boldsymbol{J}_2(\boldsymbol{\eta}_2) = \begin{bmatrix} 1 & \mathrm{s}\phi\mathrm{t}\theta & \mathrm{c}\phi\mathrm{t}\theta \\ 0 & \mathrm{c}\phi & -\mathrm{s}\phi \\ 0 & \mathrm{s}\phi/\mathrm{c}\theta & \mathrm{c}\phi/\mathrm{c}\theta \end{bmatrix}, \mathrm{c}\theta \neq 0 \tag{3.1.7}$$

where, $\mathrm{c} = \cos(\cdot)$, $\mathrm{s} = \sin(\cdot)$ and $\mathrm{t} = \tan(\cdot)$
Using the formulation given from Sørensen (2014) the 3 DOF rotation matrix becomes

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{\nu} \qquad \boldsymbol{R}^{-1}(\psi) = \boldsymbol{R}^T(\psi) \tag{3.1.8}$$

where state vectors are given to be $\boldsymbol{\eta} = [x, y, \psi]^T$ and $\boldsymbol{\nu} = [u, v, r]^T$ and the rotation matrix $\boldsymbol{R}(\psi)$ can be written as

$$\boldsymbol{R}(\psi) = \boldsymbol{C}_{z,\psi}^T = \begin{bmatrix} \mathrm{c}\psi & -\mathrm{s}\psi & 0 \\ \mathrm{s}\psi & \mathrm{c}\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1.9}$$

For more information related to mathematical modeling of CSAD, refer to Bjørnø (2016).

Figure 3.2.1: Precision Schottel drive unit 30mm courtesy of (Aero-naut, nd)

3.2 Kinetics

From Bjørnø (2016, Eq. 5.5), the moored vessel model is given as

$$\boldsymbol{M}_{RB}\dot{\boldsymbol{v}} + \boldsymbol{M}_A\dot{\boldsymbol{v}} + \boldsymbol{C}_{RB}(\boldsymbol{v})\boldsymbol{v} + \boldsymbol{C}_A(\boldsymbol{v_r})\boldsymbol{v_r} + \boldsymbol{D}(\boldsymbol{v_r})\boldsymbol{v_r} = \boldsymbol{\tau}_{\text{env}} + \boldsymbol{\tau}_{\text{moor}} + \boldsymbol{\tau}_{\text{thr}} \tag{3.2.1}$$

In this thesis the last term, $\boldsymbol{\tau}_{\text{thr}}$, is discussed which is the thruster forces. For a full description of (3.2.1), the reader is referred to Bjørnø (2016, Sec. 5.1.2).

3.2.1 Thrusters

CSAD's thrusters are driven by 6 electrical engines and 6 servos for rotation which were mentioned earlier in Chapter 2. The azimuth thrusters used on the model can be seen on Figure 3.2.1.

Thruster Angles

It's important to define a reference guide for both position and force vectors before starting on modeling the thruster system. Often, the relation between thruster angles and desired velocity is not fully intuitive. Figure 3.2.2 shows how the unit circle for each thruster is defined according to body reference frame.

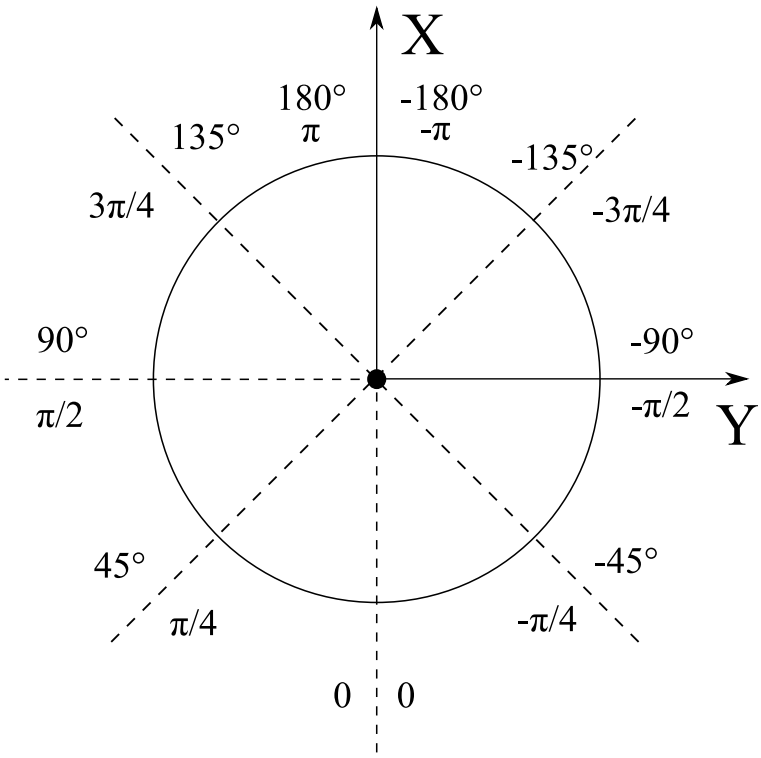

Figure 3.2.2: Unit circle defining thruster angles according to body frame

Forbidden zones

The jet produced by thrusters is rather narrow and depends on what type of thruster it is. CSAD has ducted azimuth thrusters, which means the jet stream will be more focused and narrow than an open thruster. By utilizing the Dynamic Positioning System guide (DPS) from American Bureau of Shipping (2014) it's possible to find correct forbidden sectors.

Both Figure 3.2.3 and Table 3.1 are obtained from DPS guide from American Bureau of Shipping (2014), and are showing how it's possible to find the correct angles between the different thrusters when knowing the geometry. Figure 3.2.3 shows how both x and D can be found from length between thrusters and diameter of propeller, while Table 3.1 determines how wide the angle of thruster-thruster interaction is.

Table 3.1: Range of Forbidden Zone for Different x/D courtesy of American Bureau of Shipping (2014)

x/D	Angle (degrees)	x/D	Angle (degrees)	x/D	Angle (degrees)
1	30	6	17.8	11	14.2
2	26.3	7	16.8	12	13.8
3	22.8	8	16	13	13.3
4	20.6	9	15.3	14	13
5	19	10	14.7	15	12.6

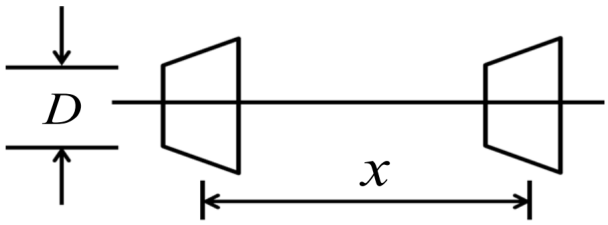

Figure 3.2.3: Thrusters Configuration in Tandem Condition courtesy of American Bureau of Shipping (2014)

Using Table 2.2 from Section 2.1 together with Figure 3.2.2, everything is provided to find the different thruster-thruster interactions. Below, the general formulation of how each constraint are found can be seen

$$X_{ij} = \sqrt{(l_{xi} - l_{xj})^2 + (l_{yi} - l_{yj})^2} \tag{3.2.2}$$

$$R_{ij} = X_{ij}/D \tag{3.2.3}$$

$$\alpha_{ic} = \text{asin}(\frac{l_{xi} - l_{xj}}{X_{ij}}) \tag{3.2.4}$$

where X_{ij} is the length between the given thrusters, l_x and l_y is thruster position from CoG and R_{ij} is the ratio. α_{ic} is the angle from thruster 'i' centered towards thruster 'j' and has to be rotated by $\pm 90°$ depending on what quadrant the other thruster is in relation.

Finding the constraint angle is done by interpolation as seen below where the resulting constraint angle α_{fij} is added and subtracted from α_{ic} to find correct forbidden zone given within α_{fi}.

$$\alpha_{fij} = \alpha_0 + (\alpha_1 - \alpha_0)\frac{R_{ij} - R_0}{R_1 - R_0} \tag{3.2.5}$$

$$\alpha_{fi} = \alpha_{ic} \pm \alpha_{fij} \tag{3.2.6}$$

The resulting constraints from (3.2.6) becomes

$$\begin{aligned}
\text{Thruster1 Constraints} = \alpha_{f1} = \quad & -140.5° \pm \frac{21.0°}{2} \quad \text{and} \quad 140.5° \pm \frac{21.0°}{2} \\
\text{Thruster2 Constraints} = \alpha_{f2} = \quad & -90.0° \pm \frac{19.7°}{2} \quad \text{and} \quad -39.5° \pm \frac{21.0°}{2} \\
\text{Thruster3 Constraints} = \alpha_{f3} = \quad & 90.0° \pm \frac{19.7°}{2} \quad \text{and} \quad 39.5° \pm \frac{21.0°}{2} \\
\text{Thruster4 Constraints} = \alpha_{f4} = \quad & -43.5° \pm \frac{18.6°}{2} \quad \text{and} \quad 43.5° \pm \frac{18.6°}{2} \\
\text{Thruster5 Constraints} = \alpha_{f5} = \quad & 90.0° \pm \frac{16.5°}{2} \quad \text{and} \quad 136.5° \pm \frac{18.6°}{2} \\
\text{Thruster6 Constraints} = \alpha_{f6} = \quad & -90.0° \pm \frac{16.5°}{2} \quad \text{and} \quad -136.5° \pm \frac{18.6°}{2}
\end{aligned}$$

From the constraints given above, it's possible to see there are no overlapping of constraints regarding thruster 2&3, 5&6 or 1&4. And since the thruster placement is similar for the couples

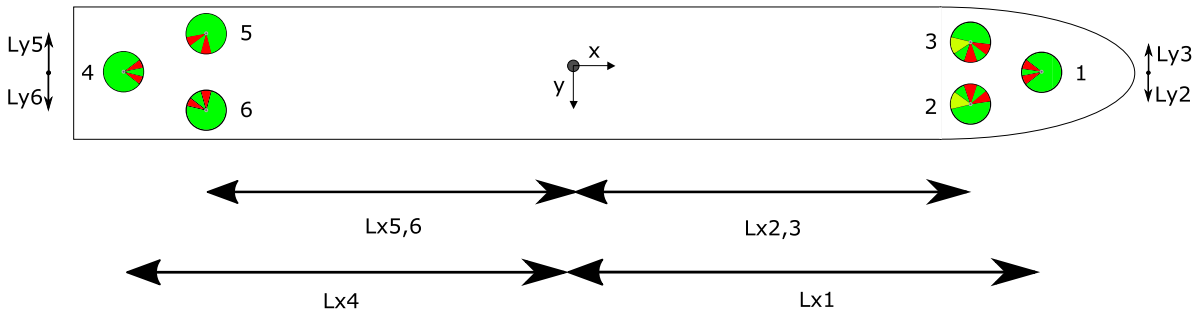

Figure 3.2.4: Thruster constraints for CSAD

it means by coupling these together, it's possible to to create a system where most of the thrusters will be able to hold desired angle. The couples which has to deviate from desired angle can then reduce the unwanted produced thrust. Using coupled constraints will help the model become more stable. Figure 3.2.4 shows how the forbidden zones are distributed for each thruster within the unit circle. Red marking means there is a thruster-thruster interaction, and the given thruster is turned off until it has crossed this point. Yellow marker shows where there will be thruster-moon pool interaction on the model if simulating moon pool position from actual vessel. This restraint has not been implemented due to differences from real vessel to scaled model regarding the thruster tilt.

Thruster Dynamics

Thruster dynamics can be divided up into three parts. There is an electrical motor, shaft with friction and hydrodynamical propeller load. From Smogeli et al. (2005) thruster dynamics is defined by the following equations

$$\dot{Q}_m = \frac{1}{T_m}(Q_c - Q_m) \tag{3.2.7}$$

$$I_s\dot{\omega} = Q_m - Q_a - Q_f(\omega) \tag{3.2.8}$$

$$Q_a = f_Q(\theta, \zeta) \tag{3.2.9}$$

$$T_a = f_T(\theta, \zeta) \tag{3.2.10}$$

$$P_a = Q_a\omega \tag{3.2.11}$$

where, T_m is the motor time constant, Q_c is commanded torque provided by thruster control (Chapter 5), I_s is the rotational inertia from propeller. Q_a, T_a and P_a is the propeller load torque, thrust and power respectively. $Q_f(\omega)$ is the shaft friction, for this thesis given as a linear function described by Smogeli et al. (2005) as $Q_f(\omega) = K_\omega\omega$ where K_ω is the linear friction coefficient.

f_Q and f_T describe Q_a and T_a. These functions can include thruster loss effects due to thruster-thruster interaction, in-and-out-of water effects, currents, winds etc. For this thesis these effects are considered negligible or are taken into account in thrust allocation. This leaves Q_a and T_a possible to write as conventional quadratic thruster characteristics described by Carlton (1994)

as

$$Q_a = \text{sign}(n) K_Q \rho D^5 n^2 \tag{3.2.12}$$

$$T_a = \text{sign}(n) K_T \rho D^4 n^2 \tag{3.2.13}$$

where K_Q and K_T is torque and thrust coefficients respectively, ρ is water density, D is propeller diameter and n is propeller shaft speed in rps.

Chapter 4

Parameter Identification

4.1 Scaling Laws

The model is a scaled replica of Inocean Cat I drillship as mentioned earlier and are set to be 1:90 scaled. In order to do accurate tests which would replicate the real vessel motion, the parameters must be scaled down. By utilizing Froude's law, the different parameters can be scaled accordingly. For this project, it's a necessity for scaling laws from the real vessel to be implemented. Both the thrusters and power availability are much higher on the model than the real vessel. Below, the different scaling laws are given to get the best fit from real vessel to model. M denotes model and F denotes real vessel parameters. $\lambda = 90$ is the scaling parameter.

$$
\begin{aligned}
L_F &= \lambda L_M = 90 L_M && [m] \\
I_F &= \frac{\rho_F}{\rho_M}\lambda^3 I_M = 747225 I_M && [Kgm^2] \\
P_F &= \frac{\rho_F}{\rho_M}\lambda^{7/2} P_M = 7088800 P_M && [W] \\
Q_F &= \frac{\rho_F}{\rho_M}\lambda^4 Q_M = 67250250 Q_M && [Nm] \\
F_F &= \frac{\rho_F}{\rho_M}\lambda^3 F_M = 747225 F_M && [N] \\
v_F &= \sqrt{\lambda} v_M = 9.48 v_M && [m/s] \\
n_F &= \frac{\sqrt{\lambda}}{\lambda} n_M = 0.1054 n_M && [RPM] \\
T_F &= \sqrt{\lambda} T_M = 9.48 T_M && [s] \\
Hz_F &= \frac{1}{\sqrt{\lambda}} Hz_M = 0.1054 Hz_M && [Rad/s] \\
a_F &= a_M && [m/s^2]
\end{aligned}
$$

where, L is length, I is moment of inertia, P is power, Q is torque, F is force, v is velocity, n is shaft speed, T is time, Hz is frequency and a is acceleration. $\rho_F = 1025$ is density of sea water, and $\rho_M = 1000$ is density of simulation tank. With these laws implemented it's now possible to find a more accurate maximum limit for each of the thrusters.

4.2 Thrust Configuration

Thrust configuration describes the relation between generalized forces/moments and the thrusters positioning from CoG and thrust angle. For a single thruster in 6 DOF, the thrust vector and positioning in body reference frame can be written as

$$\boldsymbol{f} = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \qquad \boldsymbol{l} = \begin{bmatrix} l_x \\ l_y \\ l_z \end{bmatrix} \tag{4.2.1}$$

Using notation from Skjetne (2015), the corresponding thrust for generalized forces/moments can then be written as

$$\boldsymbol{\tau} = \begin{bmatrix} f \\ l \times f \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \\ f_z \\ l_y f_z - l_z f_y \\ l_z f_x - l_x f_z \\ l_x f_y - l_y f_x \end{bmatrix} = \begin{bmatrix} \text{Surge} \\ \text{Sway} \\ \text{Heave} \\ \text{Roll} \\ \text{Pitch} \\ \text{Yaw} \end{bmatrix} \tag{4.2.2}$$

For normal seagoing vessels each thruster are normally only able to control 3 DOF which means the thrust for generalized forces/moments can be reduced to

$$\boldsymbol{\tau} = \begin{bmatrix} f_x \\ f_y \\ l_x f_y - l_y f_x \end{bmatrix} = \boldsymbol{B}_{3xr}(\boldsymbol{\alpha})\boldsymbol{K}\boldsymbol{u} \tag{4.2.3}$$

where $\boldsymbol{B}_{3xr}(\boldsymbol{\alpha})$ defines thruster configuration, $\boldsymbol{K} \in R^{rxr}$ is force coefficients and $\boldsymbol{u} \in R^{rx1}$ is control input. r defines number of thrusters.

As discussed in Section 1.2.4 there are different types of thrusters and each type of thruster has a specific thrust configuration. Since CSAD is only equipped with azimuth thrusters and have 3 controllable DOF (surge, sway yaw), the configuration can be written as

$$\boldsymbol{B}_{3x6}(\boldsymbol{\alpha}) = \begin{bmatrix} \cos(\alpha_1) & \sin(\alpha_1) & l_{x1}\sin(\alpha_1) - l_{y1}\cos(\alpha_1) \\ \cos(\alpha_2) & \sin(\alpha_2) & l_{x2}\sin(\alpha_2) - l_{y2}\cos(\alpha_2) \\ \cos(\alpha_3) & \sin(\alpha_3) & l_{x3}\sin(\alpha_3) - l_{y3}\cos(\alpha_3) \\ \cos(\alpha_4) & \sin(\alpha_4) & l_{x4}\sin(\alpha_4) - l_{y4}\cos(\alpha_4) \\ \cos(\alpha_5) & \sin(\alpha_5) & l_{x5}\sin(\alpha_5) - l_{y5}\cos(\alpha_5) \\ \cos(\alpha_6) & \sin(\alpha_6) & l_{x6}\sin(\alpha_6) - l_{y6}\cos(\alpha_6) \end{bmatrix}^T \tag{4.2.4}$$

Furthermore, from an approach by Sørdalen (1997) the thrust can be decomposed into directional force vector by an extended thrust configuration. By using an extended thrust configuration, (4.2.4) will go from being a nonlinear in $\boldsymbol{\alpha}$ to become linear thrust configuration. The extended thrust forces for CSAD are described as

$$\boldsymbol{\tau} = \boldsymbol{B}_{3x12}\boldsymbol{K}_e\boldsymbol{u}_e \tag{4.2.5}$$

The thrust configuration then becomes

$$\boldsymbol{B}_{3x12} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & l_{x1} & l_{y2} & l_{x2} & l_{y3} & l_{x3} & 0 & l_{x4} & l_{y5} & l_{x5} & l_{y6} & l_{x6} \end{bmatrix} \tag{4.2.6}$$

Thrust coefficient $\boldsymbol{K}_e$ now becomes $\in R^{12x12}$ and control input $\boldsymbol{u}_e$ becomes $\in R^{12x1}$ written as

$$\boldsymbol{u}_{ie} = \begin{bmatrix} u_{ix} \\ u_{iy} \end{bmatrix} = \begin{bmatrix} F_i \cos(\alpha_i) \\ F_i \sin(\alpha_i) \end{bmatrix} \tag{4.2.7}$$

In Chapter 6, these configurations will be discussed in more detail and used to find both optimal thrust allocation along with optimal thruster angles.

Table 4.1: Thruster coefficients

	Thruster 1	**Thruster 2**	**Thruster 3**	**Thruster 4**	**Thruster 5**	**Thruster 6**
K_T	0.3763	0.3901	0.3776	0.5641	0.4799	0.5588
K_Q	0.0113	0.0117	0.0113	0.0169	0.0144	0.0168

4.3 Thrust Coefficients

Thruster coefficients are normally found by "open water test" preformed in a towing tank or cavitation tunnel. The coefficients is found from the following parameters as described by Oosterveld and Oossanen (1975)

$$K_T = f_1(J, P/D, A_E/A_O, Z, R_n, t/c) \tag{4.3.1}$$
$$K_Q = f_2(J, P/D, A_E/A_O, Z, R_n, t/c) \tag{4.3.2}$$

where, J is advance ratio, P/D is pitch ratio, A_E/A_O is blade area ratio, Z is blade number, R_n is Reynolds number and t/c is blade thickness profile at a characteristic radius.
A towing test was conducted where the thruster belt was disconnected such that the propeller was able to move freely around. The towing test was done up to 0.7 m/s, but due to possibly resistance in shaft rotation, there was no shaft speed produced. Thruster coefficients could therefore not be found directly from a towing test.
Instead, the results from a bollard pull test done in the next section is used where both thrust and shaft speed were measured in both directions. From van Lammeren et al. (1969), the thruster coefficients can be found from

$$K_T = \frac{T_a}{\text{sign}(n)\rho D^4 n^2} \tag{4.3.3}$$
$$K_Q = \frac{Q_a}{\text{sign}(n)\rho D^5 n^2} \tag{4.3.4}$$

For this thesis, only the mean value of each thruster coefficient is used, but in Appendix E.5, the thruster curves are provided for further use if needed. The reason for only choosing the mean value and not the curve is due to discrepancies occurring at low thrust commands.

Due to lack of measurements in propeller shaft torque, K_Q is only approximated from K_T and with regards to propeller diameter. This is not an optimal solution by any means, but provides a reasonable torque coefficient. The coefficient is found by taking mean value from each thrust coefficient graph $\overline{K}_T$ and multiply with propeller diameter D. Coefficients are given in Table 4.1.

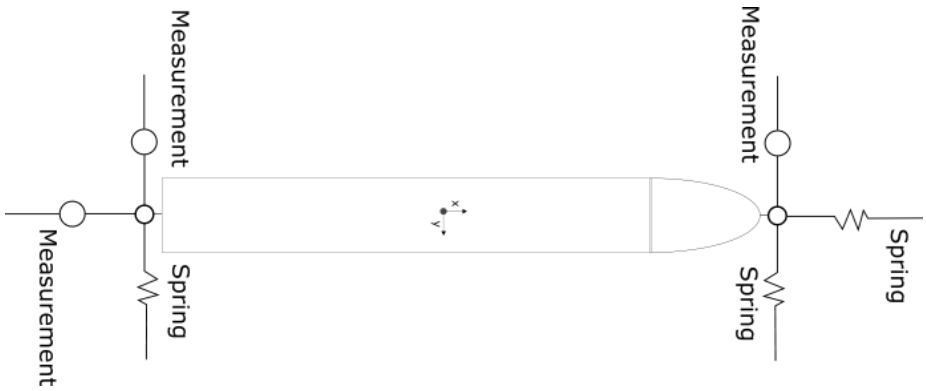

Figure 4.4.1: Constraints on CSAD for Bollard test

4.4 Bollard pull

Bollard pull test is normally done in open waters where a vessel is connected up to a tow-line attached on a shore-mounted bollard. The tow-line force is measured over time with the vessel running on maximum capacity.

In order to find relation between pulse width modulation (PWM), also known as a control signal, sent to the thrusters and force produced from control signal, a slightly modified bollard pull test is used.

In order to get correct mapping, the model is fastened with springs and rope connected to ring force transducers provided by the MC Lab. Sketch 4.4.1 shows how the model is restrained for forward thrust. For backward thrust, the spring on bow and force ring at stern are switched around. Measurements are more accurate when the vessel is fully restrained. The test also includes verification of hall effect sensor and volt measurements as described earlier in the thesis, together with relating both measurements to control signals.

The belt from motor to thruster is not equally fastened on every thruster. This leads to different resistance on each thruster, which again leads to each thruster needed to be tested and mapped accordingly.

In order to retrieve most accurate results, the data is filtered by a Butterworth filter. Butterworth filters is designed to make noisy signals more "flat", and are therefore ideal to use on the data from this test where the signal is very noisy. The filter formula is given as

$$G(\omega) = \sqrt{\frac{1}{1+\omega^{2n}}} \tag{4.4.1}$$

where $n = 3$ which is order of filtering and $\omega = 2/(200 * 2) = 0.005$ which is half the sampling frequency. Since the sampling frequency is constant for all data measured, the filter remains the same for all results found in the next sections.

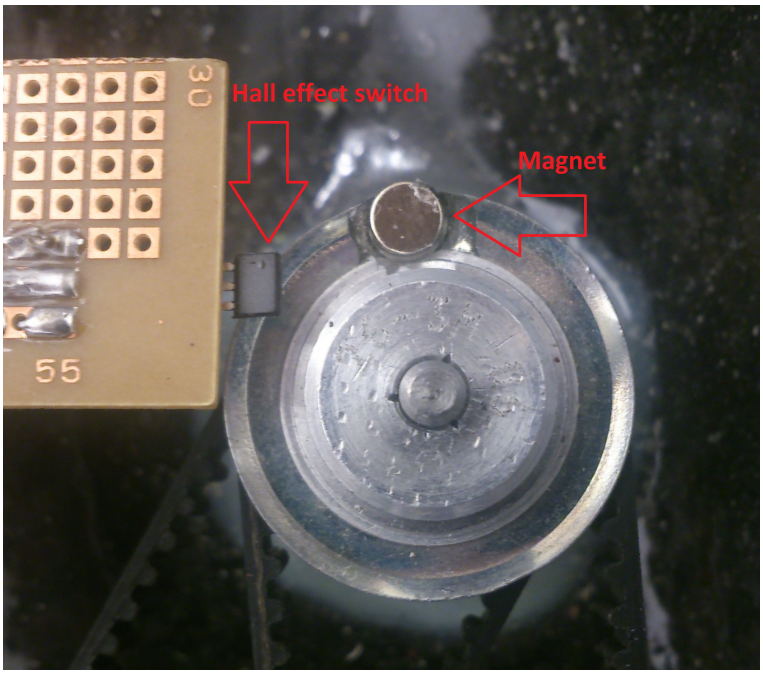

Figure 4.4.2: Experimental setup of shaft speed measurement

4.4.1 Control Signal to Speed Mapping

In order to find a mapping between control signal sent to the thrusters and propeller speed, a way of measuring gear rotation on the thruster is needed. The solution is chosen to consist of a hall effect sensor which varies its output signal depending on magnetic fields. The setup can be seen on Figure 4.4.2. The basic idea is whenever the magnet passes the hall effect sensor, a signal will be sent to LabVIEW where it's processed and sent out as a rising counter. From here, the signal is mapped over to rounds-per-minute (RPM) through a function seen in Appendix A.5 and figure in Appendix C.1.

The results from control signal to speed mapping on thruster 1 can be seen on Figure 4.4.3 for forward speed and Figure 4.4.4 for backward speed. For the other thrusters, the results are provided in Appendix B.

From the results, the error from curve fitting is relatively low. For both results, the error is within ±10% which is acceptable. The same is also true for results provided in Appendix B with an exception on thruster 4 at forward speed where a wild point caused an error of 25%. This originates from ensuring there is no negative speed mapped and by forcing the curve positive.

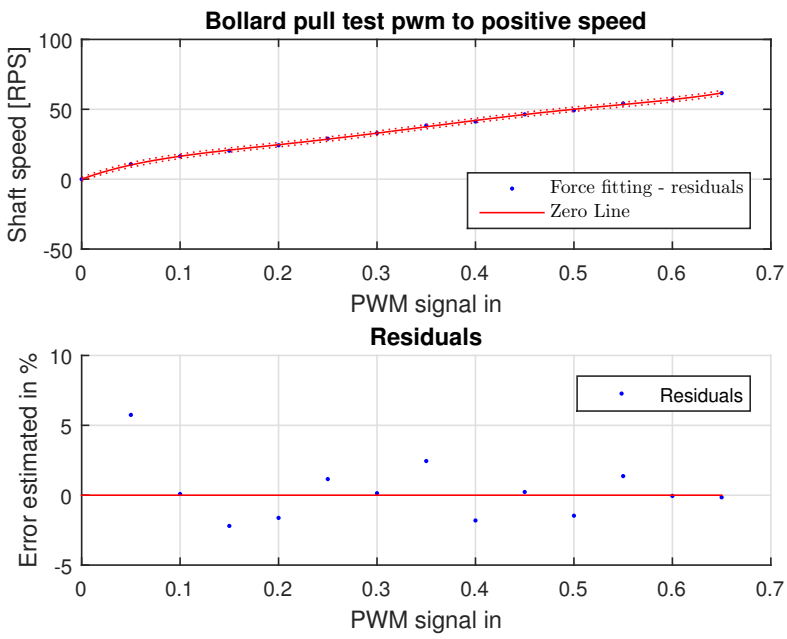

Figure 4.4.3: Results from control signal to forward speed mapping

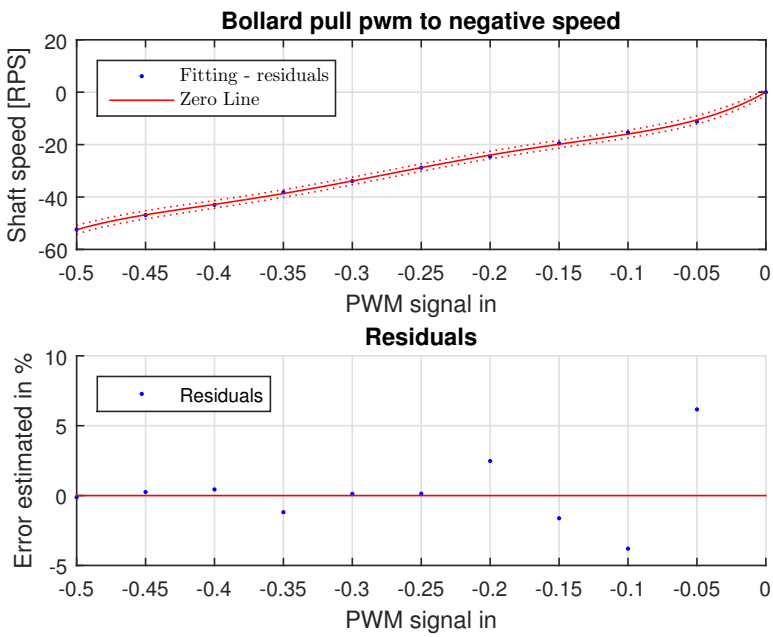

Figure 4.4.4: Results from control signal to backward speed mapping

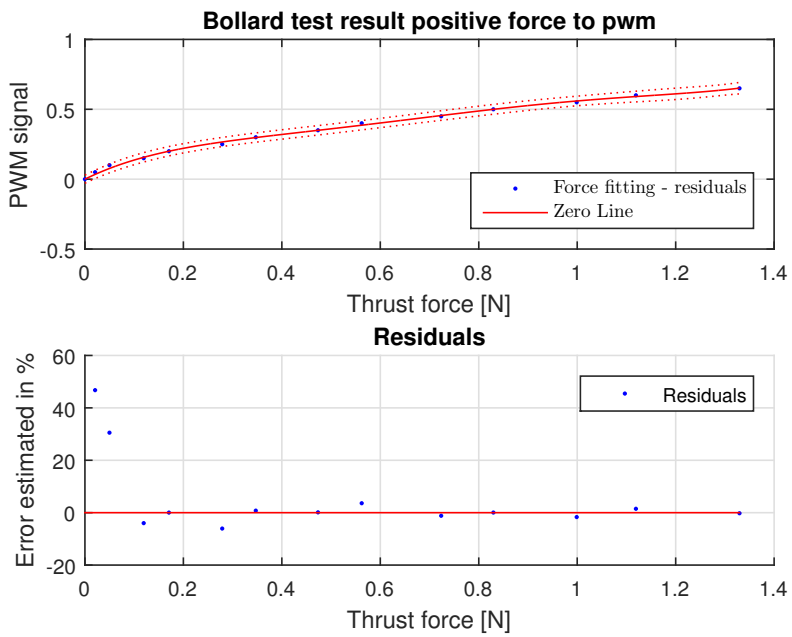

Figure 4.4.5: Results from control signal to forward force mapping

4.4.2 Control Signal to Force Mapping

The control signal to force mapping is done in the same way as seen in the last section. Each thruster was given a control signal one at the time to ensure correct mapping. It's important to note some of the thrusters are placed outside center of x-axis which provides a slight moment on the model. Through processing of data, it's found to have a negligible impact on the results and is therefore not taken into account.

The results from thruster 1 forward and backward produced thrust can be seen on Figures 4.4.5 and 4.4.6 respectively. For the other thrusters, the results are provided in Appendix B. Figure 4.4.5 and 4.4.6 shows how the control signal and thrust are related. From the residual plot, it's possible to see that there is some discrepancy on both figures when commanding a low control signal. The same results are also similar for the other thrusters seen in Appendix B at low control signal. Here, the error is as high as 100 % at the first measurements. This is normally not acceptable, but considering the commanded force at this signal is under 1% of max allowed thrust it will not create major impact. It is also necessary to have this high discrepancy in order to prevent any values going below zero.
The test was also initially conducted with lower steps between signals at 0 and 0.1, but due to sensor disturbance, the results did not provide any improvement on the curves.

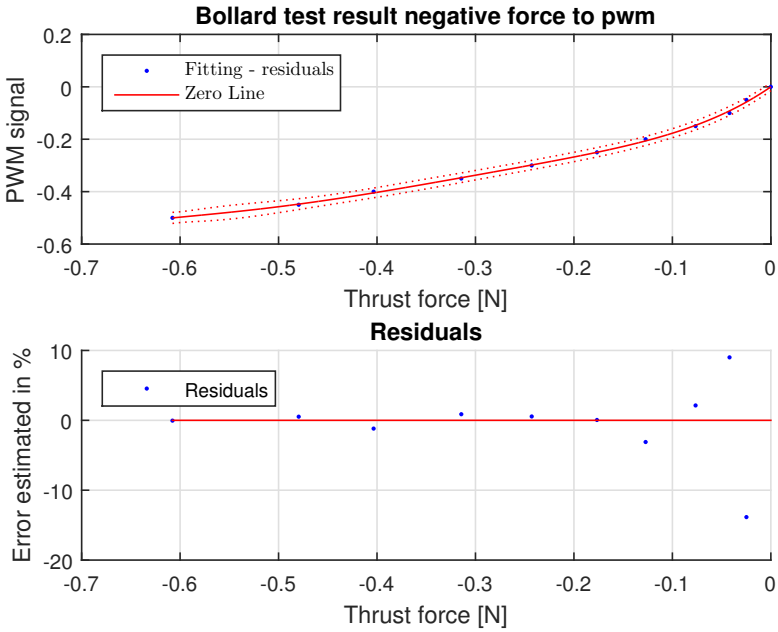

Figure 4.4.6: Results from control signal to backward force mapping

4.4.3 Control Signal to Current Mapping

Figure 4.4.7: Experimental setup of current measurement

Control signal to current mapping is done through a current sensor seen on Figure 4.4.7 by threading the electrical motor cable through it. Current sent through to the motor is then possible to measure at different control signals. There is only one current sensor equipped on CSAD for the moment of this thesis which means only one thruster can be utilized for power control.

Testing through bollard pull was also done at an early stage in the thesis where power was assumed would be very similar forward and backward, leaving results only with forward control signal mapping.
Curvefitting seen in Figure 4.4.8 is accurate with only ±5% error. The result is achieved by twisting the cable 8 times around the current sensor which means the signal is boosted approximately 8 times.

While the results are promising for further use, there are currently no mapping between current and ampere for CSAD. For control signals below 0.07, the measured result consist mainly of disturbance as seen on Figure 4.4.9 which makes mapping the signal not viable.

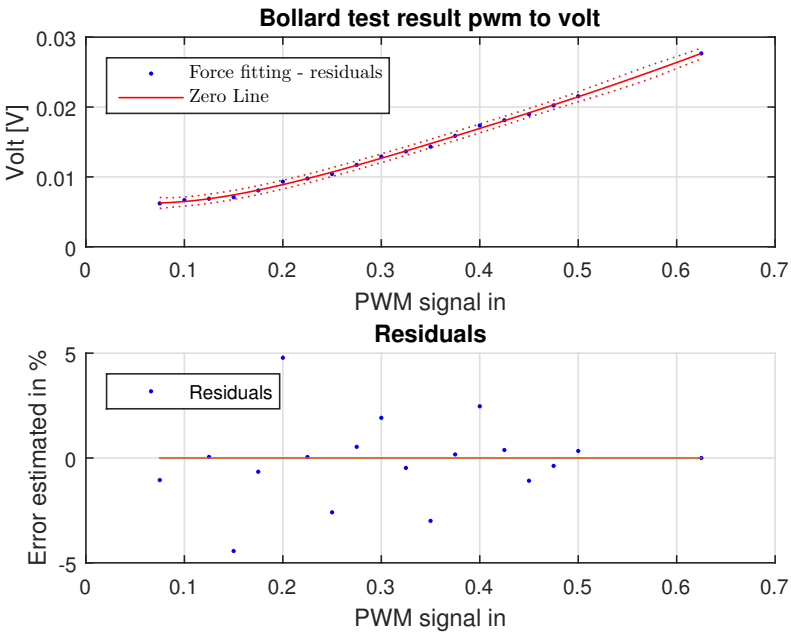

Figure 4.4.8: Results from control signal to current mapping

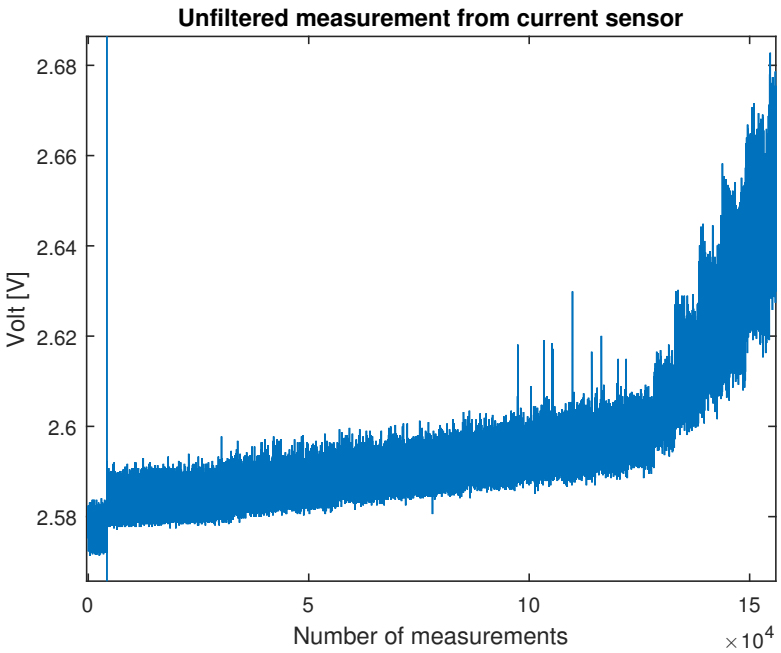

Figure 4.4.9: Measurements from current sensor with control signal from 0 to 1.125

Chapter 5

Thruster Control

5.1 Problem Formulation

The purpose of a thruster control is to relate given desired force $\boldsymbol{u}_d$ from thrust allocation in each thruster to a commanded motor torque Q_c. Through this chapter, core controllers will be described with both their upsides and flaws. The theory which is used for these controllers is covered by Sørensen (2014, Sec 9.8).

The thruster controllers are possible to switch between in real-time through a simplified switching system with a supervisor based upon the work of Hespanha (2002). Switching is done manually through VeriStand but uses logic to ensure power control is not possible to use close to zero shaft speed which will be discussed later in Section 5.4.

The testing is done with regular sine waves given as

$$F(t) = 0.06 \sin(\frac{2\pi}{1.5} t) + 0.15 \tag{5.1.1}$$

Which provides a regular wave with interval each 1.5 seconds. The thrusters are fixed in surge and thrust allocation is not taken into consideration. Only results for thruster 1 are shown in this chapter, but the behavior seen through the tests are valid for all thrusters given same conditions.

Results through the chapter is given by mean force value combined with error produced from desired value and commanded value.

5.2 Shaft Speed Control

The shaft speed control has the revolution speed of propeller controlled to a desired value. This is done by using a higher level controller which computes a desired thrust which is then mapped to the corresponding revolution speed based on the thruster model. The shaft speed is very easy to measure and will hold the same revolution speed if ventilation or out-of-water occurs. The problem with shaft speed mapping is the thruster model can be degraded where there are varying flow conditions around the propeller (thruster-thruster interaction for example). This can lead to revolution speed not corresponding to desired thrust and fluctuations in the shaft torque can cause wear and tear.

Shaft speed controller does not need to take friction and inertia compensation, discussed in Section 3.2.1, into account since it's regulating entirely based upon shaft speed. Although it's not necessary, the resulting controller torque is therefore taken into account for this thesis in order to have a core controller which can be switched between.

5.2.1 Controller

The controller is a feedback controller where both desired shaft speed found from reference generator and measured shaft speed is utilized in a PD-controller as seen below

$$Q_{cn} = K_p e + K_i \int_0^t e(\tau)\,\mathrm{d}\tau \tag{5.2.1}$$

where $e = n_r - n_m$ is the error between reference shaft speed and measured shaft speed. K_p is the controller's proportional gain and K_i is the integrator gain. The integrator gain is normally given as $K_i = \frac{K_p}{0.05}$. For CSAD, $K_p = 3.3$ which provides a very low shaft speed error.

5.2.2 Simulation Results

Figure 5.2.1 show simulated results from forces applied by (5.1.1) when the shaft speed controller is used.

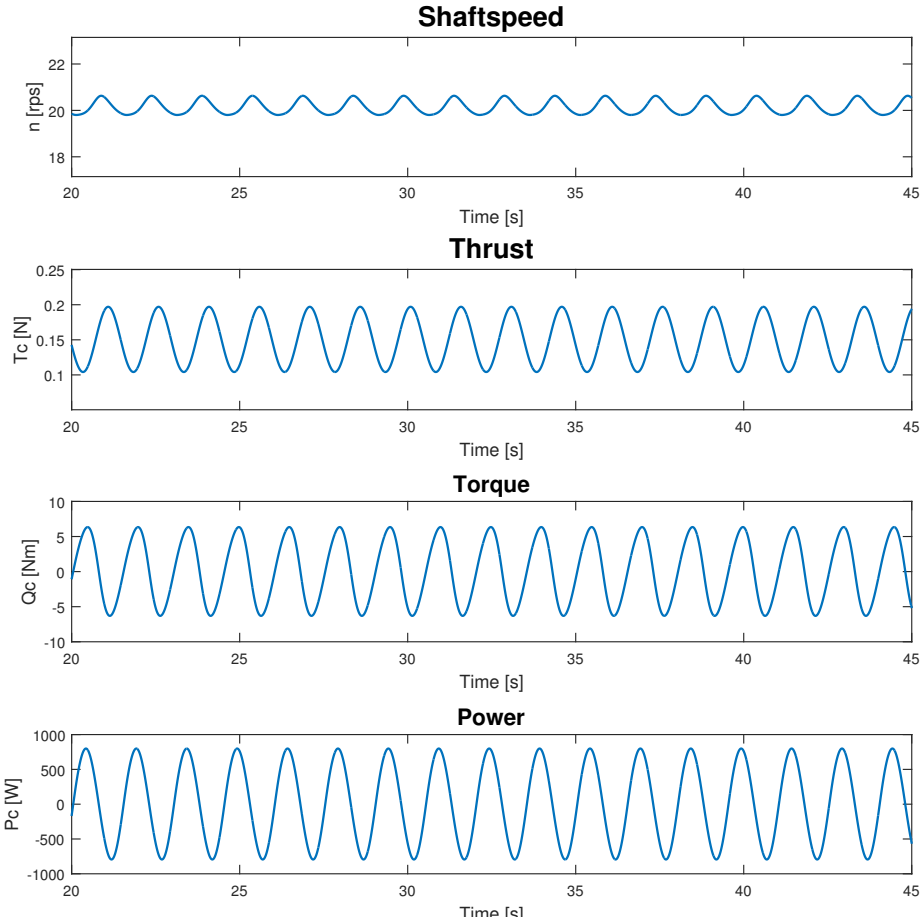

Figure 5.2.1: Shaft speed result with simulated regular waves and current

5.2.3 Experimental Results

Figure 5.2.2 show experimental results from forces applied by (5.1.1) when the shaft speed controller is used.

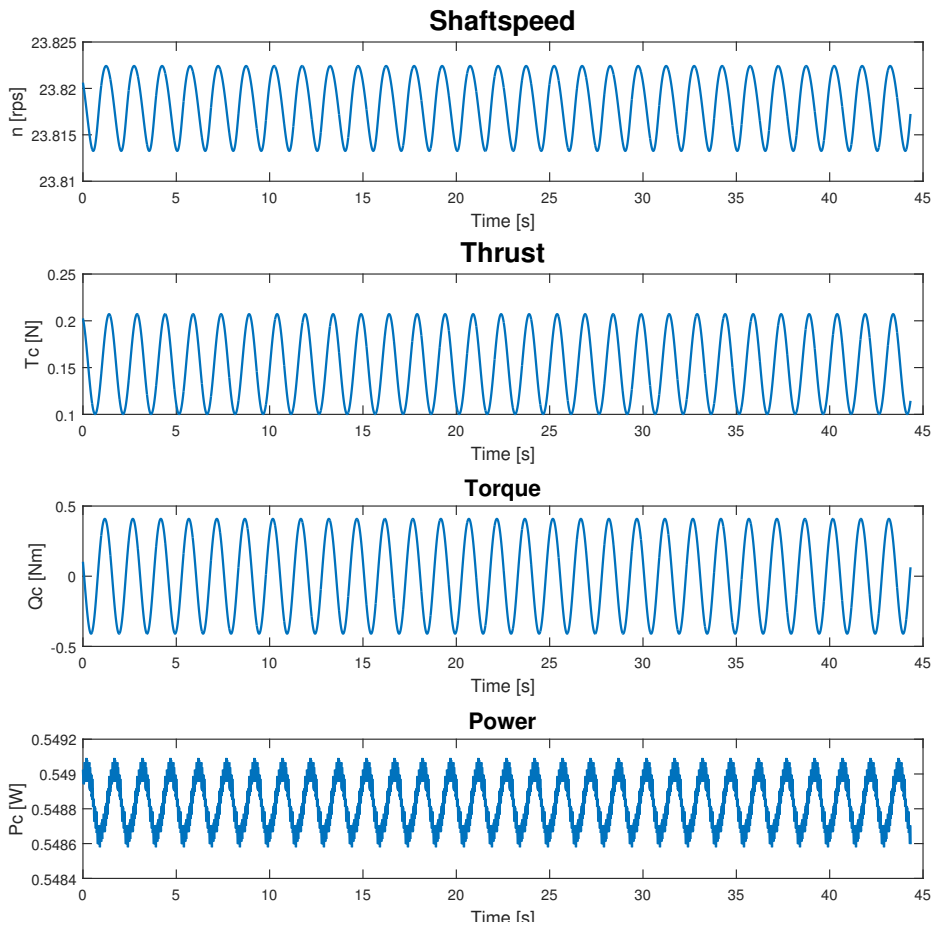

Figure 5.2.2: Shaft speed result with applied regular waves and current

5.3 Torque Control

The torque control uses the shaft torque to be controlled to a desired value. Torque is more closely related to thrust than revolution speed, which means the there will be a more accurate and smooth thrust. It can also help against wear and tear for the mechanical system because the thrust curve is more smooth. The problems is when ventilation or out-of-water occurs. Ventilation can create propeller race, which means the revolution speed increases rapidly. When the propeller hits the water again, it will create a huge load on the shaft and gives wear and tear for the mechanical system or even damage.

5.3.1 Controller

Torque controller is given as a feedforward controller and utilize the relation between thrust and thruster coefficients. The formulation can be written as

$$Q_{cq} = Q_r = \frac{K_{QC}}{K_{TC}} D T_r \tag{5.3.1}$$

Here, K_{QC} and K_{TC} is the thruster torque and force coefficients respectively, D is propeller diameter and T_r is force found by the reference generator.

5.3.2 Simulation Results

Figure 5.3.1 show simulated results from forces applied by (5.1.1) when the torque controller is used.

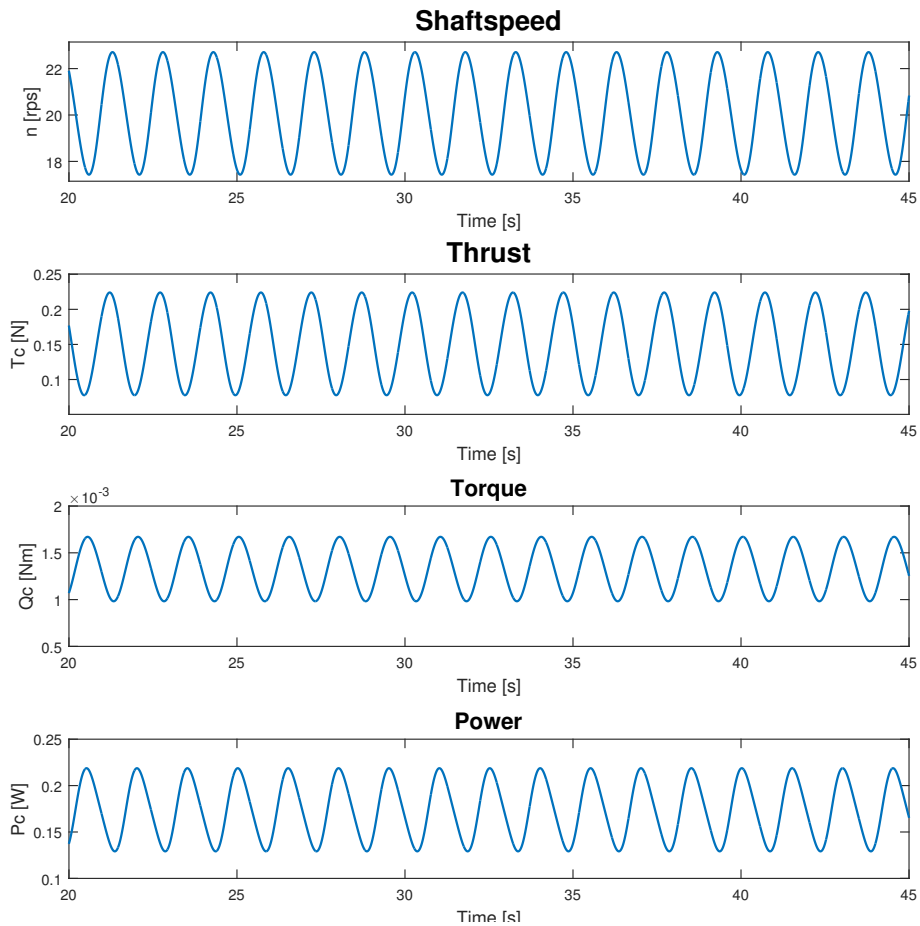

Figure 5.3.1: Torque result with simulated regular waves and current

5.3.3 Experimental Results

Figure 5.3.2 show experimental results from forces applied by (5.1.1) when the torque controller is used.

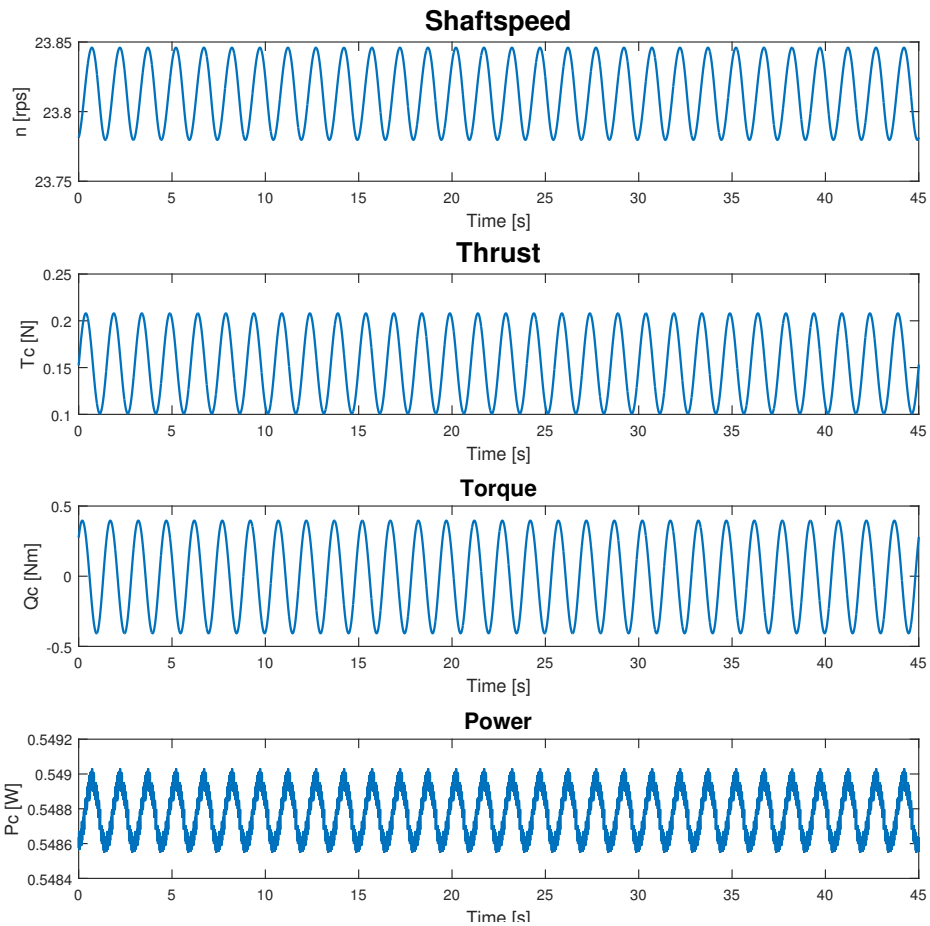

Figure 5.3.2: Torque result with simulated regular waves and current

5.4 Power Control

The power control is based upon controlling the power consumption of the thruster motor. The motor torque is found by shaft speed feedback. Power control is easier to measure than torque and useful when it comes to power management systems (PMS) with functions like load shedding. However when this method is $P = 0$ it will create a singularity problem which can result in system crash and blackout situations. In Appendix A.4 the switching function is given where for controller 3 (power) there is a fail-safe that switches to torque controller when power control is used close to zero shaft speed.

5.4.1 Controller

Power controller is given with a feedback controller and can be described with

$$P_r = Q_r 2\pi n_r = |T_r|^{3/2} \frac{2\pi K_{QC}}{\sqrt{\rho} D K_{TC}^{3/2}} \tag{5.4.1}$$

$$Q_{cp} = \frac{P_{rs}}{2\pi |n|} = \frac{K_{QC}}{\sqrt{\rho} D K_{TC}^{3/2}} \frac{\text{sign}(T_r)|T_r^{3/2}}{|n|} \tag{5.4.2}$$

5.4.2 Simulation Results

Figure 5.4.1 show simulated results from forces applied by (5.1.1) when the power controller is used.

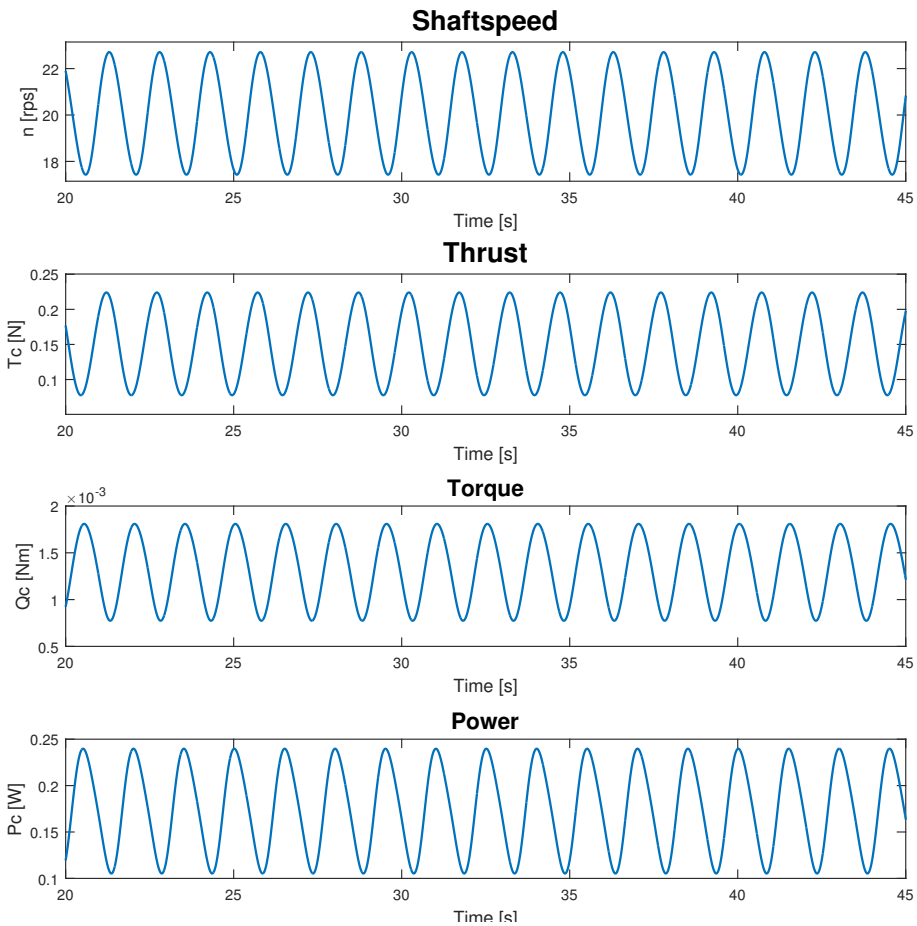

Figure 5.4.1: Power result with simulated regular waves and current

5.4.3 Experimental Results

Figure 5.4.2 show experimental results from forces applied by (5.1.1) when the power controller is used.

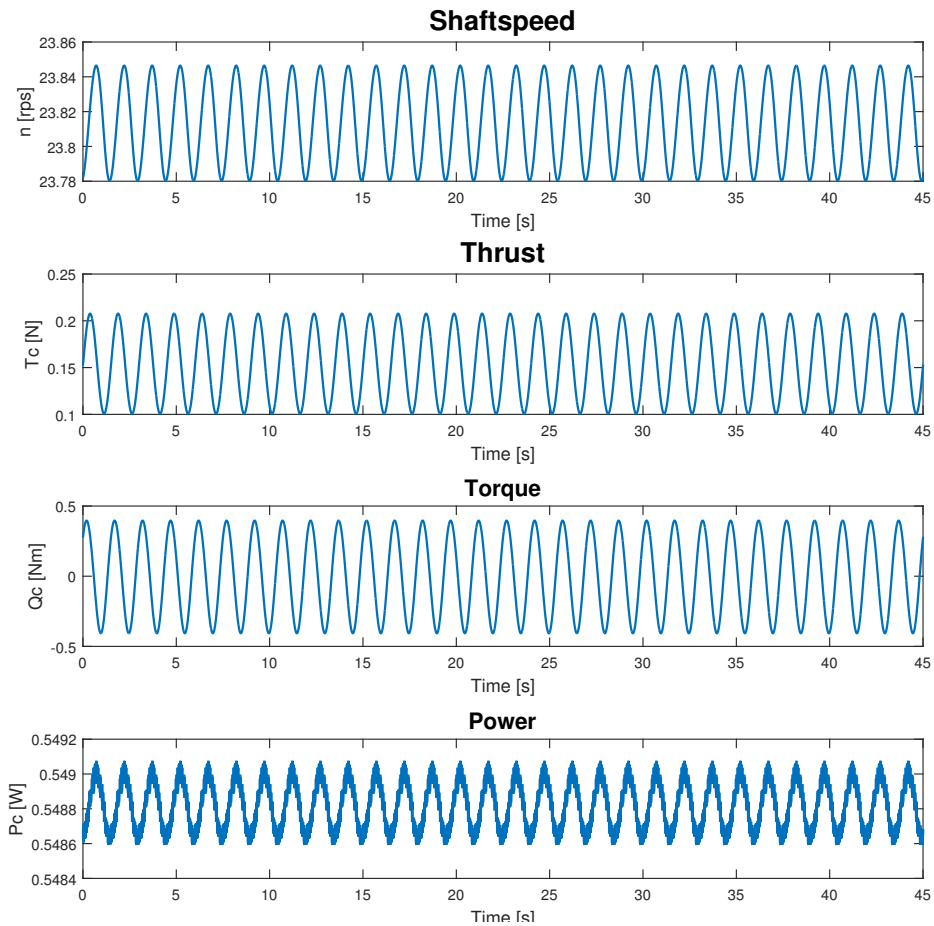

Figure 5.4.2: Power result with simulated regular waves and current

5.5 Combined Torque and Power Control

In order to counteract the singularity which occurs when shaft speed goes towards zero, a combined controller of torque and power is proposed by Sørensen (2014). Because the torque controller is fairly accurate at low shaft speeds and closely related to power, it makes the best controller to combine power with.

5.5.1 Controller

In order to combine these two controllers, a weighting function is needed to be implemented. Again, using Sørensen (2014) formulation, the function is written as

$$\alpha_c(n) = e^{-k|pn|^r} \tag{5.5.1}$$

Having $\alpha_c(n)$ for all n, it means the weighting function will become

$$\lim_{n\to\infty} \alpha_c(n) = 0$$
$$\lim_{n\to 0} \alpha_c(n) = 1$$

for k, p and r as positive constants. Combining this with (5.3.1) and (5.4.2) the equation becomes

$$Q_{cc} = \alpha_c(n)Q_{cq} + (1 - \alpha_c(n))Q_{cp} \tag{5.5.2}$$

The constants are given as $[k, p, r] = [1, 0.5, 4]$ which means between $n \in [-0.2, 0.2]$ the controller is purely torque regulated and when $n \in [-3.5, 3.5]$ the controller is purely power regulated. The switching mechanic is also shown on Figure 5.5.1.

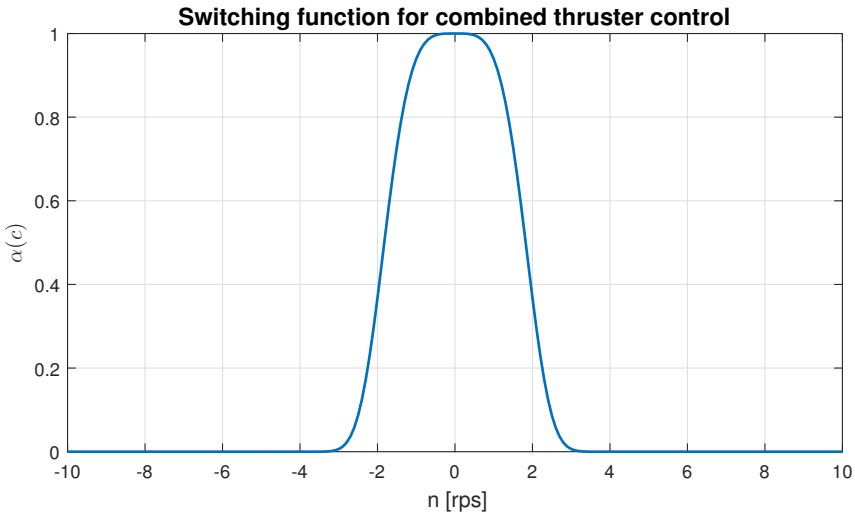

Figure 5.5.1: Switching function with constants $[k, p, r] = [1, 0.5, 4]$

5.5.2 Simulation Results

The combined controller is not subjected to the constant current force as seen in (5.1.1). Instead, only wave loads are taken into account. This is to visualize how the switching mechanic between torque and power controller impacts the results.

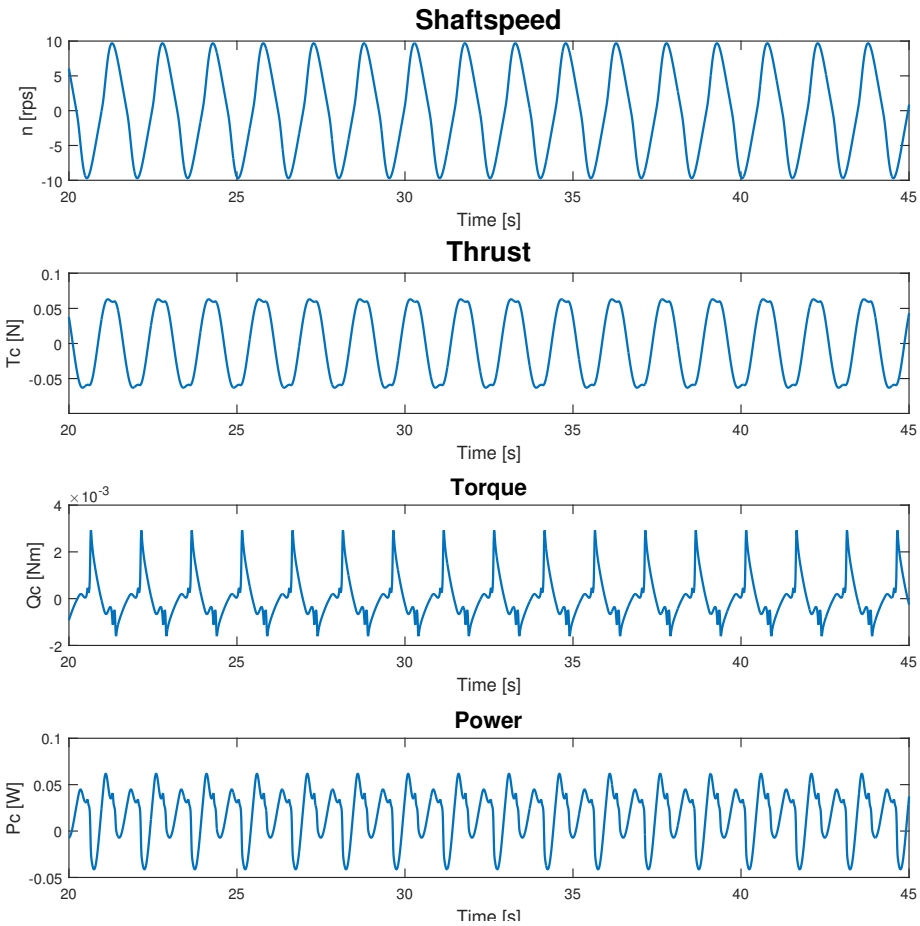

Figure 5.5.2: Combined power and torque controller result with simulated regular wave force

5.5.3 Experimental Results

Figure 5.5.3 show results during experimental testing with the combined controller.

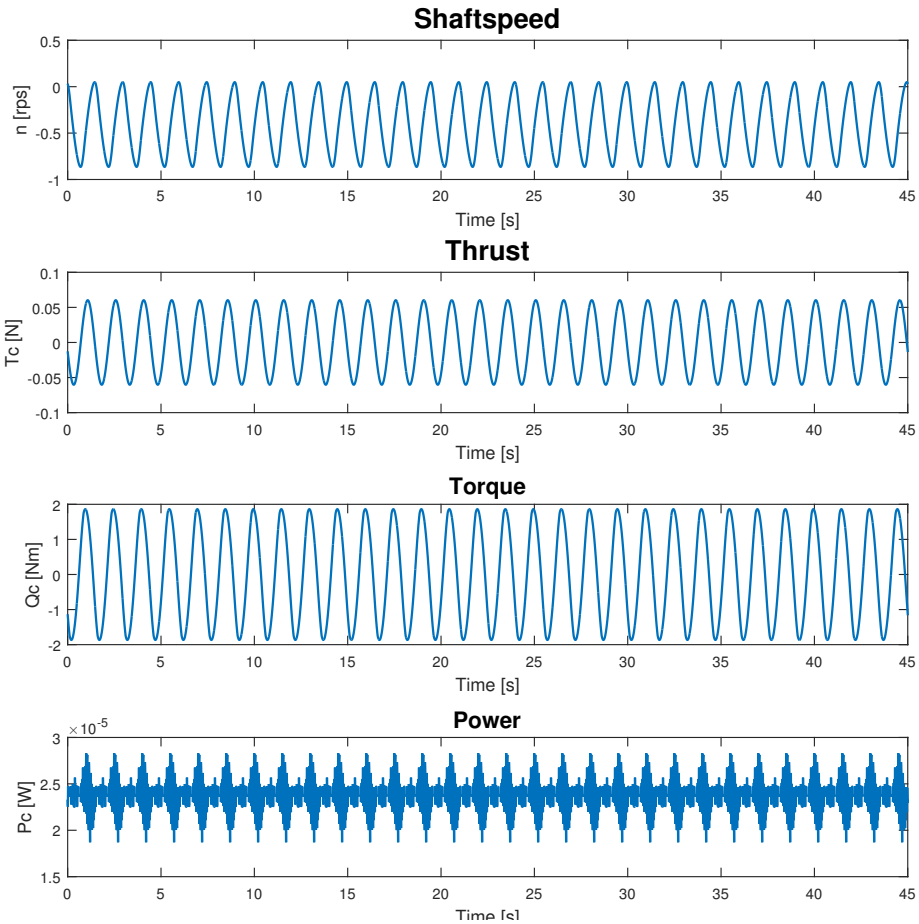

Figure 5.5.3: Combined power and torque controller result with regular wave force applied

5.6 Discussion

There are some big discrepancies between simulated results and experimental results. First and foremost is shaft speed controller. Both tests are conducted with exact parameters and force applied. Through simulation model, the PI controller provides a very high torque error which leads to a high power error. By tuning K_p very low, torque will also become lower, but provides a higher shaft speed fluctuation. Through the experimental testing, power has a high disturbance, but are at a much more consistent level. When utilizing a shaft speed controller, the desired thrust is either provided through shaft speed mapping or thrust which means in general torque and power through simulation will not be taken into account.

The torque and power controller provides a higher fluctuation in shaft speed, but varies very little in torque through simulation. Both results are very similar to what is expected since power is closely related to torque. This is also seen through the experimental results where only marginal differences are found.

The combined controller shows notably differences in shaft speed between simulation and experimental results. Since the results shown is given as the mean value combined with error from controllers, there are a high error provided in simulation while in experimental results, the error is a lot lower. The same is with shaft speed error which shows is much lower in the experimental results.

Chapter 6

Thrust Allocation

6.1 Problem Formulation

As mentioned in the introduction, most marine control systems are overactuated. This is also the case for CSAD. It has 6 azimuth thrusters and 3 DOF which are controllable.
Most marine vessel have main propellers and tunnel thrusters combined with azimuth thrusters to achieve an overactuated system. Because CSAD is only equipped with azimuth thrusters, the thruster system can potentially produce much higher thrust in a DOF than other vessels. For stationkeeping in ice, this will provide a very viable solution due to main environmental comes from current which does not have sudden changes in force direction.

Through this chapter, different algorithms will be simulated and tested on the model. The notation for the thrust allocation problem is created to coincide with the work notes from Skjetne (2015). This means there are some notation differences between this thesis and the original equations. This is done in order to have a more consistent notation through the different thrust allocation formulations.

In order to have a clear deterministic case, a function is created to check how each thrust allocation problem handles generalized forces. The function is written as

$$\boldsymbol{\tau}(t) := \begin{bmatrix} f_x(t) & = & \begin{cases} 0, & \text{if } t < 200 \\ 0.01(t-200), & \text{if } t \geq 200 \end{cases} \\ f_y(t) & = & \begin{cases} 0, & \text{if } t < 200 \\ 0.01(t-100), & \text{if } t \geq 100 \end{cases} \\ f_z(t) & = & 0.01t \end{bmatrix} \tag{6.1.1}$$

This creates a linear force vector for surge and sway and moment vector for yaw at different times such that the system can also be tested for discrepancies at directional changes in force.

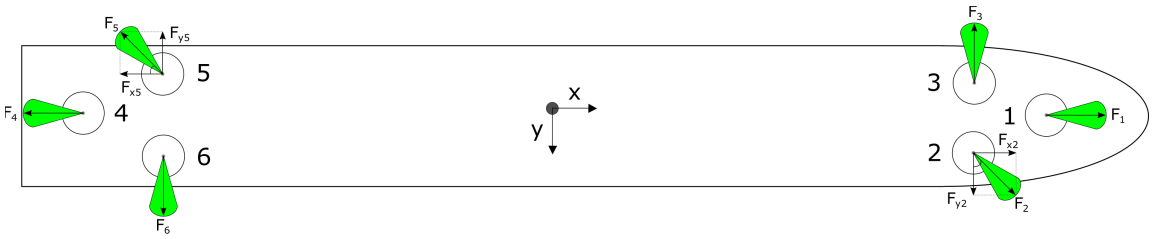

Figure 6.2.1: Thrust angle configuration with fixed thrusters

Table 6.1: Angles for fixed thruster configuration

Thruster 1	Thruster 2	Thruster 3	Thruster 4	Thruster 5	Thruster 6
180°	-135°	90°	0°	45°	-90°

6.2 Fixed Angle Pseudoinverse

Pseudoinverse optimization is considered the easiest approach for solving the thrust allocation problem. The method is often used to find a solution for linear system equations which does not have a unique solution.

Through simulation of different angles, Figure 6.2.1 shows one of the most optimal setup for having maximum control in both positive and negative surge sway and yaw.

Having the thrusters at a fixed angle creates opportunities for faster response at sudden changes in surge, sway and yaw. The downside is that not all thrusters will be able to produce thrust in desired direction at all time. Because of this, both efficiency of fuel consumption and maximum thrust in a direction will go down. The angles from Figure 6.2.1 are given in the Table 6.1.

6.2.1 Controller

The general formulation for the thrust allocation problem with fixed angles is given as

$$\boldsymbol{\tau}_c = \boldsymbol{B}_{3xr}(\boldsymbol{\alpha})\boldsymbol{u}_d \tag{6.2.1}$$

Where $\boldsymbol{\tau}_c$ is commanded torque, $\boldsymbol{B}_{3xr}(\boldsymbol{\alpha})$ is the thrust configuration matrix, r is number of thrusters and $\boldsymbol{u}_d$ is desired thrust. $\boldsymbol{B}_{3xr}(\boldsymbol{\alpha})$ for CSAD becomes $\boldsymbol{B}_{3x6}(\boldsymbol{\alpha})$ and is defined in (4.2.4). With implementation of fixed angles provided from Table 6.1, the thrust configuration becomes a constant matrix.

The desired thrust found from thrust allocation is found by rewriting (6.2.1) as

$$\boldsymbol{u}_d = \boldsymbol{B}^{\dagger}_{3x6}(\boldsymbol{\alpha})\boldsymbol{\tau}_c \tag{6.2.2}$$

Where $\boldsymbol{B}^{\dagger}_{3x6}(\boldsymbol{\alpha})$ is the pseudoinverse solution for the coinciding thrust configuration.

6.2.2 Simulation Results

Figure 6.2.2 show simulated results of function (6.1.1) when all thrusters are in fixed position described by Table 6.1.

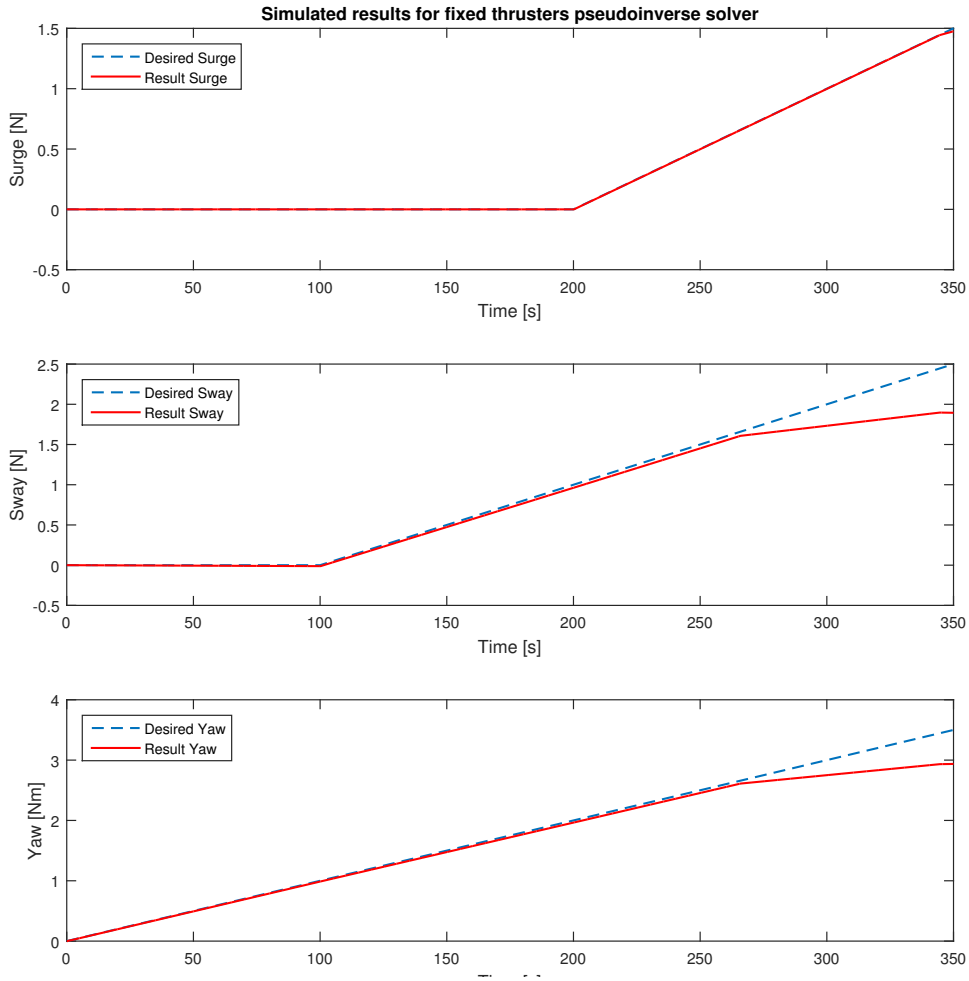

Figure 6.2.2: Simulated results from thrust allocation test at fixed thruster angles

6.2.3 Experimental Results

Figure 6.2.3 show experimental results of function (6.1.1) when all thrusters are in fixed position described by Table 6.1.

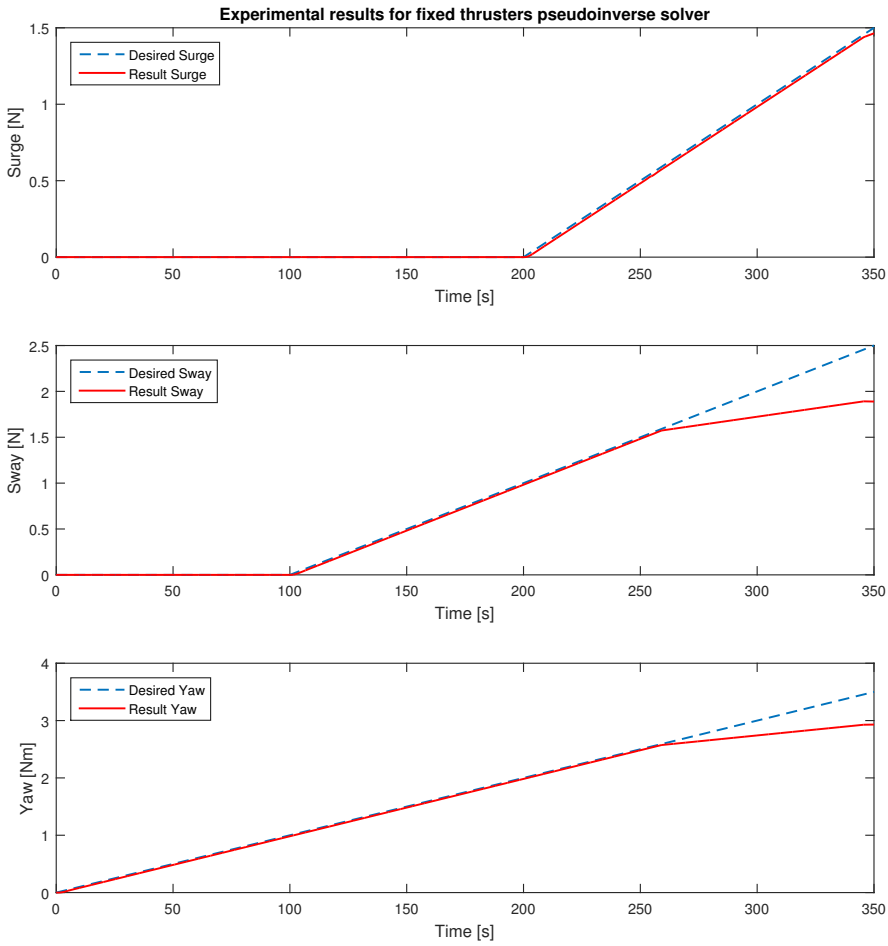

Figure 6.2.3: Experimental results from thrust allocation test at fixed thruster angles

6.3 Rotatable Angles Pseudoinverse

For rotatable thrusters, thruster configuration is no longer constant. The drawback with using a normal pseudoinverse algorithm is singularities which might lead to unreasonable high thrust demand in order to solve the thrust allocation problem. For example if all 6 thrusters are pointing in sway direction and suddenly surge forces are commanded, the thrusters will wait to produce any thrust until they are within a reasonable angle to produce thrust in surge.

This was the motivation for Sørdalen (1997) pseudoinverse optimization with singularity handling which is used to solve the following thrust allocation problem.

6.3.1 Controller

In order to incorporate singularity handling, (6.2.1) is rewritten as

$$\mathbf{u}_d = \boldsymbol{B}^{\dagger}_{3x6}(\alpha)\boldsymbol{\tau}_c - \boldsymbol{s} \tag{6.3.1}$$

Where, $\boldsymbol{B}^{\dagger}_{3x6}(\alpha)$ is the pseudo inverse configuration. A slack variable $\boldsymbol{s}$ is also incorporated to handle cases where no solution is found as the example earlier stated.

$\boldsymbol{B}^{+}_{3x6}(\alpha)$ is defined by Moore-Penrose pseudo inverse as

$$\boldsymbol{B}^{\dagger}_{3x6}(\alpha) = \boldsymbol{V}\boldsymbol{S}^{\dagger}\boldsymbol{U}^{T} \tag{6.3.2}$$

$\boldsymbol{S}^{\dagger}$ is a diagonal 3x3 matrix containing the singular values of $\boldsymbol{B}_{3x6}(\alpha)$. By ensuring large singular values are set to zero, the thruster will not produce excessive thrust if they are at a poor angle. It's through controlling $\boldsymbol{S}^{\dagger}$ possibilities for different optimization can be done. For CSAD, values of $\boldsymbol{S}^{\dagger}$ are chosen to be zero whenever one of its values are ten times as high as the others. The impact tuning of this variable can have is seen on Figure 6.3.2. Any higher values will only provide more inefficient thrust produced.

In order to account for forbidden zones within the pseudoinverse algorithm, a weighting matrix $\boldsymbol{K}$ is implemented into (6.3.1) as

$$\mathbf{u}_d = \boldsymbol{K}\boldsymbol{B}^{\dagger}_{3x6}(\alpha)\boldsymbol{\tau}_c - \boldsymbol{s} \tag{6.3.3}$$

$\boldsymbol{K}$ is a 6x6 diagonal weighting matrix which ensures there are no desired forces produced on the different thrusters when inside a forbidden zone found in Section 3.2.1. When a thruster has an angle inside a forbidden zone, the coinciding weighting gain in $\boldsymbol{K}$ becomes 0. After it has passed the forbidden zone, the weighting gain goes back to 1. This weighting function can be seen in Appendix A.2, line 109-148.

Optimal angle

Since angles are rotating, there is also need for finding optimal angles whenever the commanded forces $\boldsymbol{\tau}_c$ is changed.

The approach was also used by Sørdalen (1997) and is formulated as

$$\boldsymbol{\tau}_e = \boldsymbol{B}_{3x12}(\boldsymbol{\alpha})\boldsymbol{K}_e\boldsymbol{u}_e \tag{6.3.4}$$

Where $\boldsymbol{B}_{3x12}(\boldsymbol{\alpha})$ is the extended thrust configuration, $\boldsymbol{K}_e$ is a 12x12 identity matrix and $\boldsymbol{u}_e$ is the thrust components decomposed in X- and Y direction. By using Moore-Penrose pseudo-inverse, the extended thrust components can be found as

$$\boldsymbol{u}_e = \boldsymbol{K}_e^{-1}\boldsymbol{B}_e^{\dagger}\boldsymbol{\tau}_e \tag{6.3.5}$$

Here, $\boldsymbol{B}_e^{\dagger}$ is the pseudo-inverse. By utilizing the trigonometric relation between forces in X- and Y direction the optimal angle can be found with

$$\alpha_i = \tan2^{-1}(u_{iy}, u_{ix}) \tag{6.3.6}$$

Which is furthermore checked up against forbidden zones and limited by rotation rate (seen in Appendix A.2 line 195-368) before sent to the thrusters and looped back into thrust configuration as the new angle. The allocation can be seen in Appendix E.1 and E.2.

In Figures 6.3.3 and 6.3.5 it's also shown results from thrust allocation during simulated and experimental TAPM with irregular waves at Hs = 0.1m which is scaled up to real values as 9m. Results from the tests are also presented by Bjørnø (2016), taking into account other aspects of TAPM.

6.3.2 Simulation Results

Figure 6.3.1 show simulated results of function (6.1.1) when all thruster angles are rotatable.

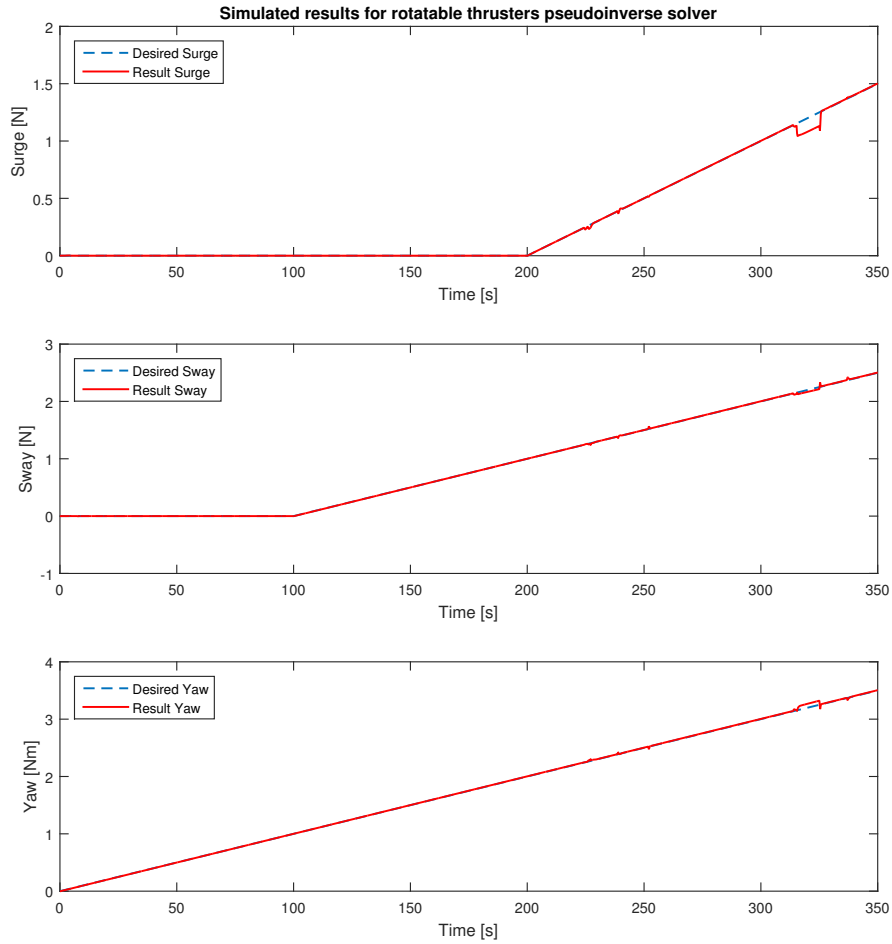

Figure 6.3.1: Simulated results from thrust allocation test

Figure 6.3.2 show the simulated difference between singular value decomposition optimization. S[3,3] is the controllable variable chosen to be limited by S[2,2].

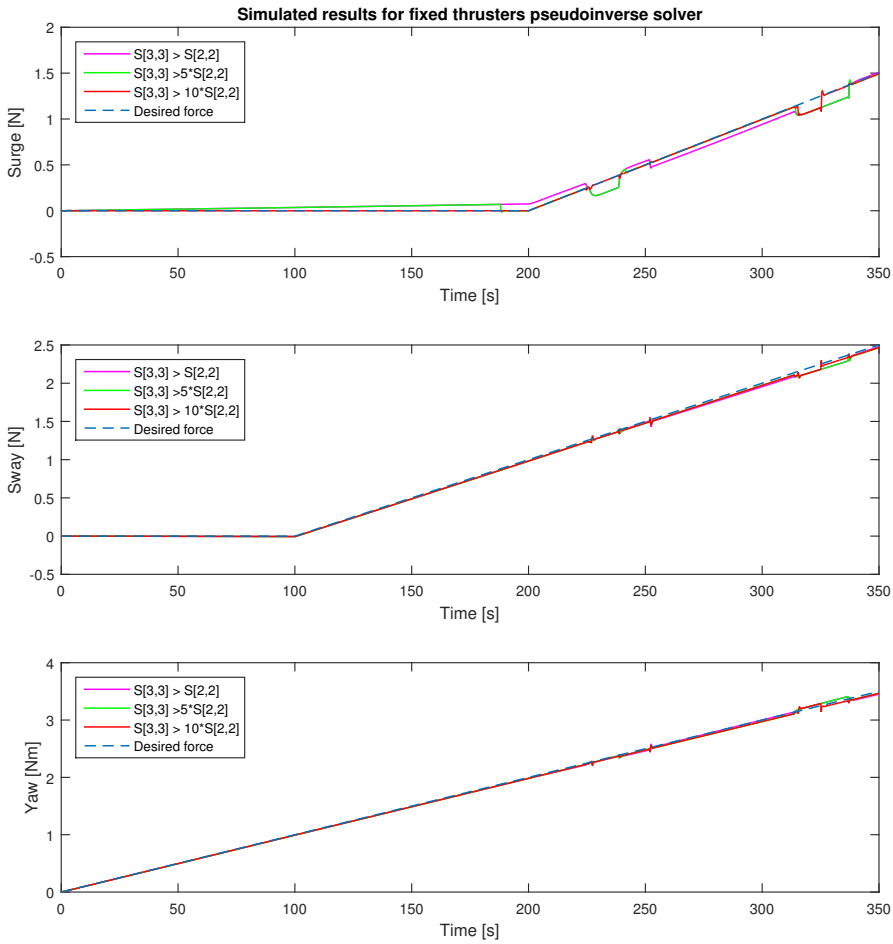

Figure 6.3.2: Simulated results from different singular value decomposition

Figure 6.3.3 show the simulated results from irregular wave spectrum where Hs = 0.1 and TAPM is applied.

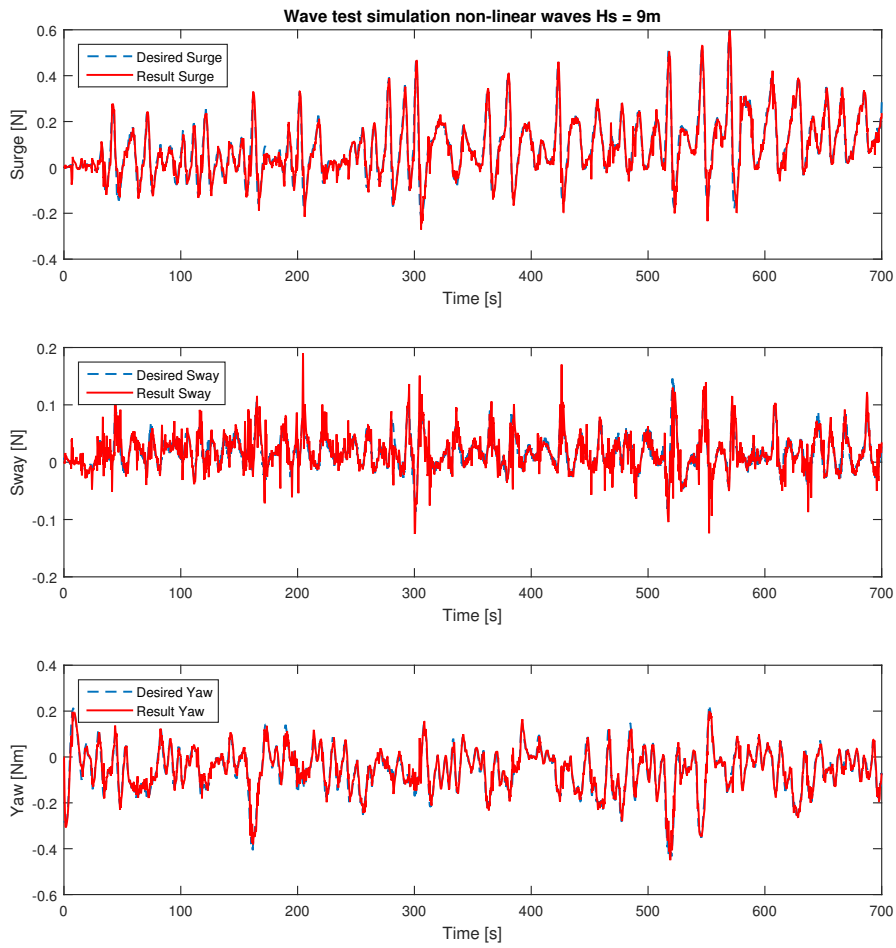

Figure 6.3.3: Simulated results with TAPM in irregular waves

6.3.3 Experimental Results

Figure 6.3.4 show experimental results of function (6.1.1) when all thruster angles are rotatable.

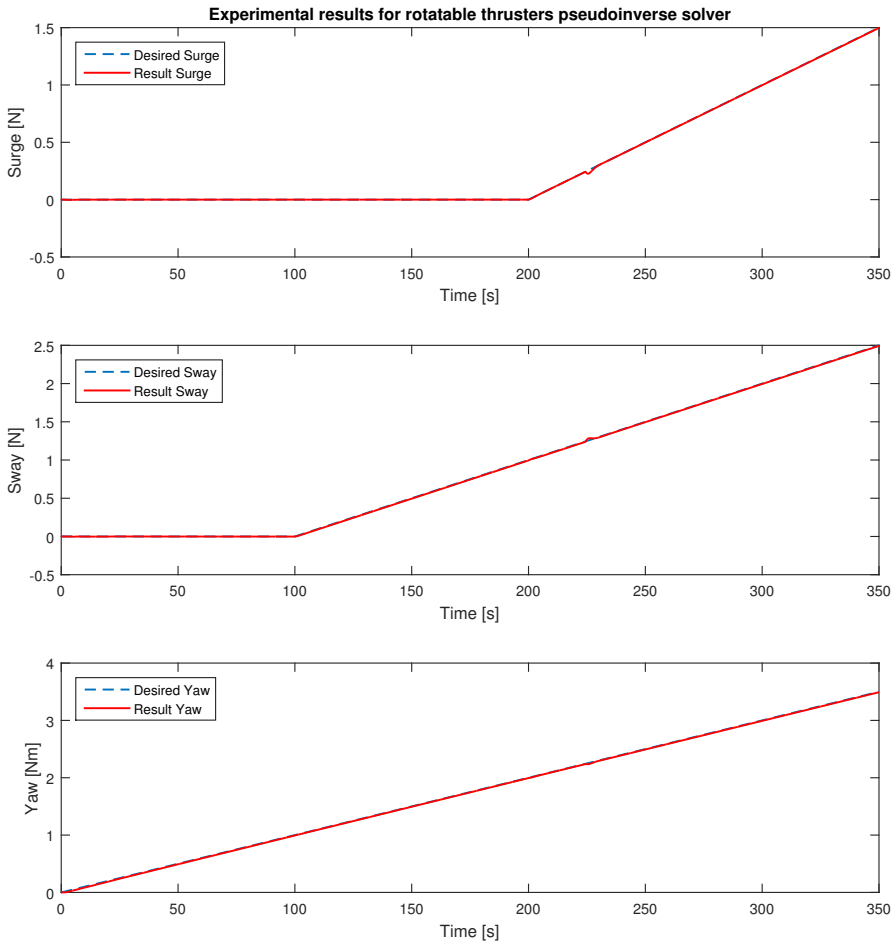

Figure 6.3.4: Experimental results from thrust allocation test

Figure 6.3.5 show the experimental results from irregular wave spectrum where Hs = 0.1 and TAPM is applied.

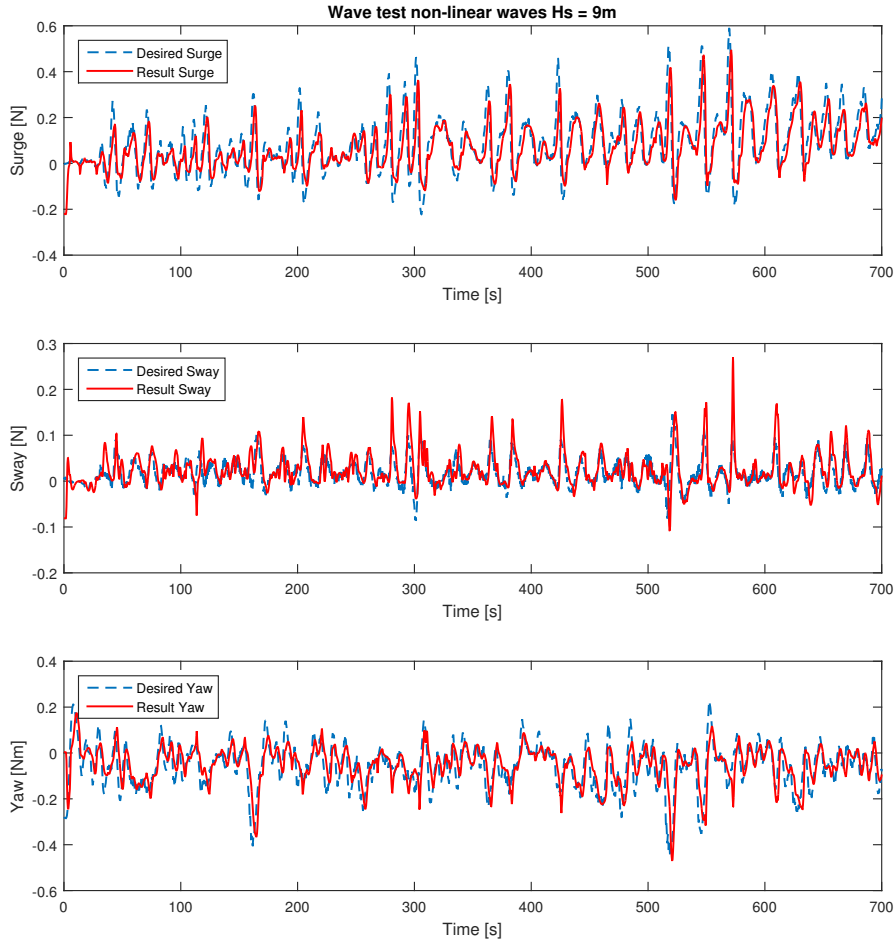

Figure 6.3.5: Experimental results with TAPM in irregular waves

6.4 Quadratic Optimization

Quadratic regulators (QR) are more complex algorithms than pseudoinverse optimization and uses cost functions to optimize the given problems. Cost functions are weighting matrices which determine what part of the optimization is most important to satisfy. For thrust allocation, there are mainly two considerations taken into account. The first consideration is how important it is for the system to maintain desired thrust even though the angles are not yet optimal, the second consideration is how much actual angle can deviate from the desired angle. There is also a third consideration as can be seen in (6.4.1) where the the slack variable are given weighting matrices too, s.t. it's possible for $\boldsymbol{B}_{3xr}(\boldsymbol{\alpha})\boldsymbol{u}_d$ to deviate a bit from τ_c on the same principle as with pseudoinverse algorithm.

6.4.1 Controller

Having prescribed angles makes the quadratic algorithm very easy to write. An explicit solution for parametric quadratic programming was first developed by Tøndel et al. (2003), but applied for marine application by Johansen et al. (2005). The optimization algorithm is formulated as

$$\boldsymbol{J}_{QR}(\boldsymbol{u}_d, \boldsymbol{s})_{\min} = \boldsymbol{u}_d^T \boldsymbol{W} \boldsymbol{u}_d + \boldsymbol{s}^T \boldsymbol{Q} \boldsymbol{s} \tag{6.4.1}$$

Which are subject to the following constraints

$$\begin{aligned} \boldsymbol{B}_{3xr}(\boldsymbol{\alpha})\boldsymbol{u}_d &= \boldsymbol{\tau}_c + \boldsymbol{s} \\ \boldsymbol{u}_{\min} 0 \leq \boldsymbol{u}_d &\leq \boldsymbol{u}_{\max} \end{aligned}$$

Here, $W \in R^{6\times 6}$ is a diagonal weighting matrix for thrust and $Q \in R^{3\times 3}$ is a diagonal weighting matrix for the slack variables. In order to ensure the slack variables are close to zero, $\boldsymbol{Q}$ is set to the typical value of 1000 times larger than $\boldsymbol{W}$ according to Ruth (2008).

Implementation of a QR proved to be difficult when the simulink model is compiled into C++. MATLAB has a stand alone solver called '*quadprog*[1]' which is described as

$$min(\boldsymbol{u}_d) = \frac{1}{2}\boldsymbol{u}_d^T H \boldsymbol{u}_d + f^T \boldsymbol{u}_d \begin{cases} \boldsymbol{B}_{3x6}^{\dagger}(\alpha) \cdot \boldsymbol{u}_d = \boldsymbol{\tau}_c + \boldsymbol{s} \\ \boldsymbol{u}_{\min} \leq \boldsymbol{u}_d \leq \boldsymbol{u}_{\max} \end{cases} \tag{6.4.2}$$

With notation corrected to fit earlier constraints.

$$\begin{aligned} \boldsymbol{H} &= \begin{bmatrix} \boldsymbol{I}_{6x6} & \boldsymbol{0}_{6x3} \\ \boldsymbol{0}_{3x6} & 1000\boldsymbol{I}_{3x3} \end{bmatrix} \boldsymbol{f} = \begin{bmatrix} \boldsymbol{0}_{9x1} \end{bmatrix} \\ \boldsymbol{u}_{\max} &= \begin{bmatrix} 1.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5 & 0.01 & 0.01 & 0.01 \end{bmatrix} \\ \boldsymbol{u}_{\min} &= \begin{bmatrix} -0.85 & -0.85 & -0.85 & -0.85 & -0.85 & -0.85 & -0.01 & -0.01 & -0.01 \end{bmatrix} \end{aligned}$$

To allow for switching between fixed angles and azimuth angles, the forbidden zones are taken into account the same way as seen in Section 6.3.1. Each angle is checked up against forbidden zones, and if they are within the section, the corresponding thruster is taken out from the equation.

By choosing to loop back $\boldsymbol{u}_d$ from last time step into the algorithm again, a "warm start" is implemented which reduces computational time greatly.

[1](Mathworks: Quadratic programming)

6.4.2 Simulation Results

Figure 6.4.1 show simulated results of function (6.1.1) when all thrusters are in fixed position described by Table 6.1.

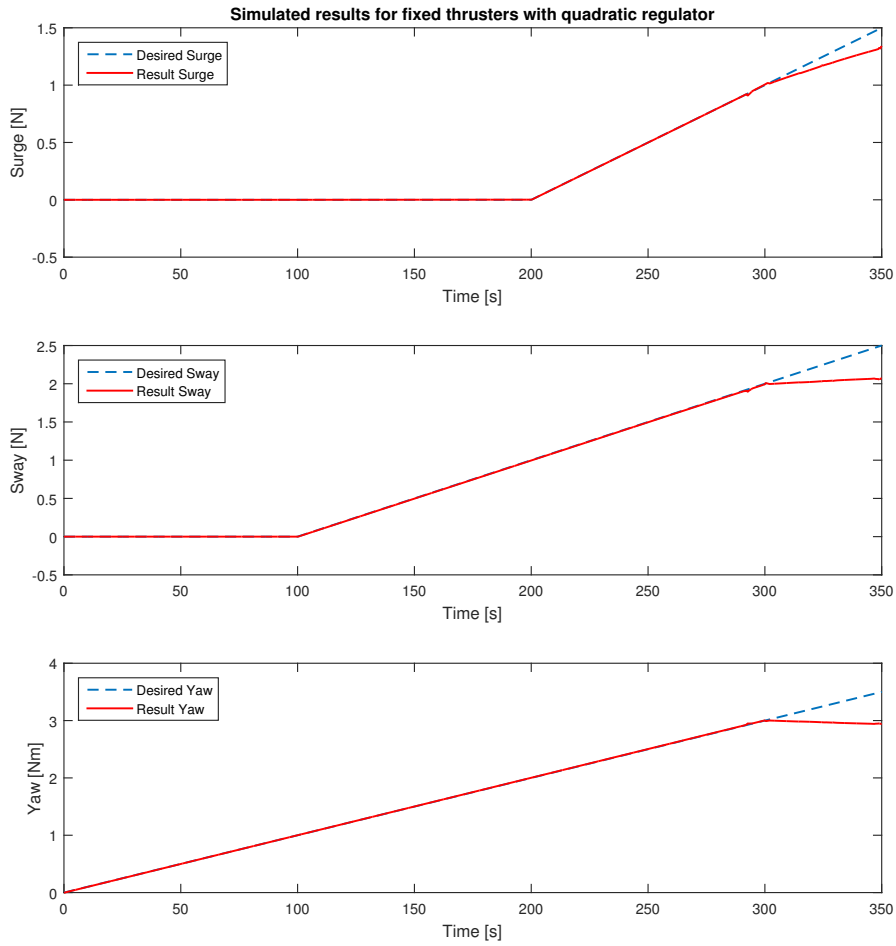

Figure 6.4.1: Simulated results from thrust allocation test for fixed thruster angles QR

Figure 6.4.2 show experimental results of function (6.1.1) when all thruster angles are rotatable.

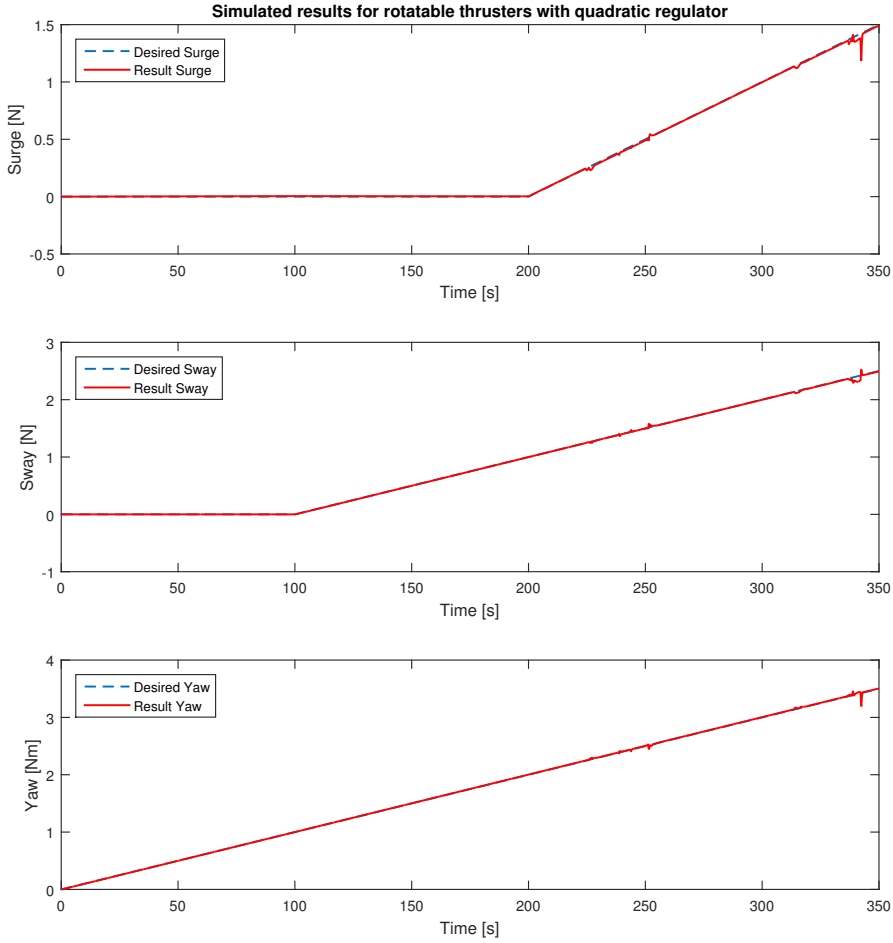

Figure 6.4.2: Simulated results from thrust allocation test for azimuth thruster angles QR

6.5 Real-time Implementation of Quadratic programs

Quadratic programming (QP) through MATLAB's Simulink is done through S-functions. By using MATLABS own stand alone solver '*quadprog*' as done in Section 6.4, the function is not possible to generate code for. This means for real time application, the function needs to be created elsewhere which proved to be a problem through this thesis.

One solution which is to use LabVIEW's own quadratic solver, see (National Instruments), which uses the same principle as MATLAB. The solver is possible to compile as a real-time program and through VeriStand it should be able to receive and provide matrices to Simulink. The method were tested, but there was a problem with input matrices to LabVIEW through VeriStand because they need to be defined as shared variables and a 1 dimensional array. Defining them as 1D matrices in LabVIEW created them as 0x1 matrices in VeriStand. To the author's best knowledge the only solution for the problem is to create a custom device which interpolates 1D arrays inside VeriStand. The tried solution can be seen in Appendix D.

Another solution might be to utilize ACADO Toolkit and implement them in C-code. This solution have been used by Ingebritsen (2016) where model predictive control (MPC) is implemented through ACADO.

A final solution is to possibly create a custom device on compactRIO which utilize an on-shore computer to run Simulink. This would possibly provide a way to work around the compiling problem.

6.6 Discussion

There are some discrepancies from simulation and testing on CSAD with azimuth thrusters. When the system is running through CSAD the allocation are able to better keep up with thrust demands, and are able to very quickly compensate when a thruster is inside a forbidden zone. For the simulated part, the thruster within a forbidden zone is much more noticeable. It's not desirable to have a gap that big in simulation, and a way to perhaps counteract this is described in Chapter 7 under further work.

When the thrusters are fixed, there is not much difference between the two methods. For both results, there is a slight delay from what is produced in thrust and what is desired. In experimental results the surge forces are dominating the error up to 0.02N. Interestingly from simulation with exact same parameters given, the error is not found in surge, but are dominating in sway with up to 0.05N discrepancy. This discrepancy can only be found when static experiments are conducted and are not shown through dynamic force simulation.

From the TAPM testing in both simulation and experimental results, the thrust allocation is able to provide good results. The key difference is in simulation it seems like the thrust demand is better held, but often overshot. While in the experimental results, the thrust is always a little bit behind. It's worth mentioning at the time of these tests, the singular value decomposition limit was set to only 2 times higher before set to zero, but has later on been increased to 10 times higher before set to zero due to the results from Figure 6.3.2.

As for the quadratic solver, there is no experimental results to compare to as described in Section 6.5, but by comparing the pseudoinverse simulation and quadratic simulation, it's possible to see some key differences. When the thrusters are rotatable, the quadratic solver is much more accurate, and are able to compensate much faster for forbidden zones. When the thrusters are fixed, the quadratic solver consistently manages to provide the correct thrust needed.

Chapter 7

Conclusion

7.1 Thrust control

The thrust control system only has an estimated torque coefficient found from bollard pull tests. This results in that the torque and power estimation might produce a slightly different torque than seen through the thrust controls here.

There is also an error which produces very high error in power and torque in shaft speed simulation. The reason for being produced in simulation and not through experimental results are not fully understood and might be a point of interest in further work.

Overall the thrust control system is fully operational and can easily switch between the four controllers while real-time experiments are conducted as seen in Appendix D. It's worth noting when switching to shaft speed controller, there is a high spike in desired thrust before it manages to correct itself. This can be seen in Appendix A.4 where the different controllers are switched between.

It's worth noting that when conducting real-time testing, switching controller to shaft speed may produce a high spike in force before settling down at desired values. This is due to the differences between each controller with regards to shaft speed.

7.2 Thrust allocation

Because the quadratic regulator was not implemented into CSAD, the allocation is only sub-optimal. Although this is true, the constrained pseudoinverse thrust allocation which is used for CSAD still produces very good results. From Figure 6.3.5, the difference in desired and actual thrust can be seen for Hs = 9m with irregular waves. The thrust allocation is able to provide a good result corresponding to commanded forces provided by the hybrid controller created by Bjørnø (2016). The largest discrepancies occurs between desired and actual thrust when there are sudden changes in desired force. This is because each thruster will rotate to their new optimal angle, and while this is happening, the thrust changes rapidly to compensate for the changes.

Constraints set on thruster angles inside forbidden zones may have been set too strict. Although the other thrusters are compensating for lack of thrust from the one inside forbidden zone, there is still going to be some discrepancies. This is mainly seen from deterministic tests as shown in Chapter 6. For testing done with fast changes in force the constraints are not as impactful as the optimal thruster angle are constantly changing. In next section there is provided an alternative solution which base upon how long it's inside a forbidden zone.

The quadratic regulator results from simulation are still giving a better performance than pseudoinverse solution. The problem with a quadratic regulator is that computation takes a very long time. With the full thruster system simulated, it uses approximately 0.5 seconds to simulate 1 second. The computation is done by a relatively new computer, which means that an old computer may not be able to keep up with the computation power needed.

Overall the system is provides good results and are as the thruster control easy to switch between fixed thruster mode and azimuth thruster mode through VeriStand, which can be seen Appendix D.

7.3 Further work

There are a few improvements which can be addressed in order to create a more optimal system. These improvements are provided through the list as seen below.

- **Find correct torque coefficients**
  As mentioned in Section 4.2, the torque coefficients are only estimated from thrust coefficients. In order to find correct values, a cavitation tank test is most likely needed to be conducted with an unbound thruster.
- **Reverse thrust coefficients**
  Through last minute testing it was found that the reverse thrust coefficients did not coincide with correct forces. Because of this, the reverse thrust coefficients is chosen to be equal to forward coefficients. In simulations this provides a correct force curve, but are not verified.
- **Implementing Quadratic regulator as a real-time solver**
  This is discussed in Section 6.5 where different solutions which possibly can work are provided.
- **Implementing current sensors on all thrusters and mapping volt to power**
  In order to have a complete mapping of power and torque control on all thrusters, more current sensors are needed. It's also needed to measure ampere while testing is undergoing to provide a correct mapping.
- **Finding sensitivity from control signal (PWM) which is sent to each thruster** In Section 4.4 it's seen the control signal is found to not being sensitive at very low changes. The result can come from each motor not being sensitive enough to pick up changes around 0.01. This might lead to minor discrepancies from commanded thrust to what the model is actually producing. Especially when low thrust is desired.
- **Reducing impact of forbidden zones in thrust allocation**
  The forbidden zones for CSAD in this thesis are only thruster-thruster interaction. The current way used to deal with these zones is to switch thrust off and force the desired angle outside the zone. By measuring how much losses each thruster experience it might be possible to map out how much extra force is needed to counteract the effect. Some theory about this is provided by the American Bureau of Shipping (2014).

Bibliography

Aero-naut (n.d.). Aero-naut. http://www.aero-naut.de/en/home/[Accessed last: 10/07-2016].

American Bureau of Shipping (2014). Guide for dynamic positioning systems.

Bjørnø, J. (2016). Thruster-assisted position mooring of C/S Inocean Cat I Drillship. Norwegian University of Science and Technology (NTNU), Trondheim.

Carlton, J. (1994). *Marine Propellers and Propulsion*. Butterworth-Heinemann, 1st edition.

Fossen, T. I. (2012). *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley.

Hespanha, J. (2002). *Tutorial on Supervisory Control*. Department of Electrical and Computer Engineering. University of California.

Ingebritsen, T. B. (2016). *Scenario- and Optimization-Based Control of Marine Electric Power Systems*. PhD thesis, Norwegian University of Science and Technology (NTNU), Trondheim.

Johansen, T. A., Fossen, T. I., and Berge, S. P. (2004). Constrained nonlinear control allocation with singularity avoidance using sequential quadratic programming. volume 12. IEEE Transaction on Control System Technology.

Johansen, T. A., Fossen, T. I., and Tøndel, P. (2005). Efficient optimal constrained control allocation via multi-parametric programming. AIAA Journal of Guidance, Control and Dynamics.

Mathworks: Quadratic programming. Quadprog. http://se.mathworks.com/help/optim/ug/quadprog.html [Accessed last: 02/07-16].

National Instruments. Quadratic programming vi. http://zone.ni.com/reference/en-XX/help/371361M-01/gmath/quadratic_programming/[Accessed last: 04/07-2016].

Oosterveld, M. W. and Oossanen, P. V. (1975). Further computer-analyzed data of the wageningen b-screw series.

Robotis (n.d.). Dynamixel mx-106. http://support.robotis.com/en/product/dynamixel/mx_series/mx-106.htm [Accessed last: 25/05-16].

Ruth, E. (2008). *Propulsion and thrust allocation on marine vessels*. PhD thesis, Norwegian University of Science and Technology (NTNU), Trondheim.

Scibilia, F. and Skjetne, R. (2012). Constrained control allocation for vessels with azimuth thrusters. volume 45. 9th IFAC Conference on Manoeuvring and Control of Marine Craft.

Skjetne, R. (2015). *Work notes: ROV control modes*. Department of Marine Technology, NTNU.

Skorpen, T. (2014). Designing for the arctic- cat I arctic drilling unit. St. Johns.

Smogeli, Ø. N., Sørdalen, O. J., and Ruth, E. (2005). Experimental Validation of Power and Torque Thruster Control. *13th Mediterranean Conference on Control and Automation, Cyprus.*

SNAME (1950). Nomenclature for treating the motion of a submerged body through a fluid, technical and research bulletin. Number 1-5. The society of Naval Achitects and Marine Engineers (SNAME).

Sørdalen, O. J. (1997). Optimal Thrust Allocation for Marine Vessels. *Control Engineering Practice*, 5(9):1223–1231.

Sørensen, A. J. (2014). *Marine Control Systems, Propulsion and Motion Control of Ships and Ocean Structures.* Department of marine technology.

Thrustmaster (2014). Thrusters on semi-submersibles and the coanda effect. https://www.thrustmaster.net/semi-submersibles-coanda-effect/ [Accessed last: 17/04-16].

Tøndel, P., A, J. T., and Bemporad, A. (2003). An algorithm for multi-parametric quadratic programming and explicit mpc solutions. Automatica.

Van Dijk, R. R. T. and Aalbers, A. B. (2001). 'what happens in water' the use of hydrodynamics to improve DP. Maritime Research Institute Netherlands (Wageningen).

van Lammeren, W. P. A., van Manen, J. D., and Ooseterveld, M. W. C. (1969). The Wagening B-Screw Series. SNAME.

Wassink, A. and van der List, R. (2013). Development of solutions for arctic offshore drilling. Society of Petroleum Engineers (SPE) International.

Appendix A

Matlab Scripts

A.1 Startup File

This is the script implemented on CSAD in order to run the thruster system.

```
%% Startup file for CSAD thruster control system

%% Model vessel parameters
lambda = 90;          % Scaling parameter
Rho_m = 1000;                              % Water density tank [Kg/m^3]
D = 3.0/100;                                % Propeller diameter model vessel  [m]
T_max = 1.5034;       % Max produced thrust [N]
I_s = 25000 /747225;                       % Moment of inertia [kg*m^2]
Rho = Rho_m;
Max_rotation = 12*sqrt(lambda);             % Max rotation of thrusters [1/s]
Max_thrust = 1.5;                          % maximum force for each thruster
Min_thrust = -0.85;
Max_Acceleration = 5;                       % Max propeller acceleration [1/s^2]
Thruster.T1 = [ 96.1/90  0          180   2];     %[m,m,Deg,Deg/s]
Thruster.T2 = [ 84.1/90  9.9/90     -135  2];     %[m,m,Deg,Deg/s] 45
Thruster.T3 = [ 84.1/90 -9.9/90     90 2];     %[m,m,Deg,Deg/s] -45
Thruster.T4 = [-104.8/90 0          0 2];     %[m,m,Deg,Deg/s] 179.5
Thruster.T5 = [-89.2/90 -14.9/90    45 2];     %[m,m,Deg,Deg/s] 135
Thruster.T6 = [-89.2/90  14.9/90    -90 2];     %[m,m,Deg,Deg/s] -135

% Thruster control
eps_c = 5;          % Constant for switching between positive and negative thrust
n_c = 5;            % Switching width between K_tc and K_tcr
omega_r0 = 0.90;    % Natural frequency [1/s]
zeta_r = .78;       % Damping ratio [-]
H1 = tf(omega_r0^2,[1 2*zeta_r*omega_r0 omega_r0^2]);
hd = c2d(H1,0.01,'foh');
```

```
% Transfer function for Engine
H2 = tf(1,[0.02 1]);
hd2 = c2d(H2,0.01,'zoh');

Kp = 3.3;          % Core controller gain [-]
nd_slew = 2*90; % Max RPS rate [1/s^2]
K_omega = 0.3;     % Linear friction coefficient [-]
epsilon = 0.5;     % Constant for friction component
Q_f0 = 0.3;        % Friction constant

% Core thruster parameters for combined torque and power
k_cc = 1;
p_cc = 0.5;
r_cc = 4;

% Thrust coefficients
K_T1f = 0.3763;
K_T2f = 0.3901;
K_T3f = 0.3776;
K_T4f = 0.5641;
K_T5f = 0.4799;
K_T6f = 0.5588;
K_T1r = K_T1f;
K_T2r = K_T2f;
K_T3r = K_T3f;
K_T4r = K_T4f;
K_T5r = K_T5f;
K_T6r = K_T6f;

% Torque coefficients
K_q1f = 0.0113;
K_q2f = 0.0117;
K_q3f = 0.0113;
K_q4f = 0.0169;
K_q5f = 0.0144;
K_q6f = 0.0168;
K_q1r = K_q1f;
K_q2r = K_q2f;
K_q3r = K_q3f;
K_q4r = K_q4f;
K_q5r = K_q5f;
K_q6r = K_q6f;
eps = 1E-5;          % Constant for avoiding singularities

% constraints on thrusters:
%ABS GUIDE Table
x = [ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15];
v = [ 30 26.3 22.8 20.6 19 17.8 16.8 16 15.3 14.7 14.2 13.8 13.3 13
    12.6];

% Thruster 1
```

```
x13 = sqrt((Thruster.T1(1)-Thruster.T3(1))^2+(Thruster.T1(2)-Thruster.T3(2))^2);
Ratio13 = x13/D;
C13 = interp1(x,v,Ratio13);
C13 = round(C13,1);
x12 = sqrt((Thruster.T1(1)-Thruster.T2(1))^2+(Thruster.T1(2)-Thruster.T2(2))^2);
Ratio12 = x12/D;
C12 = interp1(x,v,Ratio12);
C12 = round(C12,1);
C1 = [atan2d((Thruster.T1(2)-Thruster.T3(2)),(Thruster.T1(1)-Thruster.T3(1)))+C13/2 atan2d((Thruster.T1(2)-Thruster.T3(2)),(Thruster.T1(1)-Thruster.T3(1)))-C13/2 ...
      atan2d((Thruster.T1(2)-Thruster.T2(2)),(Thruster.T1(1)-Thruster.T2(1)))-C12/2 atan2d((Thruster.T1(2)-Thruster.T2(2)),(Thruster.T1(1)-Thruster.T2(1)))+C12/2];
C1 = round(C1,1);


% Thruster 2
x21 = sqrt((Thruster.T2(1)-Thruster.T1(1))^2+(Thruster.T2(2)-Thruster.T1(2))^2);
Ratio21 = x21/D;
C21 = interp1(x,v,Ratio21);
C21 = round(C21,1);
x23 = sqrt((Thruster.T2(1)-Thruster.T3(1))^2+(Thruster.T2(2)-Thruster.T3(2))^2);
Ratio23 = x23/D;
C23 = interp1(x,v,Ratio23);
C23 = round(C23,1);
C2 = [ atan2d((Thruster.T2(2)-Thruster.T3(2)),(Thruster.T2(1)-Thruster.T3(1)))+C23/2 atan2d((Thruster.T2(2)-Thruster.T3(2)),(Thruster.T2(1)-Thruster.T3(1)))-C23/2 ...
     atan2d((Thruster.T2(2)-Thruster.T1(2)),(Thruster.T2(1)-Thruster.T1(1)))+C21/2 atan2d((Thruster.T2(2)-Thruster.T1(2)),(Thruster.T2(1)-Thruster.T1(1)))-C21/2];
C2 = round(C2,1);

% Thruster 3
x31 = sqrt((Thruster.T3(1)-Thruster.T1(1))^2+(Thruster.T3(2)-Thruster.T1(2))^2);
Ratio31 = x31/D;
C31 = interp1(x,v,Ratio31);
C31 = round(C31,1);
x32 = sqrt((Thruster.T3(1)-Thruster.T2(1))^2+(Thruster.T3(2)-Thruster.T2(2))^2);
Ratio32 = x32/D;
C32 = interp1(x,v,Ratio32);
C32 = round(C32,1);
C3 = [atan2d((Thruster.T3(2)-Thruster.T2(2)),(Thruster.T3(1)-Thruster.T2(1)))-C32/2 atan2d((Thruster.T3(2)-Thruster.T2(2)),(Thruster.T3(1)-Thruster.T2(1)))+C32/2 ...
```

```
    atan2d((Thruster.T3(2)-Thruster.T1(2)),(Thruster.T3(1)-Thruster.T1(1))-C31/2 atan2d((Thruster.T3(2)-Thruster.T1(2)),(Thruster.T3(1)-Thruster.T1(1)))+C31/2];
 C3 = round(C3,1);


% Thruster 4
x45 = sqrt((Thruster.T4(1)-Thruster.T5(1))^2+(Thruster.T4(2)-Thruster.T5(2))^2);
Ratio45 = x45/D;
C45 = interp1(x,v,Ratio45);
C45 = round(C45,1);
x46 = sqrt((Thruster.T4(1)-Thruster.T6(1))^2+(Thruster.T4(2)-Thruster.T6(2))^2);
Ratio46 = x46/D;
C46 = interp1(x,v,Ratio46);
C46 = round(C46,1);
C4 = [atan2d((Thruster.T4(2)-Thruster.T6(2)),(Thruster.T4(1)-Thruster.T6(1)))-C46/2 atan2d((Thruster.T4(2)-Thruster.T6(2)),(Thruster.T4(1)-Thruster.T6(1)))+C46/2 ...
    atan2d((Thruster.T4(2)-Thruster.T5(2)),(Thruster.T4(1)-Thruster.T5(1))+C45/2 atan2d((Thruster.T4(2)-Thruster.T5(2)),(Thruster.T4(1)-Thruster.T5(1)))-C45/2];
C4 = round(C4,1);

% Thruster 5
x54 = sqrt((Thruster.T5(1)-Thruster.T4(1))^2+(Thruster.T5(2)-Thruster.T4(2))^2);
Ratio54 = x54/D;
C54 = interp1(x,v,Ratio54);
C54 = round(C54,1);
x56 = sqrt((Thruster.T5(1)-Thruster.T6(1))^2+(Thruster.T5(2)-Thruster.T6(2))^2);
Ratio56 = x56/D;
C56 = interp1(x,v,Ratio56);
C56 = round(C56,1);
C5 = [atan2d((Thruster.T5(2)-Thruster.T6(2)),(Thruster.T5(1)-Thruster.T6(1)))-C56/2 atan2d((Thruster.T5(2)-Thruster.T6(2)),(Thruster.T5(1)-Thruster.T6(1)))+C56/2 ...
    atan2d((Thruster.T5(2)-Thruster.T4(2)),(Thruster.T5(1)-Thruster.T4(1))-C54/2 atan2d((Thruster.T5(2)-Thruster.T4(2)),(Thruster.T5(1)-Thruster.T4(1)))+C54/2];
C5 = round(C5,1);

% Thruster 6
x64 = sqrt((Thruster.T6(1)-Thruster.T4(1))^2+(Thruster.T6(2)-Thruster.T4(2))^2);
Ratio64 = x64/D;
C64 = interp1(x,v,Ratio64);
C64 = round(C64,1);
x65 = sqrt((Thruster.T6(1)-Thruster.T5(1))^2+(Thruster.T6(2)-Thruster.T5(2))^2);
Ratio65 = x65/D;
```

```
C65 = interp1(x,v,Ratio65);
C65 = round(C65,1);
C6 = [atan2d((Thruster.T6(2)-Thruster.T5(2)),(Thruster.T6(1)-Thruster.T5(1)))+C65/2 atan2d((Thruster.T6(2)-Thruster.T5(2)),(Thruster.T6(1)-Thruster.T5(1)))-C65/2 ...
      atan2d((Thruster.T6(2)-Thruster.T4(2)),(Thruster.T6(1)-Thruster.T4(1)))+C64/2 atan2d((Thruster.T6(2)-Thruster.T4(2)),(Thruster.T6(1)-Thruster.T4(1)))-C64/2];
C6 = round(C6,1);

C = [C1;C2;C3;C4;C5;C6]; % All constraints for Thruster-Thruster Interaction

%% Curves for mapping commanded force to commanded PWM signal

load('signal mapping\force_to_pwm_thr1_forward')
pwm_thr1_forward = coeffvalues(ForceToPWM);

load('signal mapping\force_to_pwm_thr2_forward')
pwm_thr2_forward = coeffvalues(ForceToPWM);

load('signal mapping\force_to_pwm_thr3_forward')
pwm_thr3_forward = coeffvalues(ForceToPWM);

load('signal mapping\force_to_pwm_thr4_forward')
pwm_thr4_forward = coeffvalues(ForceToPWM);

load('signal mapping\force_to_pwm_thr5_forward')
pwm_thr5_forward = coeffvalues(ForceToPWM);

load('signal mapping\force_to_pwm_thr6_forward')
pwm_thr6_forward = coeffvalues(ForceToPWM);

load('signal mapping\force_to_pwm_thr1_backward')
pwm_thr1_backward = coeffvalues(ForceToPWM);

load('signal mapping\force_to_pwm_thr2_backward')
pwm_thr2_backward = coeffvalues(ForceToPWM);

load('signal mapping\force_to_pwm_thr3_backward')
pwm_thr3_backward = coeffvalues(ForceToPWM);

load('signal mapping\force_to_pwm_thr4_backward')
pwm_thr4_backward = coeffvalues(ForceToPWM);

load('signal mapping\force_to_pwm_thr5_backward')
pwm_thr5_backward = coeffvalues(ForceToPWM);

load('signal mapping\force_to_pwm_thr6_backward')
pwm_thr6_backward = coeffvalues(ForceToPWM);
```

A.2 S-function Quadratic Regulator

```
function [sys,ThrusterPos,str,ts] = thrustallocationQuadraticFixed(t,x,u,
    flag,Thruster,C)
% Function for finding optimal thrust and angle
% Author: Preben Frederich
% Last edited: 09/07-2016

% Version 1.0 : Pseudo inverse optimization created
%               12/12-2015 Preben Frederich
% Version 1.1 : Desired angle implemented
%               02/03-2016 Preben Frederich
% Version 1.2 : Individual & coupled constrained thrust allocation
    implemented
%               07/04-2016 Preben Frederih
% Version 1.3 : Quadratic control allocation implemented with optimal
%               angles found by pseudoinverse allocation
%               15/04-2016 Preben Frederich
% Version 1.4 : Quadratic control allocation modified to optimize around
%               the different thrusters when they are in forbidden zones
%               20/04-2016 Preben Frederich
% Version 1.5 : Implemented constraints at forbidden zones varying
    depending
%               on length between each thruster. Retrieved from Initfile
    as [C].
%               22/04-2016 Preben Frederich
% Version 1.6 : Implemented possibilities for using real model values for
%               the quadratic regulator
%               25/04-2016 Preben Frederich
% Version 1.7 : Initial conditions for Quadratic regulator is updated
    with
%               the optimal solution from last time step. Makes the next
%               timestep solution run much faster
%               03/05-2016 Preben Frederich
% Version 1.8 : Cleanup and minor changes in name definitions to coincide
%               with ROV Control Modes by Skjetne 2015. Explanation of
%               the different parts of system updated.
%               04/07-2016 Preben Frederich

switch flag,

  % Initialization
  case 0,
    [sys,ThrusterPos,str,ts]=mdlInitializeSizes(Thruster);

  % Output
  case 3,
    sys=mdlOutputs(t,x,u,Thruster);

  % Update
  case 2,
    sys=mdlUpdate(t,x,u,Thruster,C);

```

```
      case {1,4,}
    sys=[];

  % Case 9 used for terminate signal if errors where to occure.
  case 9,
      sys=mdlTerminate(t,x,u,Thruster);

  % Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);

end

function [sys,x0,str,ts]=mdlInitializeSizes(Thruster)
% Only used when starting the system
% Maps out the different states
sizes = simsizes;
sizes.NumContStates  = 0;   % Number of continuous states in the system,
sizes.NumDiscStates  = 12;  % Number of discrete states in the system
sizes.NumOutputs     = 12;  % Number of outputs (Output of each angle of each thruster and desired thrust of each thruster)
sizes.NumInputs      = 16;  % Number of inputs2 (Input: Desired angle and Desired thrust vector)
sizes.DirFeedthrough = 0;   % No element which is sent directly through the system
sizes.NumSampleTimes = 1;   % Number of sample times each itteration
sys = simsizes(sizes);

%% Determines initial angles on thrusters + initial thrust at start.
x0 = [Thruster.T1(3) Thruster.T2(3) Thruster.T3(3) Thruster.T4(3) Thruster.T5(3) Thruster.T6(3)  0 0 0 0 0 0 ]';

str = [];

ts  = [-1 0];

function sys=mdlUpdate(t,x,u,Thruster,C)
%% Input definitions used through the system
ThrusterPosX = [Thruster.T1(1) Thruster.T2(1) Thruster.T3(1) Thruster.T4(1) Thruster.T5(1) Thruster.T6(1)];%[x(1),x(2),x(3),x(4),x(5),x(6)]';
ThrusterPosY = [Thruster.T1(2) Thruster.T2(2) Thruster.T3(2) Thruster.T4(2) Thruster.T5(2) Thruster.T6(2)];%[x(7),x(8),x(9),x(10),x(11),x(12)]';

tc =  [u(1),u(2),u(3)]';
alpha = [u(4),u(5),u(6),u(7),u(8),u(9)]';
epsilon = u(10);


%% Thrust allocation algorithm
n_t = length(ThrusterPosX); % number of thrusters
B_surge = zeros(n_t,1);     % Shells for faster computing
B_sway = zeros(n_t,1);      % Shells for faster computing
```

```
B_yaw = zeros(n_t,1);        % Shells for faster computing

for i = 1:1:n_t
  B_surge(i) = cosd(alpha(i));      %Thrust in x-direction for cartesian
  B_sway(i) = sind(alpha(i));       %Thrust in y-direction for cartesian
  B_yaw(i) = (sind(alpha(i))*ThrusterPosX(i) - cosd(alpha(i))*
      ThrusterPosY(i));
end

% Constraints for Thruster-Thruster interaction.
C1 = C(1,:);     % Angle constraints Thruster 1
C2 = C(2,:);     % Angle constraints Thruster 2
C3 = C(3,:);     % Angle constraints Thruster 3
C4 = C(4,:);     % Angle constraints Thruster 4
C5 = C(5,:);     % Angle constraints Thruster 5
C6 = C(6,:);     % Angle constraints Thruster 6

K = ones(1,6); % Resets K For each time iteration

% Thruster 1 on/off
if (alpha(1) < C1(1)-0.1  && alpha(1) > C1(2)+0.1) || (alpha(1) > C1(3)
    +0.1 && alpha(1) < C1(4)-0.1)
    K(1) = 0;
else
    K(1) = K(1);
end
% Thruster 2 on/off
if (alpha(2) < C2(1)-0.1  && alpha(2) > C2(2)+0.1) || (alpha(2) < C2(3)
    -0.1 && alpha(2) > C2(4)+0.1)
    K(2) = 0;
else
    K(2) = K(2);
end
% Thruster 3 on/off
if (alpha(3) > C3(1)+0.1  && alpha(3) < C3(2)-0.1) || (alpha(3) > C3(3)
    +0.1 && alpha(3) < C3(4)-0.1)
    K(3) = 0;
else
    K(3) = K(3);
end
% Thruster 4 on/off
if (alpha(4) > C4(1)+0.1  && alpha(4) < C4(2)-0.1) || (alpha(4) < C4(3)-
    0.1 && alpha(4) > C4(4)+0.1)
    K(4) = 0;
else
    K(4) = K(4);
end
% Thruster 5 on/off
if (alpha(5) > C5(1)+0.1  && alpha(5) < C5(2)-0.1) || (alpha(5) > C5(3)
    +0.1 && alpha(5) < C5(4)-0.1)
    K(5) = 0;
else
    K(5) = K(5);
```

```
end
% Thruster 6 on/off
if (alpha(6) < C6(1)-0.1  && alpha(6) > C6(2)+0.1) || (alpha(6) < C6(3)
    -0.1 && alpha(6) > C6(4)+0.1)
    K(6) = 0;
else
    K(6) = K(6);
end

K = [K(1) K(2) K(3) K(4) K(5) K(6)];

%% Finding Optimal angle from desired forces tau
B = [B_surge,B_sway,B_yaw]';
B = B*diag(K);
u_e = [1 0 1 0 1 0 1 0 1 0 1 0;
      0 1 0 1 0 1 0 1 0 1 0 1;
      ThrusterPosY(1) ThrusterPosX(1) ThrusterPosY(2) ThrusterPosX(2)
          ThrusterPosY(3) ThrusterPosX(3) ...
      ThrusterPosY(4) ThrusterPosX(4) ThrusterPosY(5) ThrusterPosX(5)
          ThrusterPosY(6) ThrusterPosX(6)];
[U_singular,S_singular,V_singular] = svd(u_e,'econ');
S_singular_cross = 1./S_singular;
S_singular_cross(~isfinite(S_singular_cross))=0;
u0 = V_singular*S_singular_cross*U_singular'*tc;
alpha1 = [atan2d(u0(2),u0(1)) atan2d(u0(4),u0(3)) atan2d(u0(6),u0(5))
    atan2d(u0(8),u0(7)) atan2d(u0(10),u0(9)) atan2d(u0(12),u0(11))];

% Ensuring the angles are between [-180 180] degrees
for i=1:1:n_t
  if alpha1(i) > 180
      alpha1(i) = alpha1(i) - 360;
  elseif alpha1(i) < -180
      alpha1(i) = alpha1(i) + 360;
  else
      alpha1(i);
  end
end

W = eye(6);
q = sum(W);
Q = diag([100 100 10000]);
zero2H = zeros(6,3);
H = [W zero2H; zero2H' Q];
f = zeros(9,1);
Aeq = [B -eye(3)];
Beq = tc;
ub = [1.5 1.5 1.5 1.5 1.5 1.5 0.01 0.01 0.001]';
lb = -[0.85 0.85 0.85 0.85 0.85 0.85 0.01 0.01 0.001]';
X0 = [u(11),u(12),u(13),u(14),u(15),u(16)]';
options = optimoptions('quadprog',...
    'Algorithm','interior-point-convex','Display','off');
X = quadprog(H,f,[],[],Aeq,Beq,lb,ub,X0,options);

```

```
% If no solution exist:
if numel(X) <= 6
    X = [0 0 0 0 0 0 0 0 0];
end
u_d = [X(1) X(2) X(3) X(4) X(5) X(6)] ;       % Optimized Thrust

%% Constraints
%% Thruster 1
%C1 = [49.7 29.4 -151 -130 ];
if (alpha1(1) < C1(1)  && alpha1(1) > C1(2) && u_d(1) > 0) || (alpha1(1)
    < C1(1)  && alpha1(1) > C1(2) && u_d(1) < 0)
    c1 = [C1(1) C1(2)];
    [C_1,alpha_new1] = min(abs(c1-alpha1(1)));
    Alpha1(1) = c1(alpha_new1);
elseif (alpha1(1) > C1(3) && alpha1(1) < C1(4) && u_d(1) > 0) || (alpha1
    (1) > C1(3) && alpha1(1) < C1(4) && u_d(1) < 0)
    c1 = [C1(3) C1(4)];
    [C_1,alpha_new1] = min(abs(c1-alpha1(1)));
    Alpha1(1) = c1(alpha_new1);
else
    Alpha1(1) = alpha1(1);
end
if alpha(1) < C1(1)-0.1  && alpha(1) > C1(2)+0.1 || alpha(1) > C1(3)+0.1
    && alpha(1) < C1(4)-0.1
    u_d(1) = 0;
else
    u_d(1) = u_d(1);
end

%% Thruster 2
%C2 = [99.85 80.15 50 29];
if (alpha1(2) < C2(1)  && alpha1(2) > C2(2) && u_d(2) > 0) || (alpha1(2)
    < C2(1)  && alpha1(2) > C2(2) && u_d(2) < 0)
    c2 = [C2(1) C2(2)];
    [C_2,alpha_new2] = min(abs(c2-alpha1(2)));
    Alpha1(2) = c2(alpha_new2);
elseif (alpha1(2) < C2(3) && alpha1(2) > C2(4) && u_d(2) > 0) || (alpha1
    (2) < C2(3) && alpha1(2) > C2(4) && u_d(2) < 0)
    c2 = [C2(3) C2(4)];
    [C_2,alpha_new2] = min(abs(c2-alpha1(2)));
    Alpha1(2) = c2(alpha_new2);
else
    Alpha1(2) = alpha1(2);
end
if (alpha(2) < C2(1)-0.1  && alpha(2) > C2(2)+0.1) || (alpha(2) < C2(3)
    -0.1 && alpha(2) > C2(4)+0.1)
    u_d(2) = 0;
else
    u_d(2) = u_d(2);
end

%% Thruster 3
% C3 = [-99.85 -80.15 -50 -29];
```

```
if (alpha1(3) > C3(1)  && alpha1(3) < C3(2) && u_d(3) > 0) || (alpha1(3)
    > C3(1)  && alpha1(3) < C3(2) && u_d(3) < 0)
    c3 = [C3(1) C3(2)];
    [C_3,alpha_new3] = min(abs(c3-alpha1(3)));
    Alpha1(3) = c3(alpha_new3);
elseif (alpha1(3) > C3(3) && alpha1(3) < C3(4) && u_d(3) > 0) || (alpha1
    (3) > C3(3) && alpha1(3) < C3(4) && u_d(3) < 0)
    c3 = [C3(3) C3(4)];
    [C_3,alpha_new3] = min(abs(c3-alpha1(3)));
    Alpha1(3) = c3(alpha_new3);
else
    Alpha1(3) = alpha1(3);
end
if (alpha(3) > C3(1) + 0.1  && alpha(3) < C3(2) - 0.1) || (alpha(3) > C3
    (3) + 0.1 && alpha(3) < C3(4) - 0.1)
    u_d(3) = 0;
else
    u_d(3) = u_d(3);
end

%% Thruster 4
%C4 = [-52.8 -34.2 52.8 34.2];
if (alpha1(4) > C4(1)  && alpha1(4) < C4(2) && u_d(4) > 0) || (alpha1(4)
    > C4(1)  && alpha1(4) < C4(2) && u_d(4) < 0)
    c4 = [C4(1) C4(2)];
    [C_4,alpha_new4] = min(abs(c4-alpha1(4)));
    Alpha1(4) = c4(alpha_new4);
elseif (alpha1(4) < C4(3) && alpha1(4) > C4(4) && u_d(4) > 0) || (alpha1
    (4) < C4(3) && alpha1(4) > C4(4) && u_d(4) < 0)
    c4 = [C4(3) C4(4)];
    [C_4,alpha_new4] = min(abs(c4-alpha1(4)));
    Alpha1(4) = c4(alpha_new4);
else
    Alpha1(4) = alpha1(4);
end
if (alpha(4) > C4(1)+0.1  && alpha(4) < C4(2)-0.1) || (alpha(4) < C4(3)
    -0.1 && alpha(4) > C4(4)+0.1)
    u_d(4) = 0;
else
    u_d(4) = u_d(4);
end

%% Thruster 5
%C5 = [-98.25 -81.75 -145.8 -127.2];
if (alpha1(5) > C5(1)  && alpha1(5) < C5(2) && u_d(5) > 0) || (alpha1(5)
    > C5(1)  && alpha1(5) < C5(2) && u_d(5) < 0)
    c5 = [C5(1) C5(2)];
    [C_5,alpha_new5] = min(abs(c5-alpha1(5)));
    Alpha1(5) = c5(alpha_new5);
elseif (alpha1(5) > C5(3) && alpha1(5) < C5(4) && u_d(5) > 0)  || (alpha1
    (5) > C5(3) && alpha1(5) < C5(4) && u_d(5) < 0)
    c5 = [C5(3) C5(4)];
    [C_5,alpha_new5] = min(abs(c5-alpha1(5)));
```

```
    Alpha1(5) = c5(alpha_new5);
else
    Alpha1(5) = alpha1(5);
end
if (alpha(5) > C5(1)+0.1  && alpha(5) < C5(2)-0.1) || (alpha(5) > C5(3)
    +0.1 && alpha(5) < C5(4)-0.1)
    u_d(5) = 0;
else
    u_d(5) = u_d(5);
end

%% Thruster 6
%C6 = [98.75 81.75 145.8 127.2];
if (alpha1(6) < C6(1)  && alpha1(6) > C6(2) && u_d(6) > 0) || (alpha1(6)
    < C6(1)  && alpha1(6) > C6(2) && u_d(6) < 0)
    c6 = [C6(1) C6(2)];
    [C_6,alpha_new6] = min(abs(c6-alpha1(6)));
    Alpha1(6) = c6(alpha_new6);
elseif (alpha1(6) < C6(3) && alpha1(6) > C6(4) && u_d(6) > 0) || (alpha1
    (6) < C6(3) && alpha1(6) > C6(4) && u_d(6) < 0)
    c6 = [C6(3) C6(4)];
    [C_6,alpha_new6] = min(abs(c6-alpha1(6)));
    Alpha1(6) = c6(alpha_new6);
else
    Alpha1(6) = alpha1(6);
end
if (alpha(6) < C6(1)-0.1  && alpha(6) > C6(2)+0.1) || (alpha(6) < C6(3)
    -0.1 && alpha(6) > C6(4)+0.1)
    u_d(6) = 0;
else
    u_d(6) = u_d(6);
end

%% Coupled constraints:
% Thruster 2&3
if Alpha1(2) == C2(2)
    Alpha1(3) = alpha1(2) - (C2(2) - alpha1(2));
elseif Alpha1(2) == C2(1)
    Alpha1(3) = alpha1(2) - (C2(1) - alpha1(2));
elseif Alpha1(2) == C2(3)
    Alpha1(3) = alpha1(2) - (C2(3) - alpha1(2));
elseif Alpha1(2) == C2(4)
    Alpha1(3) = alpha1(2) - (C2(4) - alpha1(2));

elseif Alpha1(3) == C3(2)
    Alpha1(2) = alpha1(3) - (C3(2) - alpha1(3));
elseif Alpha1(3) == C3(1)
    Alpha1(2) = alpha1(3) - (C3(1) - alpha1(3));
elseif Alpha1(3) == C3(3)
    Alpha1(2) = alpha1(3) - (C3(3) - alpha1(3));
elseif Alpha1(3) == C3(4)
    Alpha1(2) = alpha1(3) - (C3(4) - alpha1(3));
end
```

```
% Thruster 5&6
if Alpha1(5) == C5(2)
    Alpha1(6) = alpha1(5) - (C5(2) - alpha1(5));
elseif Alpha1(5) == C5(1)
    Alpha1(6) = alpha1(5) - (C5(1) - alpha1(5));
elseif Alpha1(5) == C5(3)
    Alpha1(6) = alpha1(5) - (C5(3) - alpha1(5));
elseif Alpha1(5) == C5(4)
    Alpha1(6) = alpha1(5) - (C5(4) - alpha1(5));

elseif Alpha1(6) == C6(2)
    Alpha1(5) = alpha1(6) - (C6(2) - alpha1(6));
elseif Alpha1(6) == C6(1)
    Alpha1(5) = alpha1(6) - (C6(1) - alpha1(6));
elseif Alpha1(6) == C6(3)
    Alpha1(5) = alpha1(6) - (C6(3) - alpha1(6));
elseif Alpha1(6) == C6(4)
    Alpha1(5) = alpha1(6) - (C6(4) - alpha1(6));
end
% Thruster 4 & 1
if      Alpha1(4) == C4(2)
    Alpha1(1) = alpha1(4) - (C4(2) - alpha1(4));
elseif Alpha1(4) == C4(1)
    Alpha1(1) = alpha1(4) - (C4(1) - alpha1(4));
elseif Alpha1(4) == C4(3)
    Alpha1(1) = alpha1(4) - (C4(3) - alpha1(4));
elseif Alpha1(4) == C4(4)
    Alpha1(1) = alpha1(4) - (C4(4) - alpha1(4));
end
if      Alpha1(1) == C1(2)
    Alpha1(4) = alpha1(1) - (C1(2) - alpha1(1));
elseif Alpha1(1) == C1(1)
    Alpha1(4) = alpha1(1) - (C1(1) - alpha1(1));
elseif Alpha1(1) == C1(3)
    Alpha1(4) = alpha1(1) - (C1(3) - alpha1(1));
elseif Alpha1(1) == C1(4)
    Alpha1(4) = alpha1(1) - (C1(4) - alpha1(1));
end
Alpha1 = [Alpha1(1) Alpha1(2) Alpha1(3) Alpha1(4) Alpha1(5) Alpha1(6)]';
Desired_thrust = [u_d(1) u_d(2) u_d(3) u_d(4) u_d(5) u_d(6)]';

%% Update outputs
sys = [Alpha1',Desired_thrust'];

function sys=mdlOutputs(t,x,u,Thruster)
% Giving the desired angle output [ 1 2 3 4 5 6] together with desired
% thrust [7 8 9 10 11 12]
sys=[x(1),x(2),x(3),x(4),x(5),x(6),x(7),x(8),x(9),x(10),x(11),x(12)];

function sys=mdlTerminate(t,x,u,Thruster)
% If error within the algorithm, system terminates
sys = [];
```

A.3 Optimal Angle Path

Provided by Andreas Reason Dahl:

```
function psi_D   = OptimalAngle(desired,actual)

psi_D = desired-actual;
if abs(psi_D) >= 360
    psi_D = rem(psi_D,360);
end

if abs(psi_D) >= 180
    psi_D = psi_D - sign(psi_D)*360;
end
```

A.4 Thrust Control Switch

```
function u = fcn(control,input,n)
% u determines control number:
% u = 1 shaft speed
% u = 2 Torque
% u = 3 Power
% u = 4 combined power and torque
% Input provides manual switching through VeriStand
u = 1;
if control == 1
if input == 1
    u = 1;
elseif input == 2
    u = 2;
elseif input == 3 && n > 0.01 || input == 3 && n < -0.01
    u = 3;
elseif input == 4 || input == 3 && n < 0.01 || input == 3 && n > -0.01
    u = 4;
else
    u = 5;
end
end

if control == 2
if input == 1
    u = 1;
elseif input == 2
    u = 2;
elseif input == 3 && n > 0.01 || input == 3 && n < -0.01
    u = 3;
elseif input == 4 || input == 3 && n < 0.01 || input == 3 && n > -0.01
    u = 4;
else
    u = 5;
```

```
end
end

if control == 3
if input == 1
    u = 1;
elseif input == 2
    u = 2;
elseif input == 3 && n > 0.01 || input == 3 && n < -0.01
    u = 3;
elseif input == 4 || input == 3 && n < 0.01 || input == 3 && n > -0.01
    u = 4;
else
    u = 5;
end
end

if control == 4
if input == 1
    u = 1;
elseif input == 2
    u = 2;
elseif input == 3 && n > 0.01 || input == 3 && n < -0.01
    u = 3;
elseif input == 4 || input == 3 && n < 0.01 || input == 3 && n > -0.01
    u = 4;
else
    u = 5;
end
end

if control == 5
if input == 1
    u = 1;
elseif input == 2
    u = 2;
elseif input == 3 && n > 0.01 || input == 3 && n < -0.01
    u = 3;
elseif input == 4 || input == 3 && n < 0.01 || input == 3 && n > -0.01
    u = 4;
else
    u = 5;
end
end
```

A.5 RPM Controller

Provided by Jon Bjørnø:

```
function [rpm,measurment_tmp,revolutions_out,timeold_out] = fcn(rpm_tmp,
    timer,measurments,measurment_old,revolutions_tmp,timeold_tmp)

revolutions = revolutions_tmp;
```

```
timeold = timeold_tmp;


if measurments > measurment_old
   if revolutions >= 20
     %//Update RPM every 10 counts, increase this for better RPM
         resolution,
     %//decrease for faster update
     rpm_tmp = 600*revolutions/(timer - timeold);
     timeold_tmp = timer;
     revolutions_tmp = 0;
   else
     revolutions_tmp = revolutions + 1;
   end
elseif measurments == measurment_old && timer-timeold > 100
    rpm_tmp = 0;
end

timeold_out = timeold_tmp;
measurment_tmp = measurments;
rpm = rpm_tmp;
revolutions_out = revolutions_tmp;

end
```

Appendix B

Control Signal Mapping

This chapter shows the different mappings done for each control signal related to shaft speed and force. The corresponding coefficient values are not provided in the appendix, but can be found in attachment folder "signal mapping" described in appendix E.

The following mapping provided in this appendix is given for each thruster

- Force to control signal forward and backwards
- Control signal to force forward and backwards
- Shaft speed to control signal forward and backwards
- Control signal to shaft speed forward and backwards

The curvefitting is done by using polynomials. Because of varied results and ensuring no negative values is given at positive control signal and vica versa, the polynomials varies from 3 up to 7.

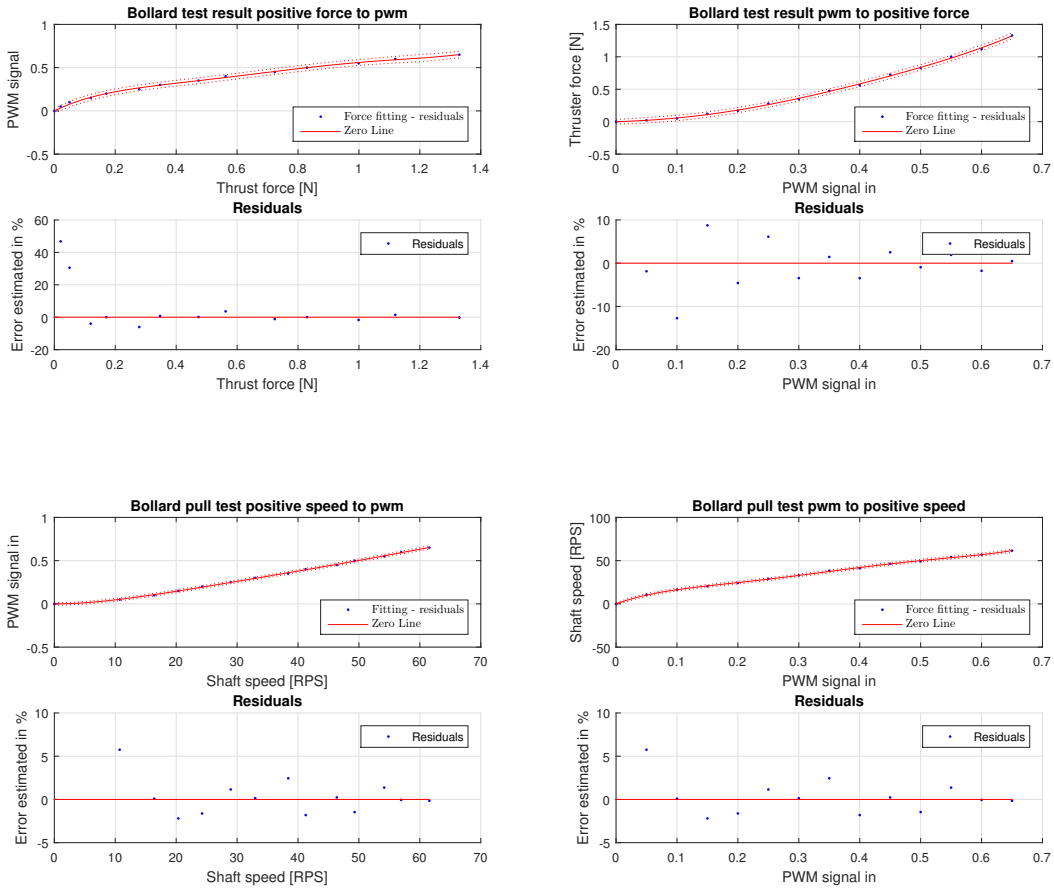

Figure B.0.1: Thruster 1 forward bollard pull results

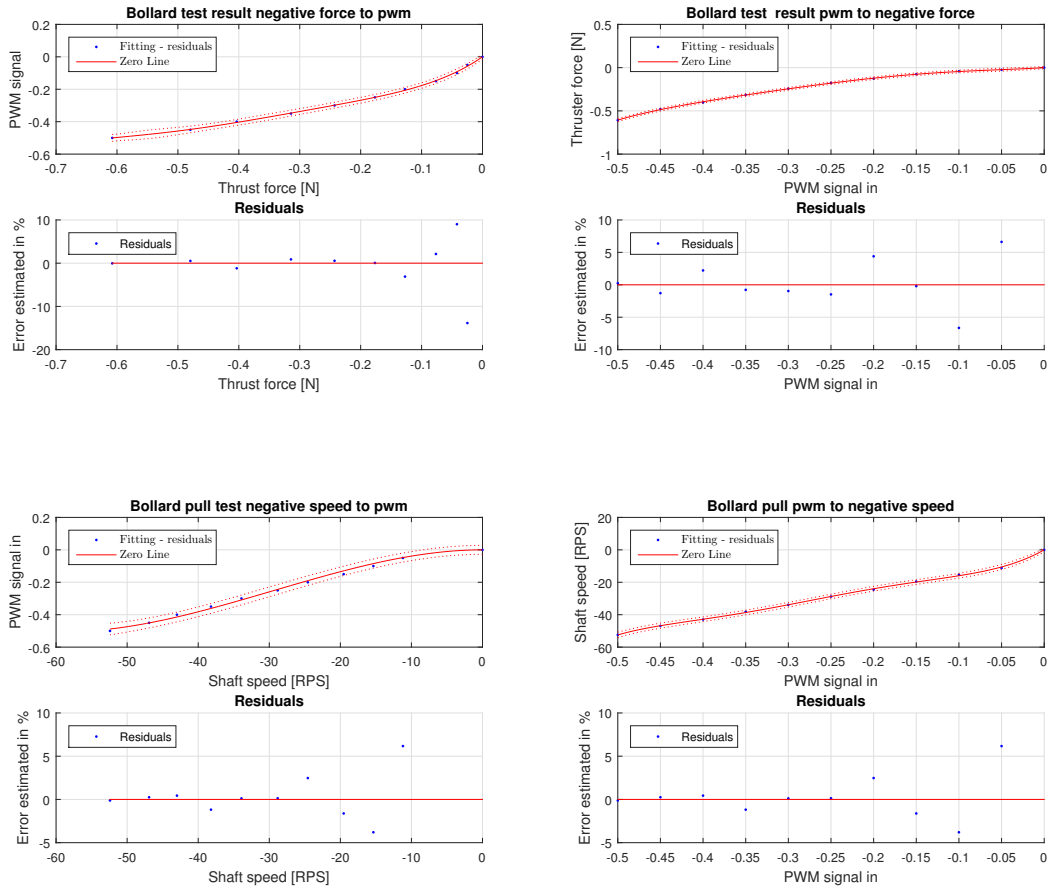

Figure B.0.2: Thruster 1 negative bollard pull results

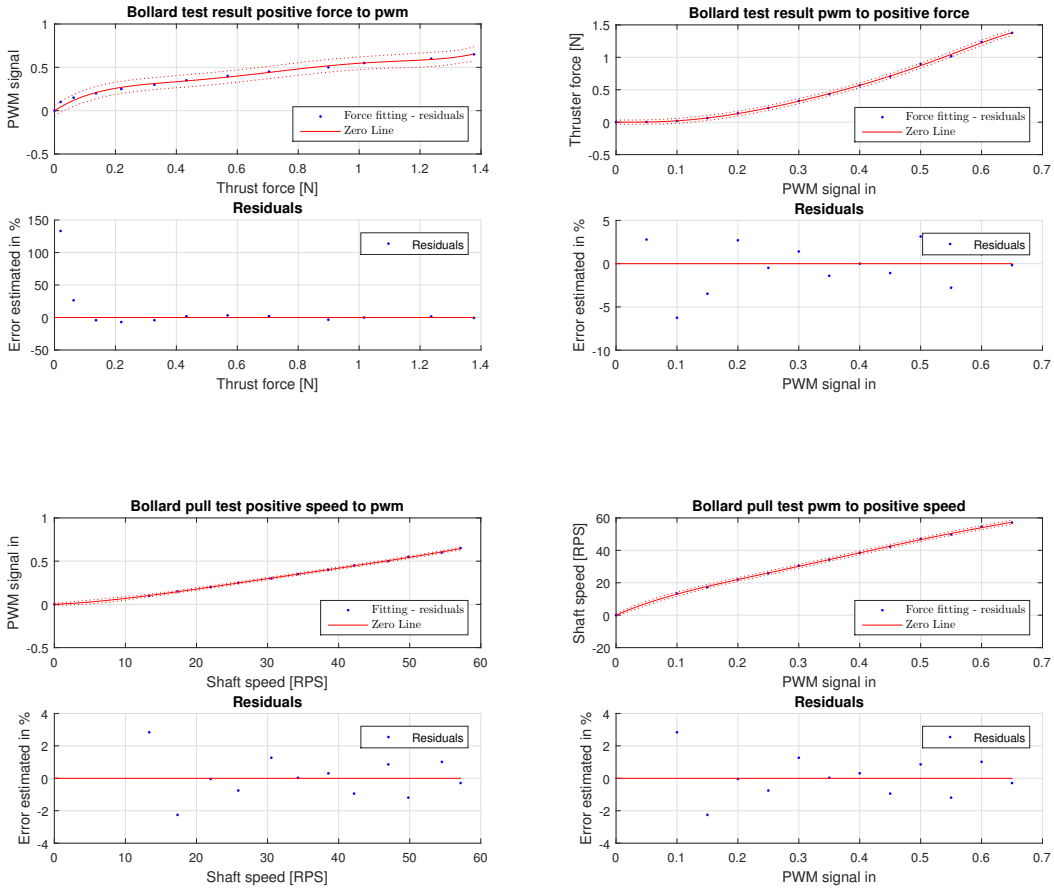

Figure B.0.3: Thruster 2 forward bollard pull results

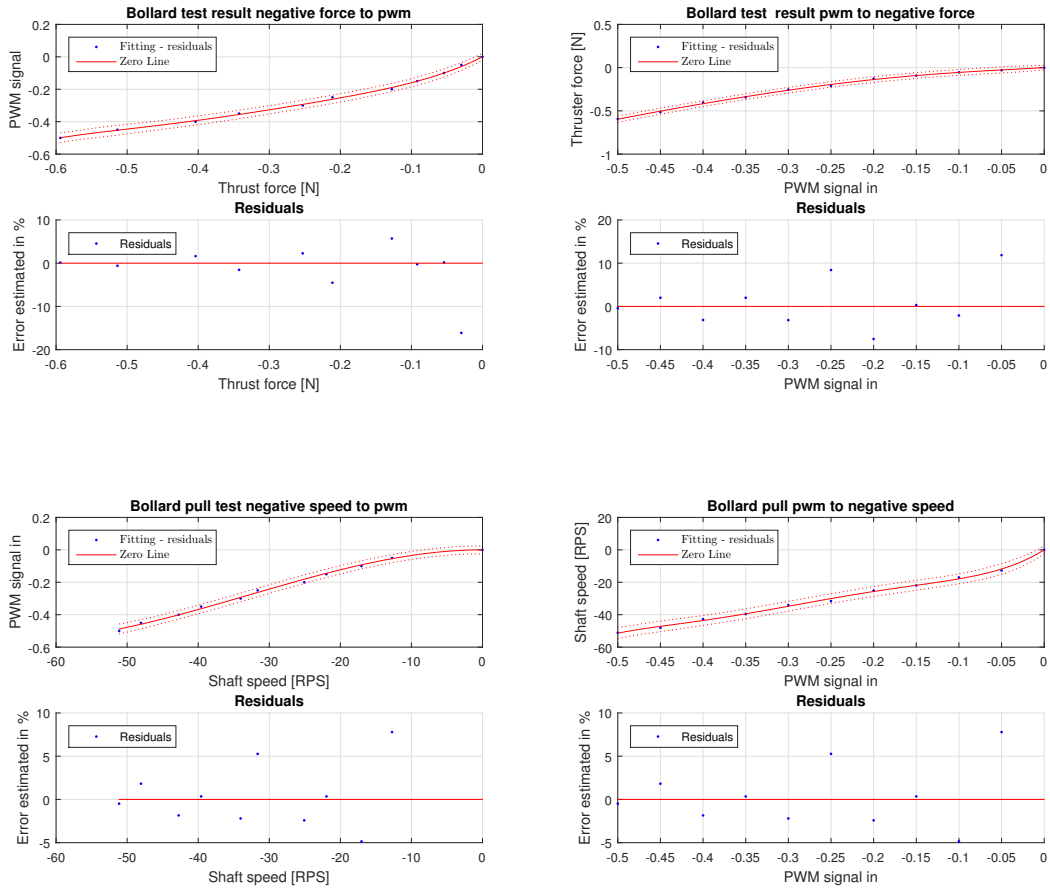

Figure B.0.4: Thruster 2 negative bollard pull results

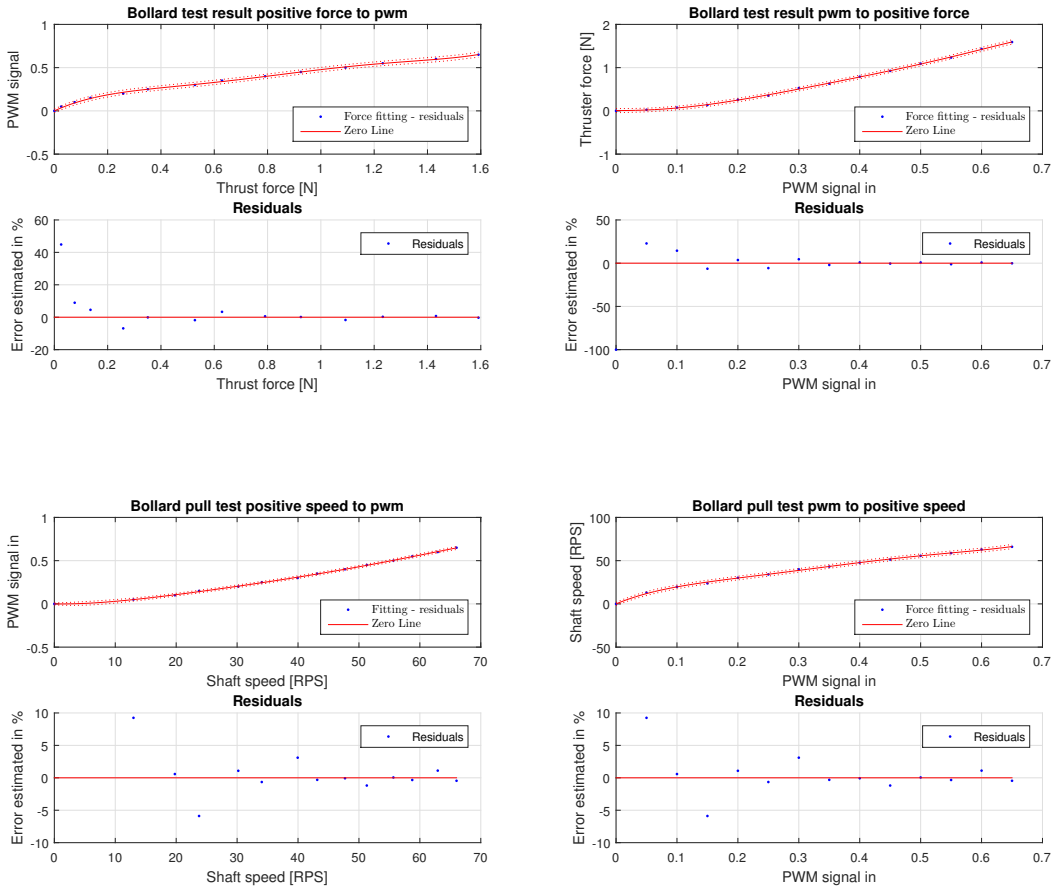

Figure B.0.5: Thruster 3 forward bollard pull results

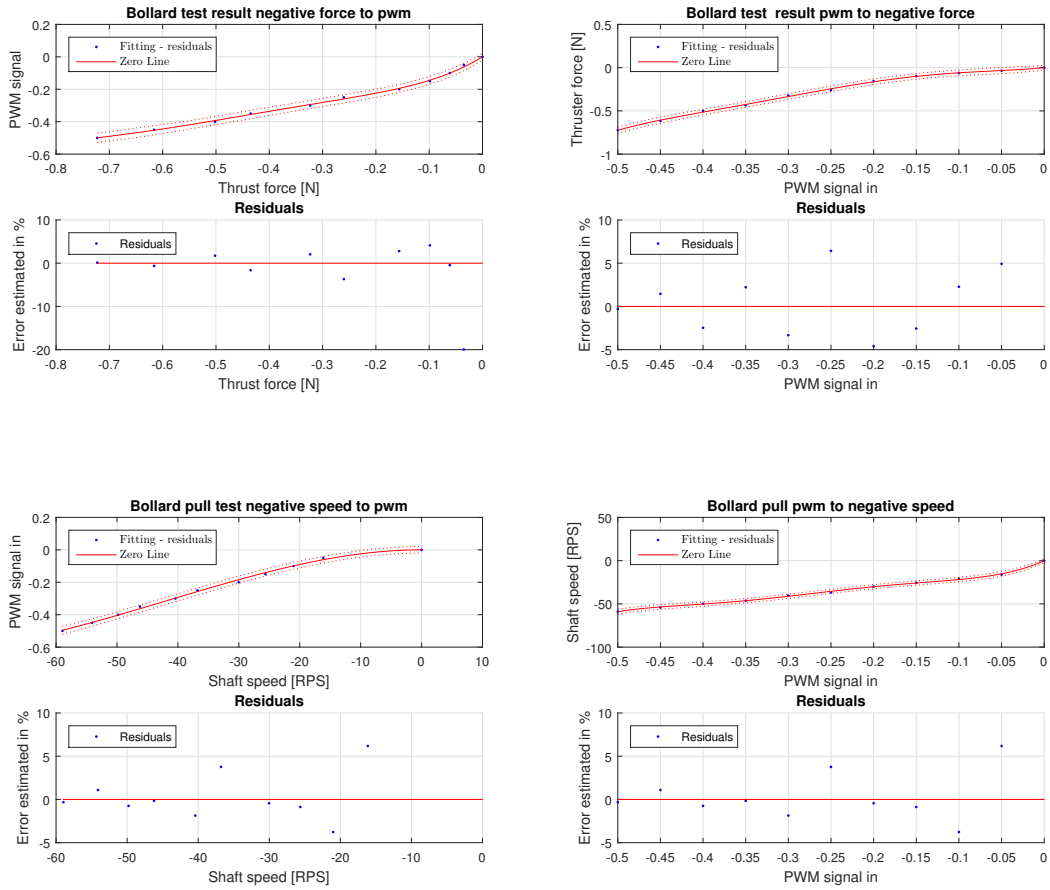

Figure B.0.6: Thruster 3 negative bollard pull results

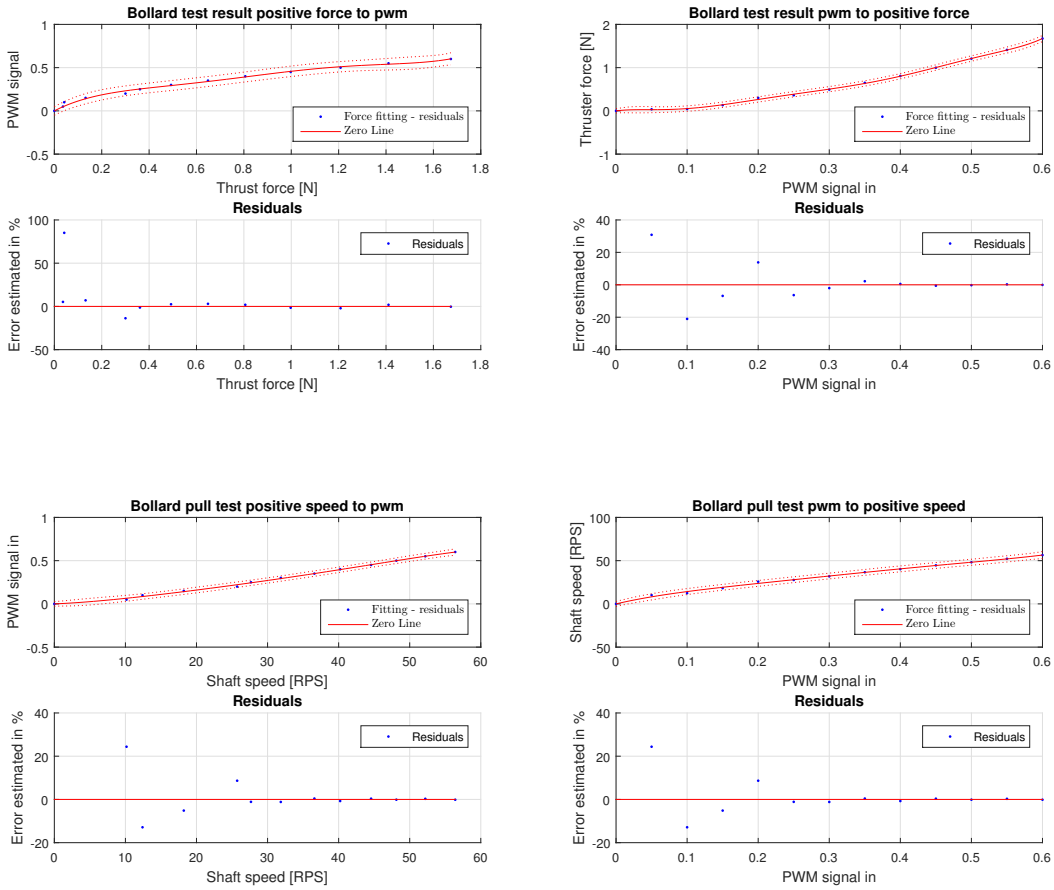

Figure B.0.7: Thruster 4 forward bollard pull results

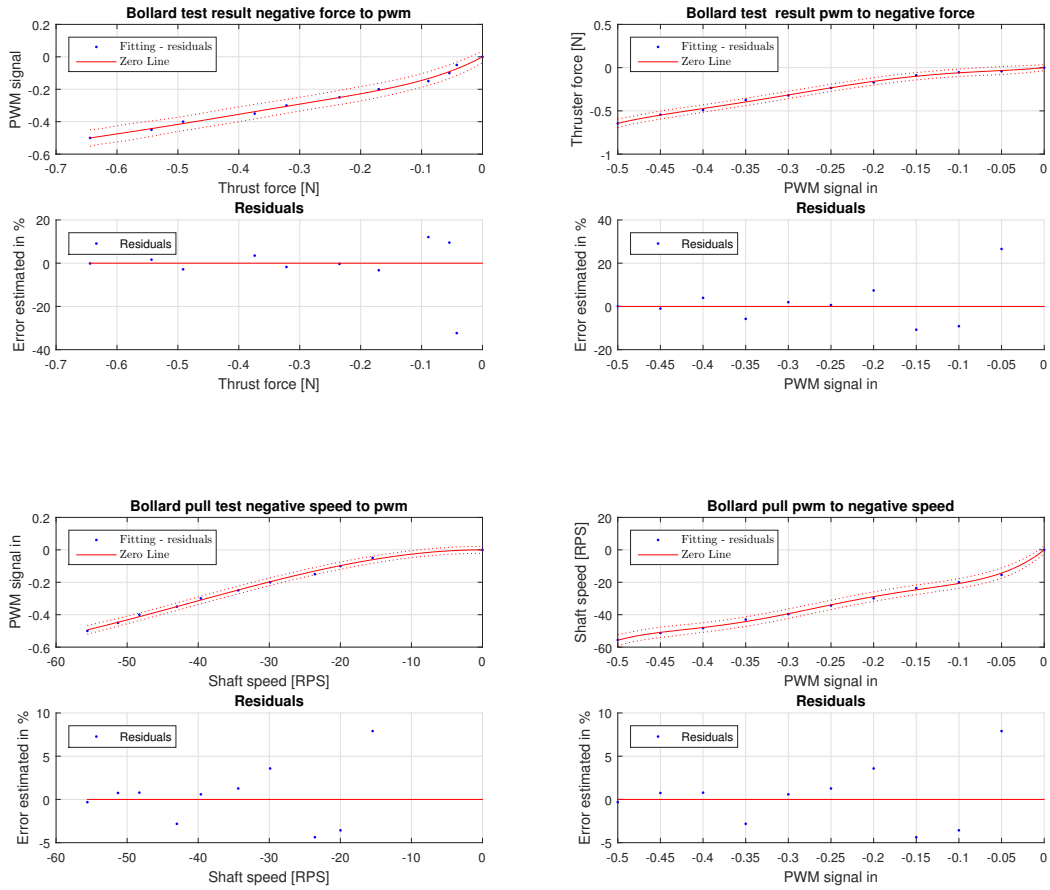

Figure B.0.8: Thruster 4 negative bollard pull results

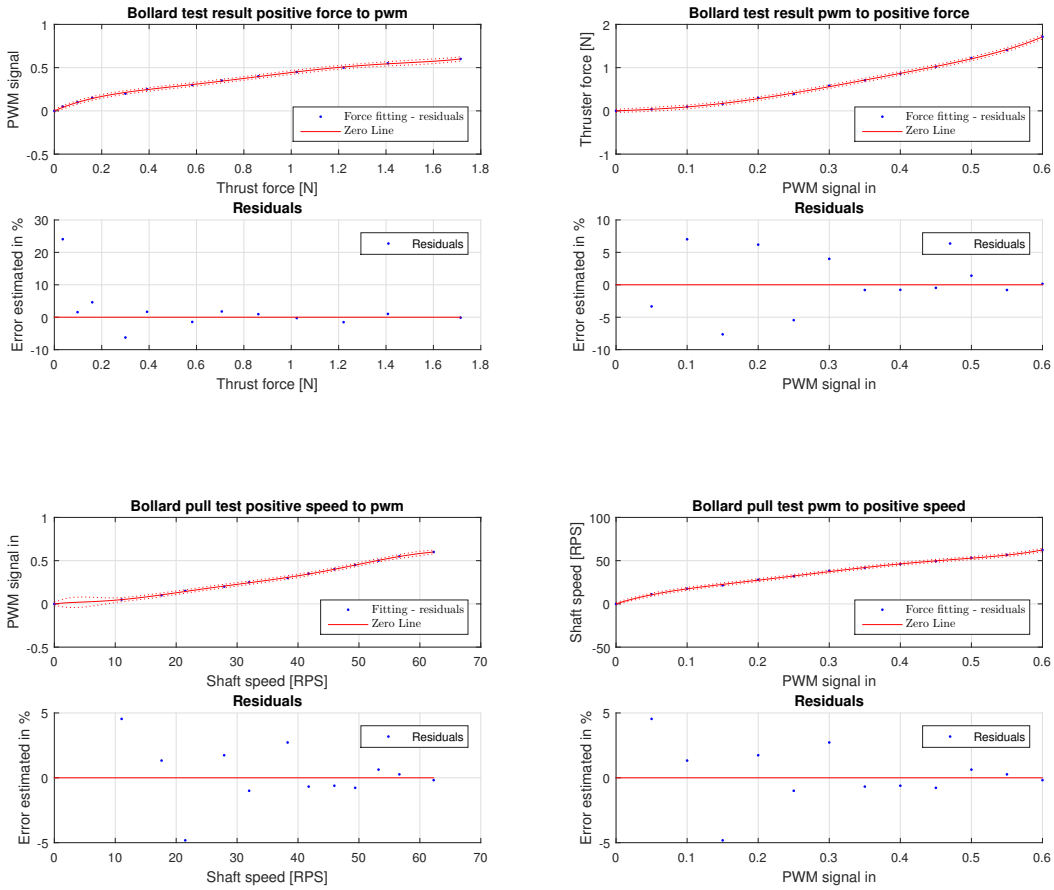

Figure B.0.9: Thruster 5 forward bollard pull results

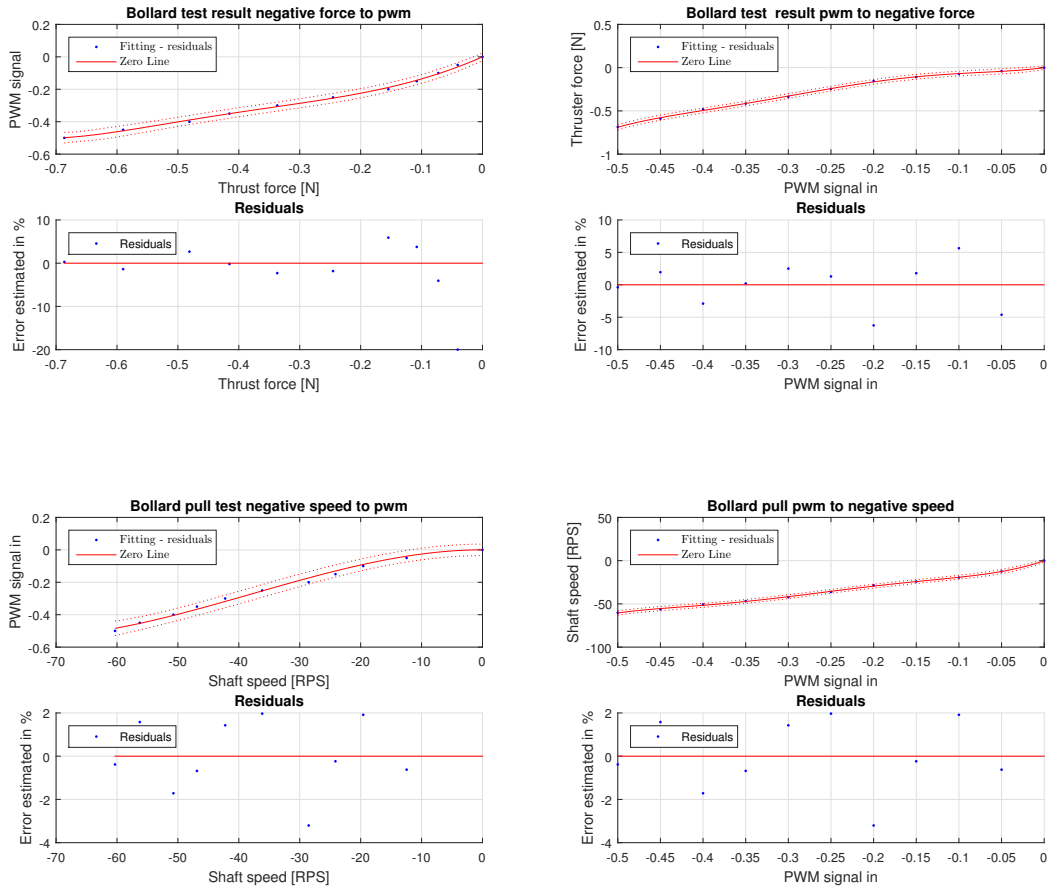

Figure B.0.10: Thruster 5 negative bollard pull results

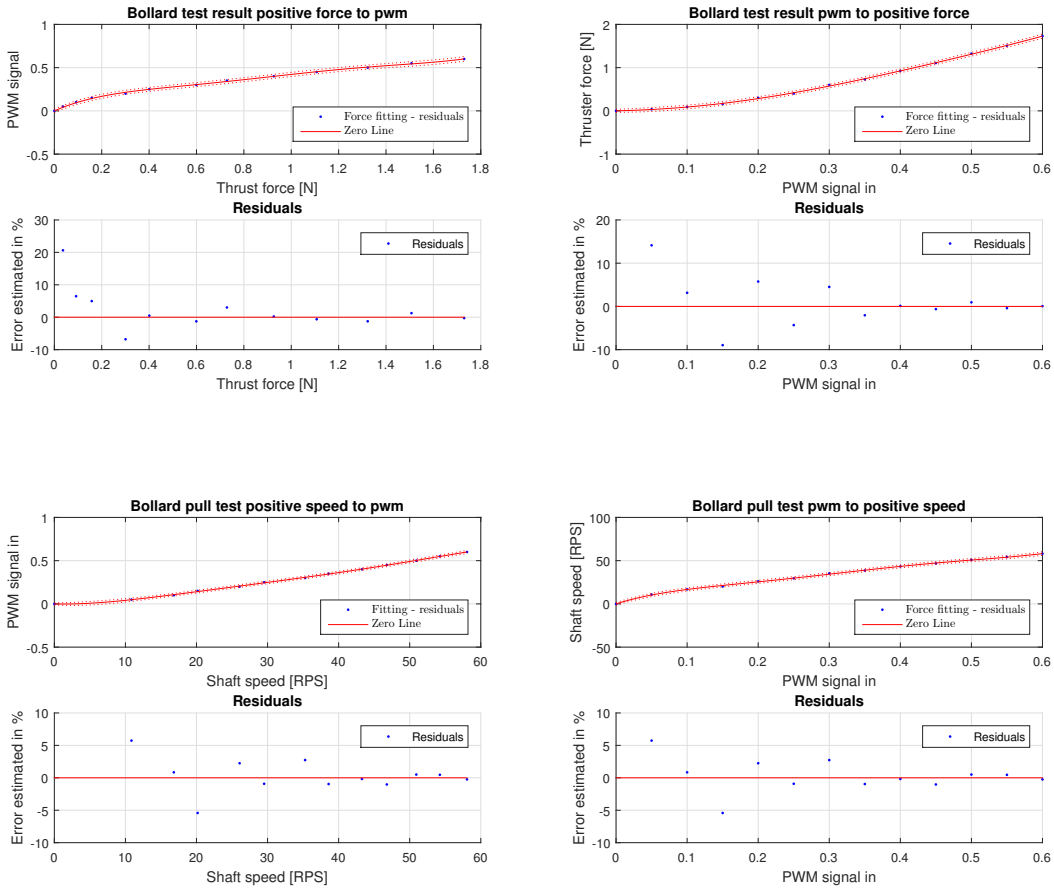

Figure B.0.11: Thruster 6 forward bollard pull results

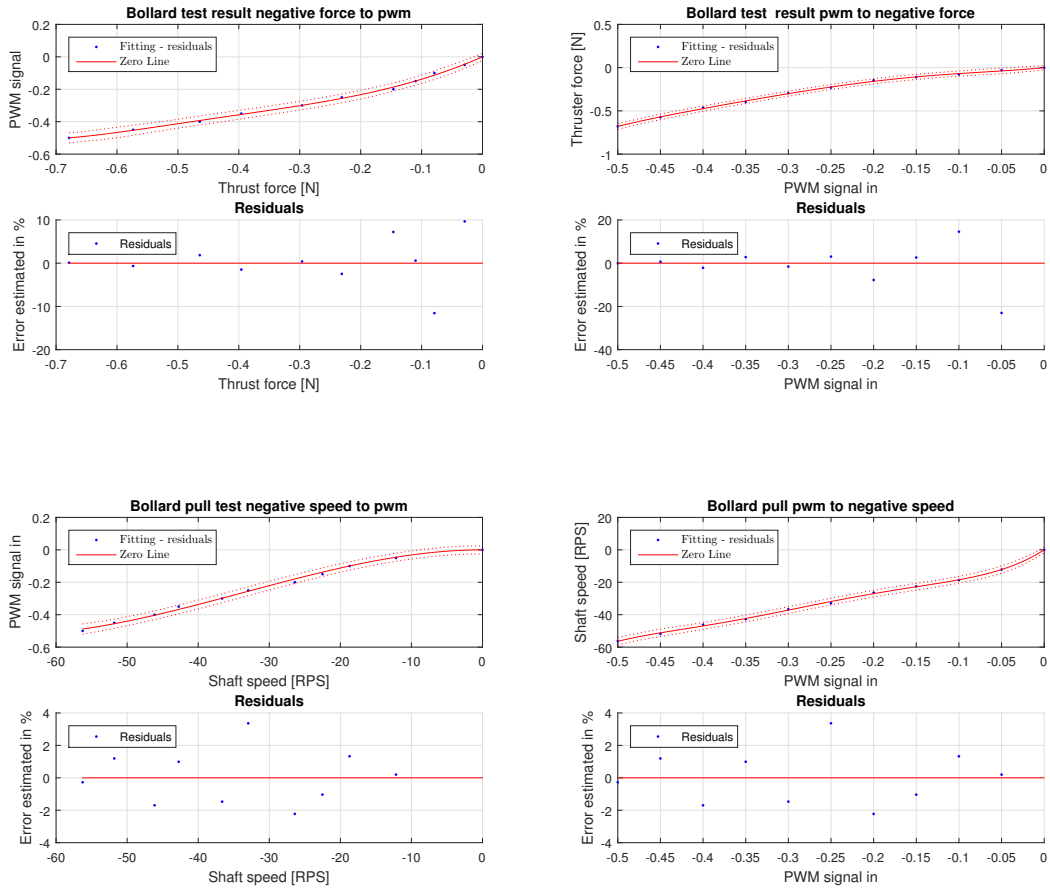

Figure B.0.12: Thruster 6 negative bollard pull results

Appendix C

Simulink Block Diagrams

C.1 RPM Measurements

Core function provided by Jon Bjørnø, extension to all thrusters and gaining to correspond with correct shaft speed is done by author

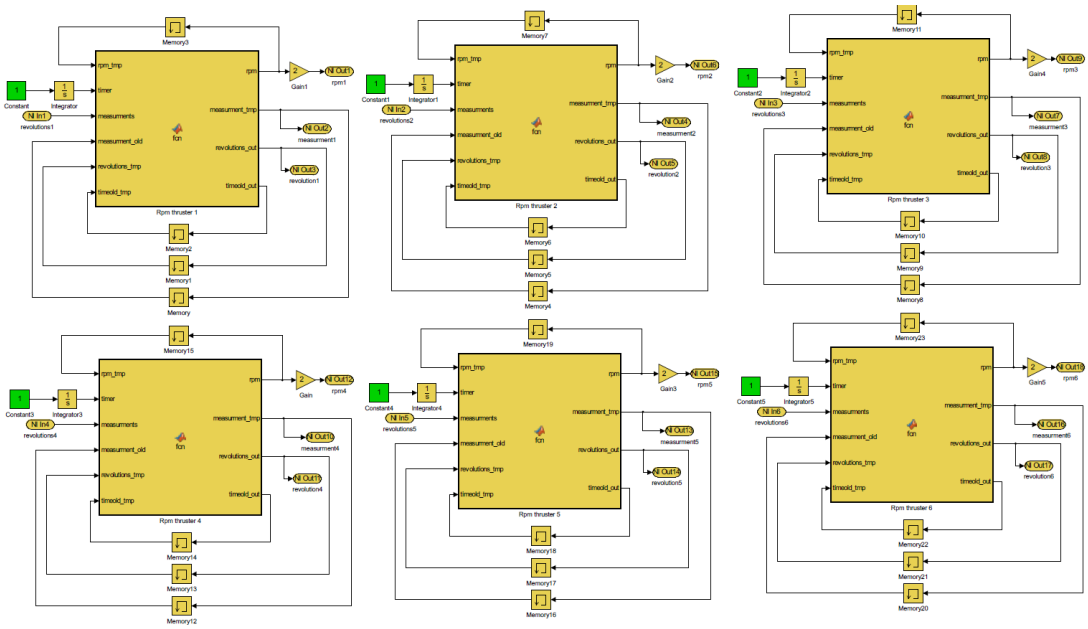

Figure C.1.1: Simulated results from thrust allocation test at fixed thruster angles

C.2 Core Thrust Control

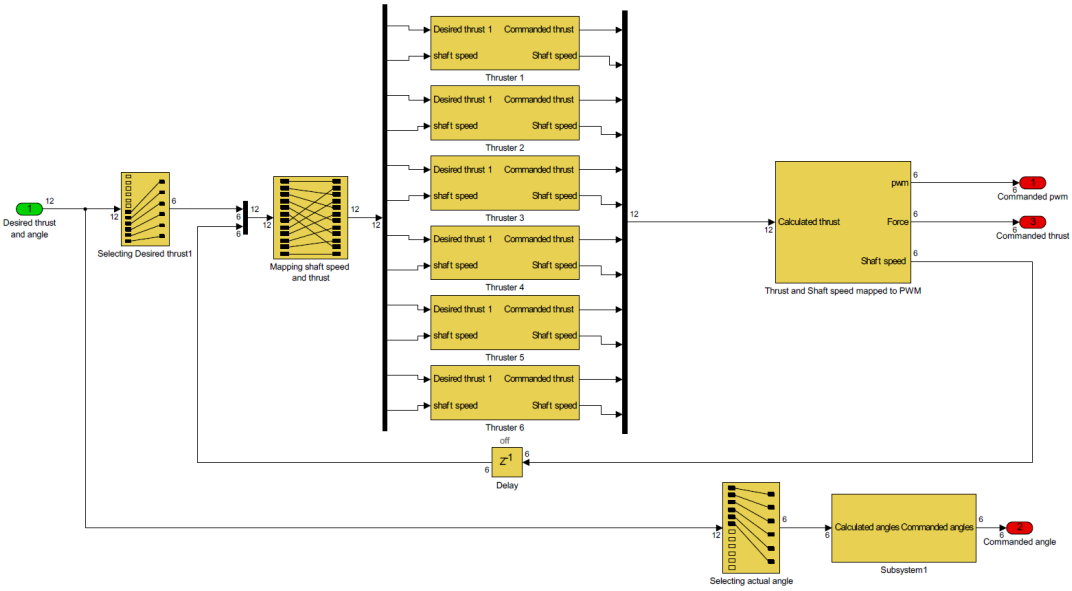

Figure C.2.1: Overview of the thruster control system

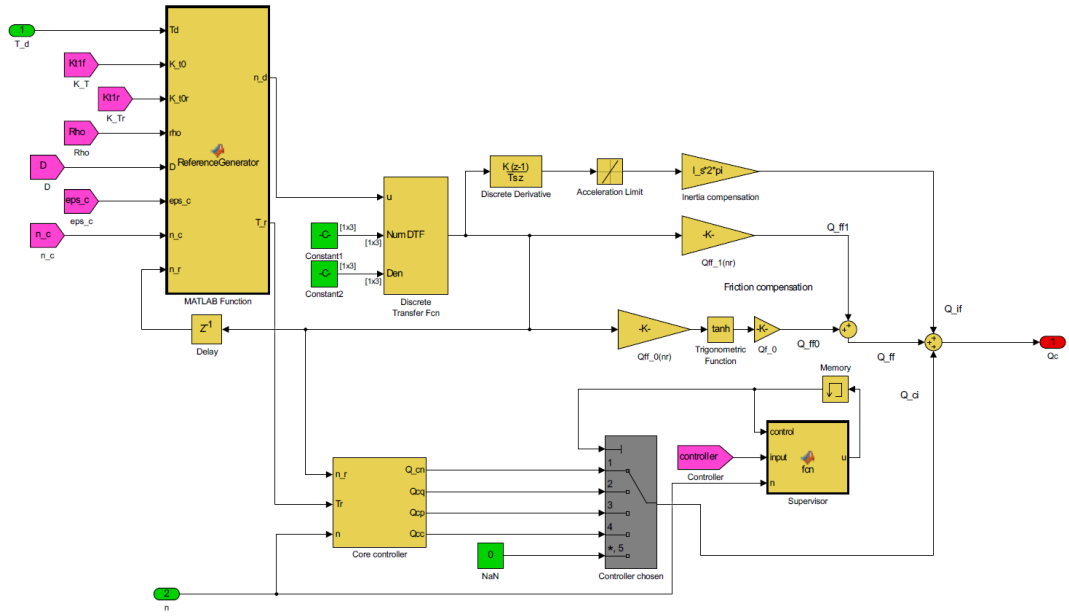

Figure C.2.2: Thruster control

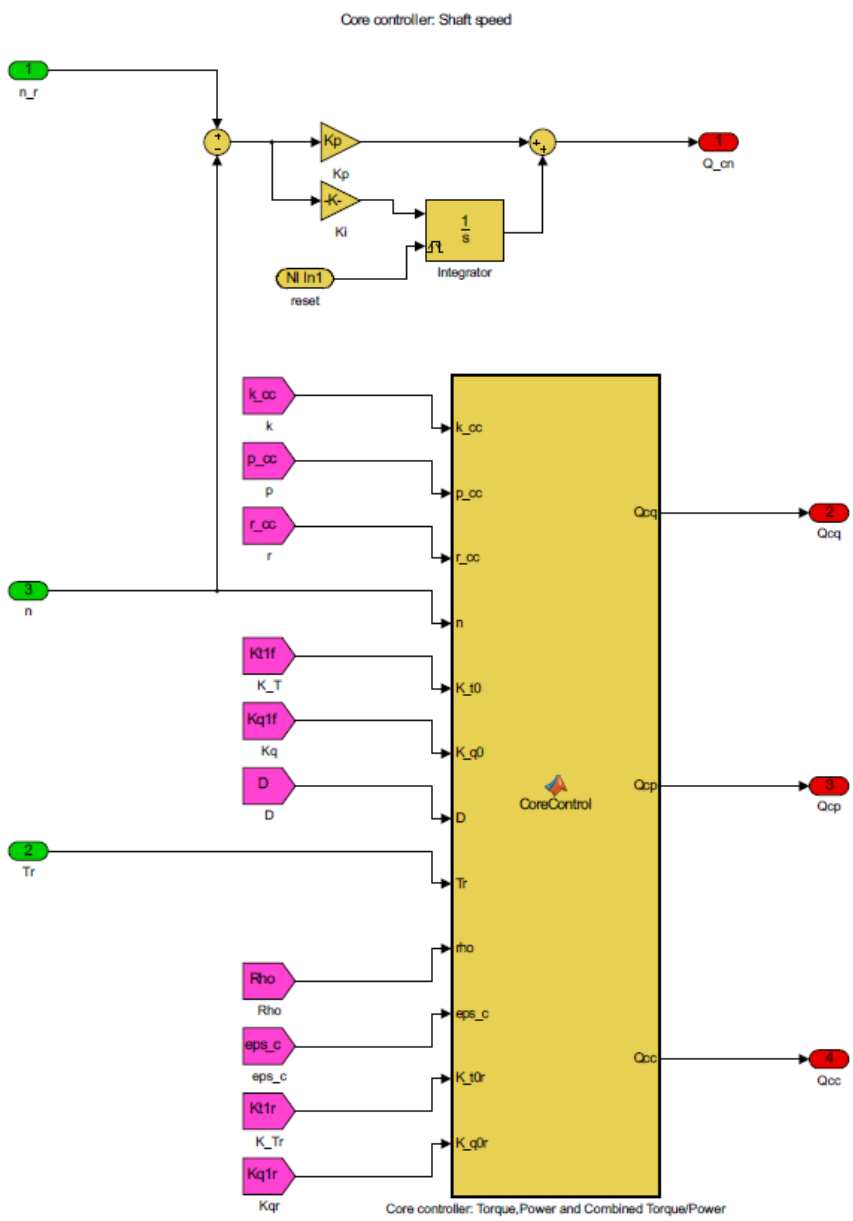

Figure C.2.3: Core thruster controller

C.3 Pwm Mapping CSAD

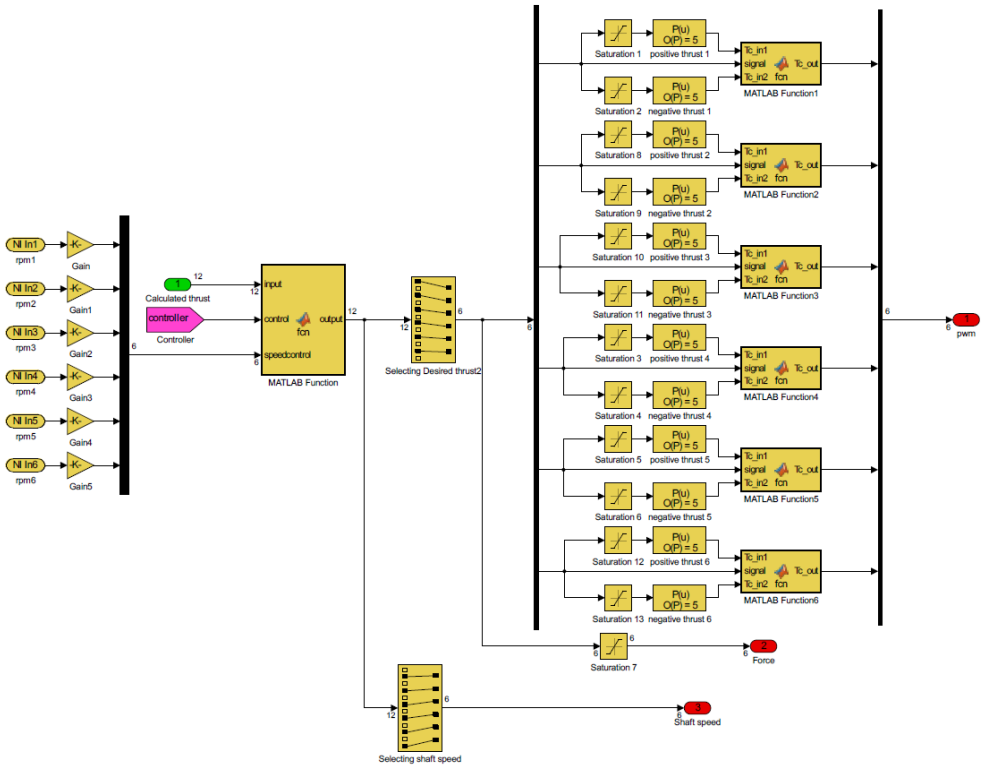

Figure C.3.1: Pwm signal mapped to CSAD for experimental testing

C.4 Thrust Allocation

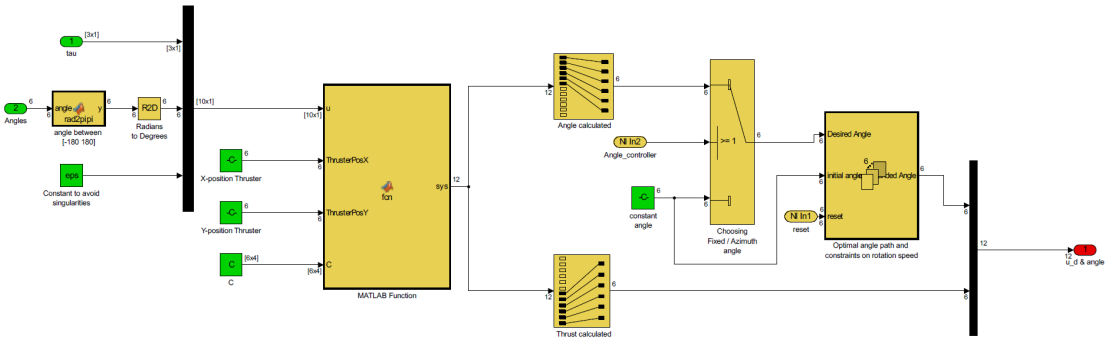

Figure C.4.1: Thrust allocation setup for experimental testing

C.5 Optimal Angle Path

Rad2pipi function which maps signal from [-pi pi] is provided by MSS Toolbox

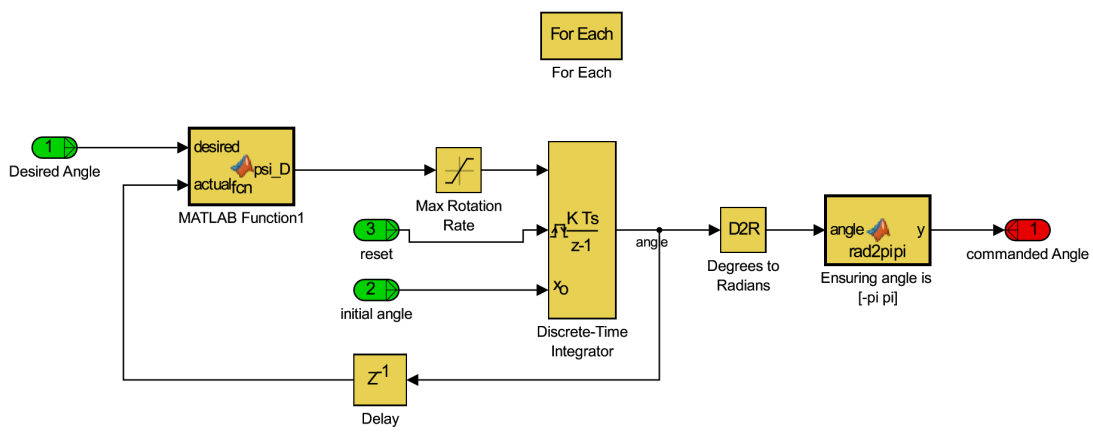

Figure C.5.1: Optimal angle path

Appendix D

LabVIEW Test Case and Veristand HMI

This shows how the LabVIEW test case was setup for implementing quadratic program. The first figure shows command window with the input and outputs, while the second figure shows the block diagram and its setup. "QP:AS" Seen in Figure **??** is the quadratic regulator.

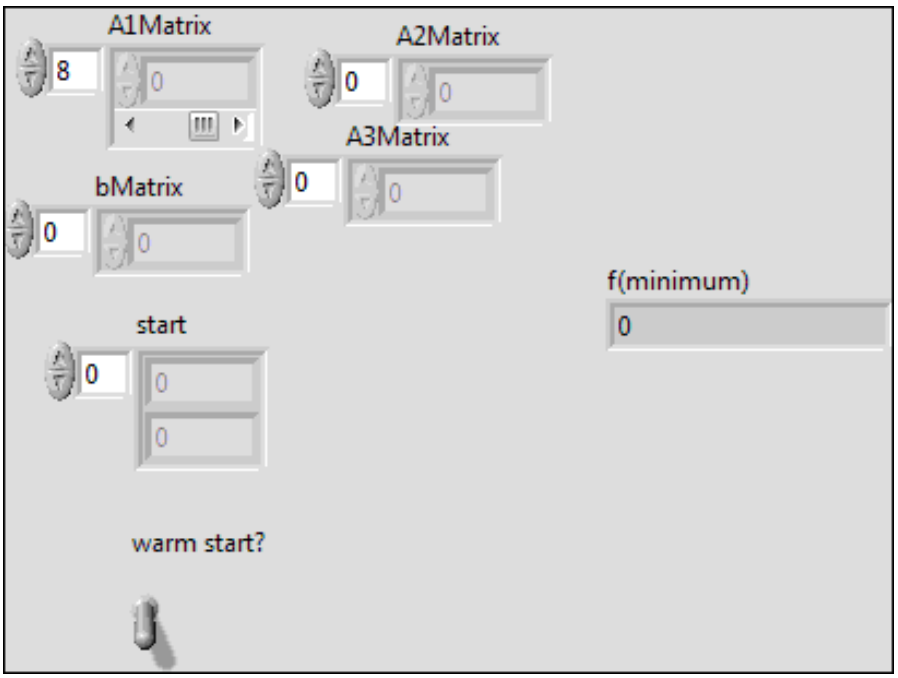

Figure D.0.1: Command window LabVIEW quadratic regulator

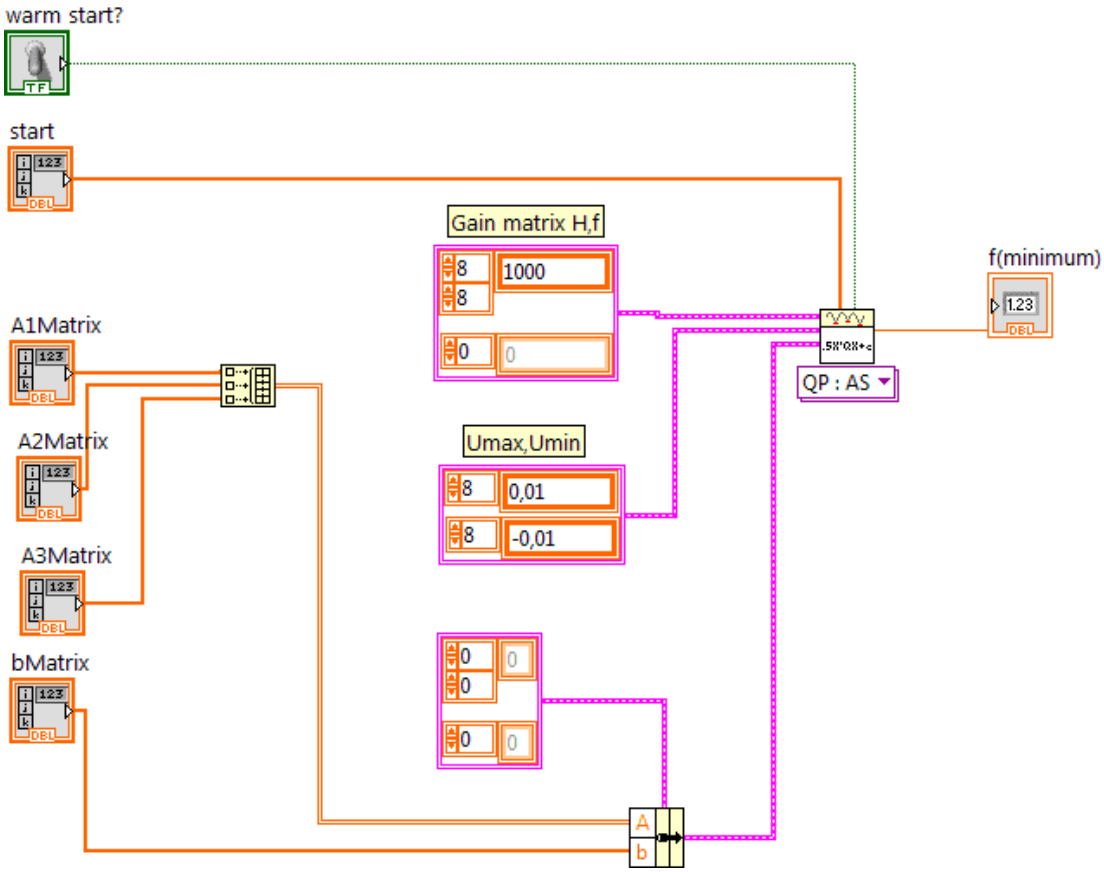

Figure D.0.2: Block diagram LabVIEW quadratic regulator

The figure below shows thruster control system setup implemented in Veristand. It provides control over fixed/azimuth angles and the different core controllers.

Monitoring blocks are provided for thrust (u_i), angle ($alpha_i$) and rpm (rpm). The blocks are positioned to visualize where each thruster are positioned on CSAD. The three graphs provides an oversight of desired forces/moment and commanded forces/moment.

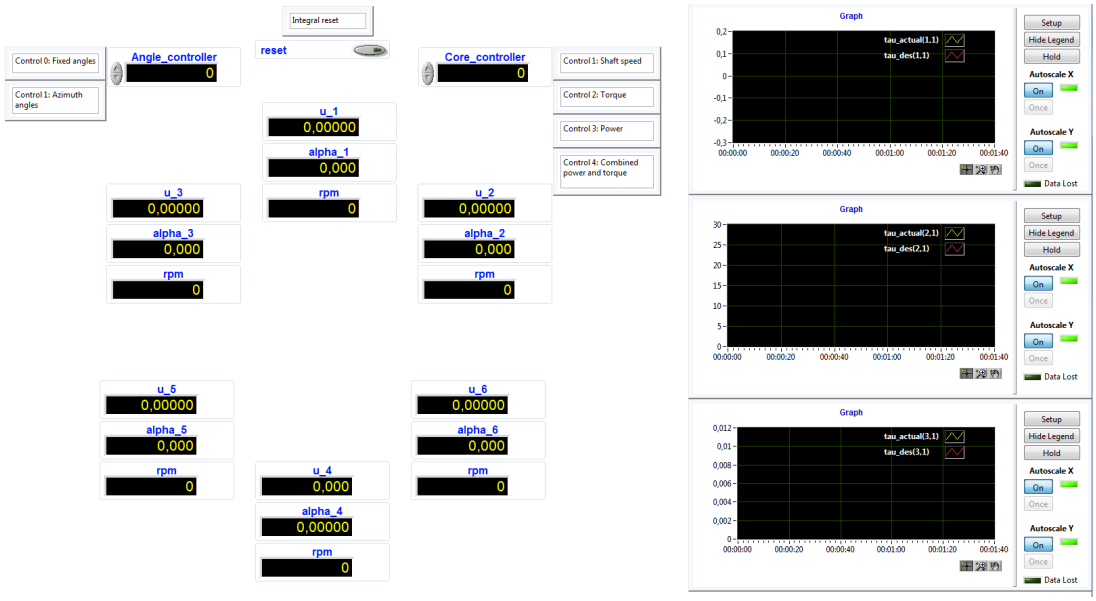

Figure D.0.3: HMI setup in Veristand for thrust control

Appendix E

Attachments

All attachments described in this chapter is provided in the .zip file delivered with the thesis.

E.1 Simulation Model

The folder consists of the whole simulation system with both quadratic and pseudoinverse solver possible to use. The following list is provided as a guide for using it

- **Open** *CSAD_Thrust_simulation.slx*
  This is the simulink model and consists of both thrust allocation for pseudoinverse optimization and quadratic regulator as well as the thrust control.
- **Open** *Initfile.m* The initfile provides all information to the parameters and calculates constraints with respect to thruster position. There are 4 modifiable controllers provided in the file at the start. These are given as
  - **tc** = forces in surge sway and moment yaw
  - **Thruster_lock** = Azimuth or fixed thrusters. 0 = fixed thrusters, 1 = azimuth thrusters
  - **Thruster_control** = Core thruster control. 1 = shaft speed controller, 2 = torque controller, 3 = power controller and 4 = combined torque and power controller.
  - **Thrust_algorithm** = Optimization algorithm. 1 = pseudoinverse optimization and 2 = Quadratic regulator

  Rest of the file provides the parameters used for CSAD which are possible to modify at users own responsibility.

E.2 Experimental Setup CSAD

This folder provides simulink setup which is used while testing TAPM in the MC Lab.

E.3 QR With Output to Real-Time

This is a folder with simulink model of the quadratic regulator without the quadratic solver and instead outputs the matrices to a real time system which can solve the QP in real-time.

E.4 LabVIEW Quadratic Testing

In this folder, the LabVIEW program tried to be utilized as a solution for implementing real-time quadratic regulator can be found. In order to open it, the computer must be equipped with LabVIEW 2015.

E.5 Bollard Pull

Here, all the tests related to bollard pull can be found. Files named thruster(i)_forward/backward ((i) name of thruster) loads and filters the data.In order to run these files, the path name for each thruster needs to correspond to the folder the data from tests are in. The two files named run_file_forward/backward creates the mapping seen in Appendix B.

In sub folder "signalmapping" are all the curve fit results provided with names corresponding to their function. In the folder there is a file called Mapping.m which provides the polynomial coefficients from each result.

E.6 Misc

This folder provides the following items

- Data sheet for OS OMA-2820-950 (motor for the thrusters)
- Data sheet for current sensor used in signal mapping
- Data sheet for hall effect sensor used in signal mapping
- Speed-thrust graph provided by Inocean for propulsion unit UUC505 FP for both 5.5MW and 6.0 MW