Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Ocean Operations and Civil Engineering

Rahul Nath Raghunathan

Development of a Deep Learning based Thrust Allocator for Dynamic Positioning of Fully Actuated Vessels

Master's thesis in Ship Design
Supervisor: Guoyuan Li
Co-supervisor: Robert Skulstad, Houxiang Zhang
June 2021

NTNU
Norwegian University of
Science and Technology

Rahul Nath Raghunathan

Development of a Deep Learning based Thrust Allocator for Dynamic Positioning of Fully Actuated Vessels

Master's thesis in Ship Design
Supervisor: Guoyuan Li
Co-supervisor: Robert Skulstad, Houxiang Zhang
June 2021

Norwegian University of Science and Technology
Faculty of Engineering
Department of Ocean Operations and Civil Engineering

NTNU
Norwegian University of
Science and Technology

MASTER'S THESIS 2021
FOR
RAHUL NATH RAGHUNATHAN

Development of a Deep Learning based Thrust Allocator for Dynamic Positioning of Fully Actuated Vessels

The Dynamic Position (DP) system of ships plays an important role in the modern maritime operations which are undertaken in territories farther and farther from the shore. Operations such as offshore windmill installation, offshore oil exploration, and construction require extensive use of vessels with DP capabilities. Thus, DP capabilities of vessels have always seen improvement by the constant demand in the market. A DP system has seen advances over its controller aspect as well as its thrust allocator from time to time. While the advancement of the controller aspect is out of the scope of this thesis, the advances in thrust allocation have been achieved through ingenious use of numerical optimisation techniques. These numerical optimisation-based allocators have been able to account for operational constraints such as reduced power consumption, avoid thruster-hull interaction, singularity avoidance, etc. In this thesis, the development of a novel thrust allocator based on Deep Learning (DL) is being investigated. Deep Learning which has revolutionized every aspect of the modern engineering field holds great prospect in being used in allocation problems. This aspect has already seen success in the aeronautic and aerospace industry and is being developed for Maritime applications. The research in the maritime field for DL thrust allocation is gaining traction with DL allocators being developed for DP operations in NTNU, Alesund. This thesis will try to utilize NTNU's R/V Gunnerus in a co-simulation environment to develop a DL thrust allocator and contribute to this research field and hope to attract more research and pave the path to real-world applications in the future.

Objective/Research Questions:

- Develop a Deep Learning (DL) based thrust allocator for DP operation of NTNU's Research Vessel R/V Gunnerus with a focus on operations in station-keeping mode and in low-speed maneuvering mode. The allocator shall have features of power minimisation, forbidden zone management for azimuth thrusters, force saturation constraints, and magnitude rate constraints.
- The DL allocator is to be tested against a classic Sequential Quadratic Programming (SQP) based allocator to identify performance differences.
- The implementation and verification of the DAE and SQP based allocator to be done in a co-simulation environment. This means each allocator should be packaged into FMU for deployment in the co-simulation.

Work Tasks:

- Carry out a literature review on the topic.
- Develop a Deep Learning thrust allocator after referring to the current state-of-the-art methods and add features to the existing ones to increase robustness and performance.
- An SQP allocator is to be created that meets the same constraint as the DL allocator.
- Both allocators to be packaged as FMU and tested in co-simulation environment Vico.
- Analyse the performance of the DL allocator compared to the SQP allocator and comment on performance and add suggestions for future work.

The scope of work may prove to be larger than initially anticipated. Subject to approval from the advisor, topics from the list above may be deleted or reduced in extent.

The thesis should be written as a research report with summary, conclusion, literature references, table of contents, etc. During preparation of the text, the candidate should make efforts to create a well arranged and well written report. To ease the evaluation of the thesis, it is important to cross-reference text, tables and figures. For evaluation of the work a thorough discussion of results is needed. Discussion of research method, validation and generalization of results is also appreciated.

The thesis shall be submitted in electronic version according to standard procedures. Instructions are found on the NTNU website (Inspera) and on Blackboard. In addition, one paper copy of the full thesis together with an electronic device carrying all relevant documents and files shall be submitted to your supervisor.

Prof. Guoyuan Li
Supervisor

Robert Skulstad, PhD candidate
Co-supervisor

Delivery: 25.06.2021 Signature candidate:

Rahul Nath Raghunathan

Preface

This thesis presents the work done for the course work: **IP501909** - Ship Design Master's Thesis at NTNU(30 ECTS), and represents the final delivery for a Master of Science in Ship Design. The thesis was written in its entirety by Rahul Nath Raghunathan during the time period of January to June 2021.

The work was motivated by the ongoing research at NTNU Ålesund, where Machine Learning is applied to the various maritime application. There have been fewer works related to thrust allocation using machine learning in the maritime field. Researchers at NTNU have been able to use machine learning-based thrust allocation for marine vessels and the thesis builds upon this foundation.

The thesis has contributed to producing a Deep Learning-based thrust allocator that can take into account power minimisation, magnitude and angle saturation constraints, rate constraints, and forbidden zone management for thrusters in a fully actuated vessel. The allocator has been packaged into an FMU for testing in a co-simulation environment. To benchmark the performance, a classic SQP thrust allocator has also been made and packaged into FMU for testing. The test results show comparable performance for the Deep learning-based allocator against the classic allocator by meeting the prescribed constraints.

In the thesis, sufficient information regarding background knowledge used for building up the thesis has been provided. The reader is requested to refer to relevant literature prescribed in different sections to be informed about more details. The reader is assumed to have some knowledge of Machine Learning and thrust allocation to get a full picture of the thesis.

Acknowledgement

This thesis is the culmination of the efforts that I have put in during the 2 years of my stay in Norway for pursuing my Master's Degree at NTNU. Pursuing a Master's Degree at NTNU has been a dream and fulfilling it brings immense pleasure and satisfaction.

First of all, I would like to thank my supervisors, Prof. Guoyuan Li and Prof. Houxiang Zhang for allowing me to do a Master's thesis under their supervision and providing me with the right support at the right time. They have been a constant source of motivation during meetings and interactions. Their motivation had a great impact during the last 6 months of the Master's thesis period.

Secondly, I would like to thank my co-supervisor Robert Skulstad-PhD Candidate at NTNU for his immense support during the thesis. I have to thank him for providing his Machine Learning framework for thrust allocation upon which I built my thesis. Without this foundation, the scope of the thesis could have been quite low. He has tolerated my stupid questions and bugs in the code and always supplied great information that solved my problems. I have enjoyed all discussions with him through Microsoft Teams and in his cabin and I would miss it after the end of the thesis.

I have to thank a few other persons from NTNU Ålesund who had great influence in my 2 years of stay. I want to thank Assoc. Prof. Henry Peter Piehl for his initial projects in Python during the first semester at NTNU for motivating me to take programming more seriously. I have been part of NAVO NTNU - a student robotic boat project at NTNU Ålesund and I have to thank Assoc. Prof. Henrique M. Gaspar for his willingness to co-ordinate NAVO NTNU during its initial stage.

Life at NTNU would have been quite different without friends that I have made through NAVO NTNU project. I have to thank my friends Michal, Mingda, Sai, Fahim, and Saravanan who have been a great support for me both in happiness and sorrow. I have to also thank my friends in India namely Shana, Umair, Balu, and Joel for all conversations that made me happy. A big thanks to Ramees-a 2019 NTNU Ålesund graduate who motivated me to come to Norway.

Finally, I am forever indebted to my Mom, Dad, and Brother who supported me throughout my stay here. They have sacrificed many things to sent me here and their unconditional support and love have been my motivation as always.

For all things lost and gained in pursuit of passion...

Rahul Nath Raghunathan
June 25, 2021

Summary

Dynamic Position (DP) System is an important development in the history of marine vessels and has contributed to the development of various fields such as offshore oil and gas, offshore renewable energy, subsea pipelaying, offshore construction, and general marine research. The ability of a marine vessel to hold its position or track a predefined path has helped humans to tolerate ever-varying sea conditions and undertake marine operations with confidence.

A simple DP system consists of a motion controller that requests generalised forces and moments to thrust allocator that distributes the request into various actuators in the vessel. It is the constraints put on the thrust allocator that brings great improvement to the DP system. Constraints such as power minimisation, saturation constraints, and rate constraints are crucial to prevent blackout in the ship and to reduce wear and tear of the actuators. Classically this has been solved used techniques such as pseudo-inverse and other optimisation theory-based schemes.

In this thesis, a notion of using Deep Learning(DL) for thrust allocation is put to use. It is considered that a DL model, a subset of Machine Learning(ML) model that learns representation from data through the principle of optimisation can learn features from data and perform the role of a thrust allocator. The literature survey investigating this showed that fewer works have been done in this direction and especially in the maritime field. Researchers at NTNU Ålesund have been investigating this topic and this thesis builds on top of the foundation set by these researchers.

In the thesis, a thrust allocator based on Deep Autoencoder(DAE) network has been developed. The allocator has been designed for NTNU's R/V Gunnerus vessel-a fully actuated ship with two azimuth thrusters and a tunnel thruster. Using a DAE network solved the problem of training data for the ML model. By generating thruster commands for three actuators using pseudo-random numbers for a sample size of 3 million, the forces in Surge and Sway and moment in Yaw that would have been produced by the random combination of these commands were obtained using the thrust configuration matrix. This Surge, Sway, and Yaw are the input values of the encoder part of the network and the decoder part tries to reconstruct the input at the output. This network has simplified the process of having a previous allocator generate training data by logging forces and different command values.

In the DAE network, the commands for the thrusters are obtained in the latent code layer - a representation of input data in a different form. This latent code layer brings in the possibility to add power minimisation, saturation constraint for magnitude, rate constraint, and forbidden zone management through different loss functions and a custom layer in the encoder part of the network. Thus the developed DAE allocator has these features.

To compare the performance of the new allocator, a classic Sequential Quadratic programming(SQP)

based allocator obeying the same constraints was made and put to test. The testing has been conducted in a Co-simulation setting. Co-simulation employs FMU(functional mockup units) for different components of the simulation and eases the process of data exchange. The decision to use co-simulation is based on the research strategy at NTNU Ålesund to promote co-simulation. Both the allocators have been made into FMU and simulated using the Vico co-simulation framework.

The allocators were put to test in three scenarios namely: a low-speed four corner test, stationkeeping test, and stationkeeping test with thruster failure. The DAE allocator has a comparable performance similar to the SQP allocator in the 4-corner test and has better performance in the Stationkeeping test. During a stationkeeping test with thruster failure, an azimuth thruster is turned off to imitate a thruster failure and both allocators fail to meet the requested force by the motion controller. Despite failing to meet the controller request, the SQP allocator returns the vessel to its origin position quite quickly whereas the DAE allocator has a jitter motion when it tries to return to the original position. Thus DAE allocator can be said to have inferior performance only in this case which is considered a future scope of the thesis. In all cases, the DAE allocator was seen to meet the constraints in a robust manner.

The future work of the thesis includes exploring a better data generation strategy different from pseudo-random numbers, improving the performance in thruster fail cases, and exploring a solution of fault - detection thrust allocation strategy. Extending the method to other actuators like rudders, Voith-Schneider propeller, etc. remains a new territory to be explored and validated. The use of hybrid allocation strategies combining DL and numerical approach or aiding numerical allocation using machine learning model can also be considered as future work.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction and Motivation

Dynamic Position (DP) maintains the position and heading of floating structure in reference to a fixed position or pre-defined track for marine operations by only using active thrusters[1]. In the modern-day, where marine operations are not only limited to cargo transport but extended to deep offshore exploration, construction, and autonomous shipping, more and more marine vessels are being fitted with DP systems to meet the required operational profiles of the maritime industry. Many vessels such as Mobile Offshore Drilling Units, Offshore supply vessels, Offshore Wind Farm Vessels, etc. extensively utilize the DP system without which it cannot meet its operational requirement of station keeping. Special path tracking applications are done using the help of DP System in cable and pipe laying vessels and in the case of ROV operations[1].

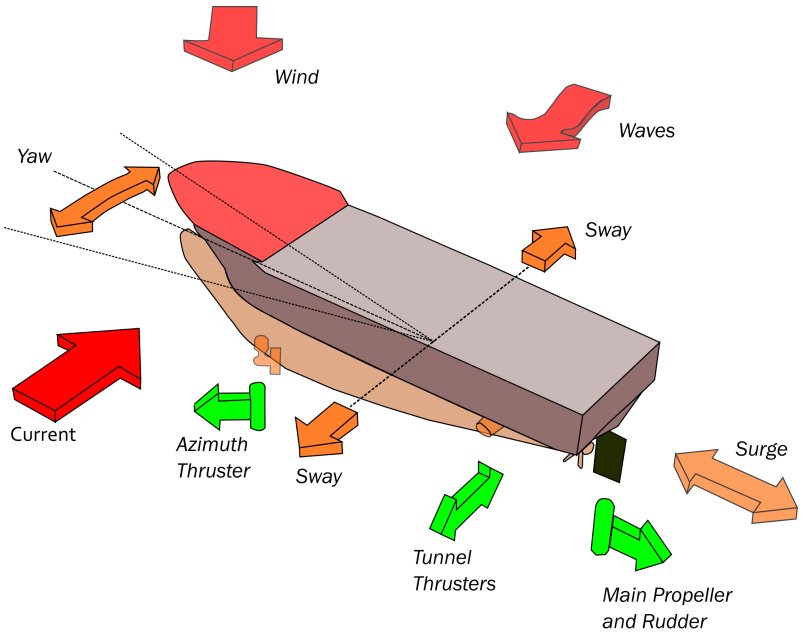


Figure 1.1: DP System Forces

Source: Adapted from [2]

Figure 1.1 shows the forces acting on a vessel and the three Degree of Freedom(DOF) that the DP system tries to control using the thrusters. The DP system that is currently operational has seen

more than 30 years of continuous research and development by many companies and thus each company offers a slightly different implementation of the DP system in the market. But the basic principle of the system design can be seen in Figure 1.2 where the author has presented the entire system architecture.

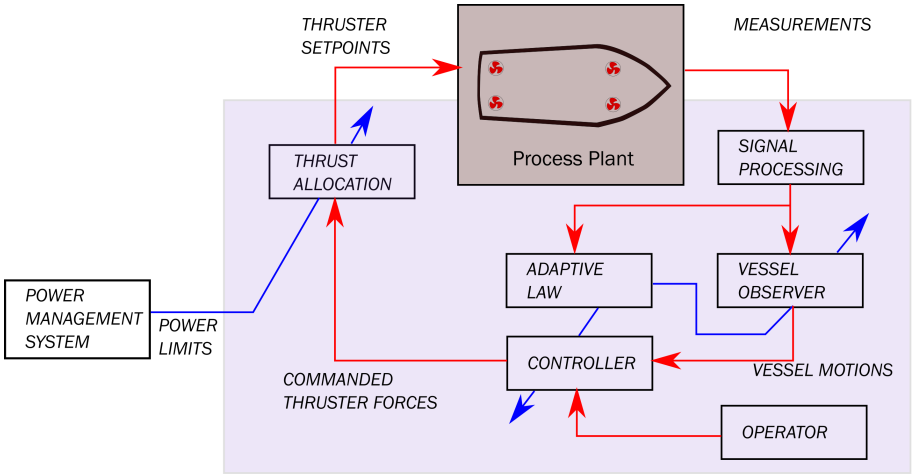


Figure 1.2: DP Architecture

Source: Adapted from [1]

In this thesis, the focus would only be on the Thrust Allocation module. A definition of Thrust Allocation(TA) can be found in [1] which states: *The high-level positioning controller produces a commanded thrust vector* $\tau_c \in \mathbb{R}^3$ *in surge, sway and yaw. The problem of finding the corresponding force and direction of the thrusters that meets the high-level thrust commands is called thrust allocation.* According to Fossen [3], a fully actuated marine craft has equal or more actuators than the DOF.

Most of the marine vessels have more actuators than their DOF under control for DP for the sake of redundancy against failure. The thrust allocation module takes into consideration the desired forces computed by the high-level controller and imposed constraints such as minimised power consumption, avoid forbidden zones, avoid thruster-thruster interaction, and thruster-hull interactions[4] and allocate commands to each actuator present in the ship to produce generalised forces and moments in the surge, sway and yaw directions.

In Figure 1.3, it can be seen that for a vessel with a commanded force of 500kN, the thrust allocator may decide to produce individual actuator forces in two ways: one with 1954 kW power and the other with 1371 kW. If the allocator is optimised for power, then allocation with 1371 kW is preferred. This can be considered as the basic intuition for a power optimised allocator.

In this thesis, the terms thrust allocation and control allocation will be used interchangeably. Also, Machine Learning(ML) model, Deep Learning(DL) model, and Neural Network (NN) are used interchangeably without loss of meaning.

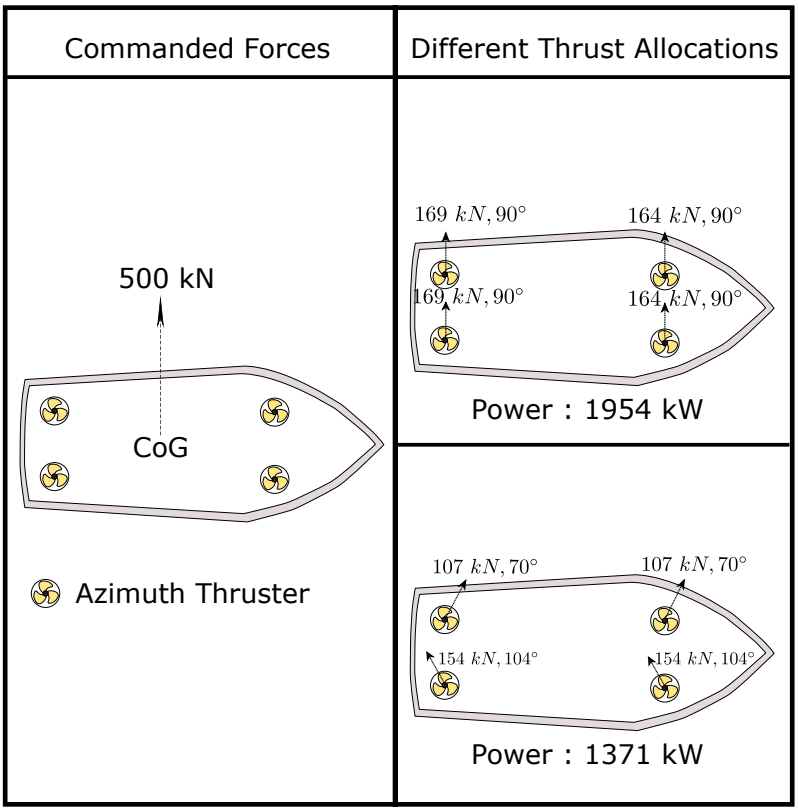


Figure 1.3: Intuition of thrust allocation

Source: Adapted from [5]

1.1 Motivation

Machine Learning(ML) has become a driving force in the context of research and is showing better performances than the existing methods in many domains. For example, one of the recent great advancements in molecular biology came through the use of ML. Alphafold [6] - a DL model was able to predict the structure of the protein in 3D when the input was a 1D sequence of amino acids. The model was able to score more than 90% in Global Distance Test in the CASP (Critical Assessment of protein Structure Prediction) challenge which is a breakthrough in recent times where the model is able to predict the 3D protein structure on the same level of performance as a complicated experimental method. Breakthroughs like this solidify the prospect of ML as a suitable means to discoveries and findings in the context of research.

ML has found its way into maritime field with application to navigation, control, fault detection etc. The transition and focus to a fully autonomous shipping concept is driving the adoption of ML and its products into the maritime field.

The main motivation of this thesis lies in the research activities at NTNU Ålesund where ML is applied to different aspects of marine operations to solve different challenges or to add support to numerical methods. During the literature survey for thrust allocation using ML, very little work has been done exploring this research path. Research publications were mainly in the Aircraft domain and less in the maritime domain. To explore such a topic seemed very interesting and was an added motivation. Fortunately, work was done by researchers at NTNU [7],[8] which introduced

thrust allocation using Neural Networks for DP operations. This thesis is the continuation of this research path.

The use of digital twin technology is increasing in all domains of engineering and the same trend is seen in the maritime industry. This thesis also uses the digital twin technology to embrace the use of such technology for realistic simulations. To test and verify the performance of the allocator, the simulations are done on a co-simulation environment using a digital twin of NTNU's R/V Gunnerus[9]. The application of digital twin for testing and verification is part of the research strategy at NTNU Ålesund to bolster the use of co-simulation as a maritime standard. A similar trend can be seen in the industry [10] where multiple companies have partnered to develop and use co-simulation and set it as the standard. As a result of this strategy, the deliverables of the thesis will be the Functional Mockup Unit(FMU) of the allocator that can be used in co-simulation. It is hoped that in the future more contributions can be made in form of FMUs that can be tested and validated in co-simulation. The thesis hopes to motivate such efforts.

1.2 Objectives

In this thesis, the following would be considered as the objectives:

1. **Objective 1**: Develop a Deep Learning (DL) based thrust allocator for DP operation of NTNU's Research Vessel R/V Gunnerus[9] with a focus on operations in stationkeeping mode and low-speed maneuvering mode. The allocator shall have features of power minimisation, forbidden zone management for azimuth thrusters, force and angle magnitude constraints and rate constraints.
2. **Objective 2**: The DL allocator is to be tested against a classic Sequential Quadratic Programming(SQP) [11] based allocator to identify performance differences.
3. **Objective 3**: The implementation and verification of the DL and SQP based allocator to be done in a co-simulation environment. This means each allocator should be packaged into FMU for deployment in the co-simulation.

1.3 Contributions

The contributions of the thesis can be summarised as:

1. **Contribution 1**: The method prescribed in work [8] has been extended and robustness and performance are added with respect to Power minimisation, forbidden zone management, force and angle magnitude constraints and rate constraints.
2. **Contribution 2**: A robust hard constraint on rate change values has been added to the allocator through the use of custom layers in the DL model which is considered a novel approach.
3. **Contribution 3**: The DL allocator performance has been compared against an industry-standard numerical approach to find the performance difference.

4. **Contribution 4**: The DL allocator and the industry-standard SQP allocator have been implemented as an FMU that can be run in co-simulation environments aiding future research.

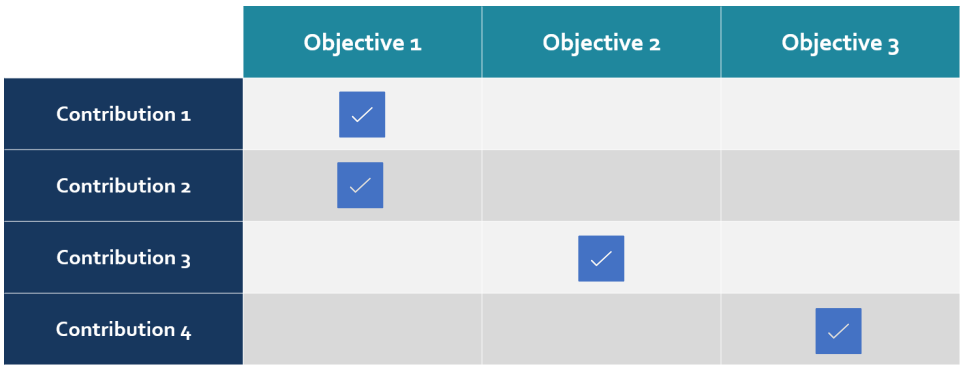


Figure 1.4: Objective and contribution matrix

A matrix for how the objectives of the thesis have been met through different contributions can be seen in figure 1.4.

1.4 Scope

The scope of this thesis could be simplified as shown in Figure 1.5. This thesis tries to combine the field of Deep Learning, Optimisation theory, and Thrust allocation. While the crux of the thesis is to incorporate a DL model for thrust allocation, simple optimisation strategies are used inside the DL model to incorporate desired features of thrust allocation.

The developed DL allocator will be tested for different cases against a benchmark numerical thrust allocation method to find the difference between the two.

All testing will be done using the co-simulation environment Vico [12] for NTNU's R/V Gunnerus. The programming language for the thesis will be Python and the machine learning framework will be Keras with Tensorflow backend.

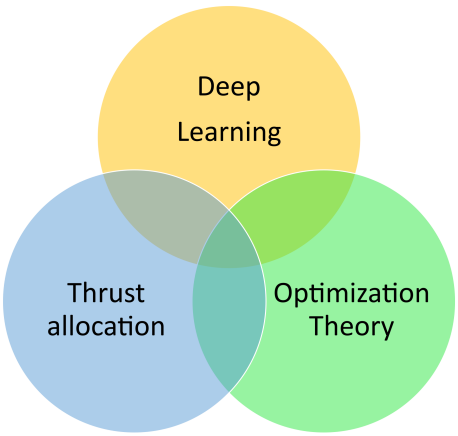


Figure 1.5: Scope

1.5 Thesis Outline

Chapter 1 introduces the topic of the thesis and motivation for pursuing the thesis. In this thesis, relevant theory and basics of thrust allocation using DL and numerical scheme is presented in two chapters. Literature Review in **Chapter 2** explores relevant literature on thrust allocation whereas **Chapter 3** introduces the basic concept of Machine learning used in the thesis for developing the DL allocator. **Chapter 4** details the development approach followed in the thesis with details to the development of DL allocator and SQP allocator. Before the two allocators are put to test in simulation, a short introduction to the co-simulation environment is provided in **Chapter 5**. In **Chapter 6**, the result of testing the two allocators in the different scenarios is presented along with the discussion of the results. The thesis concludes with **Chapter 7** summarising the thesis and putting forward suggestions for improvement.

Chapter 2

Literature Review

This chapter summarises various literature referred to get an understanding of the development of thrust allocation for DP operation of ships. After this, a short section to introduce basic control theory for the marine vessel is given followed by the method of classic control allocation strategy.

2.1 Thrust Allocation in Literature

2.1.1 Numerical Optimisation Approaches

In 2004, the authors of [11] have developed a single step constrained non-linear control allocator for DP operation of a ship using Sequential Quadratic Programming. The scheme takes into account singularity avoidance of azimuth thrusters in addition to power minimisation, constraints on force and angle saturation, and constraints on force rate and angle rates. This work is considered as a reference classical control allocation for benchmarking in this thesis. In the Master's thesis [4] published in 2009, the author introduces a thrust allocation for DP operation that employs Quadratic programming with the Disjunctive programming technique. This method linearises the constraints imposed on thrusters by polygon approximation[13]. The allocation scheme takes into account saturation values of forces and angles of azimuth thrusters. The method is also applicable for rudders.

The work titled "*Control allocation-A survey*" [14] published in 2013 is an excellent reference that gives a broad overview of control allocation schemes applied in various domains. The work discusses control allocation strategies for linear and non-linear effectors with and without constraints. The works also discuss different linear, quadratic, and non-linear optimisation techniques used in these schemes. In the work [13] published in 2013, the authors propose a fuel optimal thrust allocation for DP operation. They include a term in the cost function which accounts for fuel consumption of diesel generators in contrast to the usual power minimisation term in the cost function. The scheme puts constraints on force saturation values, azimuth angles, force change rates, azimuth angle change rates, and power constraints on the electrical bus. The optimisation scheme used is a convex Quadratic programming scheme with linear constraints. In the Master's thesis [15] published in 2013, a thrust allocation scheme for DP operation with a feature of using azimuth thruster for ice clearance is discussed. In the thesis, results are given for a vessel with 4 azimuth

thrusters and two bow thrusters where two azimuth thrusters in the stem of the vessel are used for ice clearance. To achieve this feature, the thrust allocation formulation in [11] is modified to decouple the two azimuth thrusters and added as separate terms in the objective function to allow the vessel operator to direct the azimuth thrusters. These decoupled thrusters are still included in the singularity avoidance feature of allocation and constraints such as saturation of forces and azimuth angles, rate values of forces, and angles are considered.

In 2015, authors of [5] have developed a thrust allocation method based on the Sequential Quadratic Programming technique that takes into account hydrodynamic effects such as thruster-hull interaction and thruster-thruster interaction. The allocation scheme takes into account power minimisation and saturation limits of the thrusters in addition to the above-mentioned hydrodynamic interactions. In the work, they were able to demonstrate a performance improvement of 2-5 % power consumption reduction and 5 % higher water current handling when compared to a normal forbidden zone managing thrust allocation algorithm for the vessel tested. In 2016, authors of [16] have developed a thrust allocation scheme that takes into account the load variation in the power plant of a ship due to varying force requirements from thrusters owing to the unpredictable marine environment. This scheme allows small deviation from the requested force value to improve load variation on the powerplant. It uses a quadratic formulation for power optimisation with consideration to thruster saturation values. In 2017, authors of [17] have developed a Model Predictive Control(MPC) based thrust allocation scheme for DP operations. Their non-angular MPC scheme has an optimisation horizon larger than the single-step method[11] and achieves an environmental disturbance filtering feature. This filtering allows reducing the thrust and thruster rate commands for azimuth and non-azimuth thrusters reducing power consumption and load variation on power plants.

2.1.2 Learning Based Approaches

In the work [18] presented in 2010, the authors introduce an adaptive genetic algorithm(GA) based thrust allocation. Their thrust allocation scheme takes into account power minimisation, force and angle saturation and constraints on the rate of change of forces and angles of the azimuth thrusters. Their scheme is tested on a semi-submersible offshore platform fitted with 8 azimuth thrusters and demonstrates power minimisation while obeying all the prescribed constraints. In the work published in 2015 [19], the authors present a hybrid thrust allocation scheme utilising GA and SQP method. In the method, GA is used to find a global optimum which is then refined by the SQP method to find a local optimum. They achieve this by providing the SQP with a starting point for search from the values obtained with GA optimisation. The scheme is tested on a drilling rig with 8 azimuth thrusters with features such as power minimisation, force and angle saturation handling and constraint on the rate of change of forces and angles.

In the work [7] presented in 2018, Skulstad et al. employ a supervised neural network model for control allocation for DP operation of a ship. The allocator is prescribed for a vessel with non-rotatable thrusters. The work describes two allocators: Allocator 1 that takes in only force request from a controller and Allocator 2 that takes in force request from current time step and previous time step. Allocator 2 is able to meet force saturation constraint and force rate change constraints while Allocator 1 is able to meet only force constraints.

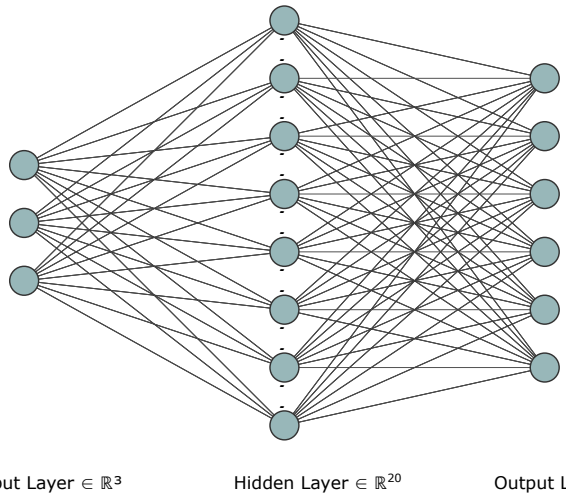


Figure 2.1: Neural Network Architecture for TA by Skulstad et al.

The learning method used in the work is the Extreme Learning Machine [20] method. The data for the NN was generated for the vessel using a professional simulator from the company named Offshore Simulator Centre (OSC) AS in Norway. The vessel in the simulator was pushed to its saturation force level and subsequently, a negative saturation force command is issued to expose the maximum change rate of thrusters. This data is recorded and fed to NN for training. The NN structure has 3 input neurons, 20 hidden layer neurons and 6 output neurons. The six output neurons correspond to the 6 control variable for the thrusters of the vessel. This neural network architecture is provided in figure 2.1.

The work in 2018 by the authors of [21] is the motivational element in the thesis. Their method of solving the control allocation problem of an aircraft using Deep learning is the foundation of this thesis. In their work, the authors develop a Deep Autoencoder(DAE) based control allocator for an aircraft solving the problem of any Machine learning-based model-generating the data to train the model beforehand. In usual cases, a method is used to generate data that is then fed to the network to learn the representation and features of the dataset. This lead to an awkward problem of creating a numerical method beforehand to generate data. Using DAE, artificial data can be generated and the network can be trained without labels in a supervised learning scheme. This work is termed as Original DAE work in the thesis.

Their deep learning-based control allocator has features of saturation handling of the actuators and power minimisation. They train their network in a specific way-they pre-train the decoder part of the autoencoder that takes in actuator commands and outputs forces in a supervised learning manner. Then they use a transfer learning approach of freezing the optimal decoder layers and train the encoder part of autoencoder in an unsupervised learning manner. They use a custom activation function at the end of encoder layers to obtain saturation handling functionality. The neural network architecture for this work is given in figure 2.2.

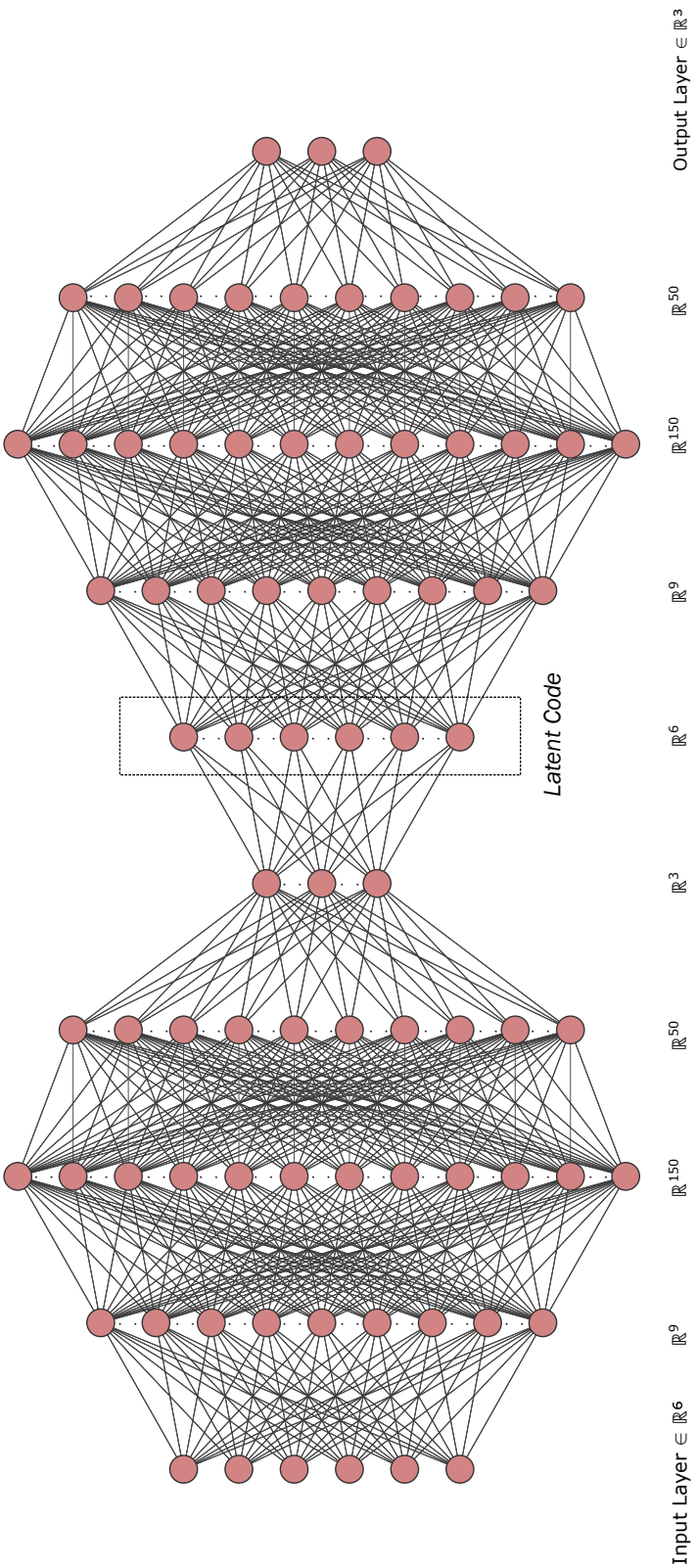


Figure 2.2: Thrust Allocator Architecture from Original DAE Work

In another work [22] by authors in 2019, they develop a control allocation formulation using Reinforcement learning for another aircraft. The work uses a tabular Q-learning approach where the agent is trained in offline simulation. In the simulation, the aircraft is initialised at an offset

position from the target and learns to reach the target. They show that without prior knowledge about the model, the RL based control allocation scheme can enable the aircraft to reach the desired altitude by using up to four control effectors and optimised for reducing drag by avoiding opposite deflection of control effectors.

In the Master's thesis [23] presented in 2020, the author introduces a Deep Reinforcement Learning(DRL) based Dynamic positioning system for a small scale Autonomous Surface vessel. The method incorporates both motion control and control allocation into the system once it obtains signals from the guidance system. The learning method used for Reinforcement learning is Proximal policy optimisation. The method was tested in a simulation environment for the ASV as well as in the real world. The ASV is configured to have a bow thruster and two azimuth thrusters at the stern. In the thesis, a comparison of the proposed DRL method against classic control allocation methods such as Quadratic programming(QP) and the Psuedoinverse(PI) method is presented. The DRL method is shown to have improved performance than QP and PI method when subjected to tests for large setpoint changes, four corner test, four corner test with ocean current and stationkeeping tests with wind and currents.

The next approach of using Machine learning for control allocation is based on Reinforcement Learning(RL). In the work [24], the authors present a novel approach to solving dynamic control allocation problems using Reinforcement learning. The approach is tested of an F-16 airplane simulation. They use an H_∞ controller to learn optimal control allocation online using the measured data without the requirement of knowing the system dynamics.

2.2 Basic Control Theory of Marine Vessels

A marine vessel is subject to motion in 6 Degree of freedoms. Their notation in SNAME convention is given in Table:2.1

DOF	*Forces and moments*	*Linear and angular velocities*	*Positions and Euler angles*
1:Motion in x-axis (Surge)	X	u	x
2:Motion in y-axis (Sway)	Y	v	y
3:Motion in z-axis (Heave)	Z	w	z
4:Rotation about x-axis (Roll)	K	p	ϕ
5:Rotation about y-axis (Pitch)	M	q	θ
6:Rotation about z-axis (Yaw)	N	r	ψ

Table 2.1: SNAME(1950) notation for marine vessels

Source: [3]

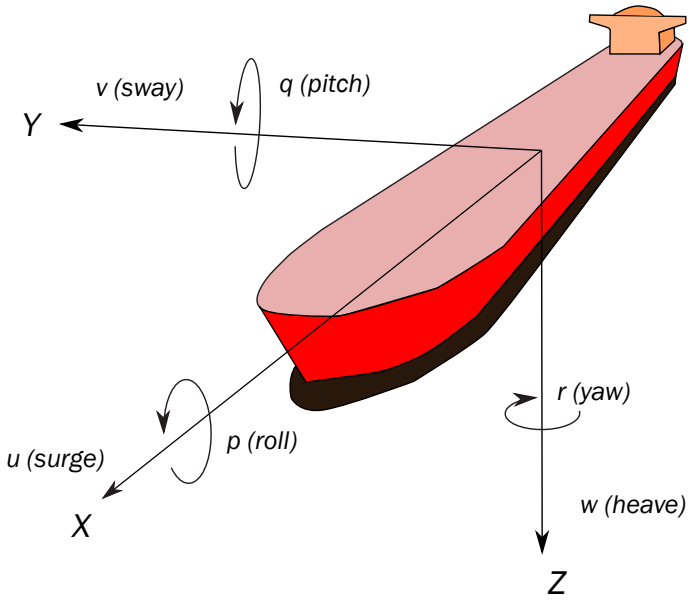


Figure 2.3: Vessel motion and Body Coordinate System

Different Coordinate system used for motion study are given below[3]:

1. **Earth-Centered Inertial(ECI) Frame**: This is a reference frame with origin at the centre of Earth and is non-accelerating. It is denoted by frame i (x_i, y_i, z_i)
2. **Earth-centered Earth-fixed (ECEF) Frame** : This frame has it's origin at the center of earth and rotates with the motion of Earth. Denoted by $e = (x_e, y_e, z_e)$
3. **North-East-Down Frame**: This frame is defined relative to Earth's reference ellipsoid. It is defined as a tangent plane on the surface of the surface of the Earth moving with the vessel. This frame is denoted by $n = (x_n, y_n, z_n)$ with x-axis pointing to true North, y-axis pointing towards East and z-xis pointing downward perpendicular to the tangent plane.
4. **Body Coordinate System**: The body-fixed reference frame $b = (x_b, y_b, z_b)$ is a coordinate system that is fixed to the vessel and moves with it. Its origin is usually set to the midship in the waterline of the vessel. This frame is given in figure 2.3

While ships are subject to motion in 6 DOF's, in DP operation of ships, the main concern is to control motion in Surge,Sway and Yaw. For this a 3-DOF model can be represented. This model is aimed at low speed operation of speed upto 2 m/s [3]. 3-DOF model can be represented as given below:

$$\dot{\eta} = R(\psi)\nu \tag{2.1}$$

$$M\dot{\nu} + C(\nu)\nu + D(\nu)\nu = \tau + \tau_{wind} + \tau_{wave} + \tau_{current} \tag{2.2}$$

where $\nu = [u, v, r]^T$ and $\eta = [x, y, \psi]^T$. The definition of notations used in equation 2.1 and 2.4 are given below:

1. M - Inertia matrix including added mass terms

2. C - The matrix of coriolis and centripetal terms
3. D - Damping matrix
4. τ - the control force and moment acting on the body-fixed frame
5. τ_{wind},τ_{wave},$\tau_{current}$ - environmental disturbances from wind,wave and current respectively.
6. $\eta = [x, y, \psi]^T$ - the position and heading in Earth-fixed coordinate system
7. $\nu = [u, v, r]^T$ - Surge, sway and yaw velocities in body-fixed frame.
8. $R(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$ - Rotation matrix transforming velocities from body-fixed coordinate system to earth-fixed coordinate system.

2.2.1 Control System Structure for DP

The Control system used for DP system in ship can be viewed broadly through the flowchart given in figure 2.4. The flowchart include a high-level motion control,followed by the Control Allocation strategy and low-level controls for the actuators. Here the effectors and actuators can be distinguished by their definition from [14]: Effectors are devices that generates force on the mechanical system(Eg: Rudder,Fin) and actuators are electromechanical device that controls the magnitude and direction of these forces.

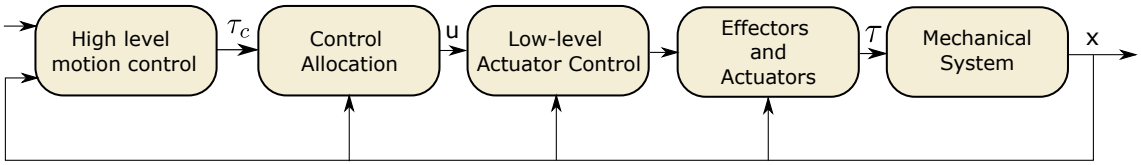


Figure 2.4: Control system structure including control allocation

Source: Modified from [14]

In the thesis, the term actuator and effector refers to thrusters.The mechanical system in this case refers to the vessel itself. The other blocks in figure 2.4 are discussed in subsequent sections.

2.2.1.1 High-level Motion Control

A high-level motion controller takes in a desired control signal and compares the current state of the system to produce a virtual control input τ_c (forces and moments corresponding to the degree of freedom) to the control allocation scheme.

The high level motion controller could be a simple PID (Proportional-Integral-Derivative) controller that takes in setpoints for position and heading in case of DP Stationkeeping operation. The controller then compares its current state and outputs forces and moments to compensate for deviation from the setpoint. The simple equation for control through PID is given in equation 2.3

$$x(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de}{dt} \tag{2.3}$$

where the terms are

1. $x(t)$ = Control variable for PID
2. K_p = Proportional gain
3. $e(t)$ = Error value = Desired value- Current value
4. K_i = Integral gain
5. K_d = Derivative gain
6. de = Change in error
7. dt = Change in time

2.2.1.2 Control Allocation

Control Allocation or Thrust allocation in simple terms, is the process of distributing the generalised control forces $\tau_c \epsilon \mathbb{R}^n$ to the actuators in terms of control inputs $u \epsilon \mathbb{R}^r$,$n = no.\ of\ DOF$, $r = no.\ of\ actuators$ [3]. There are different type of Control allocation method starting from simple pseudo-inverse to complex quadratic programming based methods. This section have been taken from [3]. Consequent section will give a discussion for control allocation using a Sequential Quadratic Programming based approach for rotatable and non-rotatable thrusters.

The allocation is an optimisation problem, which in simple case is unconstrained. But when limitations are set on to the output produced such as magnitude constraints, this allocation becomes a constrained optimisation problem.

The force generated by a propeller, rudder or fin can be approximated linearly as $F = ku$, where k is the force coefficient and u is the control input. The various control input u and corresponding force vector from different actuator is given in Table 2.2.

Actuator	*u (control input)*	α(*control input*)	f^T (force vector)
Main propellers(longitudinal)	Pitch and RPM	-	[F,0,0]
Tunnel thrusters(transverse)	Pitch and RPM	-	[0,F,0]
Azimuth(rotatable)thruster	Pitch and RPM	Angle	[Fcos(α),Fsin(α),0]
Aft rudders	Angle	-	[0,F,0]
Stabilizing fins	Angle	-	[0,0,F]

Table 2.2: Control input for various actuators

Source: [3]

The force(Surge,Sway) and moments(Yaw moment) for a 3DOF system corresponding to force vector $f = [F_x\ F_y]^T$ can be written as equation 2.4 where $\begin{bmatrix} l_x \\ l_y \end{bmatrix}$ are the moment arms.

$$\tau = \begin{bmatrix} f \\ r \times f \end{bmatrix} \overset{3DOF}{\Longrightarrow} \begin{bmatrix} F_x \\ F_y \\ F_y l_x - F_x l_y \end{bmatrix} \tag{2.4}$$

The equation in 2.4 can be written in a general form as

$$\tau = T(\alpha)f = T(\alpha)Ku \tag{2.5}$$

where $K \epsilon \mathbb{R}^{r\times r}$ is the diagonal force coefficient matrix and $u = [u_1, ..., u_r]^\intercal$ is the vector of control inputs. $\alpha = [\alpha_1, ..., \alpha_p]^\intercal \epsilon \mathbb{R}^p$ is a vector of azimuth angles with p equal to number of azimuth thrusters.

In equation 2.5, $T(\alpha) \epsilon \mathbb{R}^{n\times r}$ is the thrust configuration matrix. This matrix relates to the locations of the actuators. An example for force and moments in Surge,Sway and Yaw respectively for a marine vessel is shown in Figure 2.5 with corresponding matrices in equation 2.6.

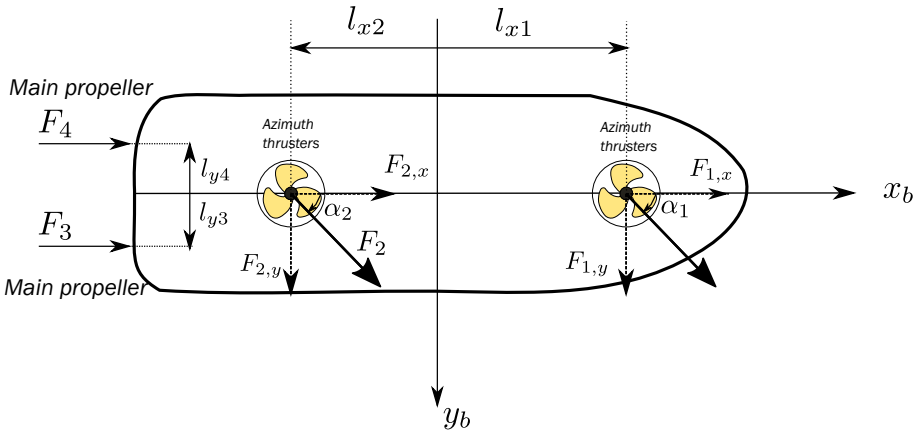


Figure 2.5: Actuator Location and Forces

Source: Adapted from [3]

$$\begin{bmatrix} X \\ Y \\ N \end{bmatrix} = \begin{bmatrix} cos(\alpha_1) & cos(\alpha_2) & 1 & 1 \\ sin(\alpha_1) & sin(\alpha_2) & 0 & 0 \\ l_{x1}sin(\alpha_1) & l_{x2}sin(\alpha_1) & -l_{y3} & -l_{y4} \end{bmatrix} . \begin{bmatrix} K_1 & 0 & 0 & 0 \\ 0 & K_2 & 0 & 0 \\ 0 & 0 & K_3 & 0 \\ 0 & 0 & 0 & K_4 \end{bmatrix} . \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{2.6}$$

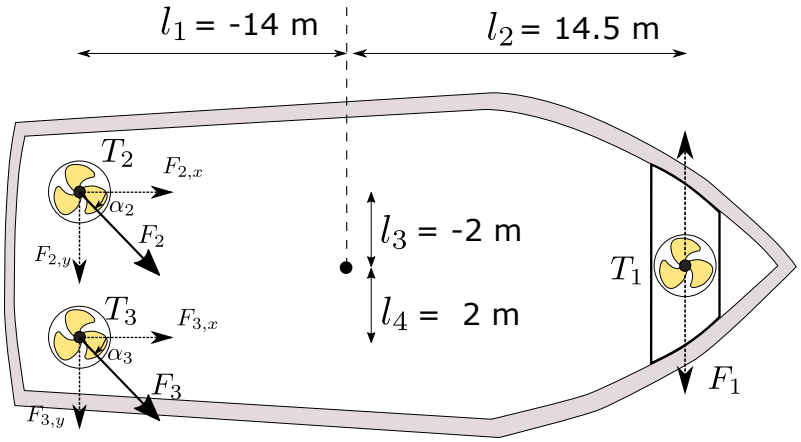


Figure 2.6: Thruster layout of vessel used in the thesis

In the thesis, the thruster arrangement of R/V Gunnerus can be visualised as in Figure 2.6. Its

corresponding thrust configuration matrix is given by equation 2.7.

$$T(\alpha) = \begin{bmatrix} 0 & cos(\alpha_2) & cos(\alpha_3) \\ 1 & sin(\alpha_2) & sin(\alpha_3) \\ l_2 & l_1 sin(\alpha_2) - l_3 cos(\alpha_2) & l_1 sin(\alpha_3) - l_4 cos(\alpha_3) \end{bmatrix} \tag{2.7}$$

An alternate way of writing the thrust configuration matrix is through extended-thrust configuration. Its details can found in [3] and no further discussion is provided as this thesis use the thrust configuration matrix discussed above.

2.2.1.3 Low-level Actuator Control

Low-level actuator control refers to a controller components that controls how the distributed control signal reaching the actuator from the thrust allocation method is used to produce motor commands that meets the request(force or torque)

For example, a low level controller could be simply a PID control that outputs a control signal(electrical current) based on reference signal to produce forces and moments.

2.3 Numerical Optimisation Scheme

From the section 2.2.1.2, it can be understood that thrust allocation refers to finding the control input vector u such that it produces the forces and moment τ as commanded by the high level controller.

This section introduces an advanced constrained allocation routine which is solved through used of Sequential Quadratic Programming(SQP) technique.

2.3.1 Constrained control allocation

In this section, a method of constrained control allocation including the use of Azimuth thruster is discussed. In real world scenario, it is important to put constraints on force saturation values, rate of change of forces and angles and feature to reduce power consumed. This is important to prevent situation of blackouts in vessel, reduce wear and tear of thrusters etc. This section has been taken from [3] and details can be found in detail in reference [3],[11].

When the rotatable thrusters are considered, control allocation becomes a non-convex optimisation problem. The primary constraint is

$$\tau = T(\alpha)f \tag{2.8}$$

where α denote the azimuth angles, $\alpha \epsilon \mathbb{R}^p$,T is the thrust configuration matrix and f is the control vector to be found out. In this scheme, the azimuth angles are computed together at each sample with control input being subjected to both amplitude and rate saturations. Constraint on azimuth angles are as follows:

$$\alpha_{i,min} \leq \alpha_i \leq \alpha_{i,max} \tag{2.9}$$

$$\dot{\alpha_{i,min}} \leq \dot{\alpha_i} \leq \dot{\alpha_{i,max}} \tag{2.10}$$

and $\dot{\alpha}$, the azimuth turning rate subject to a maximum value. For a control allocation scheme involving only non-rotating thrusters utilising Least-Square optimisation given in [3], the control force vector can be found out by using generalised inverse as given in equation 2.11.

$$f = \underbrace{W^{-1}T^T(TW^{-1}T^T)^{-1}}_{\mathrm{T_w}^\dagger}\tau \tag{2.11}$$

The equation 2.11 is modified to add azimuth angles as given in equation 2.12.

$$T_{\mathrm{w}}{}^\dagger = W^{-1}T(\alpha)^T(T(\alpha)W^{-1}T(\alpha)^T)^{-1} \tag{2.12}$$

The equation 2.12 can become singlular for some α. During singularity occurrence, several thrusters have equal azimuth angles and if sudden change of external force direction occurs, there is a possible drift because the azimuth thrusters cannot change angles quickly to compensate it [15].

In this case, the problem is locally approximated as a convex Quadratic problem as shown below [15],[3],[11].

$$\begin{aligned} J = \min_{\Delta f, \Delta\alpha, s} \{&(f_0 + \Delta f)^\intercal P(f_0 + \Delta f) + s^T Q s + \Delta\alpha^T \Omega \Delta\alpha + \\ &\frac{\partial}{\partial\alpha}\left(\frac{\rho}{\varepsilon + det(T(\alpha)W^{-1}T^T(\alpha))}\right)\Big|_{\alpha=\alpha_0} \Delta\alpha\} \end{aligned} \tag{2.13}$$

subject to:

$$s + T(\alpha_0)\Delta f + \frac{\partial}{\partial\alpha}(T(\alpha)f)\big|_{\alpha=\alpha_0, f=f_0}\Delta\alpha = \tau - T(\alpha_0)f_0 \tag{2.14}$$

$$f_{min} - f_0 \leq \Delta f \leq f_{max} - f_0 \tag{2.15}$$

$$\alpha_{min} - \alpha_0 \leq \Delta\alpha \leq \alpha_{max} - \alpha_0 \tag{2.16}$$

$$\Delta\alpha_{min} \leq \Delta\alpha \leq \Delta\alpha_{max} \tag{2.17}$$

the variable in above equations are as follows:

1. f_0 and α_0 are control force and azimuth angle from previous iteration.
2. Δf and $\Delta\alpha$ are change in control forces and azimuth angles
3. $s \,\epsilon\, \mathbb{R}^n$ are slack variables
4. $P \,\epsilon\, \mathbb{R}^{r\times r}$, $Q \,\epsilon\, \mathbb{R}^{n\times n}$, $\Omega \,\epsilon\, \mathbb{R}^{p\times p}$
5. $\rho > 0$ is a scalar weight influencing manoeuvrability and power consumption
6. $\varepsilon > 0$ is a small number to avoid division by zero

7. $W \epsilon \mathbb{R}^{r \times r}$ is a positive definite weighting matrix for the control force.

In order to obtain the optimisation problem, some simplifications are made from [11].They are:

1. Power proportional to $f^{3/2}$ is approximated as $f = f_0 + \Delta f$
2. The singularity avoidance term locally approximated by a linear term around the last azimuth angle, α_0 such that $\alpha = \alpha_0 + \Delta\alpha$

This formulation is used in the thesis with some modification for creating the benchmark SQP allocator.

Chapter 3

Machine Learning

In this chapter, an overview of Machine Learning(ML) and its associated building blocks are presented. The basics of the Autoencoder network that is used in the thesis are also presented. Deep Learning(DL) is a subset of ML and basic aspects remain the same for both.

3.1 Machine Learning

Machine learning(ML) is the study of computer algorithms that improve automatically through experience [25]. Machine Learning has grown from a subset of Artificial Intelligence(figure 3.1) and now encompasses mainly the three types of learning approach or algorithms as shown in figure 3.2.

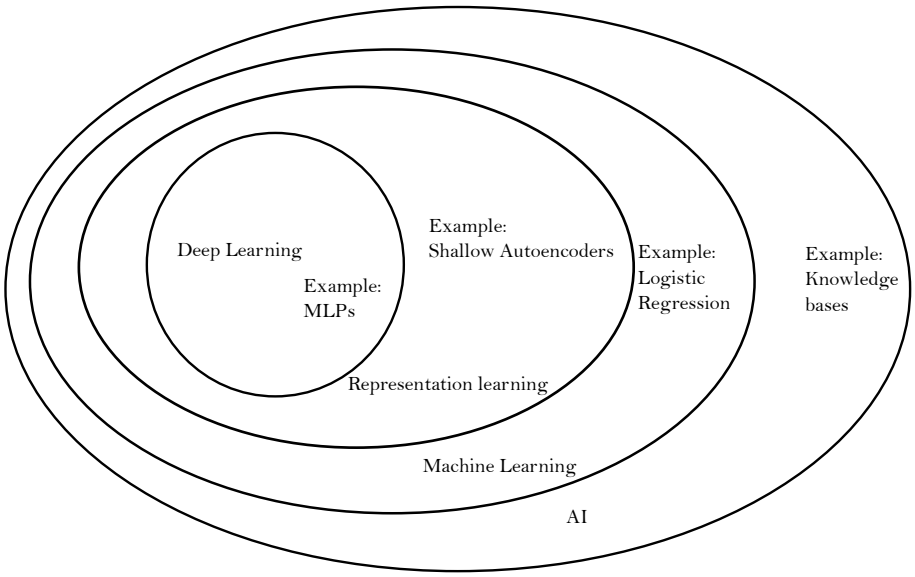


Figure 3.1: AI Venn Diagram

Source: Adapted from [26]

These algorithms are differentiated in terms of how the experience or data is provided to the machine.

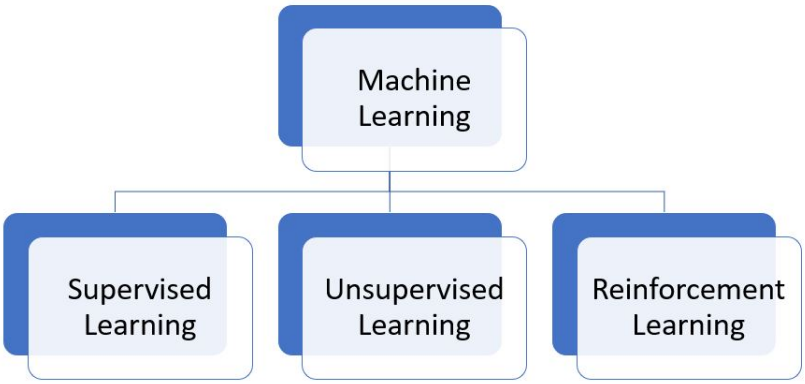


Figure 3.2: Machine Learning Hierarchy

3.2 Supervised Learning

In Supervised learning, the algorithm learns from a dataset that contains features as well as its associated label or target for each example. The aim of the algorithm is to learn a complex function that can map features to the target data. Using this learned function, it should be able to successfully predict the new target if features are provided to the function.

In the thesis, a Deep Autoencoder (DAE) network is used(Deep refers to multiple layers in the network) which can be considered as supervised learning itself because its training regime is similar to that of Supervised learning even though it does not need labeled inputs.

3.3 Machine Learning Building Block

An Artificial Neural Network(ANN) or simply Neural Network(NN) form the backbone of machine learning and its general structure is given in figure 3.3.

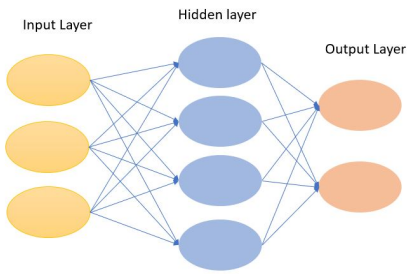


Figure 3.3: Neural Network Structure

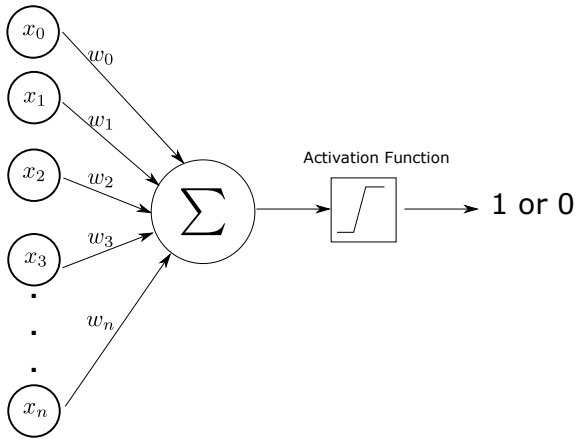


Figure 3.4: Perceptron Model

The simplest form of ANN is a Perceptron model introduced by Frank Rosenblatt [27]. It is shown in figure 3.4. A simple perceptron outputs a binary (0 or 1) when provided with weights and bias($w's$ are weights and w_0 is bias in image 3.4). The transformation in perceptron is given by equation 3.1.

$$f(x) = \begin{cases} 1, & \text{if } w^T x + b > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

A NN is termed deep or shallow based on the number of hidden layers it has. If the number of hidden layers is more than one, then the NN is termed as Deep Neural Network(DNN). There exists disagreement to this number, but in this thesis, a shallow neural network consists of only one hidden layer. In each hidden layer, the data is transformed by the activation functions and at the output layer, the transformed data is obtained. Each layer is composed of nodes or neurons. Each node transforms information and passes it on to the next layer after passing it through an activation function. Essentially, DL refers to an ML model with a large number of layers such that it can learn more representations from the data.

A network in which the connection between neurons in a layer do not make loop is called Feed Forward Neural Network(FFNN). If it makes a loop it is termed as Recurrent Neural Network(RNN) [28]. So in a FFNN, the neurons in a layer get information from only the layer before it and transfers information to the next layer.

3.3.1 Neural Network Notations

While developing Neural Networks or to say when working with ML/DL models, it is suitable to denote various terms by notations. Such notations are defined here.

- To denote different sizes we use the following notations:
  1. n_x = input size
  2. n_y = output size
  3. m = number of examples in the dataset
  4. $n_h{}^{[l]}$ = number of hidden units in the l^{th} layer

- To denote weights and bias, we use the following notations:
  1. $W^{[l]}$ = weight matrix of the l^{th} layer
  2. $b^{[l]}$ = bias vector of l^{th} layer
  3. ${w_{jk}}^{[l[}$ = weight from unit k of layer $l-1$ for the unit j of layer l
  4. ${b_j}^{[l]}$ = bias unit in layer l.
- To denote activation function and output of the neuron,the following notations are used:
  1. ${a_j}^{[l]}$ = activation of j^{th} neuron of layer l.
  2. $g^{[l]}$ = activation function of layer l.
- In NN, the number of layers are counted from the first hidden layer to the output layer.(ie,to count layers that have tunable parameters(weights and bias)).

There two basic step in Neural Network training is given below:

3.3.2 Forward Propagation

Forward Propagation or forward pass refers to the flow of information and thereby generation of intermediate values of each layer from the input layer to the output layer. During the forward pass, the information from the input layer is passed on to the first hidden layer. In the first hidden layer, the information undergoes transformation using weights and bias of that layer and then passes through an activation function to get the intermediate output of that layer. This transformation for a neuron in the layer is depicted in figure 3.5. This intermediate value is then used by the next layer and continues until the output layer.

In figure 3.5, for an arbitrary neuron j in layer l, its activated output depend on activation values from previous layer ${a_k}^{[l-1]}$ and the corresponding weights ${w_{jk}}^{[l[}$ connecting layer l and layer $l-1$. To this a bias unit is added and transformed through activation function to obtain ${a_j}^{[l]}$.

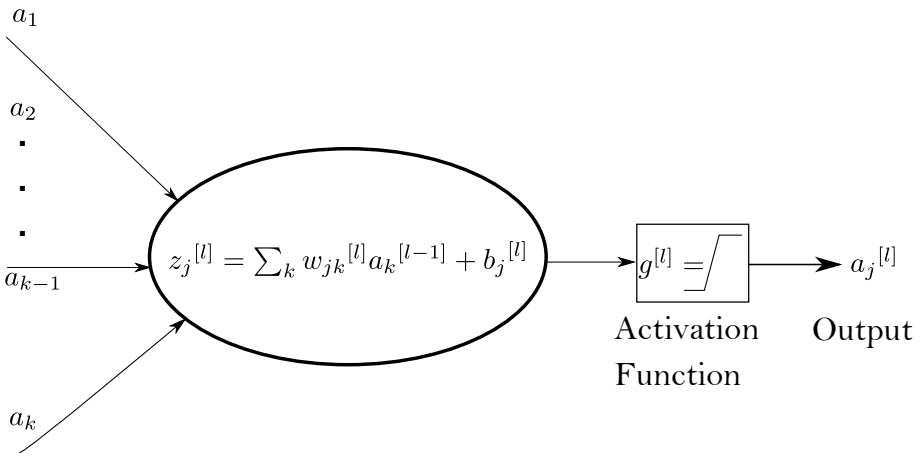


Figure 3.5: Forward Propagation Operations

To generalise the forward propagation operations, a weight matrix for layer l can be defined by stacking up the weights. w^l is a matrix containing weights that connect to the l^{th} layer of neurons.

Each entry in the weight matrix can be represented as ${w_{jk}}^{[l]}$ which is a entry in j^{th} row and k^{th} column of layer l. The bias can be stacked as a vector ${b_j}^l$. Thus the forward propagation can be written in matrix form as equation 3.2

$$z^l = w^l a^{l-1} + b^l \tag{3.2}$$

$$a^l = f(z^l) \tag{3.3}$$

where f in 3.3 denotes the activation function and z^l is termed as the weighted input in layer l [29].

Activation functions are special function that helps the neural network to learn non-linearity from the data. The aim of a NN is to learn a non-linear function that maps input to output. These functions also help keep the magnitude of the output values within a certain limit which is desirable to avoid computational issues. Some of the commonly used activation functions and its formula is provided below:

1. Sigmoid :
$$f(x) = \frac{1}{(1 + e^{-x})} \tag{3.4}$$
2. Tanh :
$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.5}$$
3. ReLU (Rectified Linear Unit) :
$$f(x) = \begin{cases} 0, & \text{if } x < 0. \\ x, & \text{otherwise.} \end{cases} \tag{3.6}$$

For more information on activation functions and their properties, the reader is encouraged to refer [30].

At the end of forward propagation, a collection of weights and bias are obtained from the network. The goal of the NN is to alter these weights and bias such that after computation through various transformations through the layers, the network is able to approximate a function $f(x)$ for its inputs. To quantify the ability of the NN in generalising this function, a cost function is introduced which measures the difference between predicted values $\hat{y}_j$ and the truth values y_j.

$$J(w, b) = \sqrt{\frac{1}{n} \sum_{j=1}^{n} (y_j - \hat{y}_j)^2} \tag{3.7}$$

In the equation 3.7, the cost function is the Root Mean Squared Error. Similarly other cost function can be defined. For example, Mean Absolute Error can be defined as Equation 3.8 and Mean Squared Error can be defined as Equation 3.9

$$MAE = \frac{1}{n} \sum_{j=1}^{n} |y_j - \hat{y}_j| \tag{3.8}$$

$$MSE = \frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2 \tag{3.9}$$

The cost function which is dependent on weights w and bias b can take on values starting from zero when all predictions and actual values match and increase to large values when the prediction is wrong. The aim is to reduce the cost as much as possible. The process to achieve this is explained next.

3.3.3 Backpropagation

It is an algorithm used for training feed-forward neural networks by use of the Gradient Descent method. The following contents have been derived from [29].

The aim of the backpropagation is to calculate partial derivatives of the cost function, $\frac{\partial J}{\partial w}$ and $\frac{\partial J}{\partial b}$, with respect to weights and bias of the NN. This is to understand how a small perturbation in values of weights and bias affects the cost function. Specifically for a neuron j in layer l, we compute $\frac{\partial J}{\partial w_{jk}{}^{l}}$ and $\frac{\partial J}{\partial b_j{}^{l}}$. This is done to introduce a concept where a small perturbation in weighted input of the layer l, $\Delta z_j{}^{l}$ can cause a change in the change in cost function by propagation of the change through the rest of the layers of the network. $\Delta z_j{}^{l}$ cause a change of $\frac{\partial J}{\partial z_j{}^{l}}\Delta z_j{}^{l}$ in the cost function of the network.

The term $\frac{\partial J}{\partial z_j{}^{l}}$ is defined as error $\delta_j{}^{l}$ of neuron j in layer l. Generalising this for a layer, δ^l is the vector of errors for layer l. Backpropagation will compute this error vector for each layer in the NN. For output layer($l = L$) this can be written as equation 3.10(for quadratic cost function)

$$\delta^L = \frac{\partial J}{\partial a_j{}^{L}} \odot f^{'}(z^L) = \nabla_a J \odot f^{'}(z^L) = (a^L - y) \odot f^{'}(z^L) \tag{3.10}$$

$\odot$ denote Hadamard product.

Now, a formulation to find error for layers previous to output layer L needs to be found out.

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot f^{'}(z^l) \tag{3.11}$$

Using equation 3.10 and 3.11, we can find error vector starting from output layer(L) and moving backwards.

To find the change in cost function with respect to change in bias, the following equation 3.12 which in turn is obtained from equation 3.11.

$$\frac{\partial J}{\partial b_j{}^{l}} = \delta_j{}^{l} \tag{3.12}$$

Similarly, the change in cost function with respect to weight can be written as equation 3.13

$$\frac{\partial J}{\partial w_{jk}{}^{l}} = a_k{}^{l-1}\delta_j{}^{l} \tag{3.13}$$

These four equations sums up the process of backpropagation. Now the weights and bias are

updated according to rules of gradient descent through equation

$$w = w - \alpha \frac{\partial J}{\partial w} \tag{3.14}$$

and

$$b = b - \alpha \frac{\partial J}{\partial b} \tag{3.15}$$

3.4 Variants of Gradient Descent Algorithms

In Machine Learning, mainly three types of Gradient Descent strategies are used.[31]They are :

1. **Batch Gradient Descent** : In Batch gradient descent, the gradient of the cost function for the different parameters of the networks(weights, bias) are computed for the entire training dataset. So a single update to the network parameters takes place only after the complete calculation of gradients for the whole dataset and then averaging the gradients. This can be quite slow and memory-consuming for the computer.
2. **Stochastic Gradient Descent(SGD)**: In this method, the parameter update of the network is performed after computing gradient for each sample in the training dataset. This results in faster updates accompanied by large variations in the gradient size.
3. **Mini-batch Gradient Descent**: In this method, parameter update is performed after computing gradients for a mini-batch (a batch size N which is less than the size of the whole training dataset). Advantages of Mini-batch Gradient Descent is that they have less variance issue of the parameters which is present in Stochastic Gradient Descent and at the same time faster than Batch Gradient Descent

3.5 Data scaling

In ML models, the data is not fed directly as it is to the network. If fed directly, the different scale of value may cause issue in computation such as gradient explosion. The two commonly used Data-scaling methods are given below:

1. **Min-max normalisation**: In this method, the data is scaled to range of [a,b]. Usually this range is [0,1] or [-1,1]
2. **Standardisation**: In this method, the data is scaled such that the mean of the sample is 0 and variance is 1. The scaling is done according to equation 3.16

$$X_{scaled} = \frac{x - \mu}{\sigma} \tag{3.16}$$

where μ is the average of the samples and σ is the standard deviation of the samples

3.6 Autoencoder

An autoencoder is a neural network that is trained to attempt to copy its input to its output.[26]. An autoencoder network has mainly three components: an **encoder**, a hidden layer **h** that describes the input representation named latent **code** and a **decoder**. The function of the encoder is to create a representation of the input in the code and the decoder's function is to reconstruct or map input from the code.

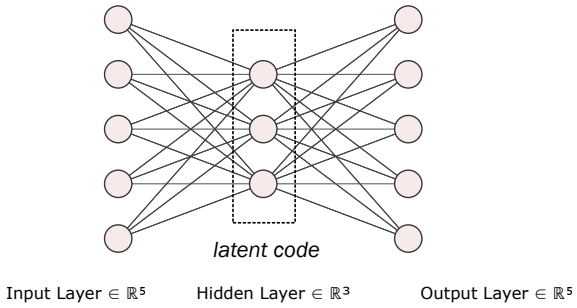


Figure 3.6: Basic Autoencoder Structure

The learning process of an autoencoder can be described simply as equtation 3.17:

$$L(x, g(f(x))) \tag{3.17}$$

where L is a loss function penalising g(f(x)) for being dissimilar from x [26].

Autoencoders can be classified broadly into two based on the dimension of the code. When the code dimension is smaller than the input dimension, the autoencoder network is termed as an **undercomplete autoencoder** as can be seen in figure 3.7. When the code dimension is higher than the input dimension, it is termed as **overcomplete autoencoder** as can be seen in figure 3.8.

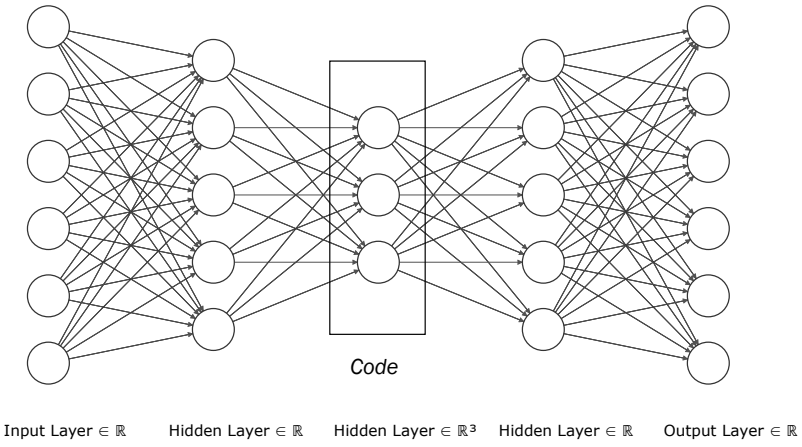


Figure 3.7: Undercomplete Autoencoder Structure

The autoencoders have been traditionally used for dimensionality reduction or feature learning[26]. The variants of autoencoders include:

1. **Sparse Autoencoders**: It is an autoencoder that has a sparsity penalty $\Omega(h)$ on the code

layer during training in addition to reconstruction error[26]. This is typically used to learn features for tasks such as classification.

2. **Denoising Autoencoders**: It is an autoencoder that tries to predict uncorrupted data points at the output when provided with corrupted input data points [26]. These kinds of the network have seen application in image filtering to remove noise from the image and produce original denoised images.

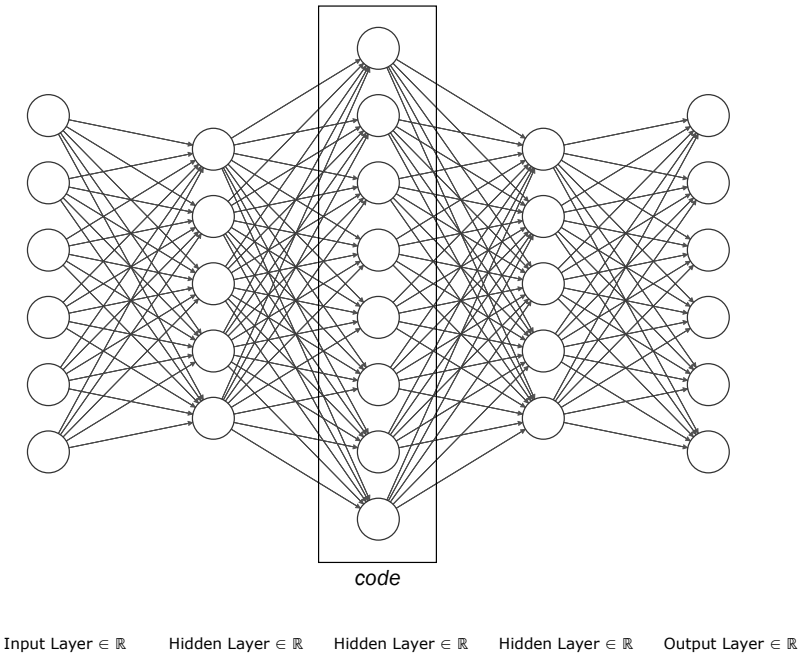


Figure 3.8: Overcomplete Autoencoder Structure

The layers of the Autoencoder could be simply fully connected dense layers or LSTM layers or Convolution layers[32]. In the use case for TA, the work in[21] had fully connected dense layers whereas, in [8], the layers are a combination of LSTM layers and fully connected layers. This thesis utilises a fully connected network architecture as can be seen in subsequent chapters for thrust allocation following the footsteps of the work in [21].

Chapter 4

Methodology

In this chapter, the approach followed for meeting the objectives of the thesis is explained followed by the details of the development of Deep learning TA and classic SQP TA.

Figure 4.1: Goal Breakup

In order to complete this thesis, it is envisioned to divide the work into 6 broad goals as seen in figure 4.1. These goals have been set up such that the learning outcome or outcome from each goal is required for the successful completion of the thesis.

These goals are subdivided into smaller sub-goals as explained in figure 4.2. The cyclic nature of goal breakup is to indicate multiple traverses during thesis work to obtain fair results.

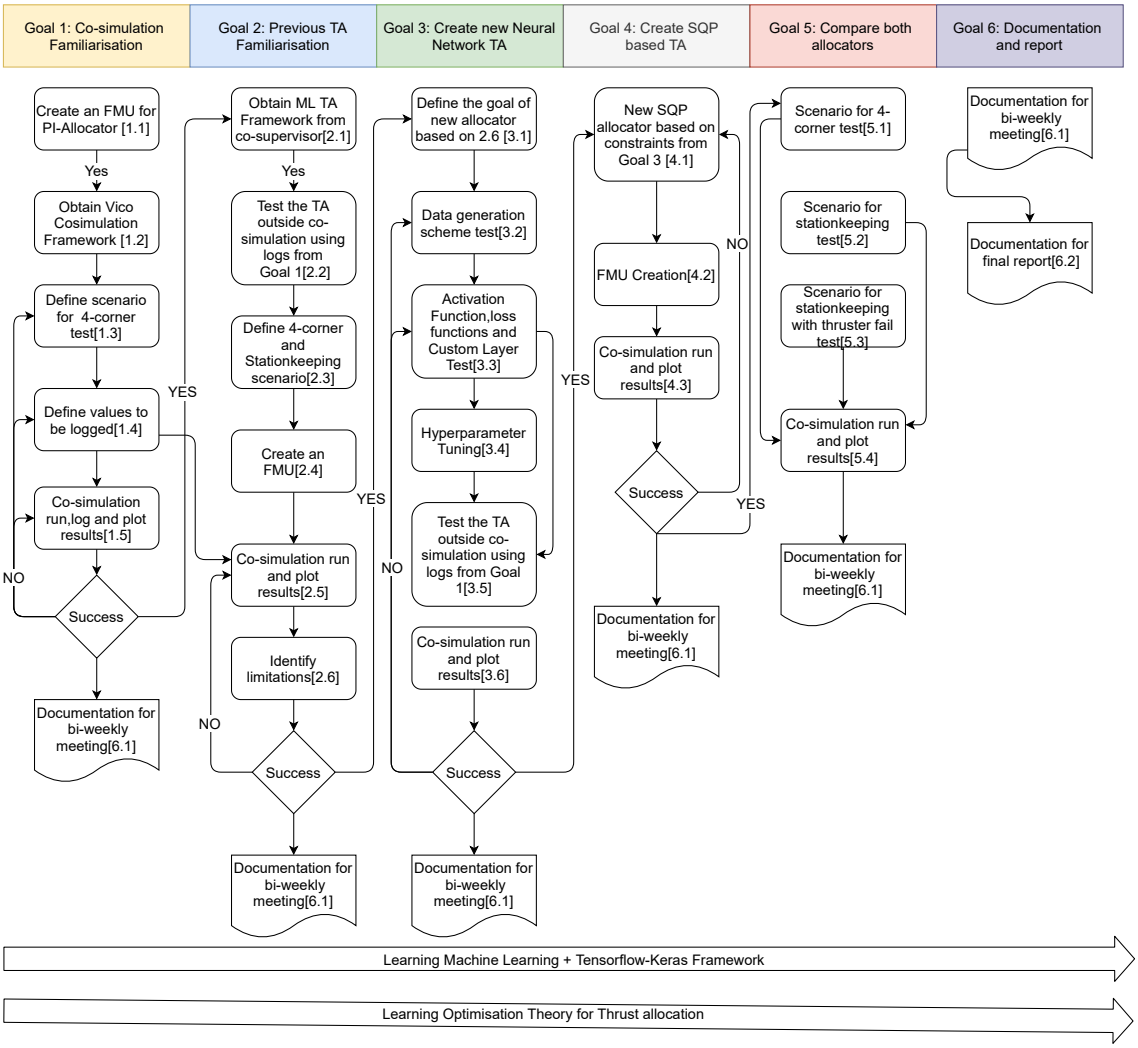


Figure 4.2: Methodology Flowchart

4.1 Methodology Flowchart explanation

1. **Goal 1** : In the methodology flowchart, the aim of goal 1 is to get familiarised with the co-simulation. In order to get familiarised with it, the first step done is to create a simple pseudo-inverse(PI) allocator for TA. This is done through the use of PythonFMU[33]. Then the framework Vico is accessed through [34] where basic steps are provided to set up co-simulation. A scenario for the 4 corner test is made as shown in Appendix A.1. The variables from FMU to be logged are specified in a separate log definition file(.xml file) and the simulation is run using Vico. An example of the shell script that runs the simulation is given in Appendix B. Defining FMU with the same variable name help to ease the process of generating the log definition file and the subsequent data plotting. The data is plotted by accessing the CSV log files at the end of the simulation. The values logged for desired forces and moments and allocated forces and moments are used for evaluating the performance of the DAE allocator outside the co-simulation environment in Goal 2.

2. **Goal 2** : This Master's thesis builds upon previous research work at NTNU [8]. The researchers had set up a framework to develop a DL allocator that makes the process of setting

up experiments easy. It is obtained from the co-supervisor in this goal. Using the framework, the TA allocator from previous work [8] is tested outside the co-simulation platform to understand the development procedure of the DL allocator. For this, the forces and moments from the PI allocator in Goal 1 are utilised. The outside co-simulation provides plots such as given in Appendix: C. The ML framework utilised in the thesis is Keras with Tensorflow backend.

After testing the allocator, a scenario for stationkeeping and 4-corner(Goal 1 scenario) is created for testing in co-simulation. The FMU for the allocator is made using PythonFMU and tests are done. The results are plotted and the limitations of the current thrust allocator from [8] is found out.

3. **Goal 3**: In this goal, the new DL allocator based on Deep Autoencoder is developed. Features of robust handling of force and angle magnitude and a hard rate constraint are proposed to be added in addition to power minimisation and forbidden zone handling.

   Subsequently, tests on artificial data generation are conducted with the existing framework to improve its force range and to test its performance. Data generation schemes to be tested include pseudo-random number generation and random walk generation strategy.

   Upon satisfactory data generation, the type of layers in the network is to be explored from normal dense layers in [21] to LSTM layer combination in [8]. In addition, the use of different activation functions and custom layers is tested to meet the constraints.

   If the network performs satisfactorily outside the co-simulation, a manual hyperparameter tuning is to be conducted to optimise the performance as in any ML model development. Key indicators such as MSE Loss, Total allocation Error, and Total mean forces are taken as metrics for the network performance. Each model is also run in co-simulation to test for performance. If this is satisfactory, the major goal of the thesis is achieved and now a benchmark allocator is created in the next goal.

4. **Goal 4**: Upon creation of a satisfactory DAE network, the benchmark classic allocator based on SQP is created using reference [15],[11],[3]. For this allocator, the constraints are features are exactly the same as the newly developed DAE allocator namely: Power minimisation, force and angle magnitude saturation handling, rate handling of forces and angles, and forbidden zone management.

   The SQP allocator FMU is created and put to test in the co-simulation environment.

5. **Goal 5**: In the thesis, the newly developed DAE allocator is tested in three scenarios- a standard 4 corner test, stationkeeping test, and a stationkeeping test with thruster failure scenario. These are tests that can identify the performance and drawbacks of the developed DAE allocator. In each case, the benchmark performance is the classic SQP allocator developed. The three scenarios are developed as seen in Appendix : A.1,A.2,A.3.

   The co-simulation is conducted and the results are to be plotted.

6. **Goal 6**: This is the final goal of the thesis where the process of development of allocator and various results are properly documented and submitted. During the thesis period, bi-weekly meetings are scheduled to present the findings and problems for discussion. For this, in each goal of the thesis, relevant sections are to be documented.

   Finally, at the end of the thesis, all documentation is presented in the form of a Master thesis report.

For achieving the objectives of the thesis, the major learning required is in the form of learning Machine learning and the framework to be used. This is a continuous process and starts from the beginning and continues till the end of the thesis. Theory about optimisation for classic allocator is also learned from the beginning and the process is finished at the end of the thesis.

4.2 Deep Learning based allocator development

In this section, the development of a Deep learning allocator using a Deep Autoencoder network with its details is provided. The development of the DAE allocator follows a development framework supplied by the co-supervisor based on his previous work on the subject [8] as mentioned before.

In the thesis, the allocator is developed for NTNU's R/V Gunnerus.(Figure 4.3)

Figure 4.3: NTNU's R/V Gunnerus

Source: Fredrik Skoglund

The parameters for the vessel is given in Table 4.1

Parameter	**Value**
Displacement(tonne)	580
LPP(m)	33.9
Beam(m)	9.6
Draught(m)	2.8
$\Delta T_1, \Delta T_2, \Delta T_3$(N/s)	1000
$\Delta T_2, \Delta T_3$(°/s)	10
T_1 max (N)	±30000
T_2/T_3 max(N)	-30000,60000
T_2/T_3 max (°)	-180,180

Table 4.1: R/V Gunnerus Specification

In the allocator developed, it is assumed that the thrusters can produce forces in both forward and backward direction.

4.2.1 Data generation

The first step in any ML project is to obtain data that can be trained to find the mapping between data and target. Since in the thesis, a DAE network is used, a method to create forces from commands of thrusters is used. For demonstration purpose the maximum force range and angle range of thrusters are restricted to the following limit:

$$\begin{aligned} u_0 &= [-8000, 8000]\ N \\ u_1 &= [-8000, 8000]\ N \\ u_2 &= [-80, 80]\ ^\circ \\ u_3 &= [-8000, 8000]\ N \\ u_4 &= [-80, 80]\ ^\circ \end{aligned} \tag{4.1}$$

In equation 4.1, u_0 corresponds to Tunnel thruster force, u_1 corresponds to Azimuth thruster 1 force, u_2 corresponds to Azimuth thruster 1 angle values, u_3 corresponds to Azimuth thruster 2 force, u_4 corresponds to Azimuth thruster 2 angle values. The angle values are limited to [-80,80]°to employ forbidden zone management considering forces can be created in both positive and negative direction. The forbidden zone management will be discussed later.

1. For each thruster, a range of pseudo-random commands are generated from a lower limit to an upper limit. As an example, for a tunnel thruster, a range of control commands from the lower limit of -10000 N and upper limit of 10000 N are generated using pseudo-random numbers. For azimuth thruster, forces from a lower limit of -10000 N to 10000 N and angles from a lower limit of -80°to an upper limit of 80°are produced. The number of samples generated is set by the user considering the training and allocation performance of the network. As a guideline to data generation, the upper and lower limit of samples generated should be above the constrained values. That is for 8000 N, a force range of 8000 N or above is to be generated. Here it is taken as 10000 N. To generate these commands, Python library *Numpy* and its function *randint* is used.

2. Once these commands for individual commands are generated, they are transformed to an individual vector of 5 commands that can be used to generate forces in Surge, Sway, and Yaw direction. The principle used to obtain the force in 3 DOF is to transform commands using Thruster configuration matrix according to formula

$$\tau = T(\alpha) \times u \tag{4.2}$$

3. The forces in 3 DOF are the input to the DL network for training. Before the forces are fed into the network, they are standardised using the theory in section 3.5. The data generation method can be visualised as shown in figure 4.4

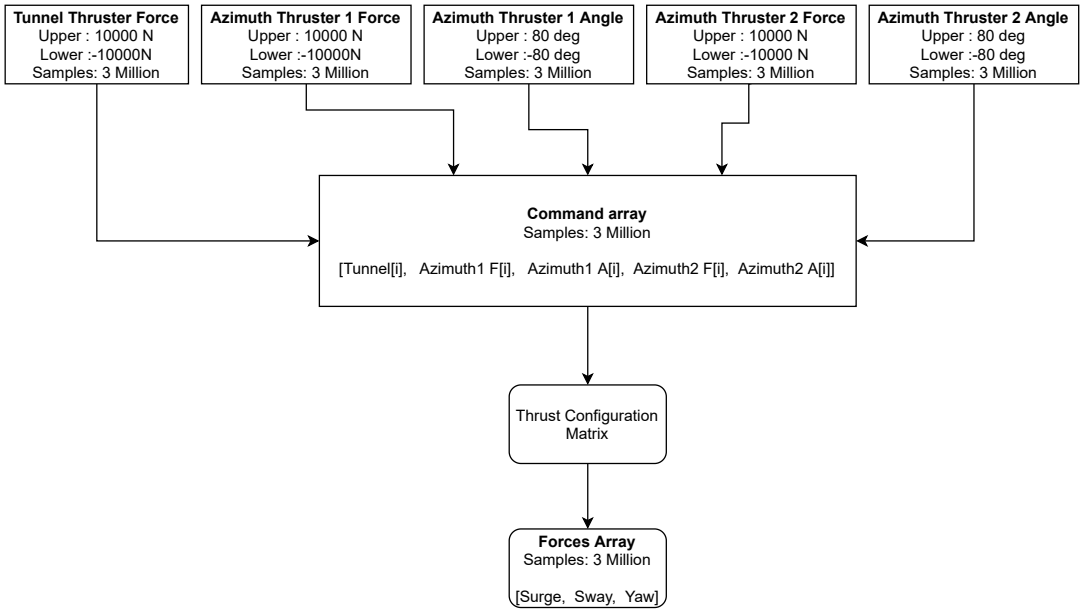


Figure 4.4: Data Generation Step

4.2.2 Network Architecture

After data generation, the neural network structure is to be investigated. The investigation is followed based on the literature [21].

Through experimentation, it is found that a Dense layer-based Deep Autoencoder(DAE) can reproduce force provided at the encoder input at the output of the decoder. The latent code in the DAE is where the control commands will be generated and taken for allocation.

For the network to learn and accommodate constraints in power, force, and angle magnitude and constraint in the rate of change of forces and angles, a few strategies are adopted from the works [21] and [8].

4.2.3 Learning Strategy

Similar to normal neural network learning features, the DAE network also learns by minimising the losses. The strategy is to formulate different loss functions that can be minimised to learn that functionality. Thus the network learning is simply a summation of different loss functions that is minimised during training. The following losses are defined for the network:

1. Loss Function for generating force at the output(L_{in-out}):
    - The function of this loss term is to ensure that the 3-DOF force and moment given to the input of the encoder is reproduced at the output of the decoder. (ie, the force is reproduced with minimal loss).
    - To achieve this, the loss function is formulated as equation 4.3 - a Mean-square error

between input forces and output forces

$$L_{in-out} = \sum_{j=1}^{3} (y_j^i - \hat{y}_j^i)^2 \tag{4.3}$$

- In equation 4.3, j represents the forces in Surge and Sway and Moment in Yaw. i represents the number of samples during training or a single sample during allocation. $\hat{y}_j^i$ is the predicted forces and y_j^i is the input forces.

2. Loss function for generating commands that produce correct force given at the input (L_{latent}):

   - The function of this loss term is to ensure that the forces given as input to the encoder can be reproduced through the commands produced at the latent code.
   - In order to establish this loss, first the commands obtained at the latent code is rescaled back to original command scale. Then this commands are used to obtain forces and moment in 3-DOF using thrust configuration matrix($\tau = T(\alpha_{rescaled}) \times u_{rescale}$). Here $\alpha_{rescaled}$ is the two azimuth angles and $u_{rescale}$ is the three forces obtained at the latent code.
   - Then these calculated forces and moment are scaled down again to input scale and a Mean absolute error between input forces are calculated as in equation 4.4.

$$L_{latent} = \frac{1}{n} \sum_{j=1}^{3} |f_j^i - f_{latent,j}^i| \tag{4.4}$$

   - In equation 4.4, f_j^i is the input forces and moments, whereas $f_{latent,j}^i$ is the forces and moments produced by the latent control commands.

3. Loss to reduce power(L_{power})

   - In this custom loss function, the control commands obtained at the latent code are rescaled back to their original form and an exponential term is used to penalise its magnitude increase. An exponential value of 1.5 is taken considering the fact the power from thruster can be estimated as $u^{3/2}$ [35]. The power loss term is given in equation 4.5.

$$L_{power} = |u_0^i|^{3/2} + |u_1^i|^{3/2} + |u_3^i|^{3/2} \tag{4.5}$$

4. Loss to reduce rate changes (L_{rate})

   - In the control allocator, the maximum rate changes per Hertz prescribed for different commands are given in equation 4.6

$$[\Delta u_0, \Delta u_1, \Delta u_2, \Delta u_3, \Delta u_4] = [\pm 50N, \pm 50N, \pm 0.5^\circ, \pm 50N, \pm 0.5^\circ] \tag{4.6}$$

   This is prescribed in such a way because the simulation runs at 20 Hz and the rates when multiplied by 20 gives the proper rate change per second as given in equation 4.7.

$$[\Delta u_0, \Delta u_1, \Delta u_2, \Delta u_3, \Delta u_4] = [\pm 1000N, \pm 1000N, \pm 10^\circ, \pm 1000N, \pm 10^\circ] \tag{4.7}$$

   - To limit the rate change, the commands obtained at the latent code are rescaled and then the command vector is shifted one step in time to compare the effect of magnitude

change between time steps. The time shift implies moving the command vector in a mini-batch by one index backward.

- The absolute difference in magnitude of commands between two time steps is found out and compared to the prescribed rate limit. If the difference exceeds the limit, a penalty is imposed on the magnitude above the change. An exponential penalty is used such that more rate changes are penalised severely compared to a slight crossing of limits.

$$L_{rate} = \sum_{l=0}^{4} max(|\hat{u_l}^i - shift(\hat{u_l}^i)| - \Delta u_{l,max}, 0)^{1.02} \tag{4.8}$$

- The exponent 1.02 was found by trial and error during training and allocation test.

5. Total Loss(L_{total})

   - As the neural network is interested only in the total loss during training, the sum of four losses discussed above are taken.
   - The sum is taken such that each of the loss is prescaled.

$$L_{total} = (k1 * Lin - out) + (k2 * L_{latent}) + (k3 * L_{power}) + (k4 * L_{rate}) \tag{4.9}$$

   - Based on prescaling of these individual loses, the performance of the allocator can be tuned. For example, a larger positive scaling for power loss try to reduce the power consumption by the allocator. The use of loss function to train the model was adopted from work in [8].

6. Hard constraint on magnitude saturation and rate changes

   - In the four learning strategies discussed above, all were soft constraints imposed on the network. They could be exceeded based on the input forces values and their dynamic change. In order to avoid this problem, hard constraints are imposed such that the prescribed value will never be crossed even if the soft constraints are broken. This idea was inspired by the work in [21].
   - In the work [21], the magnitude constraints were imposed on the allocator using custom activation function in the layer before the latent code in the encoder layer. Here also a similar strategy is used in a slightly different way. This is considered a novel approach for this use case.
   - One of the new contributions of the thesis is the implementation of hard constraints on rate change. This was considered as the future work in [21]. The implementation of hard constraints follows a method of creating a custom layer that performs these functions of robust saturation (a point discussed above) and rate handling.
   - A custom layer is created with each node in the layer dedicated to 5 control commands. The values from these 5 nodes are concatenated and passed to the latent code and subsequently to the decoder. This can be visualised in figure 4.5

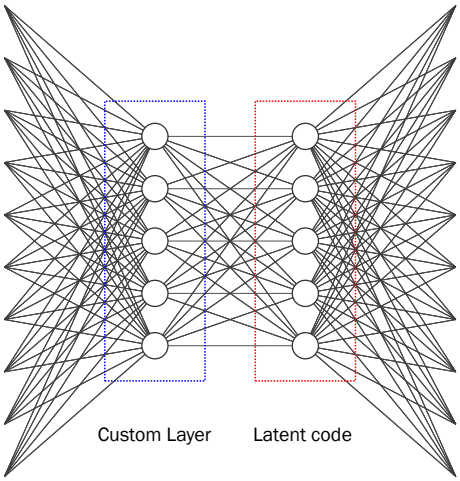


Figure 4.5: Custom layer visualised

- The custom layer is a normal dense layer with a linear activation function. The activated values are clipped for saturation during training(hard constraint) and the soft constraint for rate loss is imposed during training. During allocation, however, the hard constraint for rate is active through a simple technique of using a condition in the layer that gets evaluated based on the size of the input.
- To explain, this can be thought as during training, the network is fed in mini-batches and the size of input corresponds to mini-batch size(> 1). In the thesis,mini-batch size for training is taken as 1024 following [8]. But during allocation, at each time step, only one force command vector is fed into the network. During this time, the commands can be compared to the previous allocation using a memory feature in the layer(the previous value stored in the variable) and compared to the current value to clip against maximum or minimum increase in rate (hard constraint).
- The transformation during training when the input size is 1024 can be viewed as

$$Modif = (w * inputs) + b \tag{4.10}$$

where w and b are trainable parameters of the custom layer. First the input undergoes a linear activation. Then they undergo a generalised form of hard tanh transformation as shown in equation 4.11.

$$Modif = max(\underline{\delta}, min(\overline{\delta}, modif)) \tag{4.11}$$

The modification in equation 4.11 control the magnitude and angle constraint of the allocator.($\underline{\delta}$ and $\overline{\delta}$ indicates the allowed lower and upper magnitude of 5 control commands). After this the values are passed on to the latent code. Here soft-constraint on rate is applicable through custom loss function mentioned previously.

- During allocation(input size =1), the difference in transformation is that the output from one time step is saved in a variable

$$Previous = Modif \tag{4.12}$$

Then at the next time instant, maximum allowable rate values are added to this previous value and compared against the current input. They undergo a transformation similar

to generalised hard-tanh function. While adding the allowable rate to the previous value, the previous value is rescaled back to the original force range to avoid the issue of scaling issue using standardisation. Then the rate is added and then the values are standardised again for comparison against the incoming input values.

- This can be explained as: Let x be the Previous value.Then x is transformed according to equation 4.13.

$$x \Longrightarrow x' = (x * \sigma + \mu) \Longrightarrow x'' = x' + rate \Longrightarrow x''' = \frac{x'' - \mu}{\sigma} \tag{4.13}$$

In equation 4.13, σ and μ corresponds to the mean and standard deviation of the command vector generated for individual thrusters.ie, there are 5 mean values and 5 standard deviations corresponding to the control vector obtained during the data generation process. It is the x''' that is compared against the incoming new input and undergoes a generalised hard -tanh transformation.

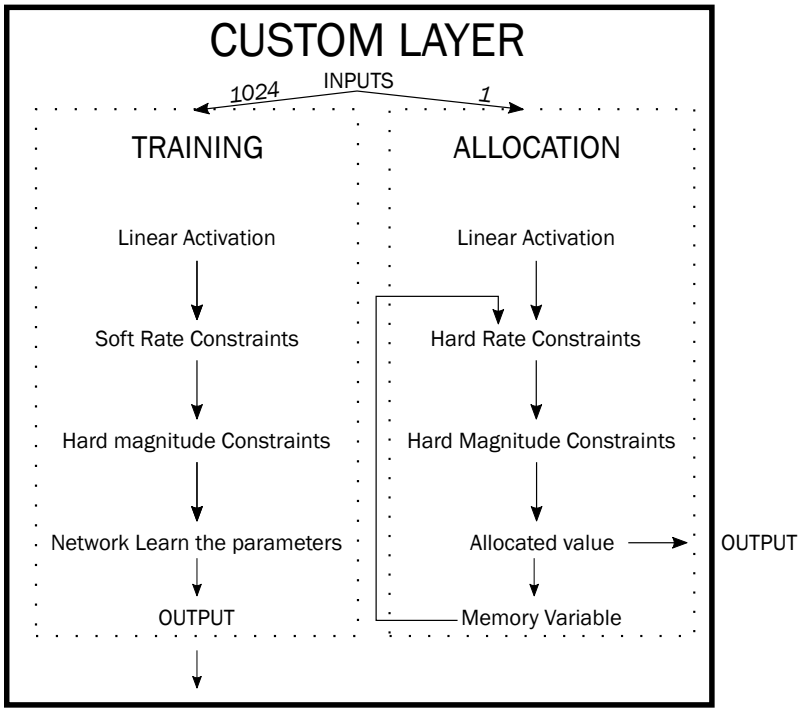


Figure 4.6: Custom Allocation Layer

- The angles are clipped at $-80°$ and $80°$ to create a forbidden zone (cut out portion of the circle) similar to one shown in figure 4.7. This can be obtained by considering the fact that forces can be produced in both positive and negative directions.

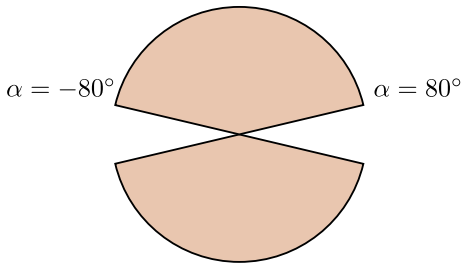


Figure 4.7: Forbidden sectors

4.2.4 Network Hyperparameters

During the proposal of any Neural Network architecture, its hyperparameters are to be chosen such that the network has an optimal performance for the task at hand. In reality, achieving optimal performance is limited to the number of tests done during choosing hyperparameters. During the hyperparameter search, there will be some combinations that may have more optimal performance than the proposed one. In this thesis, hyperparameters were chosen manually with some parameters fixed to limit the search space of the hyperparameters.

The hyperparameters to be chosen are given below:

1. Number of layers in encoder and decoder
2. Number of nodes in each of encoder layers
3. Number of nodes in each of the decoder layers
4. Activation function in the layers
5. Learning rate of optimiser(Adam[36] is used)

Since the thesis follows [21], the type of layer is taken as fully connected dense layers. Here few tests using LSTM layers have been tried following the work[8]. But due to the complexity of LSTM layers, the allocation time was high during the test in the simulator. Also, the dense layer could exhibit similar performance with faster allocation. So dense layer was chosen as the type of layer.

The optimiser was fixed as using Adam optimiser with a learning rate of 0.001. The activation function in each of the Encoder and Decoder layers was chosen to be Tanh activation function. This was chosen after a test with activation functions such as ReLU and Sigmoid. A test for ReLU activation provided inferior results compared to Tanh activation which may be explained by Tanh's symmetric nature of the function.

Now the manual search is done for finding the number of layers in the Encoder and Decoder of the network. The training is done for 20 epochs for each configuration. In the table 4.2, each test case is denoted by serial number and the corresponding number of the layer in encoder and decoder. For example, for a test case in which E:64,64;D:32,32 is written, it implies it has two hidden layers in the encoder side with 64 dense nodes in each. Similarly, the network has two layers on the decoder side with 32 nodes in each layer. The encoder is denoted by "E" and the decoder is denoted by "D". The last layer in the Encoder layer always has 5 nodes which is the custom layer designed as part of the thesis.

Sl.No	*Test Case*	*MSE*	*Total Allocation Error*	*Total Mean Force(N)*
1	[E:64,64,64,64,32,5];[D:32,32,32,10]	0.0101	778	403
2	[E:64,64,64,64,5];[D:32,32,32,10]	0.0088	2192	448
3	[E:64,64,64,32,32,5];[D:32,32,32,10]	0.0094	1539	437
4	[E:32,32,32,32,32,5];[D:32,32,32]	0.0119	1308	453
5	[E:16,16,16,16,16,5];[D:16,16,16]	0.0256	949	384
6	[E:24,24,24,24,24,5];[D:24,24,24]	0.0140	811	472
7	[E:24,24,24,24,5];[D:24,24]	0.0136	1896	520
8	[E:24,24,24,5];[D:24,24]	0.0148	817	410
9	[E:24,24,24,24,5];[D:24,24,24,24]	0.0195	685	906
10	[E:64,64,64,64,5];[D:64,64]	0.0074	643	435

Table 4.2: Hyperparameter search for number of layers and nodes

In Table 4.2, columns MSE, Total Allocation Error, and Total forces(N) represents three metrics used to select the layer configuration for the network. The layer is selected such that the selected candidate would have minimum values in each of the three metrics in the right combination.

Column MSE represents the mean square error between the input forces and the forces produced by the command vector in the latent code. If the value is minimum, it is considered that the network is able to produce the input forces accurately using the produced command vectors.

After training is done, a simple test case is made by setting up a scenario where the allocator is tested outside the co-simulation environment. For that, first, a four-corner test is conducted using a Psuedo-Inverse allocator in the simulation environment and the requested forces are recorded as a CSV file. Using the CSV file, the requested force at each time step is given as input to the DAE allocator and forces generated using the latent command vector is compared against the input. The absolute difference between the input forces and generated forces and Moments in each DOF-Surge, Sway and Yaw is found out. Then the mean of the error is taken and this value is given in the column "Total Allocator Error". The optimal allocator should have minimal mean allocation error.

Finally in column "Total Forces(N)", the mean of the sum of the absolute magnitudes of forces produced by the tunnel and two azimuth thrusters are found out for the requested forces. The optimal allocator would have minimal forces which mean minimal power for requested forces.

From the table 4.2, test case:10 has an MSE value of 0.0074(figure 4.9), Total Allocation error of 643 and Total mean forces of 435 N. This case has an optimal value when considering each of the three metrics. So it is taken as the DAE allocator layer structure.

The proposed network architecture is given in figure 4.8. After the Allocator layer was fixed, the DAE allocator was put into test in the co-simulation environment to check for performance. During the run, the time for allocation was around 3.5 ms per execution. This means that the allocator can be run in real-time for use cases in real-world scenarios.

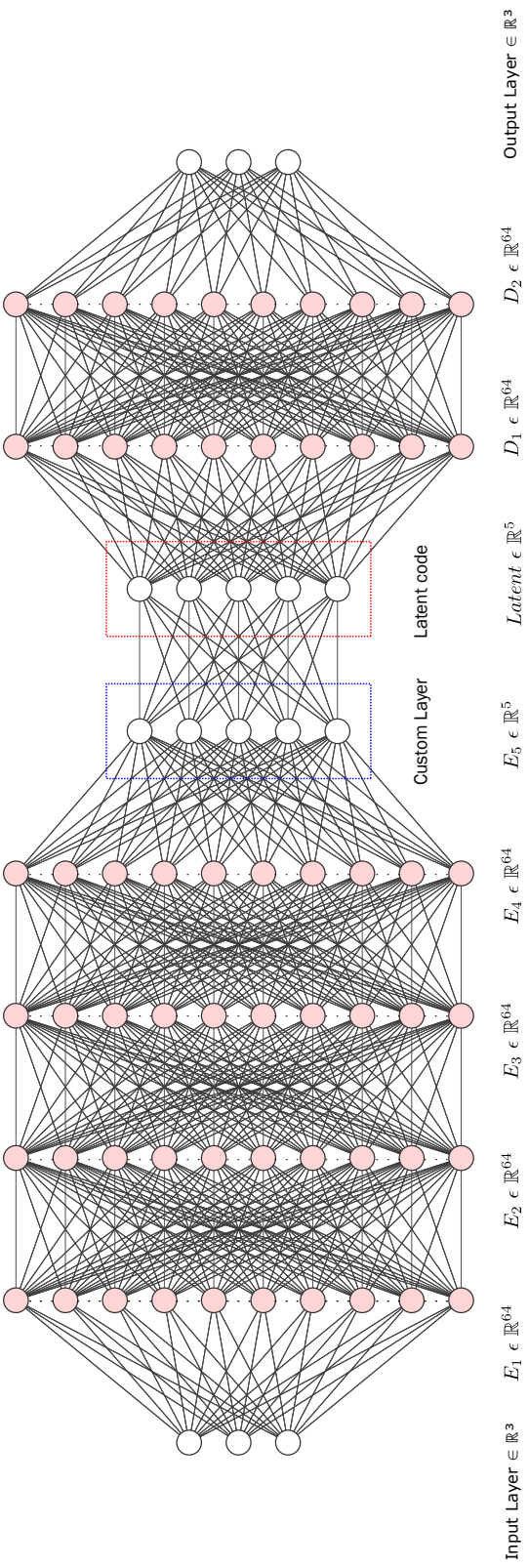


Figure 4.8: Proposed Architecture

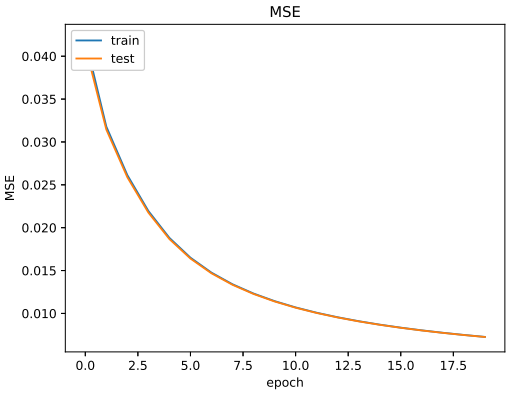


Figure 4.9: MSE Loss for Test Case 10

4.3 Classic Thrust Allocator

In this thesis, an optimisation scheme based on Sequential Quadratic programming[11] is used to create benchmark thrust allocator. Its details have been discussed before in Literature review. Here its actual implementation properties are discussed following works of [15],[3].

The control allocation problem is locally approximated as a convex Quadratic problem(QP) as shown below [15],[3],[11].

$$J = \min_{\Delta f, \Delta\alpha, s} \{(f_0 + \Delta f)^T P (f_0 + \Delta f) + s^T Q s + \Delta\alpha^T \Omega \Delta\alpha\} \tag{4.14}$$

subject to:

$$s + T(\alpha_0)\Delta f + \frac{\partial}{\partial\alpha}(T(\alpha)f)\big|_{\alpha=\alpha_0, f=f_0} \Delta\alpha = \tau - T(\alpha_0) f_0 \tag{4.15}$$

$$f_{min} - f_0 \leq \Delta f \leq f_{max} - f_0 \tag{4.16}$$

$$\alpha_{min} - \alpha_0 \leq \Delta\alpha \leq \alpha_{max} - \alpha_0 \tag{4.17}$$

$$\Delta\alpha_{min} \leq \Delta\alpha \leq \Delta\alpha_{max} \tag{4.18}$$

$$\Delta f_{min} \leq \Delta f \leq \Delta f_{max} \tag{4.19}$$

the variable in above equations are as follows:

1. f_0 and α_0 are forces and azimuth angle from previous iteration respectively.
2. Δf and $\Delta\alpha$ are change in forces and azimuth angles respectively.

3. $s \epsilon \mathbb{R}^n$ are slack variables

4. $P \epsilon \mathbb{R}^{r \times r}$, $Q \epsilon \mathbb{R}^{n \times n}$, $\Omega \epsilon \mathbb{R}^{p \times p}$

5. r = Number of actuators, p = number of azimuth thrusters, n = number of DOF (3 in our case)

6. T = Thrust configuration matrix

7. Power proportional to $f^{3/2}$ in original formulation is approximated to $f = f_0 + \Delta f$ [3].

4.3.1 Solving Using Quadratic Program Solvers in Python

The control allocation equations 4.14- 4.19 are written as an optimisation problem in matrix form [15] as shown below in equations 4.20-4.23. Then the problem is solved using open source QP solvers available in Python. In this thesis, the solver used is the *"quadprog"* package[37] . Other QP solver in Python named *CVXOPT* [38] was also used. But the execution speed of *quadprog* was better than *CVXOPT* and thus was chosen as the solver in the thesis.

$$\begin{aligned} &\min_x \{\frac{1}{2} x^T H x + g^T x\} \\ &subject\ to: \end{aligned} \tag{4.20}$$

$$Ax \ \leq \ b \tag{4.21}$$

$$A_{eq} x = b_{eq} \tag{4.22}$$

$$lb \ \leq \ x \leq \ ub \tag{4.23}$$

The associated matrices to be used in the codes is shown below. Through the formulation, we try to minimise power, avoid forbidden zone, consider rate constraints and handle both force and angle saturation. The optimisation vector $\boldsymbol{x}$ is given as :

$$x = \begin{bmatrix} \Delta f_1 \\ \Delta f_2 \\ \Delta f_3 \\ \Delta \alpha_1 \\ \Delta \alpha_2 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}_{8 \times 1} \tag{4.24}$$

where Δf_1 correspond to change in tunnel thruster force,Δf_2 correspond to change in azimuth1 force , Δf_3 correspond to azimuth 2 force, $\Delta \alpha_1$ correspond to change in azimuth 1 angle, $\Delta \alpha_2$

correspond to change in azimuth 2 angle. s_1, s_2, s_3 are change in slack variables.

$$P = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix}_{3\times3} \tag{4.25}$$

$$\Omega = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix}_{2\times2} \tag{4.26}$$

$$Q = \begin{bmatrix} 500 & 0 & 0 \\ 0 & 1000 & 0 \\ 0 & 0 & 1000 \end{bmatrix}_{3\times3} \tag{4.27}$$

$$H = \begin{bmatrix} 2P_{3\times3} & 0_{3\times2} & 0_{3\times3} \\ 0_{2\times3} & 2\Omega_{2\times2} & 0_{2\times3} \\ 0_{3\times3} & 0_{3\times2} & 2\mathrm{Q}_{3\times3} \end{bmatrix}_{8\times8} \tag{4.28}$$

$$g = \begin{bmatrix} 2 \times P \times f_0 \\ 0_{2\times1} \\ 0_{3\times1} \end{bmatrix}_{8\times1} \tag{4.29}$$

The first term in vector 4.29 come from expanding the terms in problem formulation(Equation 4.14).

$$A = \begin{bmatrix} \mathbb{1}_{3\times3} & 0_{3\times2} & 0_{3\times3} \\ -\mathbb{1}_{3\times3} & 0_{3\times2} & 0_{3\times3} \\ 0_{2\times3} & \mathbb{1}_{2\times2} & 0_{3\times3} \\ 0_{2\times3} & -\mathbb{1}_{2\times2} & 0_{3\times3} \end{bmatrix}_{10\times8} \tag{4.30}$$

$$b = \begin{bmatrix} f_{max} - f_o \\ -f_{min} + f_o \\ \alpha_{max} - \alpha_o \\ -\alpha_{min} + \alpha_o \end{bmatrix}_{10\times1} \tag{4.31}$$

$$f_{max} = \begin{bmatrix} 8 \\ 8 \\ 8 \end{bmatrix}_{3\times1} \tag{4.32}$$

.

$$f_{min} = \begin{bmatrix} -8 \\ -8 \\ -8 \end{bmatrix}_{3\times1} \tag{4.33}$$

$$\alpha_{max} = \begin{bmatrix} rad(80) \\ rad(80) \end{bmatrix}_{2\times 1} \tag{4.34}$$

$$\alpha_{min} = \begin{bmatrix} -rad(80) \\ -rad(80) \end{bmatrix}_{2\times 1} \tag{4.35}$$

$$A_{eq} = \begin{bmatrix} T(\alpha_o) & \frac{\partial}{\partial \alpha}(T(\alpha)f)\big|_{\alpha=\alpha_0, f=f_0} & \mathbb{1}_{3\times 3} \end{bmatrix}_{3\times 8} \tag{4.36}$$

$$b_{eq} = \begin{bmatrix} \tau - T(\alpha_o)f_0 \end{bmatrix}_{3\times 1} \tag{4.37}$$

$$lb = \begin{bmatrix} \Delta f_{min} \\ \Delta \alpha_{min} \\ -\infty \end{bmatrix}_{8\times 1} \tag{4.38}$$

$$ub = \begin{bmatrix} \Delta f_{max} \\ \Delta \alpha_{max} \\ \infty \end{bmatrix}_{8\times 1} \tag{4.39}$$

The vector 4.32 is the maximum forces in kN that can be produced by tunnel,azimuth 1 and azimuth 2 thrusters(top to bottom order). Similar is the vector 4.33 which is the minimum or negative forces in kN that the thrusters can produce.

The azimuth thruster can rotate 360°. But due to forbidden zones, some areas are restricted. Hence the maximum and minimum feasible angle region for thrusters are defined by vector 4.34 and 4.35(Angle in degrees converted to radians). The order corresponds to azimuth 1 and azimuth 2 thrusters. Since the azimuth thrusters are assumed to produce thrust in both direction in this thesis, by defining such an angle region $\alpha \in [-80, 80]$, we obtain a double sided pac-man thrust region(figure 4.7). In the simulation, the Azimuth 1 thruster is initialised at 45°and Azimuth 2 at -45°.

Chapter 5

Simulation Environment

This chapter describes the simulation environment used for co-simulation with details on different FMU's and their functionalities.

In this thesis, a co-simulation framework named Vico[12] is chosen for carrying out simulations. Vico is high-level co-simulation framework developed by researchers at NTNU, Ålesund with a motivation to introduce co-simulation in virtual prototyping of marine operations.

5.1 Vico

Vico is based on a software architecture named Entity-Component System(ECS) [12]. In this architecture, each object(entity) contains data(components) that can be added, removed, or changed during the run-time of co-simulation. This alteration of data alters the behavior of the object suiting the requirement of the simulation. The higher-level architecture of ECS is given in Figure 5.1.

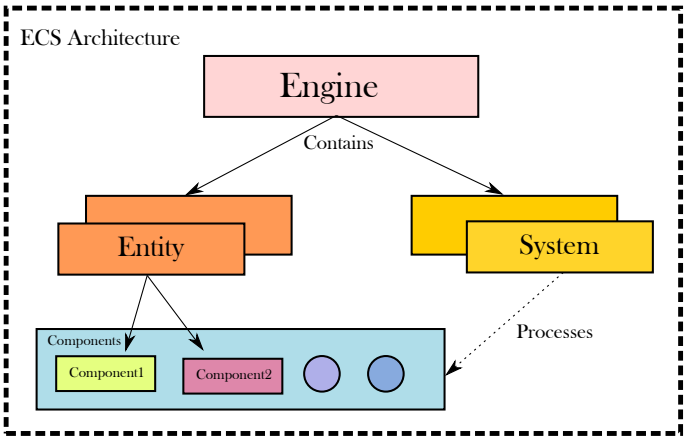


Figure 5.1: ECS Architecture in Vico

Source: Adapted from [12]

In the architecture, each entity is a container for components which is a state without any behavior. Each entity exhibits behavior when some process from the system acts on these entities. These

processes come from a family which is a set of entities with a certain set of components attached [12].

In this thesis, all the components of the digital twin model of R/V Gunnerus is used as an FMU. This thesis produces two thrust allocation FMU that works in cooperation with other FMU's.

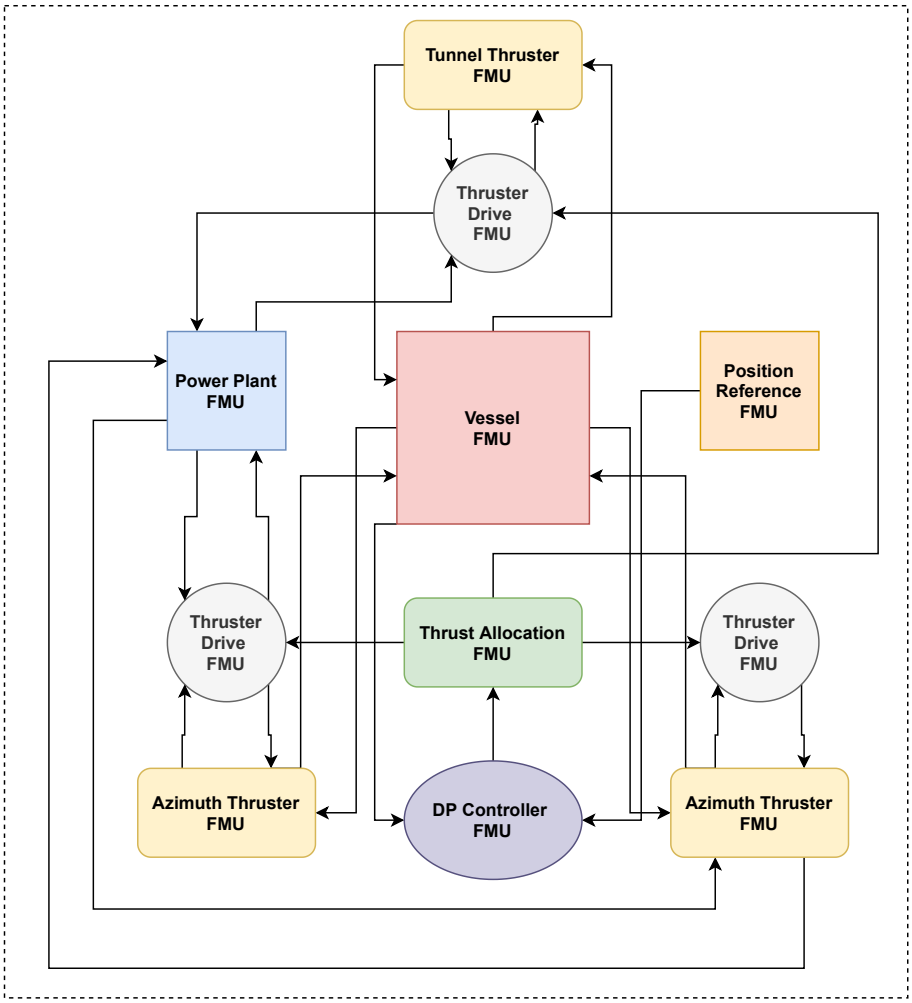


Figure 5.2: FMUs used in Co-simulation

The overview of the different FMUs used for co-simulation in the thesis is given in figure 5.2. These FMUs have been supplied by different companies or created by the researchers at NTNU Ålesund. The name of the FMU is suggestive of the function the FMU performs in the co-simulation. The connection shown in figure 5.2 is a higher level view of information transfer across the FMUs.

Providers of the FMUs used in the thesis is given in table 5.1. The information about the FMU's is given below as sourced from [39]:

1. **Vessel FMU** : This FMU outputs radiation force, mass force and restoring forces of the vessel after using its solver.
2. **Azimuth & Tunnel Thruster FMU** : This is a hydrodynamic model of the thruster that outputs forces in 3DOF when inputs are RPM,location on hull, vessel speed etc.
3. **Thrusterdrive FMU** : This FMU computes RPM for the Thruster FMUs when required

forces are input to it.

4. **DP Control FMU** : This is a standard PID controller that computes desired forces in Surge,Sway and moment in Yaw. These value are the input to the allocator.

5. **Position Reference FMU**: This FMU can be used to provide position reference during simulation. For example, during the 4-corner test, this FMU gives the setpoint for heading and position to the DP controller.

6. **PowerPlant FMU**: This is mathematical model of a generator set with auxiliary load and circuit breakers.

7. **Thrust Allocation FMU** : Thesis developed FMU written in Python that distributes commanded forces from controller to individual actuator controls

Name	*Provider*
Vessel FMU	Sintef Ocean
Tunnel Thruster FMU	Kongsberg Maritime
Power Plant FMU	Sintef Ocean
Azimuth thruster FMU	Kongsberg Maritime
Thrust allocation FMU	Thesis developed
DP Control FMU	NTNU Alesund
Position Reference FMU	NTNU Alesund
Thruster Drive FMU	Sintef Ocean

Table 5.1: FMU Providers

The simulation for all scenario has been done in a Windows computer with a specification of Intel i7-10875H CPU with 8 Core. The same simulation could also be done without much performance degradation on a relatively less powerful computer. Only reason to use a powerful computer was to use a dedicated GPU for training the deep learning model.

To setup a co-simulation using Vico framework, the following four steps are to be done. This is shown in figure 5.3

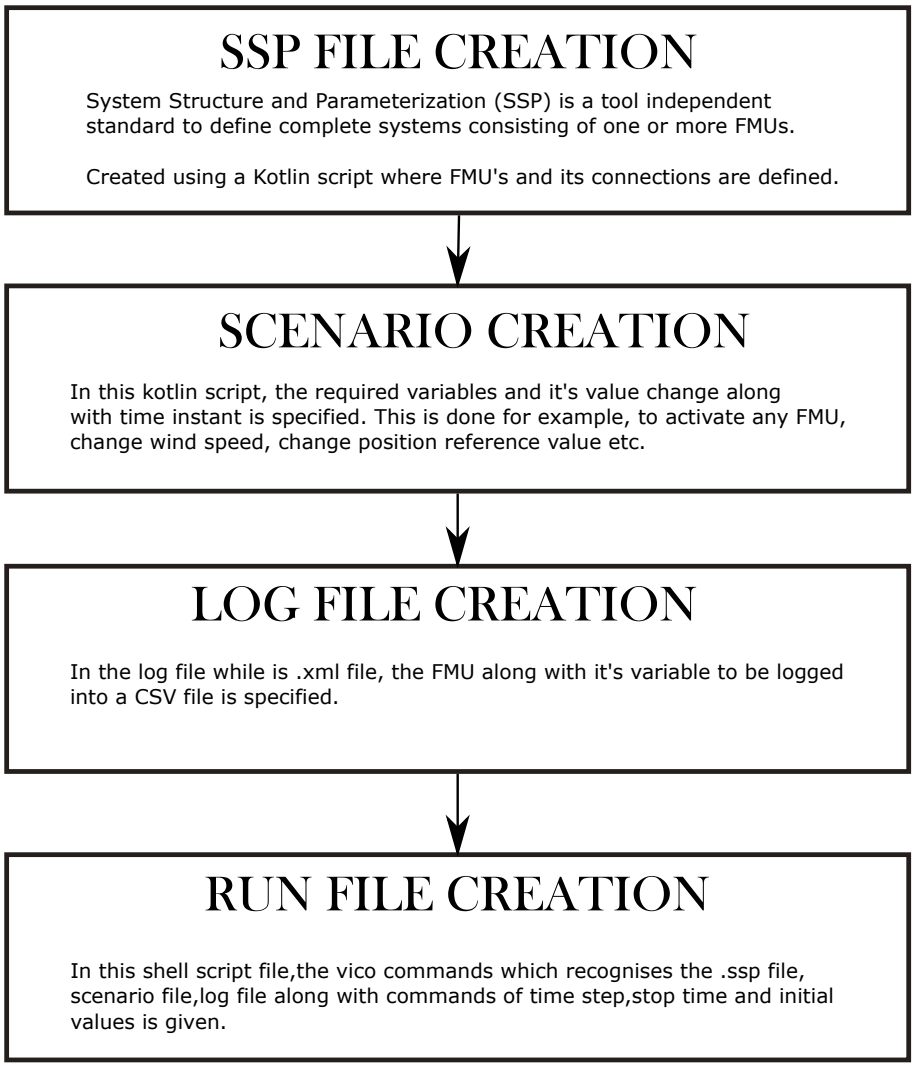


Figure 5.3: Co-simulation setup steps

Once the 4 steps in the figure 5.3 is executed, the shell script obtained at the end is run in bash environment. This executes the co-simulation and once the simulation is finished, the logged values from FMUs can be used to assess the simulation.

5.2 PythonFMU

In the thesis, to build FMU, the framework used is "PythonFMU" [33][40]. By writing Python 3.x codes using the framework, an FMU can be created that is compliant with FMI 2.0 standards. Other frameworks or tools that help in FMU creation can be found in [39].

Chapter 6

Results and Discussion

In this chapter, the result for different scenarios where the DAE allocator has been tested is presented. The allocator has been tested in three scenarios:

1. A standard 4 corner test
2. Stationkeeping test
3. Stationkeeping test with thruster failure

The assumptions are conditions for these simulations are provided here.

1. Each test is run by creating separate scenarios and running them in co-simulation. The simulation time for each test is 1400 seconds with a fixed time step of 0.05 seconds for the co-simulation master algorithm.
2. As mentioned before, after each test, a set of .csv files are obtained which contains the variables logged from each FMU that was specified. It is these values that are plotted in this chapter.
3. A comparison of the result of SQP based allocator is also presented to show performance differences.
4. The average power consumed during each scenario is also an important metric considering the fact that every allocator should try to reduce power consumption while meeting the request force commands. In the thruster failure test, this is not provided as the metric to be considered is whether the vessel can return to its original position.
5. In all the test, the allocators are tested on the basis of these metrics:
   (a) The ability of allocator to restrict the force and angles to its saturation values.
   (b) The ability of allocator to restrict rate changes for force and angle to its prescribed limit.
   (c) The ability of allocator to produce the intended result according to the mission of the test.

6. If the allocator is able to meet these requirements, then the test is considered to be a success and deficiency in meeting the requirement is considered to be a point of improvement as future work.

7. In each test, during the simulation the allocator and DP controller FMU is enabled or turned on at a time equal to 50 seconds. The only different case is in the test of 4 corner test where the SQP allocator is enabled at 0 seconds itself. But this does not affect the result in any manner. Here a point to be considered is the realistic performance of different FMU. In reality, when the allocator is turned on at 50 seconds and the allocator is suddenly allocating a value of 1000 N, the thruster FMU cannot provide that value. But in simulation, due to constraint of time of thesis, the FMUs could not be tuned properly such that they exhibit an unrealistic behavior of producing large forces(same with angles) during turn-on time of allocators. This can be generally seen in plots. Considering this, the plot for rate values are shown from time 51 second. Realistic tuning of FMU is considered as future work of the thesis.

8. Allowed force rates are
$$\dot{f} = \begin{bmatrix} \pm 50 \ N/Hz \\ \pm 50 \ N/Hz \\ \pm 50 \ N/Hz \end{bmatrix} \tag{6.1}$$
for each thrusters considering 20 Hz simulation rate.

9. Allowed angle rates are
$$\dot{\alpha} = \begin{bmatrix} \pm 0.5 \ ^\circ/Hz \\ \pm 0.5 \ ^\circ/Hz \end{bmatrix} \tag{6.2}$$

10. Allowed force for each thrusters are $\pm 8000 \ N$ and angle are $\pm 80^\circ$ as discussed previously.

6.1 4 Corner Test

A 4 corner test is a standard low-speed maneuver that is aimed at checking if the vessel can traverse a square by following the track. Large deviations from the track are not a desirable feature of the allocator and thus the performance from this test is a good indicator for a proper thrust allocation.

A four-corner test is conducted by giving the following set points in terms of position and heading at these respective times. The setpoints are provided by the Position Reference FMU to the DP controller which in turn commands the thrust allocator to produce forces and moments. The setpoints provided for the test are given in Table 6.1. The values for setpoint persist in time after it is set till further changes are made to the same variables.

Time(s)	**Setpoint**
50	North : 10 m
300	East : -10 m
500	Yaw : -45°
700	North : 0 m
900	East : 0 m
900	Yaw : 0 m

Table 6.1: 4 corner test setpoints

The test is conducted with the same configuration for all values in different FMU's except the change of allocator to retain fairness in comparison. The result of the test is provided below.

In figure 6.1, the motion of the vessel in the North-East direction is recorded for comparing the performance of both allocators. From the figure 6.1, it can be seen that the DAE allocator has a very comparable performance relative to the SQP allocator. A reference square is drawn to indicate the ideal track the vessel should follow. Both allocators only show slight deviation from the ideal track.

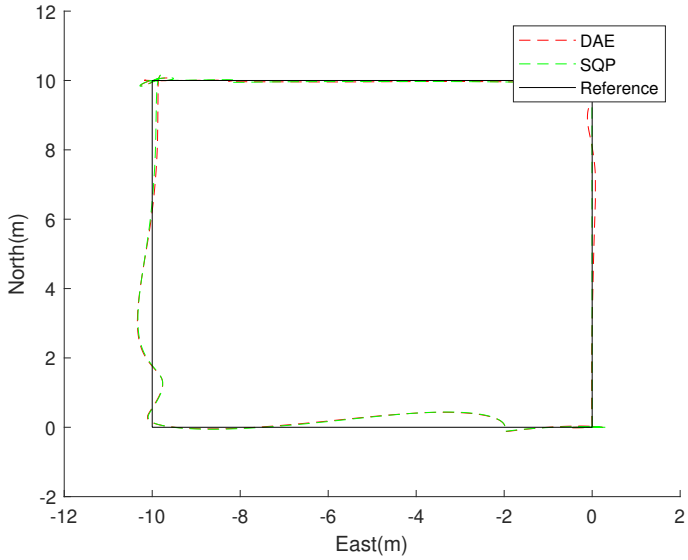


Figure 6.1: Vessel motion for DAE and SQP allocator for 4 corner test

In figure 6.2, the heading of the vessel for both allocator is recorded. Both allocators share almost the same heading value during the 4-corner test.

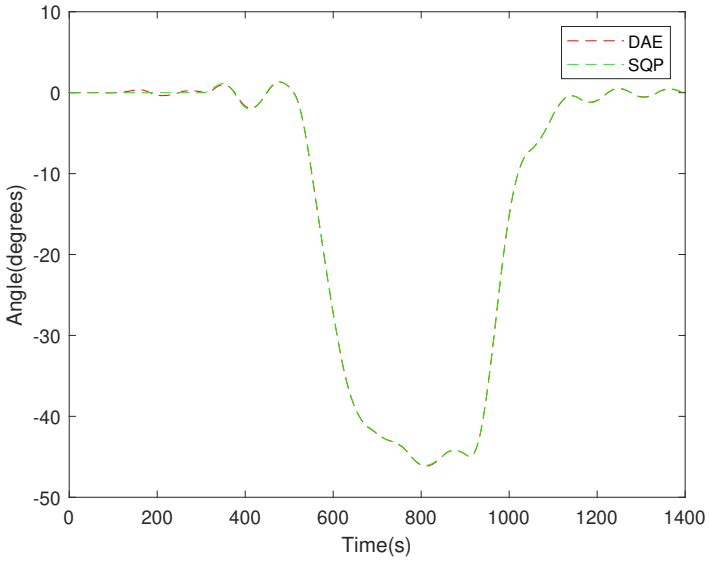
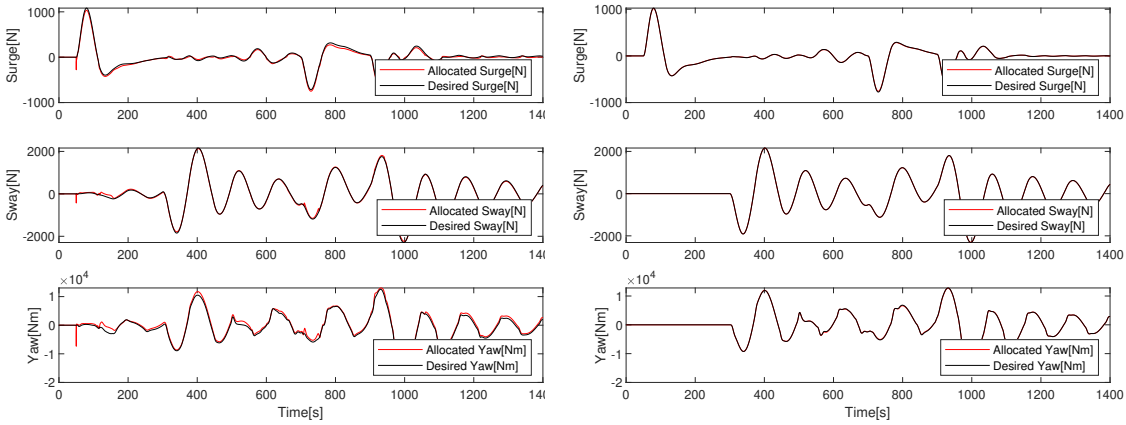


Figure 6.2: Comparison of heading angle for DAE and SQP Allocator for 4 corner test

In figure 6.3, the response of the allocators against the force request from the controller for the 4-corner test is given. The DAE allocator is able to match the request with only some deviations. The deviation is more pronounced in the Yaw torque request. Comparing this against the SQP allocator in the same figure, the SQP allocator can follow the requests without any deviations.



(a) Allocation information from DAE Allocator (b) Allocation information from SQP Allocator

Figure 6.3: Comparison of controller request and allocation from the allocators for 4 corner Test

In figure 6.4, the forces generated by the three thrusters in the vessel are given for the two allocators. The tunnel thruster is used more by both the allocators as it can produce the yaw moments which is more important for the 4 corner test. The constraint for forces is met properly by both allocators.

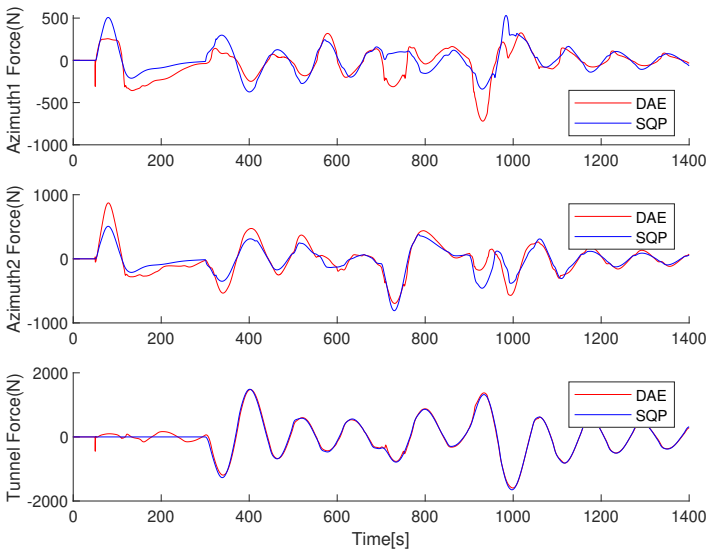
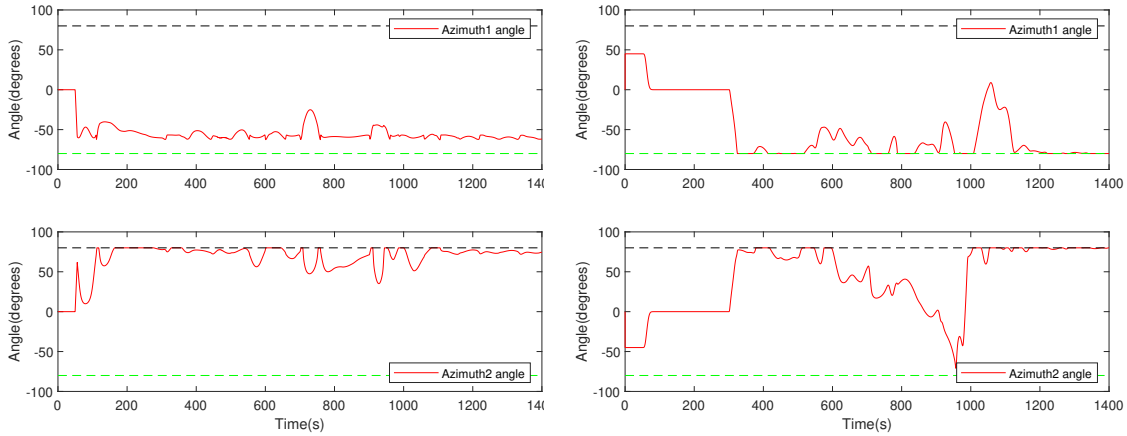


Figure 6.4: Forces generated by the two allocators during 4 corner test

In figure 6.5, the angle values for the two azimuth thrusters in the vessel during simulation are given for both the allocators. In this figure 6.5, it can be seen that for a time from 0 to 50 seconds, the angle value for the DAE allocator is zero because it has not been activated to process the force requests. Similarly for the SQP allocator, for time 0 to 50 second, the angle value is 45°and -45°for

Azimuth1 and Azimuth2 respectively. This means it was turned on from time 0 second itself. This is the warm-start value of the allocator and the allocator only starts processing the forces requests from time 50 seconds when the DP controller is activated.

The DAE allocator has a more reserved angle value for the azimuth thrusters whereas for the SQP allocator, the angle value changes covering its full allowed range of [-80°,80°]. This could be the SQP allocator's way to reduce power consumption by effective use of azimuth angles.



(a) Azimuth angles by DAE Allocator (b) Azimuth angles by SQP Allocator

Figure 6.5: Azimuth angle value from the two allocators for 4 corner test

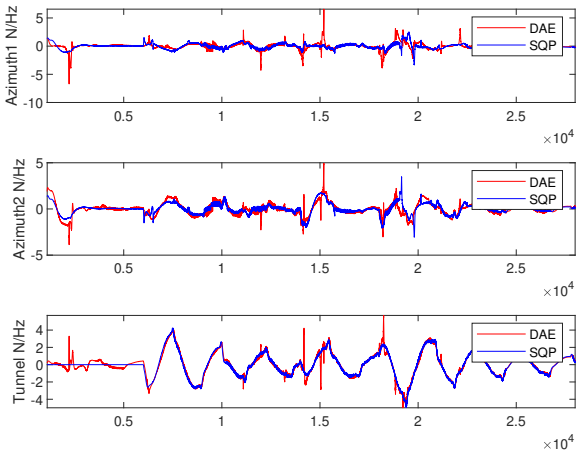
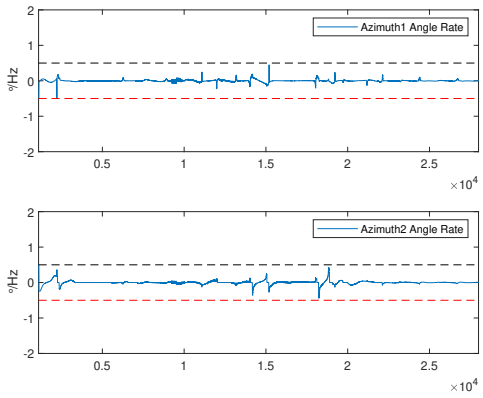


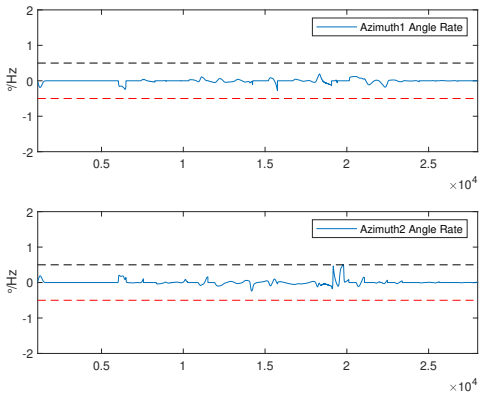
Figure 6.6: Force rate change information from the two allocators for 4 corner test

In figure 6.6, the rate of change of forces by the three thrusters for both allocators is provided. The plot is obtained by taking the difference of consecutive values in the simulation log to get the change of value per cycle of simulation. So the y-axis of the plot is (N/Hz) as can be seen. The force constraints are met properly by each allocator.

In figure 6.7, the rate of change of azimuth angle for the two allocators are provided. It can be seen that both the allocators are able to limit its angle change value to the prescribed upper(0.5°/Hz) and lower(-0.5°/Hz) limits.



(a) Angle rate change by DAE allocator



(b) Angle rate change by SQP allocator

Figure 6.7: Angle rate change information from the two allocators for 4 corner test

The average power consumption by the two allocators during the 4-corner test is given in Table 6.2. The SQP allocator has a better power consumption characteristic compared to the DAE allocator. Since the DAE allocator has a tunable power loss factor in its design, a better value could be obtained if thorough tuning is done. This applies to SQP as well to get a better result.

Allocator	*Power(kW)*
DAE Allocator	3.43
SQP Allocator	3.28

Table 6.2: Average power consumption during 4 corner test

6.2 Stationkeeping Test

In the Stationkeeping Test, the vessel is supposed to hold its position while encountering wind that changes in magnitude and direction. In this test, waves were not provided because a wave filter was not present in the simulation to remove the first-order excitation from the wave frequency. Wind conditions were provided as per table 6.3. The wind condition is stabilised to a speed of 10 m/s and angle of 45° after 440 seconds in the simulation and persists till the end of the simulation.

In the results provided for DAE and SQP allocator in the figure 6.8, it can be seen that the motion characteristics of the vessel provided by both the allocators are the same. There is almost no difference between their motion in North, East, or Heading. This is a favorable feature for DAE compared against the SQP allocator. During the transition time when the wind direction change, the vessel response is the same for both allocators. Interestingly, it can be seen that the wind direction change has more effect on motion response than the wind speed change.

Time (sec)	*Wind Speed (m/s)*	*Wind Direction (degree)*
50	1	90
60	1.5	90
70	2	90
80	3.5	90
90	4	90
100	3	90
110	5	90
120	6	90
130	8	90
160	10	90
410	10	70
420	10	60
430	10	50
440	10	45

Table 6.3: Wind direction and speed for stationkeeping test

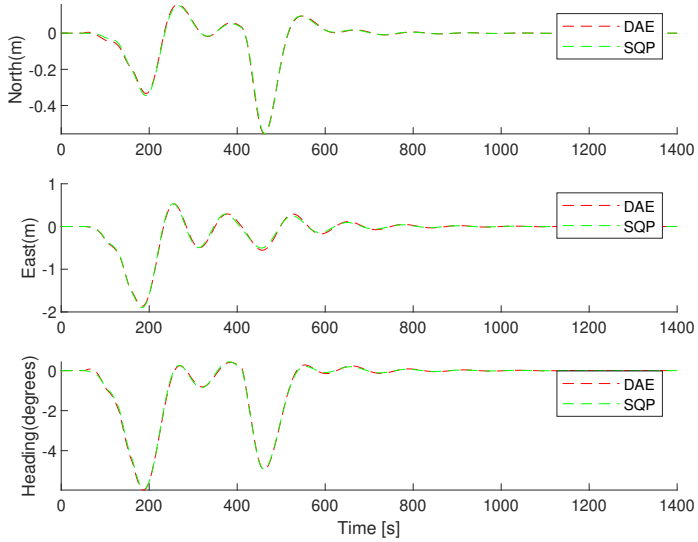
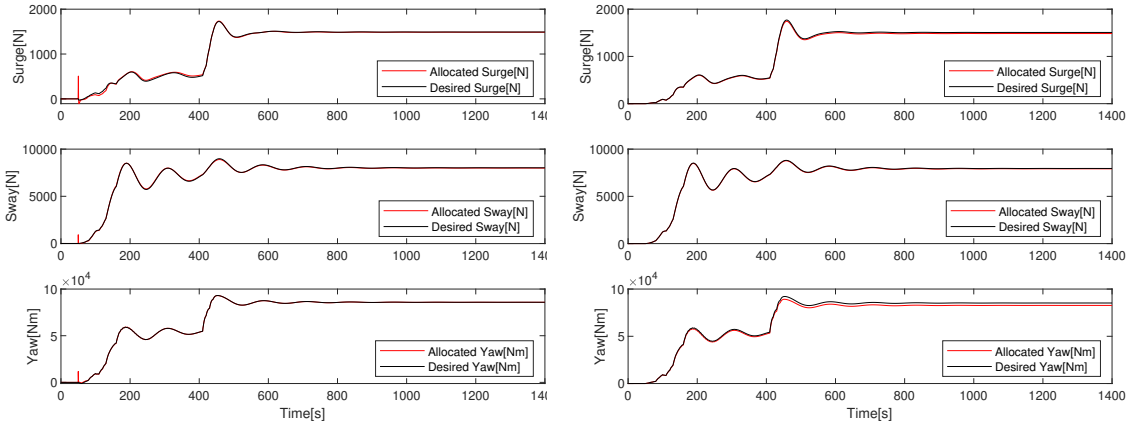


Figure 6.8: Vessel response for SQP and DAE allocators for stationkeeping test

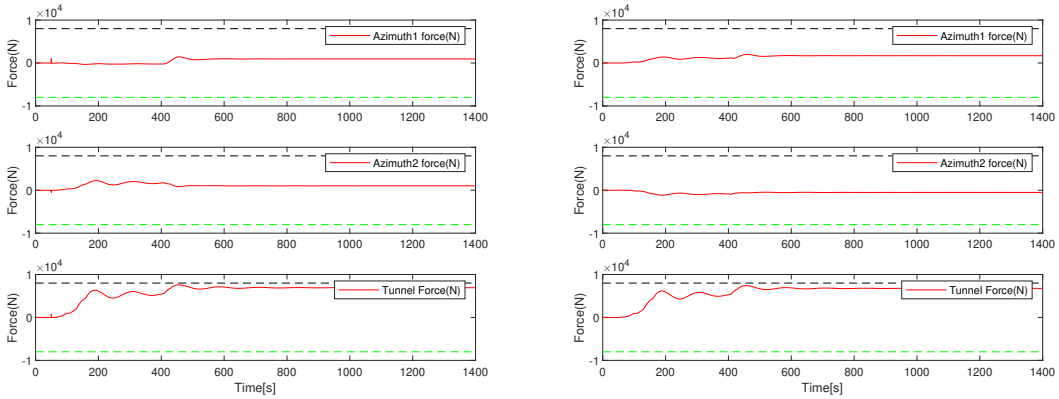
In figure 6.9, the response of allocator in providing forces (Surge and Sway) and moment in Yaw is given for both the allocators. It can be seen that the DAE allocator provides a better allocation compared to the SQP allocator. The SQP allocator has a slight deviation from the requested torque in Yaw and delivered torque after the wind condition stabilises. During the transition period where wind speed and direction are changing, the requested forces and moments are properly met by each allocator. For the DAE allocator, at time instant 50 seconds, an abrupt change in value is seen owing to its turn-on situation as discussed before. This is not observed in the SQP allocator.



(a) Allocation information from DAE allocator (b) Allocation information from SQP allocator

Figure 6.9: Comparison of controller request and allocation from the allocators for stationkeeping test

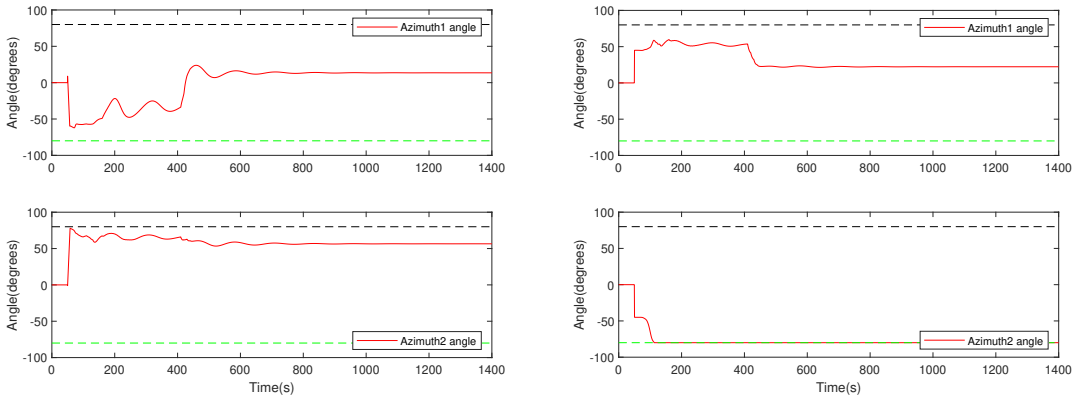
Comparing the force produced by each allocator for the three thrusters in figure 6.10, it can be seen that it is the tunnel thruster that is more active for stationkeeping. This is natural since it can produce the requested moment easily than the other two thrusters. Azimuth thrusters are put to use mainly in the transition period up to 440 seconds after which the tunnel thruster maintains a force value close to its saturation constraint magnitude. No constraints are broken in the test.



(a) Forces by DAE Allocator (b) Forces by SQP Allocator

Figure 6.10: Forces generated by the two allocators for stationkeeping test

In figure 6.11, the variation of the angle of the azimuth thrusters is given. Both allocators behave in the opposite way for the stationkeeping allocation. Azimuth thruster 1 angle starts in opposite direction for DAE allocator and SQP allocator and settle down to value close to each other. Azimuth thruster angle 2 also behaves oppositely for each allocator. For the SQP allocator, the Azimuth 2 angle can be seen to be at its magnitude saturation value.



(a) Azimuth angles by DAE allocator

(b) Azimuth angles by SQP allocator

Figure 6.11: Azimuth angle value from the two allocators for stationkeeping test

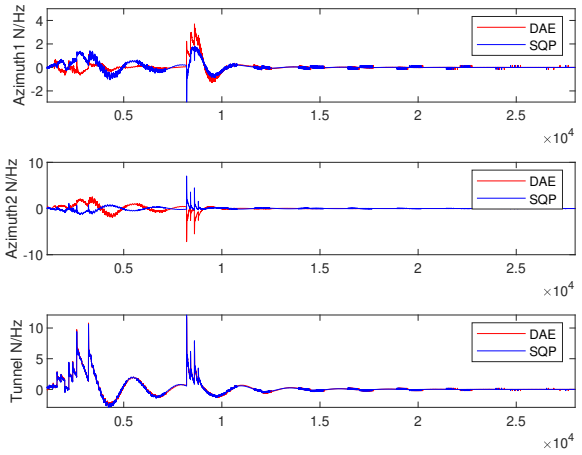


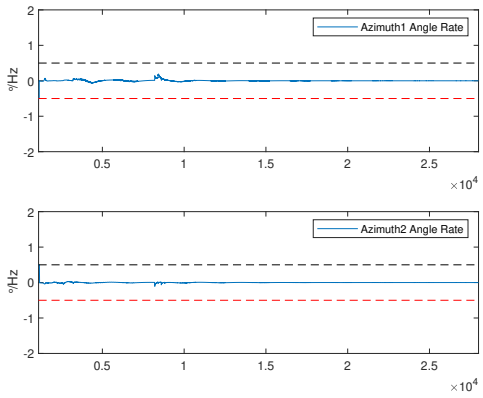
Figure 6.12: Force rate change information from the two allocators for stationkeeping test

In figure 6.12, the rate of change of forces for the three thruster by the two allocators are given. They meet the rate constraints properly. In figure 6.13, the rate of change of angle of azimuth thrusters are given. They also meet the prescribed angle rate constraints.

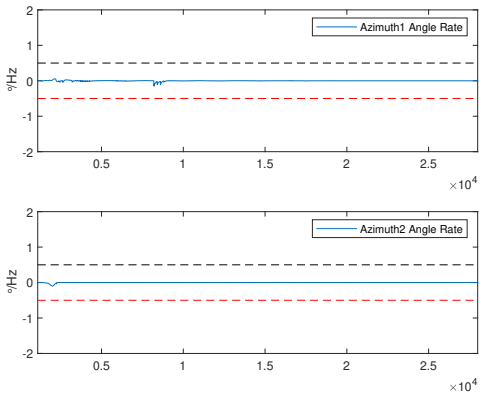
Allocator	*Power(kW)*
DAE Allocator	48.3
SQP Allocator	48.6

Table 6.4: Average power consumption during stationkeeping test

For power comparison in Stationkeeping Test, the DAE has a small margin upper hand than the SQP Allocator.



(a) Angle rate change by DAE allocator



(b) Angle rate change by SQP allocator

Figure 6.13: Angle rate change information from the two allocators for stationkeeping test

6.3 Stationkeeping Test with Thruster Failure

This test is done to exhibit one of the worst-case scenarios during the DP operation of ships: thruster failure. If during a station-keeping operation, one of the thrusters fails, the ship can lose its position and potentially jeopardize the entire operation. So thruster failure tolerance is one of the required features of the allocator. In the thesis, none of the developed allocators has a thruster fault detection feature inbuilt and if one of the thrusters fails, the allocator would see a large force and moment request from the motion controller and would have to allocate forces meeting this requirement.

For this test, the vessel is put to stationkeeping mode with the following environment wind condition as given in Table 6.5. The wind speed is set to a small magnitude to have a realistic weather condition. The simulation is run for 1400 seconds and at 700th second, the azimuth thruster 2 is cut off imitating a thruster failure scenario. Only the force value is assigned a 0 Newton and the thruster can continue to rotate.

Time (sec)	*Wind Speed (m/s)*	*Wind Direction (degree)*
50	1	90
60	1.5	90
70	2	90
80	3.5	90
90	4	90
110	5	90
120	6	90

Table 6.5: Wind direction and speed for stationkeeping with thruster failure

From the figure 6.14, the motion response of the vessel during the thruster failure scenario for both allocators can be seen. It can be seen that up to time 700 seconds, both allocators provide the very same motion characteristics to the vessel. This is similar to the stationkeeping result presented before. But once the Azimuth thruster 2 is turned off, both allocators show deviation from the stationkeeping setpoint. The motion is very high for the DAE allocator and results in a jitter

motion that continues to dampen as the simulation proceeds. The SQP allocator can return to the setpoint without much jitter motion. Its response characteristics are quite smooth compared to the DAE allocator. Even though the DAE network has a high heading change of 10°, the motion in the North and East direction remains small because the DAE is a data-driven model and it has not seen such a scenario during training.

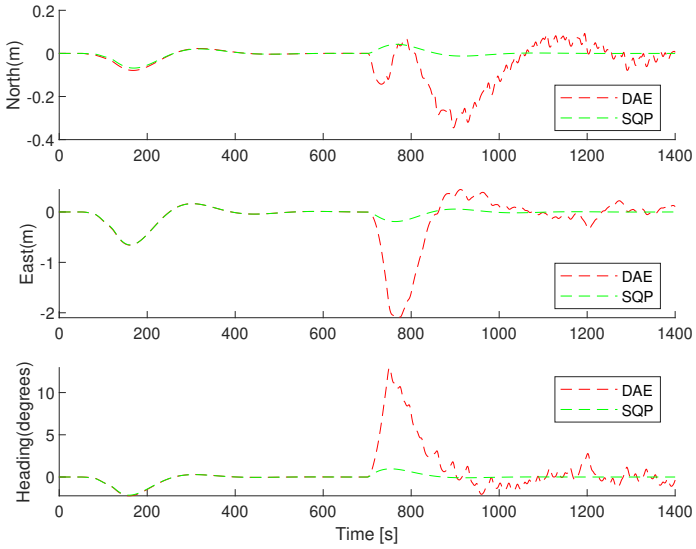
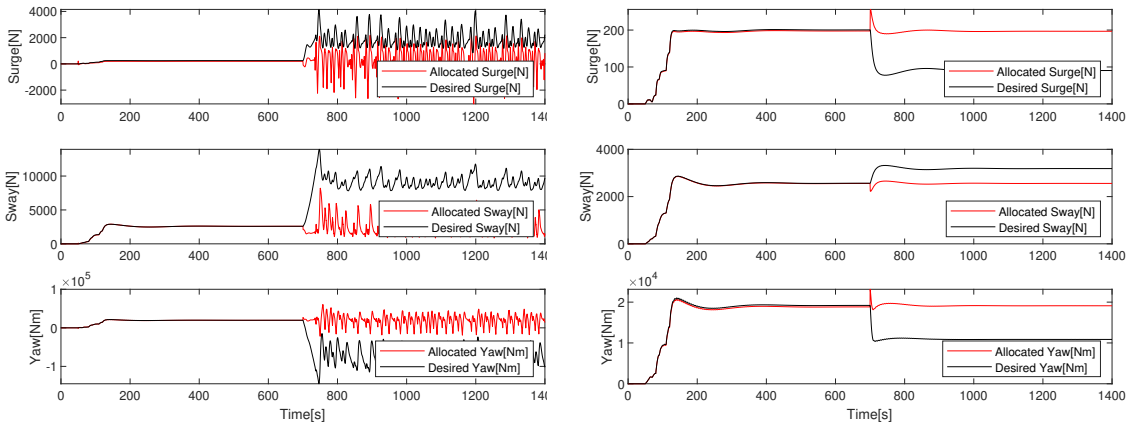


Figure 6.14: Vessel response for SQP and DAE allocators during stationkeeping thruster fail test



(a) Allocation information from DAE allocator

(b) Allocation information from SQP allocator

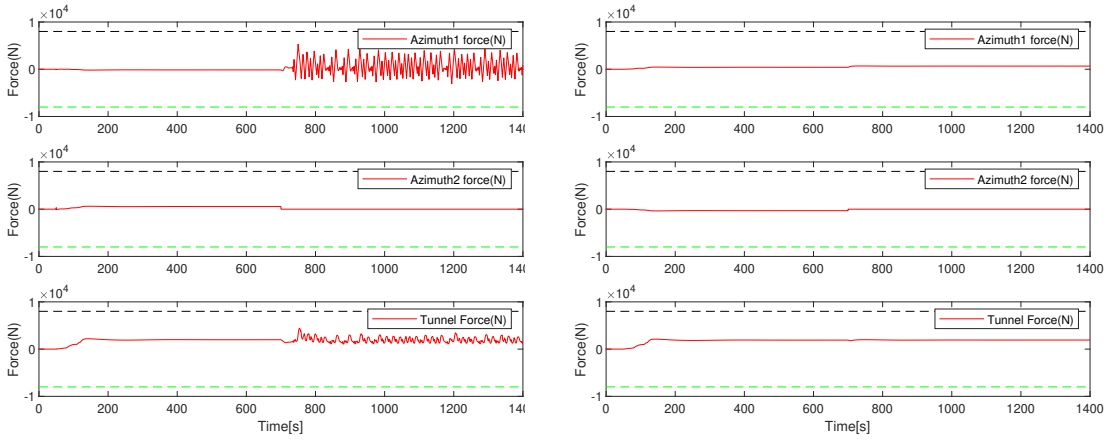
Figure 6.15: Comparison of controller request and allocation from the allocators during station-keeping thruster fail test

In figure 6.15,the response of the allocator against the controller request is provided.This figure shows one of the significant characteristics of the DAE allocator in predicting the forces when the requested commands are high and cannot meet them with the hard rate constraints getting activated. Both allocators are not able to meet the requested force request from the controller as can be seen in the figure 6.15. The DAE allocator has a hard time allocating forces properly and results in large jitters for allocation. The SQP network does not exhibit this jitter allocation

response. The removal of this jitter allocation response is considered as a future work of this Master thesis.

An explanation to the jitter response is that the DAE allocator which is a data-driven model likes fewer setpoint changes. This could be seen as an advantage in the Stationkeeping test against the SQP allocator. Also, in the thruster failure scenario, the PID controller derivative value is quite high providing large quick setpoints to DAE allocator that it return by allocating in jitter values. Proper tuning of the controller is considered as future work.

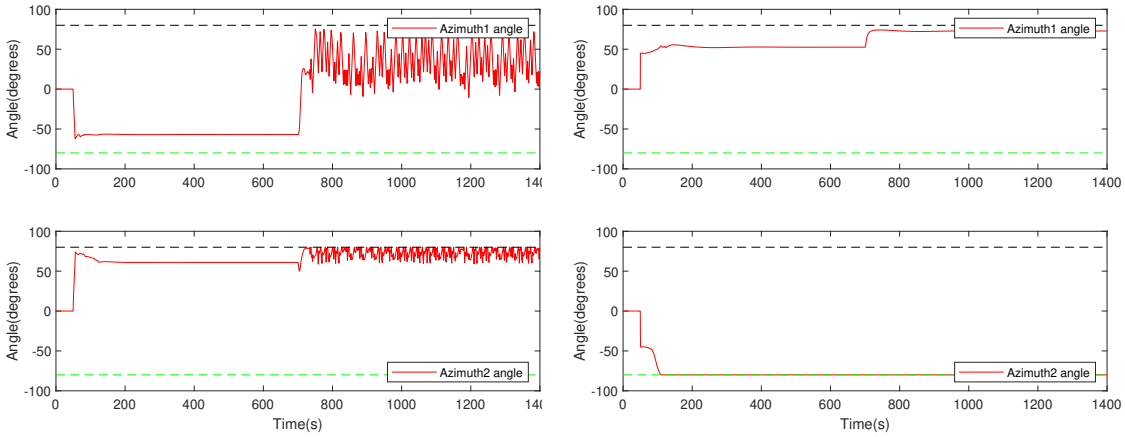
Due to a large change in the controller request, the forces produced by the thrusters also exhibit such a jitter response for the DAE allocator as can be seen in figure 6.16. The thruster cut of Azimuth 2 can be seen at time instant 700 seconds for both allocators exhibited by a step decrease to zero value in forces produced. For the DAE allocator, the jitter is present for both tunnel and azimuth 1 thruster in force produced. On the other hand, the SQP allocator has a smooth force delivery showing its superior allocation quality. In either case, forces remain within the prescribed boundary.



(a) Forces by DAE allocator

(b) Forces by SQP allocator

Figure 6.16: Forces generated by the two allocators during stationkeeping thruster fail test



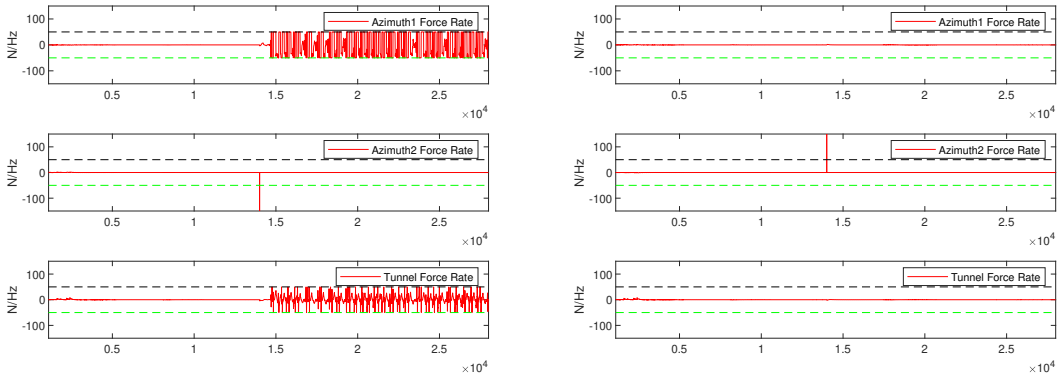
(a) Azimuth angle value by DAE allocator

(b) Azimuth angle value by SQP allocator

Figure 6.17: Azimuth angle value information from the two allocators for stationkeeping thruster fail test

The azimuth angle value for both allocators can be seen in figure 6.17. The jitter response of the DAE allocator is reflected in both azimuth thrusters. Here both azimuth thrusters continue to rotate within their prescribed constraint despite Azimuth 2 force being zero after thruster failure. There is no jitter in the angle value for the SQP allocator.

The extend of jitter of the forces and angles can be seen in figure 6.18 and figure 6.19 respectively where their rate change information is provided. A large spike in Azimuth 2 forces can be seen around the middle of the figure for both DAE and SQP allocator. This is where the thruster is turned off imitating a failure. After this instant, the rapid rate change for Azimuth1 and Tunnel force can be seen for the DAE allocator. The robust hard rate constraint allows the allocator to keep the rate within the prescribed limit of $\pm 50 N/Hz$. The DAE allocator has very smooth force rate characteristics.

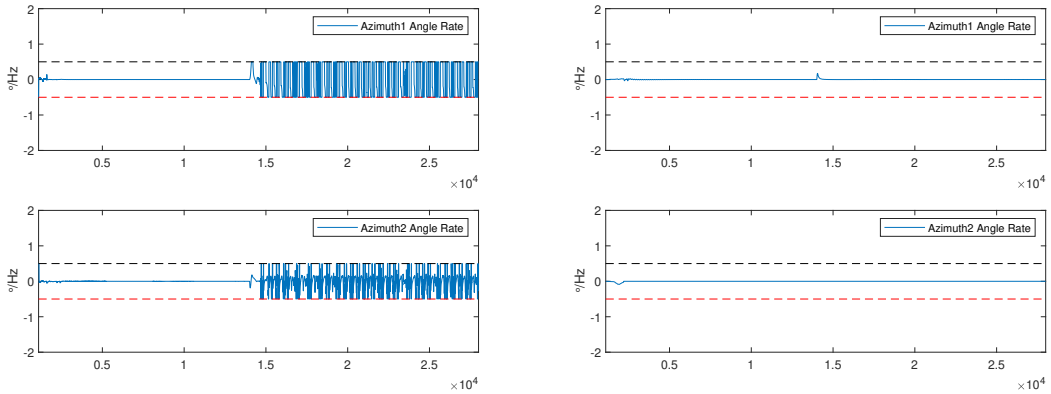


(a) Force rate change by DAE allocator

(b) Force rate change by SQP allocator

Figure 6.18: Force rate change information from the two allocators for stationkeeping thruster fail test

The angle rate change information given in figure 6.19 for the allocator also shows the jitter for DAE allocator.The SQP allocator also have a superior angle rate change characteristics.



(a) Angle rate change by DAE allocator

(b) Angle rate change by SQP allocator

Figure 6.19: Angle rate change information from the two allocators for stationkeeping thruster fail test

It is to be noted that one the drawback of DAE allocator is that, whenever the forces from DP

controller exceeds the trained values for DAE allocator, it predicts very wrong values similar to any ML model predicting wrong when data outside its trained regime is provided.

6.4 Summary

When this thesis was being developed, the existing DAE allocator [8] was proposed to small force ranges, and through this thesis, it could be found that the method can be extended to high force ranges with added robustness reassuring the validity of the existing method. The neural network architecture has been changed with the increased speed of allocation during simulation runs.

One of the main challenges of the Deep learning allocator was its robustness to meet the constraints. Without meeting the physical constraints of the thrusters, the method cannot be implemented in real-world scenarios. This thesis has proven that it is possible to implement this by making a novel custom layer that can take this into account. The approach has been tested for different scenarios with the same fairness across each test

The developed DAE allocator can perform similar to a classic SQP allocator in the 4-corner test despite having a higher power requirement. The power requirement is a tunable loss term in the DAE allocator and changes to it can change the situation favorable to the DAE allocator.

The DAE allocator has a better performance characteristic for the Stationkeeping test than the SQP allocator. From this test, it could be inferred that the DAE allocator can perform better for scenarios where large setpoint changes are not present. The power consumption of the DAE allocator is better than the SQP allocator.

The only scenario where the DAE allocator had an inferior performance was during the thruster failure scenario. This could be partly explained by the fact that the data the DAE allocator has been trained in does not have the force request regime as experienced during the thruster failure test. An improperly tuned DP controller could also be considered as adding a problem to this situation. The fixing of both these problems is considered as a future work of the thesis. Here the reader is to note that the DAE allocator has an inherent disadvantage as any Machine Learning Model: whenever the allocator sees force request above its trained regime, it allocates very poor values for forces and angles. This is similar to a Machine Learning model predicting incorrectly when inputs that are out of the trained regime are given.

Chapter 7

Conclusion and Future Works

In this Master thesis, a Deep Learning based Thrust Allocator in form of a Deep Autoencoder Network has been developed following the previous research streams([8],[21]). In this process, several improvements to the existing method have been proposed and the robustness and validity of the method have been checked through simulation.

The thesis has contributed to solving one of the future works considered in the original DAE-based allocator [21] where adding rate constraints was a challenge. Through the thesis, a robust rate handling for forces and angles has been found through the use of custom layers.

The thesis has met its objective of developing two FMUs-one is the classic control allocation FMU based on the SQP approach and the DAE allocator. Both of the FMUs have been tested extensively in the co-simulation environment and the results have been presented. These FMUs can be used for further testing and modification. The thesis has tried to point out the merits and demerits of the proposed DAE allocator based on the three tests done.

The thesis has used the Vico co-simulation environment that was developed by the researchers at NTNU Ålesund and it can be found to be working well with respect to its desired functionality. The use of co-simulation holds great prospects in plug-and-play simulation for various marine components in a vessel.

The thesis can be concluded by stating that the Deep Learning-based DAE allocator has a future once more people get involved in the development process and extensive tests are done to optimise it. The testing and optimisation start right from the data generation part to where each component FMU in co-simulation can be tuned properly to exhibit realistic behavior.

7.1 Future Works

At the end of the thesis, the following can be considered as future work of the thesis:

1. Experiment different data generation strategies to incorporate a training regime where extreme cases can be learned by the network properly.
2. A study of the tradeoff between power loss factor and rate loss factor can be done to estimate

the performance difference in different scenarios.

3. In the thesis, only thrusters with a constant pitch were considered for actuation. To use the methodology followed in the thesis for other kinds of actuators such as Rudders, Variable Pitch propellers, Voith Schneider propellers, etc. holds great possibility to explore and research.

4. The current SQP approach in literature [11] can handle singularity avoidance. This feature could be incorporated into the DAE allocator.

5. In terms of co-simulation, there are fine adjustments that can be made to each FMU in the simulation that can make the simulation more realistic. This could not be explored in the thesis due to time constraints. This could be a good path to explore.

6. Online retraining of the DAE model is considered a future work that can help alleviate the problems in extreme cases such as thruster failure

7. Including fault tolerance to the DAE allocator is another desirable feature for the future.

Bibliography

[1] Asgeir J. Sørensen. A survey of dynamic positioning control systems. *Annual Reviews in Control*, 35(1):123 – 136, 2011. ISSN 1367-5788. doi: https://doi.org/10.1016/j.arcontrol.2011.03.008. URL http://www.sciencedirect.com/science/article/pii/S1367578811000095.

[2] Kongsberg. *DYNAMIC POSITIONING BASIC PRINCIPLES*. https://www.kongsberg.com/maritime/support/themes/dynamic-positioning-basic-principles [Accessed: 20.03.2021].

[3] Thor I Fossen. *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

[4] CHRISTIAAN DE WIT. Optimal thrust allocation methods for dynamic positioning of ships. Master's thesis, Delft University of Technology, 7 2009.

[5] F. Arditti, F.L. Souza, T.C. Martins, and E.A. Tannuri. Thrust allocation algorithm with efficiency function dependent on the azimuth angle of the actuators. *Ocean Engineering*, 105: 206–216, 2015. ISSN 0029-8018. doi: https://doi.org/10.1016/j.oceaneng.2015.06.021. URL https://www.sciencedirect.com/science/article/pii/S0029801815002644.

[6] Alphafold: a solution to a 50-year-old grand challenge in biology. https://deepmind.com/blog/article/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology [Accessed: 20.06.2021].

[7] Robert Skulstad, Guoyuan Li, Houxiang Zhang, and Thor I. Fossen. A neural network approach to control allocation of ships for dynamic positioning. *IFAC-PapersOnLine*, 51(29): 128 – 133, 2018. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2018.09.481. URL http://www.sciencedirect.com/science/article/pii/S2405896318321700. 11th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2018.

[8] R.Skulstad, G.Li, T.I Fossen, and H.Zhang. *Effective Constrained Control Allocation For Dynamic Ship Positioning Using Deep Neural Network* , 2021. Submitted to IEEE Robotics and Automation Magazine.

[9] NTNU. *R/V Gunnerus* . https://www.ntnu.edu/oceans/gunnerus [Accessed: 03.03.2021].

[10] DNV GL. *Open Simulation Platform*, June 2019. https://www.dnvgl.com/feature/open-simulation-platform-osp.html [Accessed: 25.03.2021].

[11] T. A. Johansen, T. I. Fossen, and S. P. Berge. Constrained nonlinear control allocation with singularity avoidance using sequential quadratic programming. *IEEE Transactions on Control Systems Technology*, 12(1):211–216, 2004. doi: 10.1109/TCST.2003.821952.

[12] Lars I. Hatledal, Yingguang Chu, Arne Styve, and Houxiang Zhang. Vico: An entity-component-system based co-simulation framework. *Simulation Modelling Practice and Theory*, 108:102243, 2021. ISSN 1569-190X. doi: https://doi.org/10.1016/j.simpat.2020.102243. URL https://www.sciencedirect.com/science/article/pii/S1569190X20301726.

[13] Martin Rindaroey and Tor Arne Johansen. Fuel optimal thrust allocation in dynamic positioning. *IFAC Proceedings Volumes*, 46(33):43–48, 2013. ISSN 1474-6670. doi: https://doi.org/10.3182/20130918-4-JP-3022.00032. URL https://www.sciencedirect.com/science/article/pii/S1474667016461318. 9th IFAC Conference on Control Applications in Marine Systems.

[14] Tor A. Johansen and Thor I. Fossen. Control allocation—a survey. *Automatica*, 49(5):1087 – 1103, 2013. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2013.01.035. URL http://www.sciencedirect.com/science/article/pii/S0005109813000368.

[15] Henrik Emil Wold. Thrust allocation for dp in ice. Master's thesis, Norwegian University of Science and Technology, 6 2013.

[16] Aleksander Veksler, Tor Arne Johansen, Roger Skjetne, and Eirik Mathiesen. Thrust allocation with dynamic power consumption modulation for diesel-electric ships. *IEEE Transactions on Control Systems Technology*, 24(2):578–593, 2016. doi: 10.1109/TCST.2015.2446940.

[17] Stian Skjong and Eilif Pedersen. Nonangular mpc-based thrust allocation algorithm for marine vessels—a study of optimal thruster commands. *IEEE Transactions on Transportation Electrification*, 3(3):792–807, 2017. doi: 10.1109/TTE.2017.2688183.

[18] Da-wei Zhao, Fu-guang Ding, Jin-feng Tan, and Xin-qian Bian. Optimal thrust allocation based ga for dynamic positioning ship. In *2010 IEEE International Conference on Mechatronics and Automation*, pages 1254–1258, 2010. doi: 10.1109/ICMA.2010.5589933.

[19] Luman Zhao and Myung-Il Roh. A thrust allocation method for efficient dynamic positioning of a semisubmersible drilling rig based on the hybrid optimization algorithm. In *Mathematical Problems in Engineering*, 2015. URL https://doi.org/10.1155/2015/183705.

[20] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, volume 2, pages 985–990 vol.2, 2004. doi: 10.1109/IJCNN.2004.1380068.

[21] Huang Huan, Wei Wan, Chunling We, and Yingzi He. Constrained nonlinear control allocation based on deep auto-encoder neural networks. In *2018 European Control Conference (ECC)*, pages 1–8, 2018. doi: 10.23919/ECC.2018.8550445.

[22] P. D. Vries and E. Kampen. Reinforcement learning-based control allocation for the innovative control effectors aircraft. 2019.

[23] Simen Sem Øvereng. Dynamic positioning using deep reinforcement learning. Master's thesis, Norwegian University of Science and Technology, 6 2020. URL https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2731248?locale-attribute=no.

[24] Patrik Kolaric, Victor G. Lopez, and Frank L. Lewis. Optimal dynamic control allocation with guaranteed constraints and online reinforcement learning. *Automatica*, 122:109265, 2020.

ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2020.109265. URL http://www.sciencedirect.com/science/article/pii/S0005109820304635.

[25] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., USA, 1 edition, 1997. ISBN 0070428077.

[26] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[27] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[28] DeepAI. Feed forward neural network, May 2019. https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network [Accessed: 31.05.2021].

[29] Michael A. Nielsen. Neural networks and deep learning, Dec 2019. http://neuralnetworksanddeeplearning.com/chap2.html[Accessed: 30.05.2021].

[30] Tomasz Szandala. Review and comparison of commonly used activation functions for deep neural networks. *CoRR*, abs/2010.09458, 2020. URL https://arxiv.org/abs/2010.09458.

[31] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016. URL http://arxiv.org/abs/1609.04747.

[32] Lovedeep Gondara. Medical image denoising using convolutional denoising autoencoders. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pages 241–246, 2016. doi: 10.1109/ICDMW.2016.0041.

[33] Lars Ivar Hatledal and Frédéric Collonval. Pythonfmu. https://github.com/NTNU-IHB/PythonFMU, 2019.

[34] Lars Ivar Hatledal. Vico workshop. https://github.com/NTNU-mechlab/vico_workshop, 2021. accessed:[20.02.2021].

[35] Ed van Daalen, J. Cozijn, G. Loussouarn, and P. Hemker. A generic optimization algorithm for the allocation of dp actuators. *Journal of Computational and Applied Mathematics - J COMPUT APPL MATH*, 01 2011. doi: 10.1115/OMAE2011-49116.

[36] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[37] Robert T. McGibbon. quadprog. https://github.com/quadprog/quadprog. accessed:[20.02.2021].

[38] Martin S. Andersen, Joachim Dahl, and Lieven Vandenberghe. *CVXOPT- Python Software for Convex Optimization*. https://cvxopt.org/ [Accessed: 27.04.2021].

[39] Lars Hatledal, Robert Skulstad, Guoyuan Li, Arne Styve, and Houxiang Zhang. Co-simulation as a fundamental technology for twin ships. *Modeling, Identification and Control (MIC)*, 41: 297–311, 12 2020. doi: 10.4173/mic.2020.4.2.

[40] Lars Hatledal, Houxiang Zhang, and Frederic Collonval. Enabling python driven co-simulation models with pythonfmu. pages 235–239, 06 2020. doi: 10.7148/2020-0235.

Appendix

A Vico scenario file

A.1 Scenario file for 4-corner test

```
{
        @file:Repository("https://dl.bintray.com/ntnu-ihb/mvn")
@file:DependsOn("no.ntnu.ihb.vico:core:0.3.3")

import no.ntnu.ihb.vico.dsl.scenario

val tReset = 50.0

scenario {


  invokeAt(49.0){
        bool("vesselModel.reset_position").set(true)

    }
    invokeAt(tReset){
       bool("vesselModel.reset_position").set(false)

    }
  invokeAt(tReset) {
      bool("dpController.enable").set(true)
   }

    invokeAt(tReset) {
   bool("allocator.enable").set(true)
   }

    invokeAt(tReset) {
        real("vesselModel.input_global_wind_vel").set(0)

    }
    invokeAt(tReset) {
        real("vesselModel.input_global_wind_dir").set(0)

    }
    invokeAt(tReset) {
        real("posVelReference.north_wp").set(10)

    }
    invokeAt(300) {
```

```
        real("posVelReference.east_wp").set(-10)

    }

    invokeAt(500) {
        real("posVelReference.psi_wp").set(-45)

    }

    invokeAt(700) {
        real("posVelReference.north_wp").set(0)

    }
    invokeAt(900) {
        real("posVelReference.east_wp").set(0)

    }
    invokeAt(900) {
        real("posVelReference.psi_wp").set(0)

    }

}
}
```

A.2 Scenario file for Stationkeeping test

```
{
@file:Repository("https://dl.bintray.com/ntnu-ihb/mvn")

@file:DependsOn("no.ntnu.ihb.vico:core:0.3.3")
import no.ntnu.ihb.vico.dsl.scenario

val tReset = 50.0

scenario {

  invokeAt(49.0){
        bool("vesselModel.reset_position").set(true)

    }
    invokeAt(tReset){
       bool("vesselModel.reset_position").set(false)

    }
    invokeAt(tReset) {
```

```
    bool("dpController.enable").set(true)
}
invokeAt(tReset) {
    bool("allocator.enable").set(true)
}

 invokeAt(tReset) {
     real("vesselModel.input_global_wind_vel").set(1)

 }
 invokeAt(tReset) {
     real("vesselModel.input_global_wind_dir").set(90)

 }
 invokeAt(60) {
     real("vesselModel.input_global_wind_vel").set(1.5)

 }
 invokeAt(70) {
     real("vesselModel.input_global_wind_vel").set(2)

 }
 invokeAt(80) {
     real("vesselModel.input_global_wind_vel").set(3.5)

 }
 invokeAt(90) {
     real("vesselModel.input_global_wind_vel").set(4)

 }
 invokeAt(100) {
     real("vesselModel.input_global_wind_vel").set(3)

 }
 invokeAt(110) {
     real("vesselModel.input_global_wind_vel").set(5)

 }
 invokeAt(120) {
     real("vesselModel.input_global_wind_vel").set(6)

 }
 invokeAt(130) {
     real("vesselModel.input_global_wind_vel").set(8)

 }
```

```
    invokeAt(150) {
        real("vesselModel.input_global_wind_vel").set(8)

    }
    invokeAt(155) {
        real("vesselModel.input_global_wind_vel").set(8)

    }

    invokeAt(160) {
        real("vesselModel.input_global_wind_vel").set(10)

    }

    invokeAt(401) {
        real("vesselModel.input_global_wind_dir").set(90)

    }

    invokeAt(402) {
       real("vesselModel.input_global_wind_vel").set(10)

    }

   invokeAt(410) {
         real("vesselModel.input_global_wind_dir").set(70)

      }
      invokeAt(420) {
          real("vesselModel.input_global_wind_dir").set(60)

      }
      invokeAt(430) {
         real("vesselModel.input_global_wind_dir").set(50)

      }
      invokeAt(440) {
          real("vesselModel.input_global_wind_dir").set(45)

      }

}
```

A.3 Scenario file for Stationkeeping test with thruster fail

```
{
@file:Repository("https://dl.bintray.com/ntnu-ihb/mvn")

@file:DependsOn("no.ntnu.ihb.vico:core:0.3.3")
import no.ntnu.ihb.vico.dsl.scenario

val tReset = 50.0

scenario {

  invokeAt(49.0){
        bool("vesselModel.reset_position").set(true)

    }
    invokeAt(tReset){
       bool("vesselModel.reset_position").set(false)

    }
    invokeAt(tReset) {
       bool("dpController.enable").set(true)
   }
   invokeAt(tReset) {
       bool("allocator.enable").set(true)
   }

    invokeAt(tReset) {
        real("vesselModel.input_global_wind_vel").set(1)

    }
    invokeAt(tReset) {
        real("vesselModel.input_global_wind_dir").set(90)
    }

    invokeAt(60) {
        real("vesselModel.input_global_wind_vel").set(1.5)
    }

    invokeAt(70) {
        real("vesselModel.input_global_wind_vel").set(2)
        }

    invokeAt(80) {
        real("vesselModel.input_global_wind_vel").set(3.5)
    }
```

```
    invokeAt(90) {
        real("vesselModel.input_global_wind_vel").set(4)
    }
    invokeAt(100) {
        real("vesselModel.input_global_wind_vel").set(4)
    }
    invokeAt(110) {
        real("vesselModel.input_global_wind_vel").set(5)
    }
    invokeAt(120) {
        real("vesselModel.input_global_wind_vel").set(6)
    }

    invokeAt(401) {
        real("vesselModel.input_global_wind_dir").set(90)
    }

    invokeAt(402) {
       real("vesselModel.input_global_wind_vel").set(6)
    }

    invokeAt(700) {
       bool("allocator.thruster_failure").set(true)
    }
}
}
```

B Vico shell script

The example for Vico shell script for running simulation is given below.

```
{
vico simulate-ssp \
-dt "0.05" \
--stopTime "1400" \
-log "logGIGRU.xml" \
-s "scenario_dp_four_corner.main.kts" \
-p "initialValues" \
-res "results4cornerSQP8000N" \
"gunnerus-SQPStationkeeping8000.ssp"
#read -p "Press enter to continue"
}
```

C Test outside co-simulation

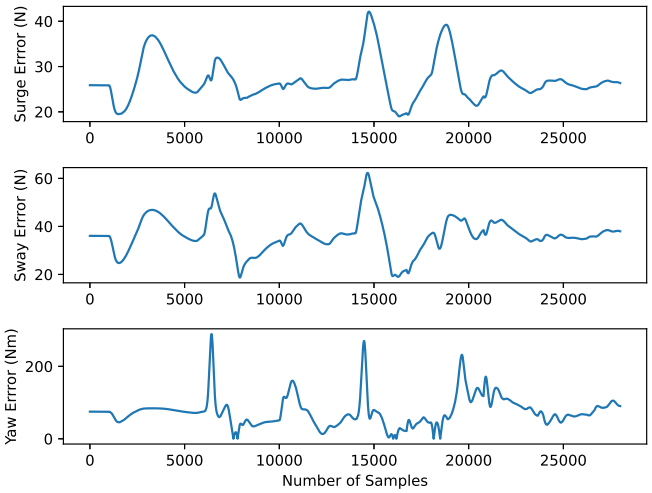


Figure 1: Error estimate in 3DOF

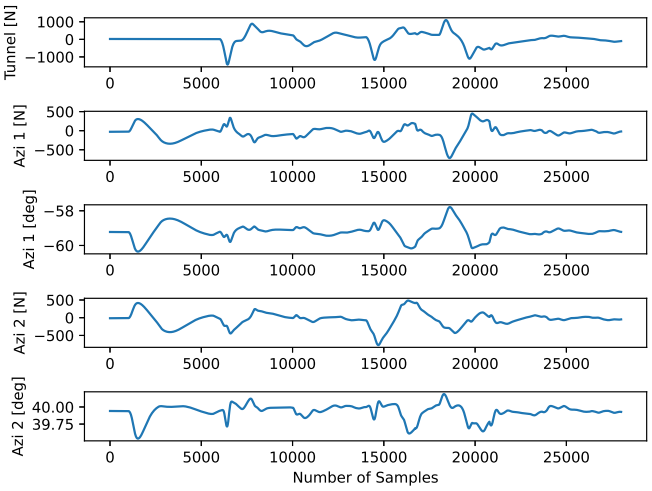


Figure 2: Thruster allocated values

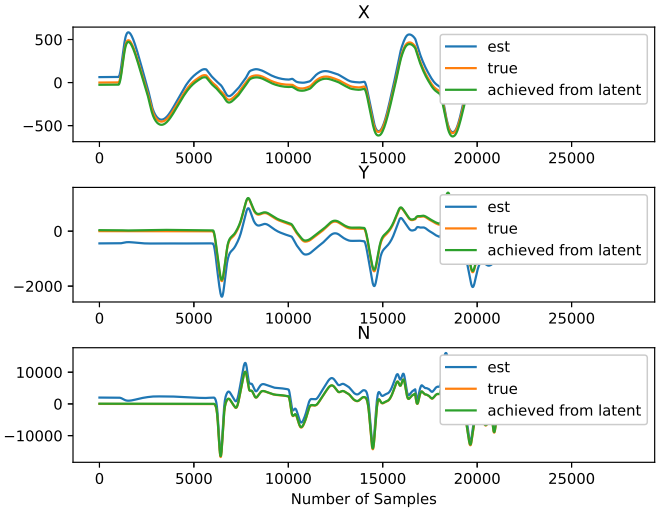


Figure 3: Allocation values in 3DOF

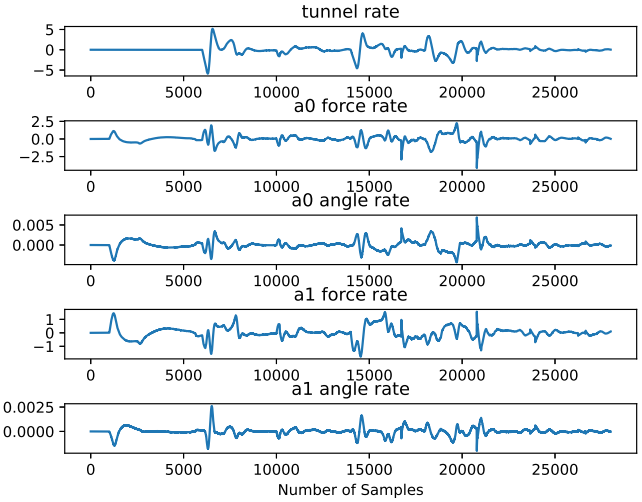
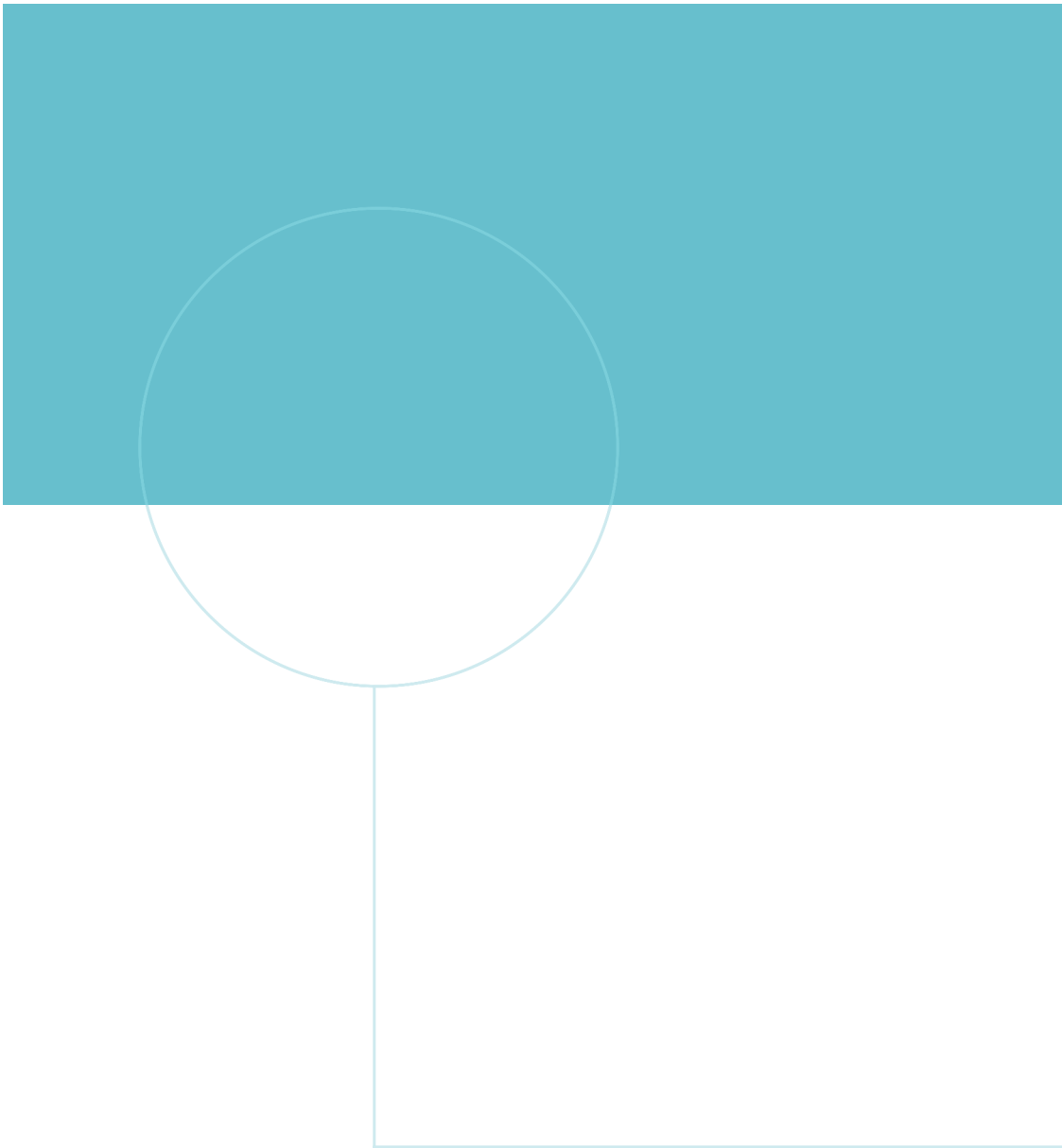


Figure 4: Allocation rate values in 3DOF



NTNU

Norwegian University of
Science and Technology