Design of Constraints for a Neural Network based Thrust Allocator for Dynamic Ship Positioning

Rahul Nath Raghunathan
Department of Ocean Operations and Civil Engineering
Norwegian University of Science and Technology
Ålesund, Norway
rahulnr@stud.ntnu.no

Robert Skulstad
Department of Ocean Operations and Civil Engineering
Norwegian University of Science and Technology
Ålesund, Norway
robert.skulstad@ntnu.no

Guoyuan Li
Department of Ocean Operations and Civil Engineering
Norwegian University of Science and Technology
Ålesund, Norway
guoyuan.li@ntnu.no

Houxiang Zhang
Department of Ocean Operations and Civil Engineering
Norwegian University of Science and Technology
Ålesund, Norway
hozh@ntnu.no

***Abstract*—Thrust allocation (TA) is an important component in dynamic positioning (DP) system of marine vessels. It plays a crucial role in offering power optimisation while meeting physical constraints of actuators in the vessel. Non-linear objective functions in TA formulation poses challenge to deriving the solution of TA schemes. While numerical optimisation techniques have tried to offer solution, the ability of neural network to model non-linearity offers a good technique of solving it. In this paper, design of constraints for TA scheme based on neural network is discussed. The allocator is based on a multi-layered autoencoder network. The allocator thus formulated is tested in various environmental and operational profiles against a numerical optimisation scheme to test its effectiveness in meeting the constraints.**

***Index Terms*—Thrust allocation, Deep learning, Ship motion control**

I. Introduction

The research on the autonomous ship is progressing at a great pace and maritime companies are competing to make products that can be integrated into autonomous ships. Dynamic Positioning (DP) systems of these future vessels are considered to be the core for controlling the stationkeeping and low speed maneuvering of these vessels and thus have been the focus of a high degree of development [1][2]. Since ship DP systems involve propellers, azimuth, and/or bow thrusters working together to produce forces and moments against environmental disturbances, it is necessary to tune them to obtain satisfactory performance.

Most marine vessels have more actuators than their degree of freedom (DOF) under control for DP for the sake of redundancy against failure. The Thrust Allocator (TA) module which is the focus of this paper takes into consideration the desired forces computed by the high-level controller and produces individual control commands to actuators. User-imposed constraints such as rate and saturation limits, minimized power consumption, forbidden zone avoidance, thruster-thruster and thruster-hull interactions can be taken into consideration by the TA module while producing the individual commands.

Research and development to solve the non-linear objective function and constraints in the TA problem have been ongoing for years. Arditti et al. [3] developed a TA method based on the Sequential Quadratic Programming (SQP) technique that takes into account, the hydrodynamic effects such as thruster-hull and thruster-thruster interaction. Veksler et al. [4] developed a Quadratic Programming (QP) based TA scheme that takes into account the load variation in the power plant of a ship due to the unpredictable marine environment. Model Predictive Control (MPC) methods have also been tried to solve TA problem. Zhu et al. [5], develops a TA scheme with fault tolerance using an MPC method. A decoupled TA scheme for inland vessels with a feedback mechanism from TA to controller is achieved by Billet et al. in [6] using an MPC formulation. Evolutionary algorithms[7] has also been used for the TA scheme. For instance, the Differential Evolution algorithm is used by Ding et al. [8] for TA development.

In this work, a Neural Network(NN) approach [9] to solving the TA problem is used as they can approximate non-linear functions efficiently [10]. SQP optimisation scheme have strong local optimisation ability but weak global optimisation ability [8]. Evolutionary algorithms suffer from slow convergence and MPC schemes suffer from computation complexity. NN TA scheme offers real time execution with low complexity. For example, Skulstad et al. employ a supervised NN model for TA for ship DP operation [11]. The allocator is prescribed for a vessel with non-rotatable thrusters and is trained with data of the force request from motion controller and implements saturation and rate constraints. In [10] , Huan et al. introduce a method of solving the control allocation problem of an aircraft using a Deep Autoencoder (DAE) network. The work employs the encoder of DAE as the TA element and decoder to produce input forces at the output, thus enabling to train the network in an unsupervised way.

The saturation limits of actuators are considered in this work. In [12], Skulstad et al. present a TA scheme for DP using DAE network while taking into consideration power optimisation, actuator saturation and rate constraints. The use of special loss function for training is the highlight of this work. In [13], Øvereng et al. employs Reinforcement Learning (RL) to develop a coupled motion controller and TA for DP. Radial Basis Function (RBF) network in combination with SQP is used for TA in [14]. RBF network in the work approximate thruster coefficient leading to improved power efficiency.

In this paper, the attention is given to designing the NN such that the user can add constraints to allocation easily. Analysis of how the thruster constraints are obeyed in allocation is not given importance in many of the above NN TA schemes. The paper tries to improve rate constraints of thrusters compared to work in [12]. Rate constraint addition was one of the future works considered in [10]. To achieve this, current paper tries to combine inputs from [12] and [10] to formulate a custom layer in NN that can handle the thruster saturation and rate constraints while managing forbidden zones.

This paper is organised as follows. Section II introduces 3-DOF TA with details on the numerical approach of TA problem formulation. Section III introduces the new custom layer designed for the allocator to meet the user-defined constraints along with the data generation scheme and loss function used. The use of the allocator is demonstrated through simulation results in section IV. It is followed by concluding remarks in section V.

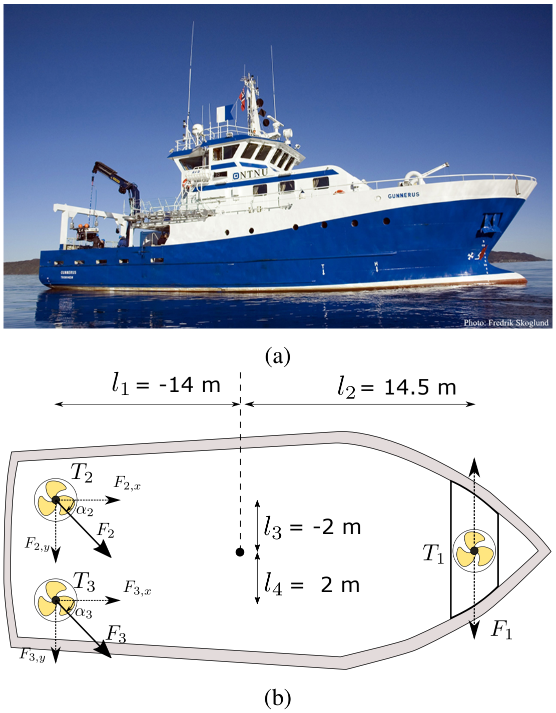
II. Constrained Thrust Allocation

In a 3-DOF DP system of a vessel, motion in surge, sway and yaw are controlled by a high-level position controller. The TA algorithm finds the corresponding forces and direction of thrusters to meet the required thrust vector generated by the position controller[1].

In this paper, the TA model for NTNU's R/V Gunnerus [15] is developed. The thruster layout for Gunnerus is given in Fig. 1b. There are two azimuth thrusters with two control variables (force and angle) and a tunnel thruster with one control variable(force). Hence 5 control variables for TA are prescribed in this paper. They are denoted as follows: u_0 corresponds to tunnel thruster force, u_1 and u_2 corresponds to azimuth thruster 1 force and angle respectively, u_3 and u_4 corresponds to azimuth thruster 2 force and angle respectively. It is assumed that the thrusters can generate the same force magnitude in both directions.

An approach from Johansen et al. [16] is used to present the numerical optimisation scheme in this paper. The cost function for optimisation from them is shown in Eqn. (1) for reading. SQP technique is used to derive the solution to this formulation and the allocator derived is termed SQP allocator.

$$J = \min_{\Delta f,\Delta\alpha,s}\{(f_0+\Delta f)^{\intercal}P(f_0+\Delta f)+s^TQs+\Delta\alpha^T\Omega\Delta\alpha + \frac{\partial}{\partial\alpha}\left(\frac{\rho}{\varepsilon+det(T(\alpha)W^{-1}T^T(\alpha))}\right)\Bigg|_{\alpha=\alpha_0}\Delta\alpha\} \tag{1}$$



(a)

(b)

Fig. 1: Test ship (a) R/V Gunnerus and (b) its thruster layout.

subject to:

$$s+T(\alpha_0)\Delta f+\frac{\partial}{\partial\alpha}(T(\alpha)f)\big|_{\alpha=\alpha_0,f=f_0}\Delta\alpha = \tau - T(\alpha_0)f_0 \tag{2}$$

$$f_{min}-f_0 \leq \Delta f \leq f_{max}-f_0 \tag{3}$$

$$\alpha_{min}-\alpha_0 \leq \Delta\alpha \leq \alpha_{max}-\alpha_0 \tag{4}$$

$$\Delta\alpha_{min} \leq \Delta\alpha \leq \Delta\alpha_{max} \tag{5}$$

The f_0 and α_0 are the control force and azimuth angle from the previous iteration. Δf and $\Delta\alpha$ are change in control forces and azimuth angles. $s\ \epsilon\ \mathbb{R}^n$ are slack variables corresponding to degree of freedom. $P\ \epsilon\ \mathbb{R}^{r\times r}$, $Q\ \epsilon\ \mathbb{R}^{n\times n}$, $\Omega\ \epsilon\ \mathbb{R}^{p\times p}$. $\rho > 0$ is a scalar weight influencing manoeuvrability and power consumption. $\varepsilon > 0$ is a small number to avoid division by zero. $W\ \epsilon\ \mathbb{R}^{r\times r}$ is a positive definite weighting matrix for the control force.

III. Neural Network Thrust allocator with custom layer

A. Custom layer for constraints

To train the network, certain loss functions are used and a custom layer is made to enforce hard constraints for force, angle limits and their rate constraints. A custom layer is the novel contribution of this paper compared to [12].

From Fig. 3, the new NN structure can be visualised. The dense layers have replaced LSTM layers compared to [12]. The new custom layer is a normal dense layer with 5 nodes corresponding to 5 control variables. The custom layer resides before the latent layer to modify inputs for effective constraint

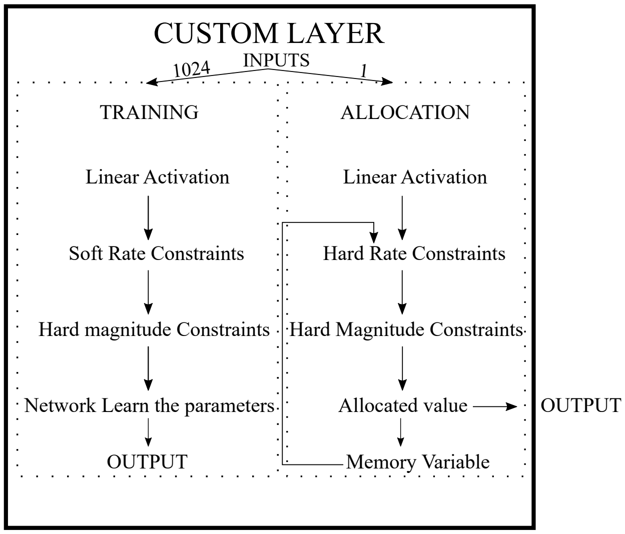


Fig. 2: Custom layer.

handling. The flow of information in the custom layer can be seen in Fig. 2.

In the custom layer, information flow occurs differently during the training phase and allocation phase. During the training phase, the layer imposes hard constraints on saturation values and soft constraints through loss function on rate changes. On the other hand, during allocation, it imposes hard constraint for both saturation and rate changes. This is done to avoid modification of training data during the learning phase of the network.

During training of network, the data is fed into mini-batch size of 1024. Meanwhile during allocation, force commands are requested once at a time and thus input size is one. During training, input(x) first undergoes a linear activation and whenever the layer encounter values for control variables above its min/max saturation values, they are clipped using a generalised hard-tanh transformation([10],[17]) as shown in (6). $\underline{\delta}$ and $\overline{\delta}$ indicates the lower and upper magnitude constraints of 5 control commands. This modification promotes the network to learn to allocate commands within the saturation limits through the latent layer. Soft constraints on rate changes are applied to the network using rate loss prescribed in section III-C.

$$x' = max(\underline{\delta}, min(\overline{\delta}, x)) \tag{6}$$

During allocation (input size =1), the difference in transformation is that the output from a time step is saved in a memory variable. Then at the next time step, maximum allowable rate values are added to this previous value and compared against the current input. They transform similarly to (6) with the difference that the $\underline{\delta}$ and $\overline{\delta}$ are minimum and maximum values after rate changes are added to the previous time step value. Magnitude constraints are also checked during this step.

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 using standardisation. The rate is added and the values are standardised again for comparison against the incoming input values.

B. Data generation scheme

The data used for training the network is the input forces from the motion controller and are synthetically generated. The data generation scheme is similar to [12] with a minor difference in force range. For each thruster, 3 million samples of the randomly generated control values are produced. For rotatable thruster, forces and angles are separately generated.

From Fig. 4, it can be seen that force values are randomly generated in the range $[-10000N, 10000N]$ and angles in the range $[-80°, 80°]$. The angles are plugged in the thrust configuration matrix($T(\alpha)$) and input forces(τ) are generated using the transformation $\tau = T(\alpha) * u$.

The formulated force τ is then fed to the DAE allocator and training is done to try to reproduce the same value at the output($\tilde{\tau}$) while producing commands at the latent space. The data is standardised before being fed to the network. A forbidden zone shown by green color for azimuth thruster is produced by the angle data range as shown in Fig. 5.

C. Specially designed losses

To train the NN, special losses are used following the work in [12]. They are mentioned here for readability.

1) MSE loss between input and output: To ensure that forces given at input(τ) are reproduced at the output ($\tilde{\tau}$), a mean square error between output and input forces is used.

$$L_{in-out} = \sum_{j=1}^{3}(y_j^i - \hat{y}_j^i)^2 \tag{7}$$

In (7), j represents the forces in surge, 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) Power loss (L_{power}): To minimise the magnitude of force commands generated, an exponential loss is designed. The force commands are penalised after being scaled to the original scale after being rescaled from latent space.

$$L_{power} = |u_0^i|^{3/2} + |u_1^i|^{3/2} + |u_3^i|^{3/2} \tag{8}$$

An exponential value of 1.5 is taken considering the fact the power from the thruster can be estimated as $u^{3/2}$ [18].

3) Soft magnitude constraint loss: To ensure that control variables are produced at the latent space, a loss function (9) is formulated.

$$L_{latent} = \frac{1}{n}\sum_{j=1}^{3}|f_j^i - f_{latent,j}^i| \tag{9}$$

In (9), 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. To obtain $f_{latent,j}^i$, the control variable obtained at the latent layer is first rescaled back from standardised form. Then these commands are used to obtain forces and moments in 3-DOF using thrust configuration matrix($\tau = T(\alpha_{rescaled}) \times u_{rescale}$).

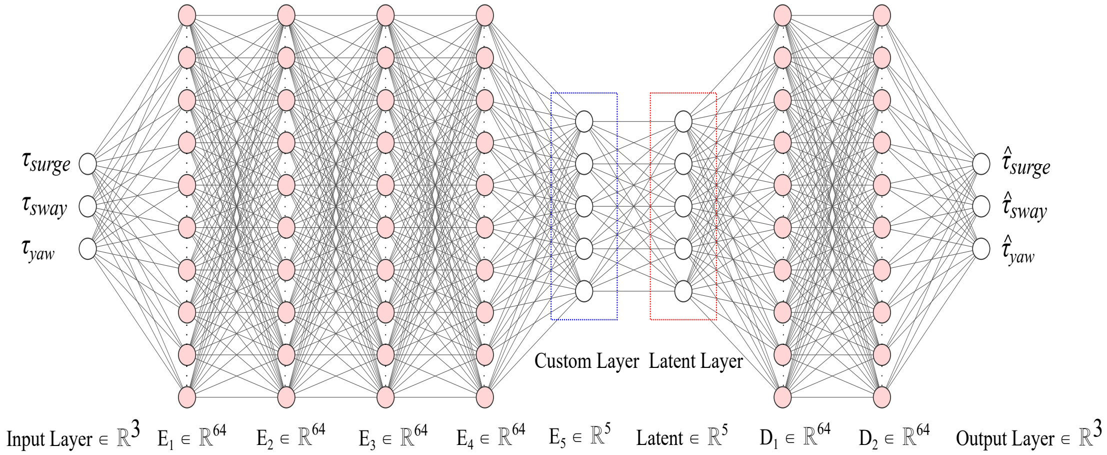

τ_{surge}
τ_{sway}
τ_{yaw}
$\hat{\tau}_{surge}$
$\hat{\tau}_{sway}$
$\hat{\tau}_{yaw}$
Custom Layer
Latent Layer
Input Layer ∈ $\mathbb{R}^3$
$E_1 \in \mathbb{R}^{64}$
$E_2 \in \mathbb{R}^{64}$
$E_3 \in \mathbb{R}^{64}$
$E_4 \in \mathbb{R}^{64}$
$E_5 \in \mathbb{R}^{5}$
Latent ∈ $\mathbb{R}^5$
$D_1 \in \mathbb{R}^{64}$
$D_2 \in \mathbb{R}^{64}$
Output Layer ∈ $\mathbb{R}^3$

Fig. 3: Proposed neural network with custom layer.

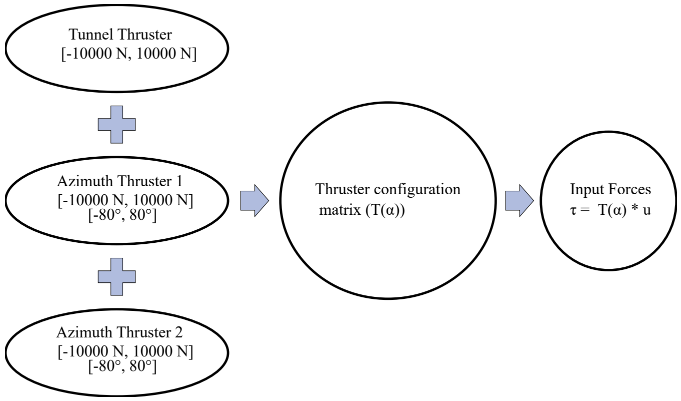


Fig. 4: Data generation scheme.

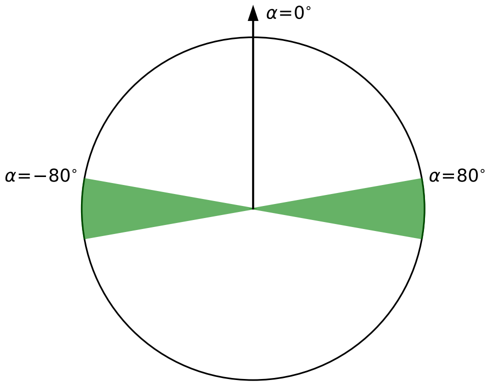


Fig. 5: Forbidden zones for azimuth thrusters.

4) Rate loss: This loss ensures that the generated commands are penalised when they have an excursion above the rate constraints prescribed.

$$L_{rate} = \sum_{l=0}^{4} max(|\hat{u}_l{}^i - shift(\hat{u}_l{}^i)| - \Delta u_{l,max}, 0)^{1.02} \quad (10)$$

The commands obtained at the latent space are rescaled and then the command vector is shifted one step in time to compare the effect of magnitude change between time steps. The exponent 1.02 was found by trial and error during the training and allocation test.

5) Total loss: During the training of the neural network, a total loss is calculated based on previously calculated losses.

$$\begin{aligned} L_{total} = (k_1 * L_{in-out}) + (k_2 * L_{latent}) \\ +(k_3 * L_{power}) + (k_4 * L_{rate}) \end{aligned} \quad (11)$$

Based on prescaling of these individual losses, the performance of the allocator can be tuned. For example, a larger positive scaling for power loss tries to reduce the power consumption by the allocator. Scaling factor used for the study are as follows : $k_1 = 1$, $k_2 = 2$, $k_3 = 1{\times}10^{-7}$, $k_4 = 8{\times}10^{-9}$.

IV. Simulation results

To test and validate the DAE allocator, simulations were conducted for a four corner manoeuvre. The simulation was conducted as co-simulation, such that all components that make up the DP system of the ship are modeled individually and put together in a global simulation where each model gets executed individually and shares information at discrete time steps[19]. In this work, a co-simulation framework developed by researchers at NTNU, Ålesund named Vico [20] was used. In the paper, a Python package named $quadprog$ is used in the SQP allocator.

A. Four corner manoeuvre

A four corner manoeuvre is a standard low-speed manoeuvre that is aimed at checking if the vessel can traverse a square by following the track. To conduct the manoeuvre, setpoints are given by the position reference generator to the DP motion controller which in turn commands the thrust allocator to produce forces and moments. The setpoints for the test are given in Table I. A steady wind of speed 3 m/s and direction of 50° is also used in the simulation.

From the Fig. 6, where motion of the vessel in North-East direction is given, the DAE allocator has a comparable performance relative to the SQP allocator. A reference square

TABLE I: Setpoints for four corner manoeuvre.

Time(s)	Setpoint [North(m),East(m),Yaw(°)]
50	10 m, 0 m, 0°
300	10 m, -10 m, 0°
500	10 m, -10 m, -45°
700	0 m, -10 m, 0°
900	0 m, 0 m, 0°

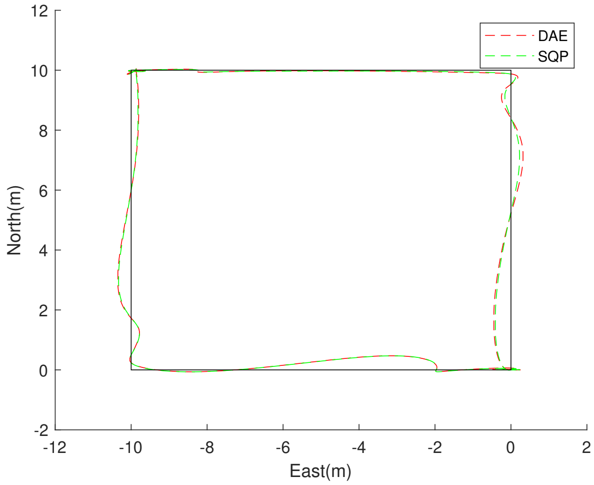


Fig. 6: Vessel motion for DAE and SQP allocator for four corner manoeuvre.

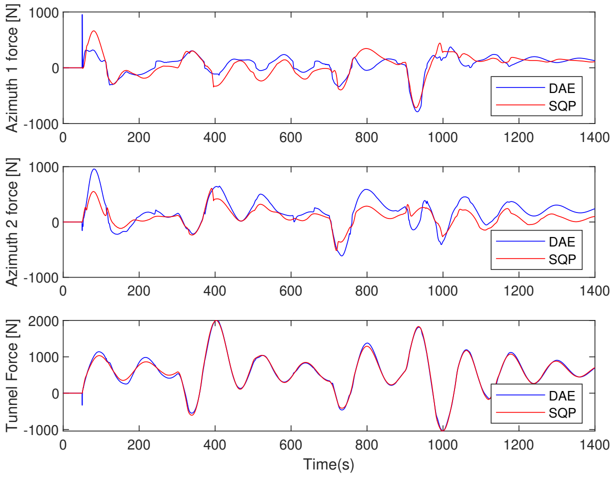


Fig. 7: Forces generated by the two allocators during four corner manoeuvre.

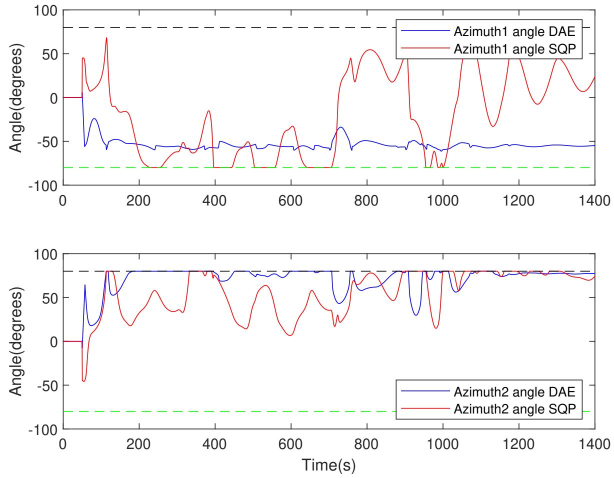


Fig. 8: Azimuth angles generated by the two allocators during four corner manoeuvre.

is drawn to indicate the ideal track the vessel should follow. Both allocators only show slight deviation from the ideal track.

In Fig. 7, the forces generated by the three thrusters are given for the two allocators. The tunnel thruster is used more by both allocators as it can produce the yaw moments which is more important for the four corner manoeuvre. The force constraints are met by both allocators as the set limits were [-8000 N, 8000 N]. The angle value for each allocator is given in Fig. 8. The DAE allocator can be seen to obey angle constraints [-80°, 80°] properly like the SQP allocator. The DAE allocator would have crossed the angle constraints if not for the use of hard constraints in the custom layer design. Like saturation constraints, angle rate constraints are also met by both allocators. In Fig. 9, azimuth angle rates for DAE allocator is given where it obey the limits between [-10°/s, 10°/s]. It can be observed that there are instances where hard constraints prevent the crossing of limits and encourage the network to generate commands below the limit after such occurrences. The SQP allocator also produce angle commands within the rate limits as can be seen in Fig. 10.

The SQP allocator can rotate the azimuth thrusters more effectively compared to the DAE allocator which uses more thrust forces at relatively fewer rotations to provide forces. This contributes to more power consumption for the DAE allocator. The average power consumption by the two allocators during the four manoeuvre is given in Table II. The SQP allocator has a better power consumption characteristic compared to the DAE allocator. Since the DAE allocator has a tuneable power loss factor in its design, a better value could be obtained if thorough tuning is done. This applies to SQP as well.

V. Conclusion

In this paper, a strategy to accommodate actuator constraints for a NN-based TA is presented. A custom layer is designed to meet this requirement. The work has been able to enforce strict saturation handling and rate constraints of the actuators while performing similarly to the SQP allocator. The forbidden zones are also met properly.

Accommodating fault tolerance to allocation is considered a future work since thruster faults leads to untrained force requests from the controller and the allocator is forced to use hard constraints rigorously. The testing of the allocator in a

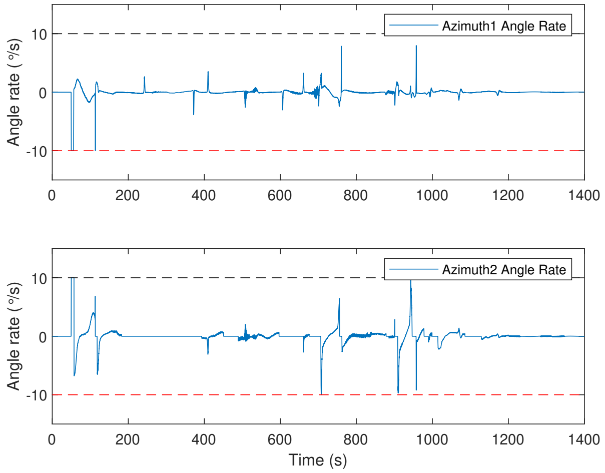


Fig. 9: Angle change rate for DAE allocator.

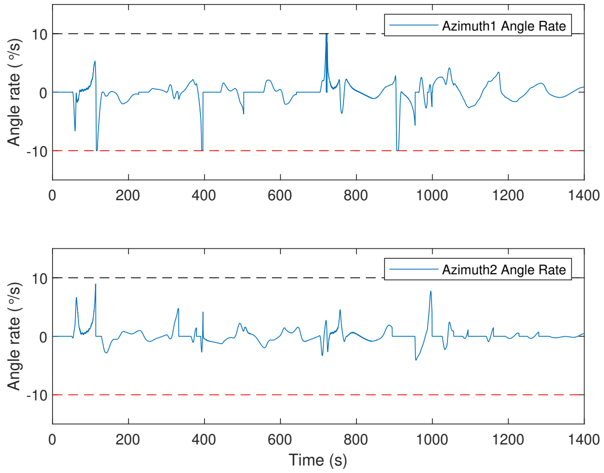


Fig. 10: Angle change rate for SQP allocator.

TABLE II: Average power consumption during four corner manoeuvre.

Allocator	Average power consumption(kW)
DAE Allocator	5.36
SQP Allocator	5.29

physical surface vessel is also considered a future work as this would help to discover the mechanical effects of thrusters on the allocator and controller.

References

[1] Asgeir J. Sørensen. A survey of dynamic positioning control systems. *Annual Reviews in Control*, 35(1):123 – 136, 2011.

[2] Robert Skulstad, Guoyuan Li, Thor I. Fossen, Bjornar Vik, and Houxiang Zhang. Dead reckoning of dynamically positioned ships: Using an efficient recurrent neural network. *IEEE Robotics & Automation Magazine*, 26(3):39–51, 2019.

[3] 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.

[4] 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.

[5] Wei Zhu, Yucheng Wang, Diju Gao, Weifeng Shi, Wanneng Yu, and Yide Wang. A thrust allocation strategy for intelligent ships based on model prediction control. *Transactions of the Institute of Measurement and Control*, 45(9):1693–1702, 2023.

[6] Jef Billet, Paolo Pilozzi, Robrecht Louw, Thibaut Schamp, and Peter Slaets. Model predictive and decoupled thrust allocation for overactuated inland surface vessels. In *2023 European Control Conference (ECC)*, pages 1–6, 2023.

[7] A. E. Eiben and J. E. Smith. *What Is an Evolutionary Algorithm?*, pages 25–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[8] Fuguang Ding, Pengju Gao, Xiaoyun Zhang, and Yuanhui Wang. Thrust allocation of dynamic positioning based on improved differential evolution algorithm. In *2020 39th Chinese Control Conference (CCC)*, pages 1368–1373, 2020.

[9] Gabriel Villarrubia, Juan F. De Paz, Pablo Chamoso, and Fernando De la Prieta. Artificial neural networks used in optimization problems. *Neurocomputing*, 272:10–16, 2018.

[10] 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.

[11] 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. 11th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2018.

[12] Robert Skulstad, Guoyuan Li, Thor I. Fossen, and Houxiang Zhang. Constrained control allocation for dynamic ship positioning using deep neural network. *Ocean Engineering*, 279:114434, 2023.

[13] Simen Sem Øvereng, Dong Trong Nguyen, and Geir Hamre. Dynamic positioning using deep reinforcement learning. *Ocean Engineering*, 235:109433, 2021.

[14] Ziying Tang, Lei Wang, Fan Yi, and Huacheng He. An optimized thrust allocation algorithm for dynamic positioning system based on rbf neural network. In *International Conference on Offshore Mechanics and Arctic Engineering*, volume 84379, page V06AT06A023. American Society of Mechanical Engineers, 2020.

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

[16] 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.

[17] Tomasz Szandala. Review and comparison of commonly used activation functions for deep neural networks. *CoRR*, abs/2010.09458, 2020.

[18] 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.

[19] Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. Co-simulation: State of the art. *CoRR*, abs/1702.00686, 2017.

[20] 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.