# Constrained Control Allocation for Dynamic Ship Positioning using Deep Neural Network

**Article** *in* Ocean Engineering · July 2023

DOI: 10.1016/j.oceaneng.2023.114434



**4 authors:**

Robert Skulstad
Norwegian University of Science and Technology
**41** PUBLICATIONS **562** CITATIONS
SEE PROFILE

Guoyuan Li
Norwegian University of Science and Technology
**159** PUBLICATIONS **1,702** CITATIONS
SEE PROFILE

Thor I. Fossen
Norwegian University of Science and Technology
**436** PUBLICATIONS **32,402** CITATIONS
SEE PROFILE

Houxiang Zhang
Norwegian University of Science and Technology
**325** PUBLICATIONS **4,502** CITATIONS
SEE PROFILE

Ocean Engineering 279 (2023) 114434

Contents lists available at ScienceDirect

Ocean Engineering

journal homepage: www.elsevier.com/locate/oceaneng

# Constrained control allocation for dynamic ship positioning using deep neural network☆

Robert Skulstad [a,*], Guoyuan Li [a], Thor I. Fossen [b], Houxiang Zhang [a]

[a] Department of Ocean Operations and Civil Engineering, Norwegian University of Science and Technology, Larsgårdsvegen 2, 6009 Ålesund, Norway
[b] Department of Engineering Cybernetics, Norwegian University of Science and Technology, O. S. Bragstads Plass 2D, 7034 Trondheim, Norway

ARTICLE INFO

*Keywords:*
Constrained control allocation
Marine robotics
Deep learning methods
Motion control

ABSTRACT

Dynamic positioning (DP) is one of the key technologies towards ship autonomy. Ships in DP mode use thruster devices to maintain position and perform low-speed maneuvering. The motion controller issues force requests according to the measured ship state and motion objectives. These requests must be translated into individual thruster commands. Due to constraints in the thrusters, such as inertia and limited angles of operation, state-of-the-art control allocation methods apply constrained optimization techniques. Although such methods readily capture the handling of constraints, they may require significant computational resources in searching for optimized commands in real time. Here we show that a neural network may be applied to offer an effective evaluation of the mapping between motion controller requests and executable thruster commands. An Autoencoder-like neural network is trained with data generated using knowledge about the configuration of the thrusters. Custom loss functions shape the weights of the network, such that the overall motion objectives and thruster constraints are met. Then, the network is applied to perform low-speed maneuvering and stationkeeping in a simulator. Comparison relative to a state-of-the-art variable angle, constrained control allocator indicate similar dynamic performance with reduced peak power consumption.

## 1. Introduction

Control allocation for normal surface vessels and autonomous ships involves distributing force requests made by a preceding motion controller. The force requests are given relative to the vessel center of gravity and therefore needs to be translated into commands specific to each thrust-producing device, hereafter termed thruster. Typically, ships involved in complex operations have redundant thrusters, which allows for increased safety in case of thruster failures, but also allows for optimizing the overall motion performance in terms of eg. power consumption. Such a thruster configuration renders the ship over-actuated, which implies that more than one set of commands may meet the overall force request. The primary goal of the control allocator is to allocate individual thruster control signals such that the thrusters jointly produce the requested force. Secondary goals include managing the limitations of the thrusters in terms of magnitude and rates, minimization of power consumption, forbidden sectors of operation (rotatable thrusters) and the avoidance of singularities (Johansen and Fossen, 2013).

To solve the constrained control allocation problem for ships, researchers have explored several methods, the most popular one being optimization based methods. Basic approaches to this domain of methods involve posing the problem as an unconstrained optimization problem (Fossen, 2021). This disregards the inherent physical constraints of a thruster, such as force magnitude and its force rate of change. If the thruster can rotate, the maximum/minimum angle and the angle change rate must be considered as well. Thus, this approach may produce infeasible commands, resulting in a failure to meet the force request of the motion controller. Optimization-based approaches have been applied to deal with several challenging problems in control allocation. In terms of maintaining maneuverability when azimuth thrusters that rotate slowly are applied, singularity avoidance is important (Johansen et al., 2004; Sørdalen, 1997). This technique avoids thruster configurations where no force can be produced in one of the ship axes. Load variation reduction on marine power plants (Veksler et al., 2016) and fuel minimization of marine power plants (Rindaroey and Johansen, 2013) have also received focus. This reduces wear and

☆ This work was supported in part by a grant from the Knowledge-Building Project for Industry "Digital Twins for Vessel Life Cycle Service" (Project 280703) and in part by a grant from the Research-Based Innovation "SFI Marine Operation in Virtual Environment" (Project 237929) in Norway. The third author was partially funded by the Norwegian Research Council (NTNU AMOS) at the Norwegian University of Science and Technology (grant no. 223254).

* Corresponding author.
*E-mail addresses:* robert.skulstad@ntnu.no (R. Skulstad), guoyuan.li@ntnu.no (G. Li), thor.fossen@ntnu.no (T.I. Fossen), hozh@ntnu.no (H. Zhang).

https://doi.org/10.1016/j.oceaneng.2023.114434
Received 24 January 2023; Received in revised form 19 March 2023; Accepted 3 April 2023

tear on the power production system, emissions and fuel costs. For ships that have fixed main propellers with rudders, the corresponding feasible thrust vector set is non-convex (Johansen et al., 2008). This is due to the reduced lift force produced by the rudder at negative propeller thrust. Thruster–thruster and thruster–hull hydrodynamic interactions were considered in Arditti et al. (2015), while fault tolerant allocation procedures were described in Witkowska and Śmierzchalski (2018) and Yu and Du (2022).

In an effort to reduce power consumption and rate changes in the face of environment disturbances, Skjong et al. proposed the use of Model Predictive Control (MPC) to optimize control allocation in the long run (Skjong and Pedersen, 2017). The use of the command horizon, inherent to the MPC domain, allowed for both negating the effect of oscillating wave forces as well as to rotate azimuth thrusters to more favorable directions (ie. thrusters that are more efficient at positive Revolutions Per Minute (RPMs)).

Neural networks used for solving control allocation problems have been researched within both the maritime and aerospace domain. In the maritime domain Reinforcement Learning (RL) has recently been applied for adaptive control of DP ship scenarios. A combined model-based tracking control and allocation approach using MPC was investigated in Martinsen et al. (2022). To optimize the controller closed-loop performance, they applied RL and system identification. In Lee et al. (2020) the RL model optimizes the parameters of a 3 degree of freedom PID controller, arguing that an end-to-end RL model for control and allocation would face difficulties in terms of action space dimensionality and complexity in learning.

In the aerospace domain supervised learning in feedforward networks have been investigated. By posing the control allocation problem as a convex nonlinear program, Chen applied a neural network to solve for the control commands for a near-space vehicle (Chen, 2016). A more direct approach was taken by Huan et al. (2018) to create a control allocator for a space re-entry vehicle. A deep autoencoder-like neural network was applied to model the relationship between the controller requests and the resulting commands to actuators. A similar network structure will be used in this paper. Although the type of layer in the network differs and additional features for constraining the allocated commands are added.

Using neural networks for control allocation directly offers a general purpose tool in terms of not imposing bounds on the convexity or linearity of the proposed system. Neural networks can provide allocation solutions at low latency, however the training phase may be time consuming, which suggests offline training. If applied to an autonomous ship, the operating system may invoke the retraining process based on the current operational requirements.

This paper elaborates on how a Deep Neural Network (DNN) can be constructed for control allocation, along with training design considerations and simulations exploring different aspects of the method. Based on knowledge about the thruster configuration and the operating range of the forces of each thruster, training data is generated. Then a model incorporating the above-mentioned constraints is trained offline according to the loss functions described in Section 3.2. A significant feature of such a training regime is that the DNN does not require real data from ship maneuvering operations (Huan et al., 2018).

A related work from the authors is published in Skulstad et al. (2018), in which a one layer feedforward neural network was applied for control allocation. The current paper has the following novelties:

- The model used for allocation consists of recurrent layers, which facilitates constraining rates without feeding thruster commands from a real operation to the network (see Section 3.2.3). This is a novel feature relative to Huan et al. (2018) and Skulstad et al. (2018).
- Implementation of constraints such as power minimization and sector constraints is enabled through the use of the Autoencoder-like structure of the network (see Section 2.2)

## 2. Background

If non-rotatable thrusters are considered, an unconstrained allocation solution may be found using a generalized matrix inverse of the thruster configuration matrix, $\boldsymbol{B}(\boldsymbol{\alpha_0})$, where $\boldsymbol{\alpha}_0$ is a fixed vector of angular displacements for each thruster (Fossen, 2021). To get the allocated thruster forces, $\boldsymbol{f}$, (1) could be used

$$\boldsymbol{f} = \underbrace{\boldsymbol{B}^T(\boldsymbol{B}\boldsymbol{B}^T)^{-1}}_{\mathbf{B}^\dagger}\boldsymbol{\tau} \tag{1}$$

where $\boldsymbol{\tau}$ is the force request from the motion controller. Note that for fixed azimuth angles the generalized-inverse-allocated control forces are optimal in terms of energy (Fossen, 2021). However, the fixed azimuth angles offer sub-optimal thrust directions for the translations and rotations of ship operations. The following sections introduce an established constrained allocation method (Section 2.1) and the proposed approach of this paper (Section 2.2).

### 2.1. Allocation using SQP

A popular constrained allocation method is the Sequential Quadratic Programming (SQP) method described in Johansen et al. (2004). It leverages local approximation of the problem such that the previous solution may be reused. The solution for the current time step is thus a sum of the previous solution plus the relative solution provided by (2). The optimization problem from (Johansen et al., 2004) is re-iterated here for readability.

$$\begin{aligned} J = \min_{\Delta\boldsymbol{f},\Delta\boldsymbol{\alpha},\boldsymbol{s}} \{&(\boldsymbol{f}_0+\Delta\boldsymbol{f})^\intercal \boldsymbol{P}(\boldsymbol{f}_0+\Delta\boldsymbol{f}) + \boldsymbol{s}^T\boldsymbol{Q}\boldsymbol{s} + \Delta\boldsymbol{\alpha}^T\boldsymbol{\Omega}\Delta\boldsymbol{\alpha} + \\ &\frac{\partial}{\partial\boldsymbol{\alpha}}\left(\frac{\varrho}{\varepsilon+\det(\boldsymbol{B}(\boldsymbol{\alpha})\boldsymbol{B}^T(\boldsymbol{\alpha}))}\right)\Big|_{\alpha=\alpha_0}\Delta\boldsymbol{\alpha}\} \end{aligned} \tag{2}$$

subject to:

$$\boldsymbol{s} + \boldsymbol{B}(\boldsymbol{\alpha}_0)\Delta\boldsymbol{f} + \frac{\partial}{\partial\boldsymbol{\alpha}}(\boldsymbol{B}(\boldsymbol{\alpha})\boldsymbol{f})\Big|_{\alpha=\alpha_0,\boldsymbol{f}=\boldsymbol{f}_0}\Delta\boldsymbol{\alpha} = \boldsymbol{\tau} - \boldsymbol{B}(\boldsymbol{\alpha}_0)f_0$$

$$\boldsymbol{f}_{min} - \boldsymbol{f}_0 \leq \Delta\boldsymbol{f} \leq \boldsymbol{f}_{max} - \boldsymbol{f}_0 \tag{3}$$

$$\boldsymbol{\alpha}_{min} - \boldsymbol{\alpha}_0 \leq \Delta\boldsymbol{\alpha} \leq \boldsymbol{\alpha}_{max} - \boldsymbol{\alpha}_0 \tag{4}$$

$$\Delta\boldsymbol{\alpha}_{min} \leq \Delta\boldsymbol{\alpha} \leq \Delta\boldsymbol{\alpha}_{max} \tag{5}$$

- $\boldsymbol{f}_0$ and $\boldsymbol{\alpha}_0$ are the allocated force and azimuth angle vectors from the previous iteration.
- $\Delta\boldsymbol{f}$ and $\Delta\boldsymbol{\alpha}$ are the changes in thruster forces and azimuth angles.
- $\boldsymbol{s}$ is a vector of slack variables.
- $\boldsymbol{P}$, $\boldsymbol{Q}$ and $\boldsymbol{\Omega}$ are weighting matrices.
- $\varrho > 0$ is a scalar maneuverability weighting factor
- $\varepsilon > 0$ is a small scalar value to avoid numerical issues in the fourth term of (2)

This method was used in the simulation example of this study, implemented using the python programming language and the package *qpsolvers*.

### 2.2. DNN

The neural network architecture used in this paper resembles an Autoencoder network. However, Autoencoder networks typically aim to reduce the dimension of the latent space, also known as the Code, to represent the input space samples using a more compact representation (ie. compression of information) (Liu et al., 2017). In this work the structure of the Autoencoder network is exploited to facilitate training and enforce constraints such that the encoder-part of the network in Fig. 1 acts as a control allocator, similar to the work of Huan et al.

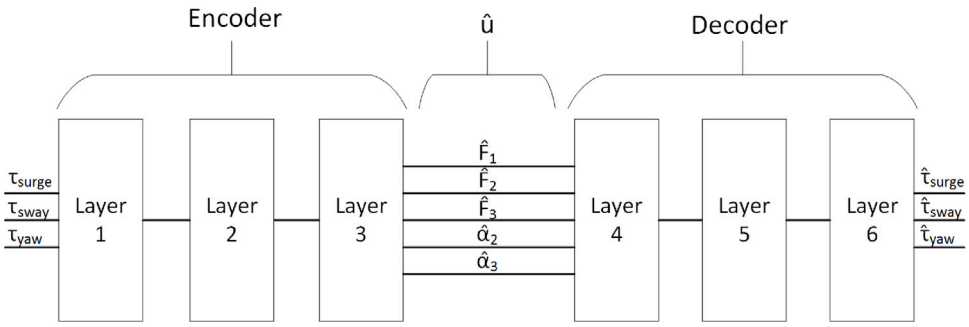



**Fig. 1.** Structure of the neural network used for modeling the control allocation problem. The input to Layer 1 is the force commanded by the motion controller while the output of Layer 3 is the commands issued to each thruster.

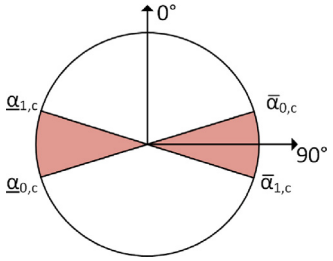



**Fig. 2.** Disallowed sectors of thrusters $T_2$ and $T_3$ are given in red.

(2018). Section 3 describes the dimensions of the input, latent and output spaces.

LSTM nodes were selected based on their ability to retain memory across input sequences (Greff et al., 2017). This functionality allows for applying rate constraints, where there is a need to store temporal information.

## 3. Neural network-based control allocation

The neural network allocator used in this study is in the form of a DNN. Fig. 1 shows the architecture of the network, which interfaces with the motion controller through the force $\boldsymbol{\tau}$ and outputs individual thruster commands through $\hat{\boldsymbol{u}}$. The latter contains force and angle commands while the former contains the force request components along the three dimensions of freedom for the vessel: surge, sway and yaw.

### 3.1. Data

The data used for training the DNN is artificially generated based on a user-defined force and angle command range for $\boldsymbol{u} = [F_1, F_2, \alpha_2, F_3, \alpha_3]^T$ shown in (7). Information about the thruster locations is also used to transform a set of force/angle commands into generalized forces, which is what the allocator receives from the motion controller. In that way, inputs and targets for the supervised training procedure are calculated according to (6), while the entries of $\boldsymbol{u}$ are used in the calculation of losses during training (see Section 3.2).

$$\begin{aligned}\boldsymbol{\tau} &= \begin{bmatrix} 0 & c(\alpha_2) & c(\alpha_3) \\ 1 & s(\alpha_2) & s(\alpha_3) \\ l_2 & l_1 s(\alpha_2) & l_1 s(\alpha_3) \\ & -l_3 c(\alpha_2) & -l_4 c(\alpha_3) \end{bmatrix} \times \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} \\ &= \boldsymbol{B}(\boldsymbol{\alpha})\boldsymbol{u}_f \end{aligned} \tag{6}$$

For the azimuth thrusters a range must be defined for both the force and the azimuth angle, while the tunnel thruster requires only a force range.

There exists a trade-off between the resolution within each range and the corresponding training performance, which hinges on the generalization ability of the DNN. The ranges shown in (7) are based on that trade-off, as well as the size of the resulting dataset and the required range in which the vessel will be operating. The required range of commands depends on the aggressiveness of the motion controller and the maneuvering regime.

An important consideration at this stage is that the ranges and variation observed in the generalized force training data must exceed the constraints applied during training. This applies to the losses described in Sections 3.2.2, 3.2.3 and 3.2.5 as these are only active if the constraints are violated. If a loss function always returns zero, it has no impact on the network training, and therefore does not contribute to learning to keep within these constraints.

The simulation examples in this study require the ship to perform low-speed maneuvering and stationkeeping with moderate wind speeds. This results in low force magnitudes relative to the corresponding maximum force magnitudes of each thruster given in Table 1.

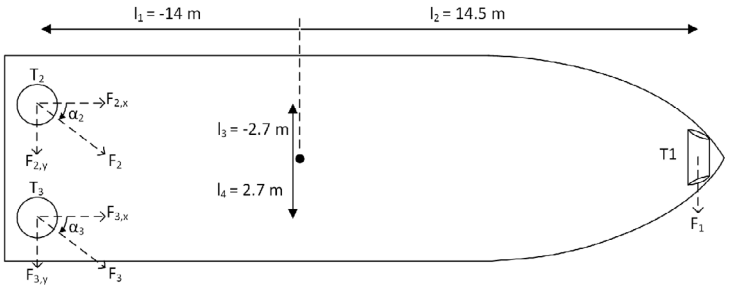
$$\begin{aligned} F_1 &\in [-10000, 10000]N \\ F_2 &\in [-5000, 5000]N \\ \alpha_2 &\in [-180, 180]^\circ \\ F_3 &\in [-5000, 5000]N \\ \alpha_3 &\in [-180, 180]^\circ \end{aligned} \tag{7}$$

$F_1$ is the force of $T_1$, $F_2$ and $\alpha_2$ are the force and thruster angle of $T_2$, and $F_3$ and $\alpha_3$ are the force and thruster angle of $T_3$. Drawing one million samples from these ranges, using a randomly initialized random walk process for each thruster command in (7) per mini-batch, yields the dataset for the commands, $\boldsymbol{u}$. Each mini-batch is thus a sequence of thruster commands that reflects the thruster physical limitations. It has a dimension of $5 \times 1000000$. Applying a continuous sequence of 80% of the columns in $\boldsymbol{u}$ to (6) results in the training dataset, $\boldsymbol{\tau}$, which holds the generalized force. A standardization scaling procedure is performed on the training dataset to facilitate proper training of the DNN.

### 3.2. Training

In order to fulfill constraints in terms of meeting the motion controller force requests, keeping the magnitude and rate of the control vector within certain bounds, and facilitating power minimization, loss functions were designed. The output of the loss functions in Sections 3.2.2–3.2.5 determine the weight updates to each layer in the Encoder-part of Fig. 1 during the backpropagation training scheme. The loss function in Section 3.2.1 applies to the weight update of all layers in the DNN. Loss functions serve the same purpose as objective functions of optimization-based allocation methods. Note that the entire training procedure should be performed offline. In the following $f_e(\boldsymbol{\tau})$ represents the Encoder-part of the network, $f_d(\hat{\boldsymbol{u}})$ represents the Decoder-part of the network while $f(\boldsymbol{\tau})$ is the overall model. For ease


(a) Starboard view of the R/V Gunnerus




(b) Thruster layout of the vessel used in this study

**Fig. 3.** A virtual model of the R/V Gunnerus was applied in this study.

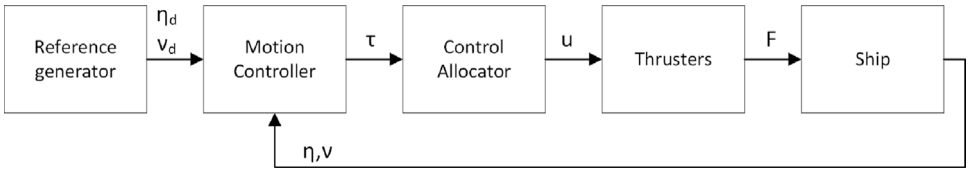



**Fig. 4.** Simulation components.

of notation all variables included in this section are connected to the DNN training process.

A key feature of this training regime is a loss function to ensure that the output of Layer 3 is expanded. Without this feature the neural allocator would not have any incentive to utilize the entirety of the feasible command region, since the weights of Layer 4–6 would absorb the training error. By re-scaling the output of Layer 3 according to the ranges seen in (7), calculating the resulting estimated generalized force and comparing it to the ground truth, the commands are forced into ranges that minimize the requested generalized force (see (9)). First the expression for the generalized force estimate (calculated from the estimated commands $\hat{\boldsymbol{u}}_f(k) = [\hat{F}_1(k), \hat{F}_2(k), \hat{F}_3(k)]^T$ and $\hat{\boldsymbol{\alpha}}(k) = [\hat{\alpha}_2(k), \hat{\alpha}_3(k)]^T$) at mini-batch index $k \in [0, j-1]$, is given in (8). $j = 1024$ is the number of samples in each mini-batch.

$$\hat{\boldsymbol{y}}_{cmd}(k) = \boldsymbol{B}(\hat{\boldsymbol{\alpha}}(k))\hat{\boldsymbol{u}}_f(k) \tag{8}$$

The result of (8) is a $3 \times j$ matrix of estimated surge/sway force and moment about the yaw axis of the ship. $\hat{\boldsymbol{y}}_{cmd}$ is applied in (9) to get the mean squared error (MSE) loss relative to the ground truth $\boldsymbol{y}$.

$$L_0 = \text{MSE}(\boldsymbol{y} - \hat{\boldsymbol{y}}_{cmd}) \tag{9}$$

This loss function does not contribute towards satisfying the constraints imposed on the neural allocator in the training process. Note that the MSE()-function calculates loss on a per-sample basis.

### 3.2.1. Minimize error in allocated force

The primary goal of any control allocation method is to minimize the difference between the motion controller force request and the combined generalized force produced by all thrusters. This goal is enforced through a loss function applied to the output layer of the DNN (Layer 6) seen in (10).

$$L_1 = \text{MSE}(\boldsymbol{y} - \hat{\boldsymbol{y}}) \tag{10}$$

where $\hat{\boldsymbol{y}} = \hat{\boldsymbol{\tau}} = f(\boldsymbol{\tau})$ is the generalized force estimate made by the final layer of the decoder. During training (10) quantifies how well the output of the network matches/reconstructs the generalized force. It offers a way to shape the weights of the DNN towards minimizing the error in allocated generalized force. However, it does not contribute to keep the secondary constraints of the allocator. This will be covered in the following sections.

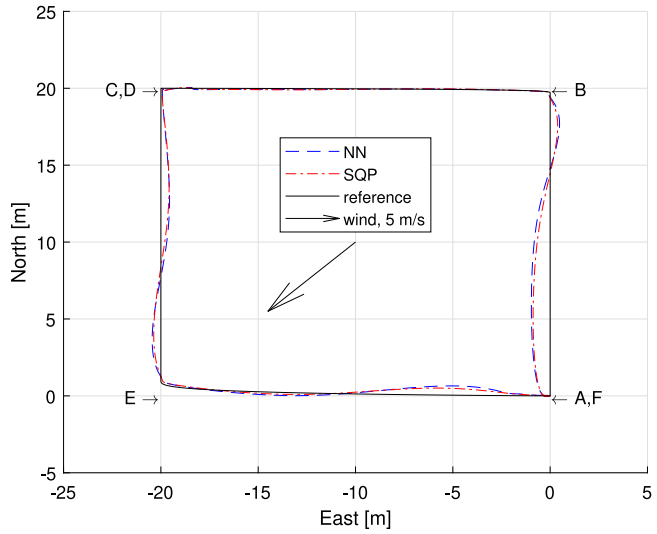



**Fig. 5.** Path taken for both the neural allocator (blue dashed line) and the SQP allocator (red dash–dot line). A uniform wind field of magnitude 5 m/s and direction 45 degrees was applied.

### 3.2.2. Thruster command magnitude

Each thruster has a finite range of operation in terms of force (for thrusters $T_1$, $T_2$, $T_3$) and azimuth angles (for thrusters $T_2$ and $T_3$). From Table 1 the maximum force of $T_2$ and $T_3$ are not symmetrical about zero. However, for simplicity and the limited force range required in the case study, a symmetrical maximum force constraint is applied according to (11)

$$L_2 = \sum_{l=0}^{4} \max((|\hat{\boldsymbol{u}}_l| - \boldsymbol{u}_{l,max}), 0) \tag{11}$$

$L_2$ penalizes commands exceeding the threshold set for each command variable, given in $\boldsymbol{u}_{l,max}$. The max()-function and the |.|-function perform element-wise operations for all five variables in $\boldsymbol{u}$, denoted by the subscripted variable $l$. Note that this results in a soft threshold for the command output and, depending on the relative weighting of the loss functions, an excursion beyond the maximum limit may occur. The same is true for the constraints described in Sections 3.2.3 and 3.2.5. A higher relative weighting is therefore applied to these constraints to keep within the specified constraint thresholds, yielding more conservative allocated commands.

### 3.2.3. Rate changes

For the network to learn to penalize large rate changes the output of the Encoder-part of the network, $\hat{\boldsymbol{u}} = f_e(\boldsymbol{\tau})$, is compared against its copy that has been shifted along the second dimension by one index. This results in a matrix containing the predicted rates of each command variable in $\hat{\boldsymbol{u}}$ within a mini-batch. If the rates exceed a certain value (see Table 1), a loss is incurred according to (12).

$$L_3 = \sum_{l=0}^{4} \max(|\hat{\boldsymbol{u}}_l - \text{shift}(\hat{\boldsymbol{u}}_l)| - \Delta\boldsymbol{u}_{l,max}, 0) \tag{12}$$

### 3.2.4. Power consumption minimization

Without explicitly telling the network to minimize power consumption, ie. penalize the use of thrust, arbitrary values within the allowable ranges of the constraints mentioned in Sections 3.2.2 and 3.2.3 may occur. Therefore, using the fact that the thrust is a quadratic function of the RPM and the consumed power is a cubic function of the RPM (Johansen et al., 2004), the loss function $L_4$ in (13) is applied. The power required to rotate the azimuth thrusters is not considered.

$$L_4 = |\hat{\boldsymbol{u}}_0|^{(3/2)} + |\hat{\boldsymbol{u}}_1|^{(3/2)} + |\hat{\boldsymbol{u}}_3|^{(3/2)} \tag{13}$$

**Table 1**
Parameters of the vessel and thrusters used in the simulation experiment.

| Parameter | Value |
|---|---|
| Deadweight ($10^3$ kg) | 107 |
| Length between perpendiculars (m) | 28.9 |
| Breadth middle (m) | 9.6 |
| Draught (m) | 2.8 |
| $T_1/T_2/T_3$ max N/s | 1000 |
| $T_2/T_3$ max °/s | 10 |
| $T_1$ max N | ±30000 |
| $T_2/T_3$ max N | −30000, 60000 |
| $T_2/T_3$ max ° | ±180 |

**Table 2**
The values of the loss scaling factors $k_0 - k_5$.

| Parameter | Value |
|---|---|
| $k_0$ | $10^0$ |
| $k_1$ | $10^0$ |
| $k_2$ | $10^{-1}$ |
| $k_3$ | $10^{-7}$ |
| $k_4$ | $10^{-7}$ |
| $k_5$ | $10^{-1}$ |

### 3.2.5. Azimuth sectors

Constraining the azimuth angles $\alpha_2$ and $\alpha_3$ is beneficial in terms of avoiding a decrease in thruster efficiency due to disturbance of inflow velocities by neighboring thrusters (Yadav et al., 2014). Two disallowed sectors are defined, since thrusters $T_2$ and $T_3$ may produce both positive and negative thrust. They are: $\underline{\alpha}_c = [\underline{\alpha}_{0,c}, \underline{\alpha}_{1,c}] = [-100, -80]$ deg and $\overline{\alpha}_c = [\overline{\alpha}_{0,c}, \overline{\alpha}_{1,c}] = [80, 100]$ deg. The disallowed sectors are visualized in Fig. 2.

When training the network to avoid these sectors the loss functions in (14) and (15) are applied.

$$\underline{L}_5 = \sum_{l\in[2,4]} (\hat{\boldsymbol{u}}_l < \underline{\alpha}_{1,c}) \times (\hat{\boldsymbol{u}}_l > \underline{\alpha}_{0,c}) \tag{14}$$

$$\overline{L}_5 = \sum_{l\in[2,4]} (\hat{\boldsymbol{u}}_l < \overline{\alpha}_{1,c}) \times (\hat{\boldsymbol{u}}_l > \overline{\alpha}_{0,c}) \tag{15}$$

The index $l \in [2, 4]$ denotes the azimuth angle of $T_2$ and $T_3$, respectively. Note that the $\times$-operator and $\langle,\rangle$-operators perform element-wise operations on $\hat{\boldsymbol{u}}_l$, resulting in per-sample loss values. The $\langle,\rangle$-operators yield a value of 1 if violating the sector constraint, otherwise zero. Note that these sectors could be replaced by efficiency functions that represent hydrodynamic interactions between thrusters if these are known (Arditti et al., 2019). This would be beneficial in terms of smoothness of the returned loss. The overall loss for the two disallowed sectors for $T_2$ and $T_3$ is given by $L_5 = \underline{L}_5 + \overline{L}_5$.

### 3.2.6. Combined loss

A combination of the aforementioned loss functions is used to meet the constraints given in Section 3.2. The combined loss $L = k_0 L_0 + k_1 L_1 + k_2 L_2 + k_3 L_3 + k_4 L_4 + k_5 L_5$ is applied to the network optimizer, which performs the adaptation of the weights in $f_e()$ (due to $L$) and $f(\boldsymbol{\tau})$ (due to $k_1 L_1$). The scaling factors $k_0 - k_5$ weight the impact of each loss such that the performance may be altered depending on user requirements. Re-scaling the output of $f_e(\boldsymbol{\tau})$ in the training procedure calls for small scaling factors in the weighted sum for $L$. Table 2 gives values for the case study presented in Section 4.

### 3.3. Allocation

Once the DNN is trained and validated offline, it may be used to obtain the commands for each thruster. This is done by performing a forward pass over $f_e()$ using the force request from the motion controller as input. The execution time is fixed and depends on the complexity of the neural allocator (number of layers, type of layers, number of nodes in each layer etc.) and the hardware it runs on.

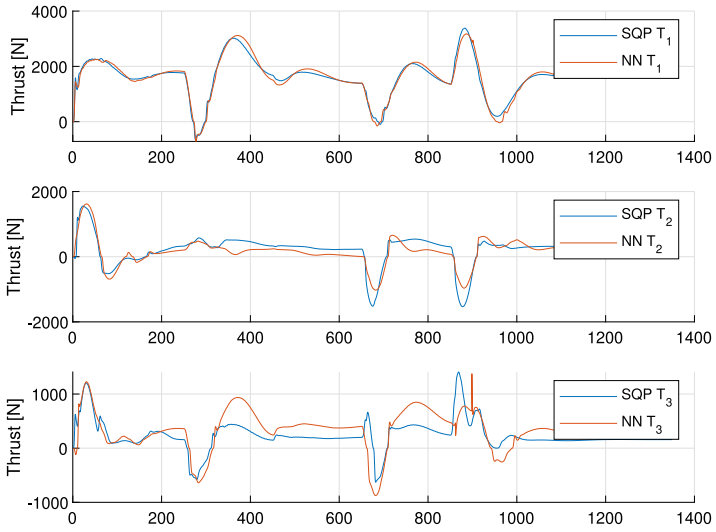



**Fig. 6.** Force commands issued by both allocators.

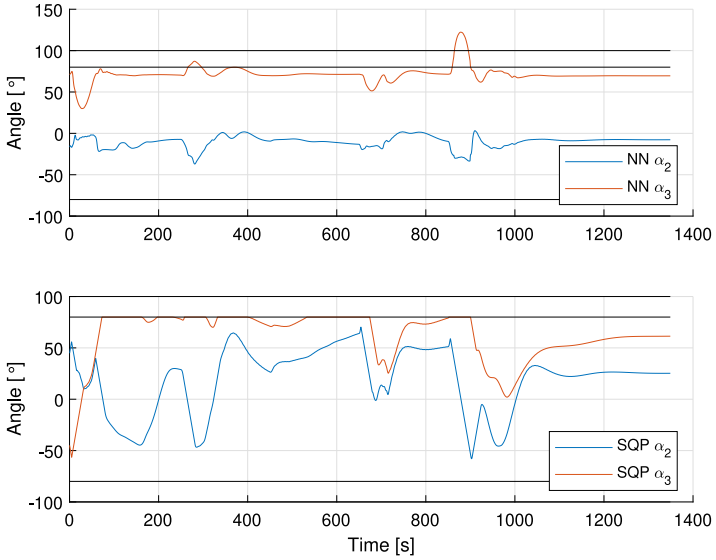



**Fig. 7.** Azimuth angle commands issued by both allocators. The solid black lines at $y = [80, 100]°$ and $y = [-80, -100]°$ show the disallowed sectors used in the case study.

In the event of thruster failure situations, operations that require thrust to be produced in certain directions only, or if operational considerations require less penalization of power consumption, re-training the neural allocator is required. For time-critical events, a set of pre-trained neural allocators may be stored and used on demand (eg. if $T_2$ fails, a neural allocator disregarding this thruster may be applied). Otherwise, the training procedure may be performed online and applied when ready. Using a low-rate graphics processing unit, the training time for the neural allocator used in this study averaged at roughly two minutes.

## 4. Case studies and results

The procedure for constrained control allocation using a DNN, described in Section 3.2, was tested in a simulator. The physical objects included in the simulation were modeled in separate containers, known as functional mock-up units. So were the software required for motion control and allocation.

According to Fig. 3(b), two thruster types appear in the vessel used in this study: One variable-speed tunnel thruster ($T_1$) and two identical, variable speed, azimuth thrusters ($T_2$ and $T_3$). The simulation models used in this study are based on the Research Vessel (R/V) Gunnerus (Hassani et al., 2015) and its propulsion system (Skjong et al., 2018). A view of the real ship is given in Fig. 3(a), and its physical parameters are given in Table 1 (see Fig. 3).

Force commands are issued from the allocator to a Proportional–Integral (PI) controller per thruster. The PI controller outputs a torque reference to an electrical motor based on the error between the estimated thrust and the reference thrust from the allocator. The motor interfaces with the hydrodynamic model of the thruster, which produces thrust, through RPM. In Fig. 4, three PI-motor-thruster subsystems are contained in the *Thrusters* block. Each thruster exerts a 3-dimensional force at its location on the ship hull. The combined thruster force is then applied, along with the additional excitation forces, in the time-domain ship simulator VeSim (Hassani et al., 2015) (see the *Ship* block of Fig. 4).

The two parts of the network; the *Encoder* and the *Decoder* were assigned three layers each based on a manual search considering training/execution time versus performance. Hyperparameters were also manually selected based on the performance of the model when applying a separate validation dataset. 64 Long Short-Term Memory (LSTM) nodes were applied in the first two layers of each part of the network.

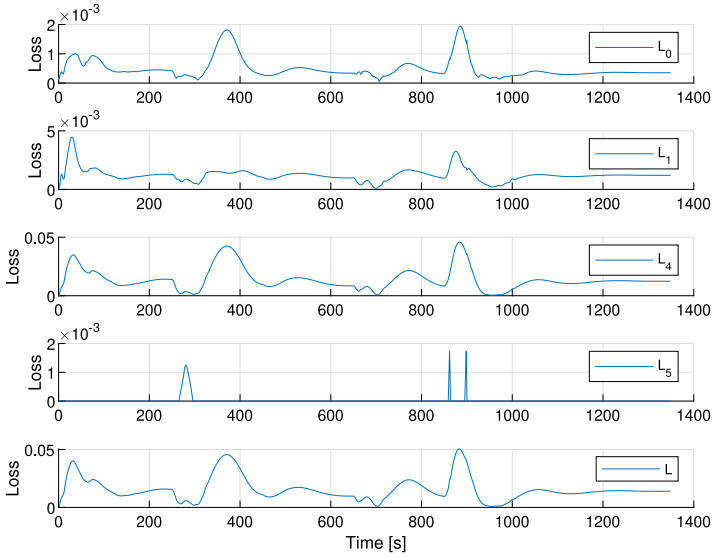



**Fig. 8.** Losses incurred by the neural allocator during the 4-corner test.

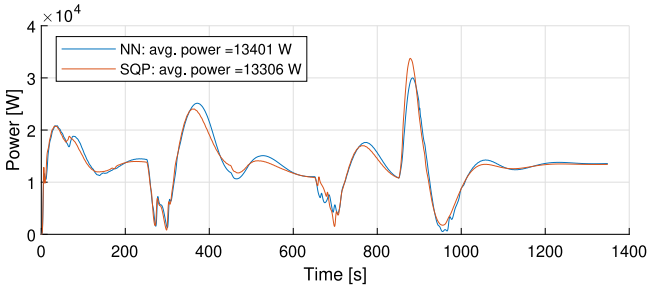



**Fig. 9.** Power consumption of the SQP allocator and the neural allocator.

A regular densely connected layer made up the final layer for both parts of the network, with a dimension of 5 for the *Encoder* and 3 for the *Decoder*. Fig. 4 shows the components of the simulation and how they interact. The motion controller was based on the PID controller described in Sørensen (2011).

### 4.1. Low-speed maneuvering

To create smooth references for the controller during a low-speed maneuvering test, a reference generator was applied, which outputs both position/heading references and their corresponding velocity references. A 4-corner DP test was constructed to gauge the performance of the neural allocator described in Section 3 (Værnø and Brodtkorb, 2017). The outline of the procedure is given below, where the letters of the itemized list correspond to letters in Fig. 5:

- A: Initiate the vessel at a heading of 0 degrees at location (0 m north, 0 m east).
- B: Move 20 m straight north. Start time: 0 s.
- C: Move 20 meters straight west. Start time: 250 s.
- D: While at location (20 m north, 20 m east), rotate the vessel to achieve a heading of 315 degrees. Start time: 450 s.
- E: While maintaining a heading of 315 degrees, move 20 m south. Start time: 650 s.
- F: Move 20 m east while also rotating to a heading of 0 degrees. Start time: 850 s.

#### 4.1.1. Results

Fig. 5 shows the simulated path for the vessel using the SQP allocator and the neural allocator. The simulation was run for both allocators using identical simulation parameters (path references, control parameters etc.) and the simulation time was 1350 s.

Thruster force commands issued by the neural allocator and the SQP allocator are shown in Fig. 6 and angular displacements in Fig. 7. We immediately see that both figures exhibit large changes at the start of each of the moves described in the itemized list in Section 4.1. However, the maximum thrust is well within the range experienced by the neural allocator during training, given in (7). Regarding the SQP-allocated angular displacements, seen in the bottom plot of Fig. 7, they have more variability than its neural counterparts (seen in the upper plot). It also has a natural way of handling constraints through its incremental online optimization scheme. This leads to no violations in the angular displacements of thrusters $T_2$ and $T_3$, while the neural allocator incurs two violations. In the time interval of the violations, the ship is involved in a lateral translation, which means that angular displacements close to $\pm 90$ degrees are efficient.

Fig. 8 shows the resulting losses observed during the 4-corner test for the neural allocator. Notably, the losses $L_2$ and $L_3$ are not included since they remain zero if constraints are not breached. They contribute during the training phase. Here we see that $L_5$ reflects the azimuth angle violations seen in Fig. 7. It is also clear that $L_4$ dominates the other three losses. This suggests that the training procedure has been successful in shaping the network, such that the autoencoder estimates the input well (low $L_1$ value). The relatively low $L_0$ value indicates that the generalized force, achieved through multiplying the allocated commands with the thruster configuration matrix, closely matches the requested generalized force. During training of the neural allocator, adjusting the weighting of $L_4$ results in both altering the power consumption (magnitude of thrust), but also the range and variability of the azimuth angles. The latter comes naturally from penalizing large thrust magnitudes and thus having to find more efficient thrust directions.

Due to maneuvering at low speed the force commands issued by the SQP allocator resulted in force rates within 60% of the maximum values given in Table 1. Similar force rates are seen for the neural allocator (within 40%), while both allocators approached the $10°/s$ limit for azimuth angle rates.

As the training procedure of the neural allocator applies several loss functions (to keep within the constraints mentioned in Section 3.2), the relative weighting between them dictates the allocation performance.

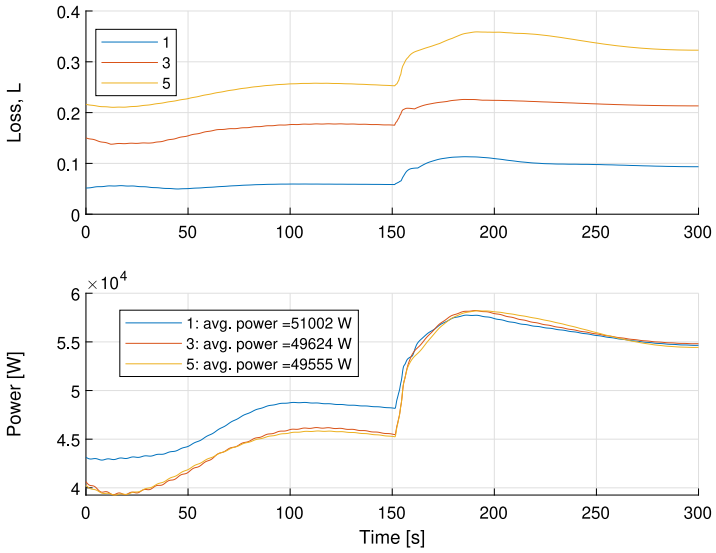



**Fig. 10.** Power consumption (bottom) and overall loss (top) observed during the stationkeeping test. The legend entries indicate: 1: $k_4 = 1 \times 10^{-7}$, 3: $k_4 = 3 \times 10^{-7}$, 5: $k_4 = 5 \times 10^{-7}$.

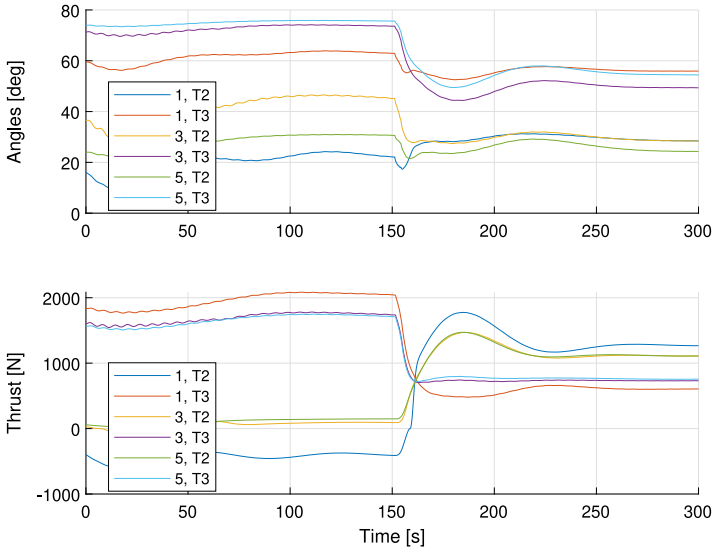



**Fig. 11.** Angles (bottom) and thrust (top) for thrusters observed during the stationkeeping test. The legend entries indicate: 1: $k_4 = 1 \times 10^{-7}$, 3: $k_4 = 3 \times 10^{-7}$, 5: $k_4 = 5 \times 10^{-7}$. $T_2$ and $T_3$ indicate the azimuth thrusters.

In this case, the values of Table 2 were used to scale the individual loss values $L_0 - L_5$.

The average power consumption for the two allocator instances are similar, with an average power decrease of 0.75% using the SQP allocator (see Fig. 9). The largest power reduction is seen in the right after the move from point E to F of Fig. 5 (850–900 s). In this period the SQP allocator makes a substantial azimuth angle change for thruster $T_2$ and simultaneously increases the thrust of all thrusters, which suggests inefficient operation (see Fig. 7). Furthermore, the relatively similar azimuth angles of $T_2$ and $T_3$ for the SQP allocator, although efficient for the present motion controller request, leaves the SQP allocator exposed to temporarily inefficient thrust directions. Fig. 9 shows the larger peak power consumption around 870 s for the SQP allocator.

Constraining the power consumption for the neural allocator effectively led to more variation in the azimuth angles of thrusters $T_2$ and $T_3$. In allowing less use of thrust, the thrusters are forced to find more effective angles to operate. The user must therefore weight the importance of power minimization versus the advantage of having low azimuth angle rate changes. At a certain point, constraining the power output also influences on the ability of the allocator to produce the requested generalized force.

In terms of computational speed the present approach takes 6 ms per execution. To put this in context, the execution time of the MPC algorithm described in Skjong and Pedersen (2017) was reported to be approximately 10 ms. The SQP approach of Johansen et al. (2004), described in Section 2.1 and implemented as a reference allocation method in this case study, had an average computational time of 0.2 ms.

### 4.2. Impact of scaling power loss

The effect of varying the scaling of the loss described in Section 3.2.4 is investigated in this section through a stationkeeping simulation test. The implementation of this loss yields a positive value for all time if the force produced by one of the thrusters is greater than zero. An increase in scaling factor $k_4$, relative to the other scaling factors, implies that the trained network favors lower thrust commands. Scaling factors other than $k_4$ remains equal to that of Table 2 in this test.

Stationkeeping involves the use of thrusters to keep the ship at a fixed location and heading (Skulstad et al., 2019). A constant external disturbance was applied in the form of a uniform wind field coming from the east at a magnitude of 10 m/s. By allowing the wind direction to change quickly (from east to northeast in five seconds), the effect of varying $k_4$ will be examined.

Fig. 10 shows the total loss, $L$, and the overall power consumption. The sensitivity of $L$ with respect to changes in $k_4$ is evident from the substantial increase in $L$ for increasing $k_4$. Increasing $k_4$ also causes azimuth angles to be used more efficiently. This suggests an indirect coupling between $L_4$ and the azimuth angle commands produced by the allocator, since only the thrust is penalized in $L_4$. The top plot of Fig. 11 shows the azimuth angles of $T_3$, for 3 and 5 times the original $k_4$, being closer to the 80 degree constraint. This means that it more efficiently counters the wind disturbance coming from the east. The coupling effect between scaling factors increases the difficulty of finding suitable parameters for $k_0 - k_5$.

## 5. Conclusion

The proposed neural network allocator provides similar functionality as optimization-based methods by accommodating constraints in its training phase. Also, a comparable performance was found relative to the SQP allocator in terms of matching the force request from the motion controller, power consumption and ability to perform low-speed maneuvering. Modeling the neural allocator using only knowledge of the thruster configuration, the user-specified constraints and loss functions with scaling, allows for pre-training of the neural allocator without the need for data from a real operation. A computation time of 6 ms was observed, which enables real-time operation.

The proposed neural network allocator does not enforce constraints in a strict way. This means that, depending on how well the training process is posed, constraints may be breached. For instance, if too little emphasis is put on penalizing azimuth angle rates, a rapid change in the motion controller output might lead to commanding unattainable azimuth angles. Means of enforcing hard constraints will hence be further investigated.

In case of failures, which implies that the neural allocator needs to be re-trained, the training time of the allocator is critical. In future developments of this work, this will be a main focus.

## CRediT authorship contribution statement

**Robert Skulstad:** Conceptualization, Methodology, Software, Writing. **Guoyuan Li:** Experiment, Methodology, Writing – review & editing. **Thor I. Fossen:** Methodology, Review & editing. **Houxiang Zhang:** Methodology, Writing – review & editing.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## References


Arditti, F., Cozijn, H., Van Daalen, E., Tannuri, E.A., 2019. Robust thrust allocation algorithm considering hydrodynamic interactions and actuator physical limitations. J. Mar. Sci. Technol. 24 (4), 1057–1070. http://dx.doi.org/10.1007/s00773-018-0605-8.

Arditti, F., Souza, F.L., Martins, T.C., Tannuri, E.A., 2015. Thrust allocation algorithm with efficiency function dependent on the Azimuth angle of the actuators. Ocean Eng. 105, 206–216. http://dx.doi.org/10.1016/j.oceaneng.2015.06.021.

Chen, M., 2016. Constrained control allocation for overactuated aircraft using a neurodynamic model. IEEE Trans. Syst. Man Cybern. Syst. 46 (12), 1630–1641. http://dx.doi.org/10.1109/TSMC.2015.2505687.

Fossen, T.I., 2021. Handbook of Marine Craft Hydrodynamics and Motion Control, second ed. John Wiley and Sons Ltd., http://dx.doi.org/10.1002/9781119994138.

Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., Schmidhuber, J., 2017. LSTM: A search space Odyssey. IEEE Trans. Neural Netw. Learn. Syst. 28 (10), 2222–2232.

Hassani, V., Ross, A., Selvik, Ø., Fathi, D., Sprenger, F., Berg, T.E., 2015. Time domain simulation model for research vessel Gunnerus. In: International Conference on Ocean, Offshore and Arctic Engineering. pp. 1–6.

Huan, H., Wan, W., We, C., He, Y., 2018. Constrained nonlinear control allocation based on deep auto-encoder neural networks. In: 2018 European Control Conference. ECC 2018, pp. 2081–2088. http://dx.doi.org/10.23919/ECC.2018.8550445.

Johansen, T.A., Fossen, T.I., 2013. Control allocation - a survey. Automatica 49 (5), 1087–1103. http://dx.doi.org/10.1016/j.automatica.2013.01.035.

Johansen, T.A., Fossen, T.I., Berge, S.P., 2004. Constrained nonlinear control allocation with singularity avoidance using sequential quadratic programming. IEEE Trans. Control Syst. Technol. 12 (1), 211–216. http://dx.doi.org/10.1109/TCST.2003.821952.

Johansen, T.A., Fuglseth, T.P., Tøndel, P., Fossen, T.I., 2008. Optimal constrained control allocation in marine surface vessels with rudders. Control Eng. Pract. 16 (4), 457–464. http://dx.doi.org/10.1016/j.conengprac.2007.01.012.

Lee, D., Lee, S.J., Yim, S.C., 2020. Reinforcement learning-based adaptive PID controller for DPS. Ocean Eng. 216, http://dx.doi.org/10.1016/j.oceaneng.2020.108053.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., Alsaadi, F.E., 2017. A survey of deep neural network architectures and their applications. Neurocomputing 234 (November 2016), 11–26. http://dx.doi.org/10.1016/j.neucom.2016.12.038.

Martinsen, A.B., Lekkas, A.M., Gros, S., 2022. Reinforcement learning-based NMPC for tracking control of ASVs: Theory and experiments. Control Eng. Pract. 120, http://dx.doi.org/10.1016/j.conengprac.2021.105024.

Rindaroey, M., Johansen, T.A., 2013. Fuel optimal thrust allocation in dynamic positioning. In: IFAC Conference on Control Applications in Marine Systems. IFAC, pp. 43–48. http://dx.doi.org/10.3182/20130918-4-JP-3022.00032.

Skjong, S., Pedersen, E., 2017. Nonangular MPC-based thrust allocation algorithm for marine vessels - a study of optimal thruster commands. IEEE Trans. Transp. Electrif. 3 (3), 792–807. http://dx.doi.org/10.1109/TTE.2017.2688183.

Skjong, S., Rindarøy, M., Kyllingstad, L.T., Æsøy, V., Pedersen, E., 2018. Virtual prototyping of maritime systems and operations: Applications of distributed co-simulations. J. Mar. Sci. Technol. 23 (4), 835–853. http://dx.doi.org/10.1007/s00773-017-0514-2.

Skulstad, R., Li, G., Fossen, T.I., Vik, B., Zhang, H., 2019. Dead reckoning of dynamically positioned ships: Using an efficient recurrent neural network. IEEE Robot. Autom. Mag. 26 (3), 39–51. http://dx.doi.org/10.1109/MRA.2019.2918125.

Skulstad, R., Li, G., Zhang, H., Fossen, T.I., 2018. A neural network approach to control allocation of ships for dynamic positioning. IFAC-PapersOnLine 51 (29), 128–133. http://dx.doi.org/10.1016/j.ifacol.2018.09.481.

Sørdalen, O.J., 1997. Optimal thrust allocation for marine vessels. Control Eng. Pract. 5 (9), 1223–1231. http://dx.doi.org/10.1016/S0967-0661(97)84361-4.

Sørensen, A.J., 2011. A survey of dynamic positioning control systems. Annu. Rev. Control 35 (1), 123–136. http://dx.doi.org/10.1016/j.arcontrol.2011.03.008.

Værnø, S.A.T., Brodtkorb, A.H., 2017. AMOS DP research cruise 2016: Academic full-scale testing of experimental dnamic positioning control algorithms Onboard R/V Gunnerus. In: Proceedings of the ASME 2017 36th International Conference on Ocean, Offshore and Arctic Engineering. pp. 1–10.

Veksler, A., Johansen, T.A., Skjetne, R., Mathiesen, E., 2016. Thrust allocation with dynamic power consumption modulation for diesel-electric ships. IEEE Trans. Control Syst. Technol. 24 (2), 578–593. http://dx.doi.org/10.1109/TCST.2015.2446940.

Witkowska, A., Śmierzchalski, R., 2018. Adaptive dynamic control allocation for dynamic positioning of marine vessel based on backstepping method and sequential quadratic programming. Ocean Eng. 163 (November 2017), 570–582. http://dx.doi.org/10.1016/j.oceaneng.2018.05.061.

Yadav, P., Kumar, R., Panda, S.K., Chang, C.S., 2014. Optimal thrust allocation for semisubmersible oil rig platforms using improved harmony search algorithm. IEEE J. Ocean. Eng. 39 (3), 526–539. http://dx.doi.org/10.1109/JOE.2013.2270017.

Yu, Z., Du, J., 2022. Constrained fault-tolerant thrust allocation of ship DP system based on a novel quantum-behaved squirrel search algorithm. Ocean Eng. 266, http://dx.doi.org/10.1016/j.oceaneng.2022.112994.