Model Predictive and Decoupled Thrust Allocation for Overactuated Inland Surface Vessels

Jef Billet[1], Paolo Pilozzi[1], Peter Slaets[1], Robrecht Louw[1] and Thibaut Schamp[1]

***Abstract*— This work presents and implements a decoupled Thrust Allocation (TA) algorithm for overactuated inland surface vessels. The utilised control strategy is Nonlinear Model Predictive Control (NMPC), which solves an Optimal Control Problem (OCP) over a control horizon. The controller optimises for minimum energy consumption, subject to the desired responsiveness. We describe the established NMPC formulation as a Nonlinear Programming Problem (NLP) employing the multiple shooting method. Here we use a variety of expressions for nonlinear constraints and saturation models of the propulsion system. Finally, we detail the implementation of the NLP. The resulting implementation has been evaluated on computational load and convergence rate, embedded on a scale model, during on land stationary tests.**

I. INTRODUCTION

Towards more sustainable freight transport, one possibility is to offload road transport to the more sustainable inland waterway transport. To that end, the development of new autonomous platforms is an active field of research. KU Leuven IMP research group currently operates a scale model of a Conférence Européenne des Ministres de Transport (CEMT)-I vessel, named the Cogge [1], to contribute to the automation of inland surface vessels. The Cogge has a bow and stern thruster that can rotate a full circle, akin to azimuth thrust, without a rudder [2]. As such, the Cogge is overactuated [3]. This scale model serves as the test platform in this paper. Automation of these vessels can, by making crew unnecessary, render inland surface vessels an excellent alternative to classic freight transport [4]. In the motion control of such (inland) surface vessels, determining the directions and thrust of the available actuators, given mechanical and other constraints, is needed. This is known as the Thrust Allocation (TA) problem [5].

TA is a critical element in motion control of autonomous vessels, which is applied extensively in Dynamic Positioning (DP) systems [6]. These are present in, for example, offshore drilling units. TA systems have to take actuator constraints into account. To this end, controllers can be tuned not to exceed constraints, or a control method that implicitly incorporates system constraints, such as NMPC, can be used.

A decoupled TA controller is developed using a Nonlinear Model Predictive Control (NMPC) strategy. NMPC allows for using a generic expression of the TA problem without requiring an expansive tuning process compared to other types of controllers [7]. The aim is to simplify the problem of scaling and verification of autonomous systems and move away from the, in this respect, more demanding ad-hoc implementations. NMPC controllers have been used scarcely in implementing TA systems. Examples are the implementations of Veskler et al. [8], Skjong and Pedersen [9], and Jayasiri et al. [10]. A difference between this work and the cited works is the targeted use of the controller. We intend the TA algorithm for use in inland shipping, for vessels that are not mainly stationary. This considerably impacts the types of models and constraints that are necessary, and hence, included. This work also further focuses on a decoupled TA strategy, in contrast to Veskler et al. [8] and Jayasiri et al. [10]. The decoupled TA strategy can further benefit from the ease of implementation and verifiability of a high-level controller without knowledge of the propulsion layout [11]. Therefore, decoupled controllers cater more to physical implementations and prompter advances in inland surface vessel autonomy, provided that performance is adequate. Finally, we use a novel numerical optimisation tool, specifically CASADI [12], combined with IPOPT [13], to solve the nonlinear optimal thrust allocation problem.

The content of this paper is structured as follows: Section II introduces general thrust allocation, and associated propulsion models, after which it outlines the role of TA in such a vessel and its workings. Subsequently, Section III discusses the NMPC controller for the TA problem in length, followed by the controller implementation in Section IV. Then, Section V presents test results for a thrust allocation scenario on the scale model. Finally, Section VI concludes the topics mentioned above and offers possibilities for the continuation of the presented work on the automation of inland surface vessels.

II. THRUST ALLOCATION

We only consider the three horizontal Degrees Of Freedom (DOFs), surge, sway, and yaw, of an inland surface vessel, assuming an earth-fixed inertial system and a body-fixed reference frame attached to the vessel in the Centre of Gravity (CG). The following equation represents the kinematics of such a surface vessel [5]:

$$\dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\boldsymbol{\nu} \tag{1}$$

Where

$$\boldsymbol{\eta} = \begin{bmatrix} x & y & \psi \end{bmatrix}^\top \tag{2}$$

[1]Corresponding author. KU Leuven IMP, Department of Mechanical Engineering, 3000 Leuven, Belgium. E-mail: jef.billet@kuleuven.be

and

$$\boldsymbol{\nu} = \begin{bmatrix} u & v & r \end{bmatrix}^\top \tag{3}$$

Their designations are summarised in Fig. I. $\mathbf{R}(\psi)$ is a transformation matrix that projects the vessel velocities in the local reference frame to the inertial reference frame. We further refer the reader to Fossen [5] for a more comprehensive elaboration on surface vessel dynamics.

TABLE I: 3-DOF notation for positioning and velocities of a surface vessel.

Symbol	Designation
x	Cartesian x-position of the vessel
y	Cartesian y-position of the vessel
ψ	Cartesian heading of the vessel
$\boldsymbol{\eta}$	the 3-DOF positioning vector
u	Surge velocity of the vessel
v	Sway velocity of the vessel
r	Yaw rotational velocity of the vessel
$\boldsymbol{\nu}$	the 3-DOF velocity vector

A general representation of propulsion-created control forces and moments, when neglecting inflow velocities, is given by Fossen [14]:

$$\boldsymbol{\tau_d} = \mathbf{T}(\alpha)\mathbf{f}(\mathbf{n}) \tag{4}$$

where

$$\boldsymbol{\tau_d} = \begin{bmatrix} X & Y & Z \end{bmatrix}^\top \tag{5}$$

Here, X, Y and Z are the desired surge and sway forces, and the yaw moment, respectively. This vector is to be brought about by the use of available actuators. A TA algorithm is responsible for converting the desired force vector $\boldsymbol{\tau_d}$ to thruster-specific outputs or commands F and α. α denotes the angle of a given thruster, whereas F is the force output. Afterwards, thrust mapping has to convert F further RPM commands n, so the complete control outputs $\mathbf{u}(F, \alpha)$ are found. Fig. 1 presents such a control scheme. The high-level controller acquires feedback from the system or surface vessel, with regard to the states $\boldsymbol{\eta}$ or $\boldsymbol{\nu}$, within the loop. The TA does not receive any feedback, and is a standalone open-loop controller. Finally, the outputs F of the thrust allocation need to be mapped to corresponding RPMs first. $\mathbf{T}(\alpha) \in \mathbb{R}^{3xM}$ represents the thruster configuration for an arbitrary amount of thrusters M in 3 DOF. $\mathbf{T}(\alpha)$ is the control effectivity matrix in a TA context. $\mathbf{f}(\mathbf{n}) \in \mathbb{R}^M$ contains the thrust magnitudes per thruster. Here holds that

$$\mathbf{T}(\alpha) = \begin{bmatrix} \mathbf{t_1}, ..., \mathbf{t_M} \end{bmatrix} \tag{6}$$

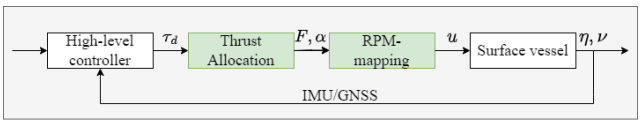

Fig. 1. Block diagram representing the location of TA in a decoupled surface vessel motion controller.

In (6), the columns are defined by

$$\mathbf{t_i}(\alpha_i) = \begin{bmatrix} cos(\alpha_i) \\ sin(\alpha_i) \\ l_{x_i} sin(\alpha_i) - l_{y_i} cos(\alpha_i) \end{bmatrix} for \;\; i = 1, ..., M \tag{7}$$

where l_{x_i} and l_{y_i} are the moment arms for every thruster i with respect to the CG. Equation (4) as an expression for the created control forces is used a lot in DP systems [6]. Dynamically positioned marine vessels commonly use a combination of azimuth and tunnel thrusters, which are used to maintain a fixed position. The CEMT-type scale model Cogge, on the other hand, has more atypical propulsion in this context [1]. Neglecting the dependency on inflow velocities and thrust angles will result in poor thrust allocation. Therefore, for usage in the NMPC-controller, the following model dependencies can increase performance.

$$\boldsymbol{\tau_d} = \mathbf{T}(\boldsymbol{\alpha})\mathbf{F}(\mathbf{n}, \boldsymbol{\alpha}, \boldsymbol{\nu}) \tag{8}$$

Here $\mathbf{F}$ remains dependent on the orientation angles, inflow velocities, and propeller speeds. Note that, the TA algorithm requires correct constraints and models in order to decide the control outputs $F(n)$[1] and α for a thruster i. Consequently, it is required to consider a model for every maximum F_i that depends on $\boldsymbol{\nu}$ and α_i. This is further elaborated in Section II-A.

A. Thruster Saturation Model and Proposed TA Layout

Thrust saturation for applications considered in this paper should not be represented as a linear constraint such as used by Veskler et al. [8]. Thrust saturation varies for each thruster across multiple variables, including α_i, n_i, and the states $\boldsymbol{\nu}$ of the ship, which other implementations neglect [8][15]. Increasing the fidelity of thrust models will also improve MPC-based TA further [15]. We first define

$$V_i = \sqrt{(u + l_{y_i} r)^2 + (v + l_{x_i} r)^2} \tag{9}$$

V_i, which is the velocity over ground of thruster i, according to the body-fixed reference frame. Equation (10)

$$\beta_i = \arctan \frac{v + l_{x_i} r}{u + l_{y_i} r} \tag{10}$$

represents the course over ground of a thruster i. Then, (11) can model the nonlinear behaviour of the thrusters, and adheres to the mentioned dependencies [3] .

$$F = [T_d(\alpha) T_{nn} + cos(\alpha - \beta) T_{nnV} V] \cdot n^2 \tag{11}$$

$T_d(\alpha)$ represents the static thrust reduction at a given angle α. T_{nn} is the quadratic relation between thrust output and the propeller speed n. Finally, T_{nnV} models the thrust reduction due to wake factors, which depends quadratically on n and linearly on V. All variables in (11) also depend on i, which was omitted for brevity. We can use this model to achieve more correctly allocated thrust.

[1]Throughout this work, F is always dependent on n, but we will bypass this notation for conciseness.

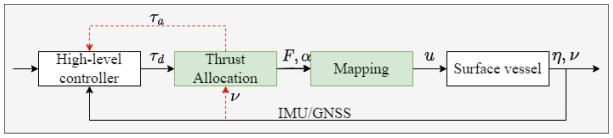

Fig. 2. Block diagram representing TA with the proposed feedback.

Note that, as presented in the motion controller structure in Fig. 1, the TA algorithm has no knowledge of the states $\boldsymbol{\nu}$ of the vessel, nor does the high-level controller know the current saturation limits on the desired thrust $\boldsymbol{\tau_d}$. A well known disadvantage of decoupled TA is the persisting mismatch between the developed thrust and the desired thrust $\boldsymbol{\tau_d}$ [15]. This is an issue directly linked to the decoupled strategy. In order to mitigate this, introducing additional feedback to and from the TA is possible, as Fig. 2 illustrates. In this control structure, the high-level controller receives feedback from the TA, denoted as $\boldsymbol{\tau_a}$, the available or feedback thrust vector. The TA can create this feedback, given its knowledge of the current thruster saturation. The thrust saturation is, in turn, dependent on the new feedback to the TA, which includes all the system states $\boldsymbol{\nu}$. The effect of $\boldsymbol{\tau_a}$ lies outside the scope of this work, however, the influence of the feedback of $\boldsymbol{\nu}$ to the TA is part of the implementation and experimental results in Sections IV and V. The motion controller structure discussed here will serve as a guide throughout the next chapter.

III. NMPC FORMULATION

We define $\mathbf{S} \in \mathbb{R}^{(3+2M)}$ as the states of the controller with

$$\mathbf{S} = \begin{bmatrix} X, & Y, & Z, & F_1, & ..., & F_M, & \alpha_1, & ..., & \alpha_M \end{bmatrix}^\top \tag{12}$$

where F_i and α_i are the thrust and thrust angle respectively of a thruster i. $\mathbf{C} \in \mathbb{R}^{(2M)}$ is the control input vector. Note that the only directly controlled[2] states are X, Y, and Z.

$$\mathbf{C} = \begin{bmatrix} \phi_1, & ..., & \phi_M, & \omega_1, & ..., & \omega_M \end{bmatrix}^\top \tag{13}$$

Here ϕ_i is the thrust rate and ω_i the angle rate for thruster i. Because the thrusters can rotate this problem is non-convex [14] and numerically hard. NMPC controllers are typically implemented discretely due to the computational complexity. As such, we define the horizon N, based on the sampling time, as follows

$$N = \frac{t_{tot}}{t_{step}} \tag{14}$$

t_{tot} signifies the total horizon time frame, t_{step} the time step per MPC iteration. Now

$$k = 1, ..., N \tag{15}$$

is the discrete time in the horizon. Before the MPC formulation can be provided, some further definitions of the states, constraints, and objective function are required.

[2]The only states that are regarded as relevant inputs in the TA algorithm.

A. States and Inputs

As seen in (12), both the thrust vector components and the thrust inputs are states in the system. When considering an M number of thrusters, the global thrust vector, is found as follows at discrete step k cfr. Skjong and Pedersen [9]:

$$\begin{aligned} X(k) &= \sum_{i=1}^{M} F_i(k) cos(\alpha_i(k)) \\ Y(k) &= \sum_{i=1}^{M} F_i(k) sin(\alpha_i(k)) \\ Z(k) &= \sum_{i=1}^{M} [F_i(k) cos(\alpha_i(k)) l_{y_i} - F_i sin(\alpha_i(k)) l_{x_i}] \end{aligned} \tag{16}$$

As presented, the created thrust vector only depends on the other states in the states vector $\mathbf{S}$. The remaining states are purely dependent on the controls $\mathbf{C}$. Consequently, we define the following discrete-time relationship:

$$\begin{bmatrix} F_1(k+1) \\ \vdots \\ F_M(k+1) \\ \alpha_1(k+1) \\ \vdots \\ \alpha_M(k+1) \end{bmatrix} = \begin{bmatrix} F_1(k) + \phi_1(k) \cdot t_{step} \\ \vdots \\ F_M(k) + \phi_M(k) \cdot t_{step} \\ \alpha_1(k) + \omega_1(k) \cdot t_{step} \\ \vdots \\ \alpha_M(k) + \omega_M(k) \cdot t_{step} \end{bmatrix} \tag{17}$$

Now, a numerical method is required to convert the above Differential-Algebraic system of Equations (DAE) into constraints for an NLP.

B. Direct Multiple Shooting and Equality Constraints

Direct multiple shooting is a method for transcribing an Optimal Control Problem (OCP) to an NLP where both the states and the controls are considered decision parameters in the NLP. This method is often considered superior to other methods, such as direct single shooting, where only the controls are decision parameters [16]. Because both the states and the controls are decision parameters, the boundary nonlinearity does not propagate throughout the horizon. Instead, at every iteration in the horizon a new Initial Value Problem (IVP) is solved. In order to build the NLP with multiple shooting, we implement the following equality constraints:

$$F_i(k+1) - h(F_i(k), \phi_i(k), t_{step}) = 0 \tag{18}$$

and

$$\alpha_i(k+1) - h(\alpha_i(k), \omega_i(k), t_{step}) = 0 \tag{19}$$

These are necessary in order to constrain the freedom of choice the solver has in choosing the states at each discrete k. Equations (18) and (19) use a fourth-order Runge-Kutta method as the DAE solver h. X, Y, and Z are also states and have to be constrained as well. Referring to (16), we use

the following constraints:

$$\begin{aligned} X_c(k) - \sum_{i=1}^{M} F_i(k)cos(\alpha_i(k)) = 0 \\ Y_c(k) - \sum_{i=1}^{M} F_i(k)sin(\alpha_i(k)) = 0 \\ Z_c(k) - \sum_{i=1}^{M} [F_i(k)cos(\alpha_i(k))l_{y_i} \\ -F_i(k)sin(\alpha_i(k))l_{x_i}] = 0 \end{aligned} \tag{20}$$

At every discrete time in the horizon, the NLP solver will not be able to deviate from the above reliance. Also adding initial state conditions, leads to a final combination of equality constraints:

$$\mathbf{g}_i(k) = 0 \tag{21}$$

where

$$\mathbf{g}_i(k) = \begin{bmatrix} X_c(k) - \sum_{i=1}^{M} F_i(k)cos(\alpha_i(k)) \\ Y_c(k) - [\sum_{i=1}^{M} F_i(k)sin(\alpha_i(k))] \\ Z_c(k) - \sum_{i=1}^{M} [F_i(k)cos(\alpha_i(k))l_{y_i} - \\ F_i(k)sin(\alpha_i(k))l_{x_i}] \\ F_i(k+1) - h(F_i(k), \phi_i(k), t_{step}) \\ \alpha_i(k+1) - h(\alpha_i(k), \omega_i(k), t_{step}) \\ F_i(1) - F_{i_1} \\ \alpha_i(1) - \alpha_{i_1} \end{bmatrix} \tag{22}$$

C. Thruster Saturation and Inequality Constraints

Section II-A presented a model for thrust saturation, which, to implement into the NLP, should be restated as an (in)equality constraint, similar to Veskler [17]. In this case, however, the constraint is not linear and depends on both the vessel and the propulsion states. We also know that the current and estimated values for n are not a part of the decision-making in the TA. In order to enable the allocation of only thrust that is feasible, based only on the available states, as defined in (12), the model has to be restated as an inequality constraint such as

$$\begin{aligned} F(k) \leq [T_d(\alpha(k))T_{nn} \\ +cos[\alpha(k) - \beta(k)]T_{nnV}V(k)] \cdot n_{max}^2 \end{aligned} \tag{23}$$

where n_{max} is the maximum available RPM and thus constant. Note that all the variables in (23) are thruster-specific. This constraint will force the MPC controller to never allocate a thrust higher than the model-based available thrust at the maximum RPM, throughout the horizon. It is softer than an equality constraint, yet has the same result. It guarantees that the mapping step as presented in Fig. 2 will find a corresponding RPM. For clarity we define

$$\boldsymbol{\gamma}_i(k) \leq 0 \tag{24}$$

where

$$\begin{aligned} \boldsymbol{\gamma}_i(k) = F_i(k) - [T_{d_i}\alpha_i(k)T_{nn_i} \\ +cos[\alpha_i(k) - \beta_i(k)]T_{nnV_i}V_i(k)] \cdot n_{max_i}(k)^2 \end{aligned} \tag{25}$$

D. Rate Constraints

As we chose the control rates as inputs, rate constraints are now arbitrary to implement, the offside being that problem becomes numerically harder to solve [18]. There are other methods available [19] which, while favourable in terms of computational efficiency, are more cumbersome to implement. In a generalised form, the rate constraints for every thruster i, are given by:

$$\underline{\mathbf{C}_{j,1}} \leq \mathbf{C}_{j,1}(k) \leq \overline{\mathbf{C}_{j,1}} \qquad \forall_j \in [1, ..., 2M] \tag{26}$$

E. Cost Function

Finally, the cost function is derived from the one proposed by Johansen et al. [20]:

$$J(\mathbf{C}, \mathbf{S}, \mathbf{S_r}, k) \tag{27}$$

$$= \sum_{k=1}^{N} [\sum_{i=1}^{M} \mathbf{W}_i (F_i(k))^2] \tag{27a}$$

$$+ \sum_{k=1}^{N} [\mathbf{S}(k) - \mathbf{S_r}(k)]^T \mathbf{Q} [\mathbf{S}(k) - \mathbf{S_r}(k)] \tag{27b}$$

$$+ \sum_{k=1}^{N} \boldsymbol{\omega}(k)^T \boldsymbol{\Omega} \boldsymbol{\omega}(k) \tag{27c}$$

$$+ \sum_{k=1}^{N} [\sum_{i=1}^{M} \frac{\rho}{\epsilon + det[\mathbf{T}(\alpha_i(k))\mathbf{Z}^{-1}\mathbf{T}^T(\alpha_i(k))]}] \tag{27d}$$

Equation (27a) is a cost on thruster-specific power consumption, with $\mathbf{W}_i \in \mathbb{R}^+$ being weights, and $F_i(k)$ the thruster-specific force. $F_i(k)$ is squared, unlike in the original cost function [20], to reduce nonlinearity.

The second term, in (27b), is a quadratic cost function representing the main objective. The objective is to minimise the difference between the reference and current states. Here, k is the sampling number as defined in (17), $\mathbf{S}$ is the states vector and $\mathbf{Q} \in \mathbb{R}^{(3+2M)\times(3+2M)}$ is a diagonal weights matrix, which specifies the cost weight of each respective state. The subscript r denotes the reference states. Note that there is no reference for the states $\mathbf{F}$ and $\boldsymbol{\alpha}$. Although these are considered states[3] in the system, they are not to be minimised compared to a reference state. Hence, weights the matrix $\mathbf{Q}$ only has weights Q_1, Q_2 and Q_3 on the first three diagonal elements, for thrust components X, Y, and Z respectively. The other diagonals remain zero.

Equation (27c) presents a cost on the rotation of thrusters. Here, $\boldsymbol{\omega}$ contains the angular velocity control inputs and

[3]They are part of $\mathbf{S}$.

$\mathbf{\Omega} \in \mathbb{R}^{(M)\times(M)}$ is another diagonal weight matrix in which the values serve to tune the rotational behaviour of individual thrusters.

A final term, (27d), serves to avoid solver singularity. Singularities occur when the layout matrix $\mathbf{T}(\alpha)$ does not have a full rank. In this case, because the system is over-actuated, the matrix is full rank. However, for some combinations of α_i, singularities can still ensue [20]. In order to mitigate singularities in the solution we built, the singularity avoiding part of the cost function is kept. Section III-F outlines the complete MPC formulation.

F. Complete NMPC Formulation

Combining the states, control parameters, constraints, and cost function defined in all the previous Sections, we can present one general NMPC problem.

$$\begin{aligned} \min_{\mathbf{C}} \quad & J(\mathbf{C}, \mathbf{S}, \mathbf{S_r}, k) \\ s.t. \quad & \forall_i \in [1, ..., M], \quad \forall_k \in [1, ..., N] \\ & and \ \forall_j \in [1,..., 2M] \\ & \mathbf{g}_i(k) = 0 \\ & \boldsymbol{\gamma}_i(k) \leq 0 \\ & \underline{\mathbf{C}_{j,1}} \leq \mathbf{C}_{j,1}(k) \leq \overline{\mathbf{C}_{j,1}} \end{aligned} \tag{28}$$

IV. NMPC IMPLEMENTATION AND EXPERIMENT SETUP

This work employs CASADI [12] in PYTHON to implement the NMPC controller, shown in (28), using the IPOPT [13] solver. CASADI provides a symbolic abstraction layer, and access to multiple third-party solvers. It utilises complete symbolic NLP expressions and substitutes the symbolics for proper values at each controller iteration of the MPC, hence increasing readability and adaptibility of the implementation. No direct interactions with the underlying solvers are required, beyond specifying a number of options.

The Cogge [1] serves as the test platform. It has an onboard MC-7200-MP-T Series maritime computer with an Intel Core i7-3555LE CPU @ 2.50GHz on which the NMPC controller was implemented in PYTHON. The actuator models and constraints were implemented in the controller as described in Sections II-A and III. Fig. II presents the relevant parameters for the thruster models of the Cogge. Note that, the thrust deductions as shown in (11), are not constant nor linear over α [3], for both the bow and stern thruster of the Cogge. As such, we modelled $T_d(\alpha)_1$, and $T_d(\alpha)_2$ using trigonometric Fourier series approximations with a period of 2π radians. The tuning of the weights was chosen for optimal reference tracking, rather than energy consumption, see Fig. III.

TABLE II: Thruster model parameters for the Cogge.

Parameter	Bow thruster	Stern thruster
n_{max}	2600 RPM	1700 RPM
T_{nn}	$7.906 \cdot 10^{-6}$	$6.52 \cdot 10^{-5}$
T_{nnV}	$-3.40 \cdot 10^{-6}$	$-2.46 \cdot 10^{-5}$
l_x	2.61 m	-0.63 m
l_y	0 m	0 m
ϕ_{max}	2.055 N/s	4.34 N/s
ω_{max}	0.698 rad/s	0.419 rad/s

To evaluate the controller, we performed a semi-virtual pure sway propulsion manoeuvre during an on land stationary run. The sampling time t_{step} of the controller was set to be one second, with a horizon N of 30. During the test we requested the required force vector, (29), which, on the basis of the models of the Cogge, would effect a 0.2 meters per second of sway velocity without any accompanying surge or yaw.

$$\boldsymbol{\tau_d} = \begin{bmatrix} 0\text{N} & -20.96\text{N} & -20.04\text{Nm} \end{bmatrix}^\top \tag{29}$$

The initial states are

$$\mathbf{S_0} = \begin{bmatrix} 0, & 0, & 0, & 0, & 0, & 0, & 0 \end{bmatrix}^\top \tag{30}$$

In this manner, we evaluate the effects of the implemented models and constraints on the mechanical system as well as embedded computational performance and stability of the decoupled system. The combined motion control performance with an external high-level motion controller is beyond the scope of this work. Section V details the results of the semi-virtual pure sway manoeuvre.

V. EXPERIMENTAL RESULTS

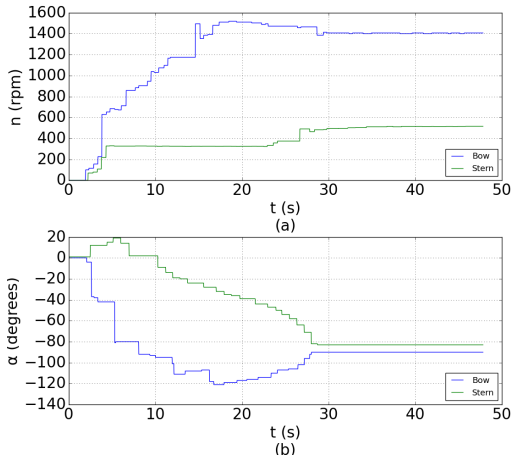

Fig. 3. TA using NMPC control. (a) Rpm and time plot of mechanical propeller speeds of bow and stern thrusters during the NMPC controlled pure sway TA manoeuvre. (b) Angle and time plot of the angles of bow and stern thrusters during the NMPC controlled pure sway TA manoeuvre.

The model-based values for the thrust components on the surface vessel contain substantial fluctuation because of a combination of the nonlinear propulsion models and sensor noise. A rolling average is included to present a basic

TABLE III: Weight values for the NMPC controller

Reference		Thrusters		Singularity	
Q_1	10000	Ω_1	1	ρ	0.001
Q_2	1000	W_1	50	ϵ	100
Q_3	10000	Ω_1	1		
		W_2	50		

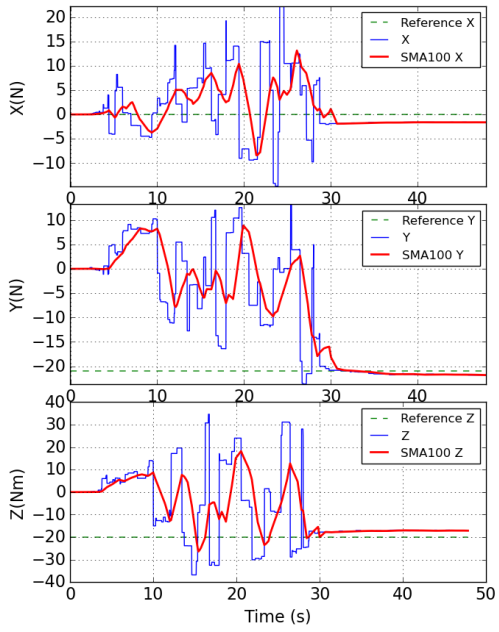

Fig. 4. Model based thrust component feedback to the NMPC controller, with a simple moving average estimation.

estimation for the real progress of the thrust components. Despite the fluctuations, the NMPC controller converges to a stable solution. One can observe that, in Fig. 3 the initial propeller acceleration is high but slows down, as the angles of the thrusters change more slowly. Fig. 4 shows that the initial Y and Z thrust components decrease, yet as the surge deviates from zero, is slowed down to compensate. Once the thrusters are rotated to the optimal angle, steady state is achieved.

The computational load averaged on 12% single core usage, with an average iteration time of 0.015 seconds, which is well below real-time.

VI. CONCLUSION

This paper presents and implements a decoupled NMPC TA algorithm for overactuated inland surface vessels, which considers thrust saturation and rate constraints, and minimises energy consumption, subject to responsiveness. The NMPC implementation employs CasADi with the nonlinear solver IPOPT. The nonlinear cost function and constraints are transcribed to an NLP using a multiple-shooting method. The result is an implementation suitable for real-time use on hardware found in surface vessels. Tests were performed on a scale model of a CEMT-I ship to evaluate convergence and computational load in an embedded environment. The experiments achieve real-time performance, allocating thrust correctly, and considering propulsion system models and constraints. Future work includes combining high and low control levels and evaluating the effect of the proposed feedback forms on the overall motion control performance.

REFERENCES

[1] G. Peeters, M. Kotzé, M. R. Afzal, T. Catoor, S. V. Baelen, P. Geenen, M. Vanierschot, R. Boonen, and P. Slaets, "An unmanned inland cargo vessel: Design, build, and experiments," *Ocean Engineering*, vol. 201, p. 107056, 4 2020.

[2] G. Peeters, T. Catoor, M. R. Afzal, M. Kotze, P. Geenen, S. V. Baelen, M. Vanierschot, R. Boonen, and P. Slaets, "Design and build of a scale model unmanned inland cargo vessel: Actuation and control architecture," vol. 1357. Institute of Physics Publishing, 11 2019.

[3] G. Peeters, M. R. Afzal, M. Vanierschot, R. Boonen, and P. Slaets, "Model structures and identification for fully embedded thrusters: 360-degrees-steerable steering-grid and four-channel thrusters," *Journal of Marine Science and Engineering*, vol. 8, 3 2020.

[4] O. A. Enezy, E. v. Hassel, C. Sys, and T. Vanelslander, "Developing a cost calculation model for inland navigation," *Research in Transportation Business & Management*, vol. 23, pp. 64–74, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210539516300992

[5] T. I. Fossen, *Marine control systems : guidance, navigation and control of ships, rigs and underwater vehicles*. Marine Cybernetics, 2002.

[6] A. J. Sørensen, "A survey of dynamic positioning control systems," *Annual Reviews in Control*, vol. 35, no. 1, pp. 123–136, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1367578811000095

[7] J. Dentler, "Real-time model predictive control of cooperative aerial manipulation," 10 2018.

[8] A. Veksler, T. Johansen, F. Borrelli, and B. Realfsen, "Dynamic positioning with model predictive control," *IEEE Transactions on Control Systems Technology*, vol. 24, pp. 1–14, 01 2016.

[9] S. Skjong and E. Pedersen, "Nonangular mpc-based thrust allocation algorithm for marine vessels - a study of optimal thruster commands," *IEEE Transactions on Transportation Electrification*, vol. 3, pp. 792–807, 9 2017.

[10] A. Jayasiri, A. Nandan, S. Imtiaz, D. Spencer, S. Islam, and S. Ahmed, "Dynamic positioning of vessels using a ukf-based observer and an nmpc-based controller," *IEEE Transactions on Automation Science and Engineering*, vol. 14, pp. 1778–1785, 10 2017.

[11] T. A. Johansen and T. I. Fossen, "Control allocation—a survey," *Automatica*, vol. 49, pp. 1087–1103, 5 2013.

[12] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, pp. 1–36, 3 2019.

[13] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming 2005 106:1*, vol. 106, pp. 25–57, 4 2005. [Online]. Available: https://link.springer.com/article/10.1007/s10107-004-0559-y

[14] T. I. Fossen, "A survey of control allocation methods for ships and underwater vehicles," in *2006 14th Mediterranean Conference on Control and Automation*. Institute of Electrical and Electronics Engineers (IEEE), 12 2006, pp. 1–6.

[15] A. Bärlund, J. Linder, H. Feyzmahdavian, M. Lundh, and K. Tervo, "Nonlinear mpc for combined motion control and thrust allocation of ships," *IFAC-PapersOnLine*, vol. 53, pp. 14 698–14 703, 1 2020.

[16] C. Kirches, "The direct multiple shooting method for optimal control," *Fast Numerical Methods for Mixed-Integer Nonlinear Model-Predictive Control*, pp. 13–29, 2011. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-8348-8202-8_2

[17] A. Veksler, T. A. Johansen, F. Borrelli, and B. Realfsen, "Cartesian thrust allocation algorithm with variable direction thrusters, turn rate limits and singularity avoidance," *2014 IEEE Conference on Control Applications, CCA. Part of 2014 IEEE Multi-conference on Systems and Control, MSC 2014*, pp. 917–922, 12 2014.

[18] J. T. Betts, "Practical methods for optimal control and estimation using nonlinear programming," *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 1 2010.

[19] Y. Nie and E. C. Kerrigan, "How should rate constraints be implemented in nonlinear optimal control solvers?" *IFAC-PapersOnLine*, vol. 51, pp. 362–367, 1 2018.

[20] 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*, vol. 12, 2004.