Reinforcement Learning-based NMPC for Tracking Control of ASVs: Theory and Experiments

Andreas B. Martinsen[a,*], Anastasios M. Lekkas[a,b] and Sébastien Gros[a]

[a]Department of Engineering Cybernetics, Norwegian University of Science and Technology, NO-7491, Trondheim, Norway
[b]Centre for Autonomous Marine Operations and Systems, Norwegian University of Science and Technology, NO-7491, Trondheim, Norway

ARTICLE INFO

Keywords:
Dynamic positioning
Model predictive control
Optimal control
Reinforcement learning
Surface vessels
System identification
Trajectory tracking

ABSTRACT

We present a reinforcement learning-based (RL) model predictive control (MPC) method for trajectory tracking of surface vessels. The proposed method uses an MPC controller in order to perform both trajectory tracking and control allocation in real-time, while simultaneously learning to optimize the closed loop performance by using RL and system identification (SYSID) in order to tune the controller parameters. The efficiency of the method is evaluated by performing simulations on the unmanned surface vehicle (USV) *ReVolt*, as well as simulations and sea trials on the autonomous urban passengers ferry *milliAmpere*. Our results demonstrate that the proposed method is able to outperform other state of the art methods both in tracking performance, as well as energy efficiency.

1. Introduction

In recent years we have seen a growing interest in developing methods for automatic and autonomous marine operations, with applications such as surveying and mapping, surveillance, and transportation, being of interest both for commercial and government use. This has lead to the need for robust high precision motion control systems, for performing operations such as docking and berthing Martinsen et al. (2020a), trajectory tracking Bitar et al. (2020), and collision avoidance Eriksen (2019).

Efficient control system design for marine vessels poses a number of challenges including the development of accurate mathematical models to describe complex vessel dynamics, the estimation of hydrodynamic coefficients that can vary significantly during operation, and the unpredictable nature of the marine environment. Consequently, extensive research has taken place in the past using ideas from almost all branches of control engineering. Linear, nonlinear, stochastic, optimal, intelligent, fuzzy, and adaptive control, to name a few, approaches have been developed and tested via simulations and field trials (Fossen and Grovlen, 1998; Pettersen and Nijmeijer, 2001; Skjetne and Fossen, 2001; Lefeber et al., 2003; Do et al., 2004; Aguiar and Pascoal, 2007; Caccia et al., 2008; Bibuli et al., 2009; Ahmed and Hasegawa, 2016). In order to simplify the control design process, a common approach is to choose control strategies based on operating conditions. This has led to station keeping and dynamic positioning (DP) controllers for low speed maneuvers, and path following or trajectory tracking controllers for higher speeds and transit. However, using this approach has the drawback of requiring multiple controllers and/or models with different properties. In order to achieve performance diversity with conventional methods, the two most common approaches are to design multiple controllers and switch between them, or to use adaptive control methods. To this end, research effort has been dedicated to developing methods for learning the vessel model and model parameters by, for instance, using parameter estimation, system identification or adaptive control (Hwang, 1980; Burns, 1995; Aguiar and Hespanha, 2007; Rajesh and Bhattacharyya, 2008; Wang et al., 2015; Ramirez et al., 2018; Martinsen et al., 2020b). In most of these works, model-based approaches exploiting knowledge on hydrodynamics and the laws of motion were considered.

Reinforcement learning (RL) is a subfield of machine learning (ML) which tackles the problem of optimal sequential decision making under uncertainty. The roots of RL can be traced back to the Artificial Intelligence (AI) community in the 60's (Sutton and Barto, 2018; Bertsekas, 2019). Since then the field has come a long way, evolving in several directions to become one of the most active research areas at the intersection of machine learning, artificial intelligence, neural network and control theory. Contrary to other machine learning methods, RL does not rely on prerecorded datasets, but rather learns by following a trial and error process, from which is receives evaluative feedback. Similarly to optimal control, this feedback comes in the form of a hand-engineered reward or cost function, which assigns a reward, or penalty, to the actions that result in desired, or undesired, outcomes, respectively. Given the reward or cost function, the job of the RL algorithm is to find a state-action mapping, known as the policy (the analog of a controller, in control engineering terminology), that optimizes the reward or cost given the problem constraints and uncertainties. To sum up, RL algorithms learn through feedback from the reward function, using trial and error in order to learn a policy that optimizes the given reward. In recent years, RL has also been shown to be a useful as an adaptive control approach for marine vehicles (Kamalapurkar et al., 2018; Martinsen and Lekkas, 2018a,b; Meyer et al., 2020; Wang et al., 2020).

Nonlinear model predictive control (MPC) is a popular approach for optimizing the closed loop performance of complex systems subject to constraints, which includes tra-

*Corresponding author
andreas.b.martinsen@ntnu.no (A.B. Martinsen)
ORCID(s): 0000-0002-6047-1715 (A.B. Martinsen)

jectory tracking and control of surface vessels (Li and Sun, 2011; Zheng et al., 2014; Liu et al., 2015; Veksler et al., 2016). MPC works by solving an optimal control problem (OCP) at each control interval in order to find an optimal policy. The optimal control problem seeks to minimize a the sum of stage costs over a horizon, provided a model of the system and the current observed state. While MPC is a well-studied approach, and an extensive literature exists on analysing its properties Mayne et al. (2000); Rawlings and Amrit (2009), the closed loop performance heavily relies on the accuracy of the underlying system model, which naturally presents challenges when significant unmodeled uncertainties are present.

In this work, we propose a model based RL approach for trajectory tracking of surface vessels. The approach builds on the work in Martinsen et al. (2020b), and extends it use a nonlinear MPC (NMPC) in order to perform the trajectory tracking in combination with control allocation. In order to optimize performance, the NMPC and model parameters are updated using RL Gros and Zanon (2019) and system identification (SYSID) Martinsen et al. (2020c). This allows the proposed method to compensate for model mismatch and environmental forces, with a focus on optimizing the closed loop performance of the trajectory tracking controller, rather than simply fitting the MPC model to the real system dynamics. In order to run the proposed control scheme in real-time, we implemented it using advanced-step NMPC (asNMPC). Additionally, simulations as well as sea trials were performed on the unmanned surface vehicle (USV) *ReVolt*, and the autonomous urban passengers ferry *milliAmpere*. The main contributions of this work are:

- A NMPC-based controller which combines trajectory tracking and control allocation for surface vessels (Section 2.2).
- The addition of RL and SYSID to the NMPC, in order to update the controller on-line. Making the controller able to compensate for model mismatch and environmental forces, and optimize the closed loop performance (Section 2.3).
- An implementation of the method using asNMPC, allowing for the controller to run in real-time (Section 3.1).
- Simulation study on two different vessel models, demonstrating how the approach outperforms our previous method from Martinsen et al. (2020b), and a traditional PID based controller (Section 4).
- Experimental results on an autonomous urban passenger ferry, demonstrating that NMPC based tracking control for surface vessels is real-time feasible, and is able to outperform a traditional PID based controller (Section 4).

The rest of the article is structured as follows. In Section 2 we show how reinforcement learning-based NMPC can be used for trajectory tracking for surface vessels. In Section 3 we outline the implementation of the proposed control scheme. Section 4 discuss the simulation and experimental results, while Section 5 concludes the paper.

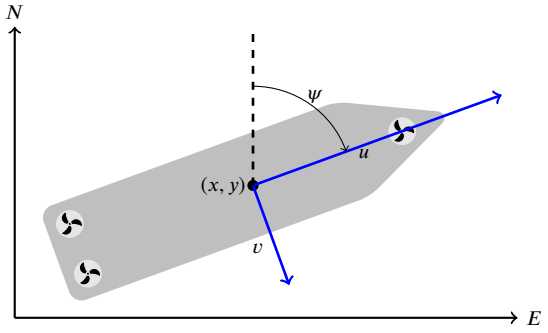

Figure 1: 3-DOF vessel centered at (x, y) in the North-East-Down (NED) reference frame, with surge velocity u, sway velocity v and heading ψ.

2. Reinforcement learning-based trajectory tracking NMPC

In this section, we will outline the proposed control scheme, as well as providing background on modeling of surface vessels, reinforcement learning-based NMPC and system identification.

2.1. Modeling of surface vessels

In order to accurately perform trajectory tracking, it is important to have a good model of the system we want to control. In this section we will provide background on how to model a surface vessel, including kinematics, dynamics, and thrusters. We will also show how the model can be made parametric, in a way that keeps the parameters linear in the model. This is useful, as it gives some nice properties when learning the model parameters.

2.1.1. Kinematics and dynamics

For control purposes, it is beneficial to keep the vessel model reasonably simple, this can be done by limiting the degrees of freedom, to the planar position and orientation of the vessel. Given $\mathbb{R}$ as the set of real numbers, $\mathbb{S} = [0, 2\pi]$ as the set of angles, and $SO(n) = \{\boldsymbol{R} | \boldsymbol{R} \in \mathbb{R}^{n\times n},\ \boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{R}\boldsymbol{R}^\top = \boldsymbol{I},\ \det(\boldsymbol{R}) = 1\}$ as the special orthogonal group in n dimensions, the motion of a surface vessel can be represented by the pose vector $\boldsymbol{\eta} = [x, y, \psi]^\top \in \mathbb{R}^2 \times \mathbb{S}$, and velocity vector $\boldsymbol{\nu} = [u, v, r]^\top \in \mathbb{R}^3$. Here, $\boldsymbol{p} = [x, y]^\top$ describe the Cartesian position in the Earth-fixed reference frame, ψ is yaw angle, (u, v) is the body fixed linear velocities, and r is the yaw rate, an illustration is given in Figure 1. Using the notation from Fossen (2011), a 3-DOF vessel can be modeled as follows

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu}, \\ \boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} &= \boldsymbol{\tau}_{\text{Thrust}} + \boldsymbol{\tau}_{\text{Env}}, \end{aligned} \tag{1}$$

where $\boldsymbol{M} \in \mathbb{R}^{3\times3}$, $\boldsymbol{D}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$, $\boldsymbol{C}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$, $\boldsymbol{\tau}_{\text{Thrust}} \in \mathbb{R}^3$, $\boldsymbol{\tau}_{\text{Env}} \in \mathbb{R}^3$ and $\boldsymbol{J}(\boldsymbol{\eta}) \in SO(3)$ are the inertia matrix, damping matrix, Coriolis matrix, thruster forces, environmental forces and transformation matrix respectively. The transformation matrix $\boldsymbol{J}(\boldsymbol{\eta}) \in SO(3)$ is given by

$$\boldsymbol{J}(\boldsymbol{\eta}) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

and is the rotation from the body frame to the earth-fixed North East Down (NED) reference frame. For notational brevity, we can express the vessel dynamics in (1), in terms of the implicit continuous time dynamics:

$$\begin{bmatrix} \dot{\boldsymbol{\eta}} - \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu} \\ \boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{\tau}_{\text{Thrust}} - \boldsymbol{\tau}_{\text{Env}} \end{bmatrix} = 0. \tag{3}$$

2.1.2. Thrust configuration

In order to find the thrust vector $\boldsymbol{\tau}_{\text{Thrust}}$ we must consider the mapping between the actuators present on the vessel, and how they translate into the surge, sway and yaw forces and moments acting on the vessel. This mapping can be represented by the thrust configuration matrix $\boldsymbol{T}(\boldsymbol{\alpha}) \in \mathbb{R}^{3\times n_{\text{thrusters}}}$ which maps the thrust $\boldsymbol{f}$ from each thruster into the surge, sway and yaw forces and moments in the body frame of the vessel given the thruster azimuth angles $\boldsymbol{\alpha}$.

$$\boldsymbol{\tau}_{\text{Thrust}} = \boldsymbol{T}(\boldsymbol{\alpha})\boldsymbol{f} \tag{4}$$

Each column $\boldsymbol{T}_i(\alpha_i)$ in $\boldsymbol{T}(\boldsymbol{\alpha})$ gives the configuration of the forces and moments of a thruster i as follows:

$$\boldsymbol{T}_i(\boldsymbol{\alpha}) f_i = \begin{bmatrix} F_x \\ F_y \\ F_y l_x - F_x l_y \end{bmatrix} = \begin{bmatrix} f_i \cos(\alpha_i) \\ f_i \sin(\alpha_i) \\ f_i(l_x \sin(\alpha_i) - l_y \cos(\alpha_i)) \end{bmatrix} \tag{5}$$

where α_i is the orientation of the thruster in the body frame, and f_i is the force it produces. Selecting the orientation $\boldsymbol{\alpha}$ and force $\boldsymbol{f}$ of the thrusters in order to generate the desired force $\boldsymbol{\tau}$ is called the *thrust allocation* problem. While there are numerous ways of solving the thrust allocation problem (Johansen and Fossen, 2013), for our purpose we want to include the thrust allocation as part of the optimization for performing path tracking. This allows us to take into account physical thruster constraints such as force saturation and feasible azimuth sectors:

$$\begin{aligned} \underline{\alpha}_i \leq \alpha_i \leq \overline{\alpha}_i \\ \underline{f}_i \leq f_i \leq \overline{f}_i, \end{aligned}$$

without limiting the trajectory tracking control scheme to fully-actuated vessels. We may additionally take into account thruster dynamics, this is especially useful for azimuth thrusters, where the rotation of the truster from one orientation to an other can be quite slow. Given a setpoint for the azimuth angle $\boldsymbol{\alpha}_s$ and thrust force $\boldsymbol{f}_s$, we can express the thruster dynamics as follows:

$$\begin{aligned} \dot{\boldsymbol{\alpha}} &= f_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}, \boldsymbol{\alpha}_s) \\ \dot{\boldsymbol{f}} &= f_{\boldsymbol{f}}(\boldsymbol{f}, \boldsymbol{f}_s) \end{aligned} \tag{6}$$

Adding the thruster dynamics we get the following continuous time implicit model dynamics.

$$\underbrace{\begin{bmatrix} \dot{\boldsymbol{\eta}} - \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu} \\ \boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{T}(\boldsymbol{\alpha})\boldsymbol{f} - \boldsymbol{\tau}_{\text{Env}} \\ \dot{\boldsymbol{\alpha}} = f_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}, \boldsymbol{\alpha}_s) \\ \dot{\boldsymbol{f}} = f_{\boldsymbol{f}}(\boldsymbol{f}, \boldsymbol{f}_s) \end{bmatrix}}_{f(\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{u})} = 0, \tag{7}$$

where the state $\boldsymbol{x}$ consists of the pose $\boldsymbol{\eta}$, velocity $\boldsymbol{\nu}$, thruster force $\boldsymbol{f}$ and azimuth angles $\boldsymbol{\alpha}$. And the control inputs $\boldsymbol{u}$ are given in terms of the thrust forces setpoint $\boldsymbol{f}_s$ and azimuth angle setpoint $\boldsymbol{\alpha}_s$.

2.1.3. Parametric model

While the model structure for a surface vessel is well known, estimate the model parameters can be quite difficult. For our approach we try to make as few assumptions on the parameters of the vessel model as possible, and use online learning in order to model the vessel based on gathered data. For this we assume that we know the model structure as given in (7), but that the model parameters in the inertia matrix $\boldsymbol{M}$, coriolis matrix $\boldsymbol{C}$ and damping matrix $\boldsymbol{D}$ are unknown. Assuming the vessel has port starboard symmetry, from Fossen (2011) we get the following structure:

$$\boldsymbol{M} = \begin{bmatrix} m_{1,1} & 0 & 0 \\ 0 & m_{2,2} & m_{2,3} \\ 0 & m_{2,3} & m_{3,3} \end{bmatrix}, \tag{8}$$

$$\boldsymbol{C}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & -m_{2,2} \cdot v - m_{2,3} \cdot r \\ 0 & 0 & m_{1,1} \cdot u \\ m_{2,2} \cdot v + m_{2,3} \cdot r & m_{1,1} \cdot u & 0 \end{bmatrix}, \tag{9}$$

$$\boldsymbol{D}(\boldsymbol{\nu}) = \begin{bmatrix} -X_u - X_{|u|u} \cdot |u| & 0 & 0 \\ 0 & -Y_v - Y_{|v|v} \cdot |v| - Y_{|r|v} \cdot |r| & -Y_r - Y_{|v|r} \cdot |v| - Y_{|r|r} \cdot |r| \\ 0 & -N_v - N_{|v|v} \cdot |v| - N_{|r|v} \cdot |r| & -N_r - N_{|v|r} \cdot |v| - N_{|r|r} \cdot |r| \end{bmatrix}, \tag{10}$$

where $m_{1,1}$, $m_{2,2}$, $m_{2,3}$, $m_{2,3}$, $m_{3,3}$ are the mass and added mass in the inertia matrix and X_u, $X_{|u|u}$, Y_v, $Y_{|v|v}$, $Y_{|r|v}$, Y_r, $Y_{|v|r}$, $Y_{|r|r}$, N_v, $N_{|v|v}$, $N_{|r|v}$, N_r, $N_{|v|r}$, $N_{|r|r}$ are the linear and nonlinear dampening terms. For the damping matrix $\boldsymbol{D}(\boldsymbol{\nu})$, both linear and nonlinear terms are included. The linear terms are important for low speed maneuvering and station keeping, while ensuring the velocity converges exponentially to zero. The nonlinear terms are required as they dominate at higher velocities. This ensures that the model is able to handle a wide range of velocities. To more accurately capture the dynamics, it is possible to include higher order terms, however this also increases the complexity of the model, and may in some cases lead to overfitting of the model.

In addition to learning the vessel dynamics, it is also useful to compensate for environmental forces, such as wind and current. This can be done by modeling the environmental forces as bias vector $\boldsymbol{w} \in \mathbb{R}^3$, which can be learned online together with the model parameters. Assuming that $\boldsymbol{w}$ is

given in the the NED frame, the resulting force in the body frame can then be modeled as follows:

$$\boldsymbol{\tau}_{\text{Env}} = \boldsymbol{W}\boldsymbol{J}^{\top}(\boldsymbol{\eta})\boldsymbol{w}, \tag{11}$$

where $\boldsymbol{W}$ is a weighting matrix representing how environmental forces act on the different dimensions of the vessel. In this approach, we choose a weighting based on the cross sectional area of the vessel:

$$\boldsymbol{W} = \text{diag}([w, l, 1]^{\top})$$

where l and w are the length and width of the vessel respectively, note that for better accuracy, a possibly state dependant $\boldsymbol{W}$ based on the hull geometry may be used instead of the length and width.

Choosing the parameters $\boldsymbol{\theta}_{\text{model}}$ as the vector of model parameters from the mass matrix ($m_{1,1}$, $m_{1,2}$, $m_{2,2}$, $m_{2,3}$, $m_{3,3}$), dampening matrix (X_u, Y_v, Y_r, N_v, N_r, $X_{|u|u}$, $Y_{|v|v}$, $Y_{|v|r}$, $Y_{|r|v}$, $Y_{|r|r}$, $N_{|v|v}$, $N_{|v|r}$, $N_{|r|v}$, $N_{|r|r}$) and environmental forces (w_1, w_2, w_3), we get a parametric continuous time model, which is linear in the parameters, on the following form:

$$f_{c,\theta}(\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{u}) = 0. \tag{12}$$

2.1.4. Parametric discrete time model

In order to use the vessel model for control, we discretize the continuous time dynamics in (7). While many options for discretizing the continuous time model exist, we chose to use the forward Euler integration for two main reasons.

- Forward Euler results in a discretization that is computationally cheap to evaluate. This is important when used in a real time NMPC setting, where increased model complexity results in increased evaluation time.
- Forward Euler preserves the linear in the parameters model structure from the the continuous time model. This is a highly desired property when performing parameter updates.

Given a sampling time T_s, the discretization resulting from the forward Euler method is given by replacing the derivative of the state $\dot{\boldsymbol{x}}$ with the approximation $\frac{\boldsymbol{x}^{+}-\boldsymbol{x}}{T_s}$, where $\boldsymbol{x}^{+}$ is the state at the next timestep. Using this we can formulate an implicit discrete time model on the form:

$$f_{d,\theta}(\boldsymbol{x}^{+}, \boldsymbol{x}, \boldsymbol{u}) = f_{c,\theta}(\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{u})\Big|_{\dot{\boldsymbol{x}}=\frac{\boldsymbol{x}^{+}-\boldsymbol{x}}{T_s}} = 0. \tag{13}$$

2.2. Trajectory tracking NMPC

For trajectory tracking control, the objective is to find a control policy which is able to make the vessel converge to a desired trajectory $\boldsymbol{x}_d(t)$. For a vessel with the dynamics given in (13), the control policy is the mapping from the vessel position $\boldsymbol{x}(t)$ and desired trajectory $\boldsymbol{x}_d(t)$, to the individual thruster force $\boldsymbol{f}_d$ and azimuth $\boldsymbol{\alpha}_d$ setpoints. Formulating this as an OCP, we get the following:

$$\begin{aligned}
\min_{\boldsymbol{x},\boldsymbol{\alpha},\boldsymbol{f}} \quad & \lambda_{\boldsymbol{\theta}}(\boldsymbol{x}_0, \boldsymbol{x}_{d,0}) + \sum_{i=0}^{N-1} \gamma^i L(\boldsymbol{x}_i, \boldsymbol{x}_{d,i}, \boldsymbol{u}_i) \\
& + \gamma^N V_{\boldsymbol{\theta}}^{f}(\boldsymbol{x}_N, \boldsymbol{x}_{d,N}) && \text{(14a)} \\
\text{s.t.} \quad & f_{d,\boldsymbol{\theta}}(\boldsymbol{x}_{i+1}, \boldsymbol{x}_i, \boldsymbol{u}_i) = 0, && \text{(14b)} \\
& \underline{\boldsymbol{f}} \leq \boldsymbol{f}_{s,i} \leq \overline{\boldsymbol{f}}, && \text{(14c)} \\
& \underline{\boldsymbol{\alpha}} \leq \boldsymbol{\alpha}_{s,i} \leq \overline{\boldsymbol{\alpha}}, && \text{(14d)} \\
& \boldsymbol{x}_0 = \boldsymbol{s}. && \text{(14e)}
\end{aligned}$$

Due to the nonlinear nature of the kinematics and dynamics of the vessel model, as well as the nonlinear cost function, we should note that the above OCP is a nonlinear optimization problem, and can be solved using a NMPC. The goal is to minimize the objective function (14a), consisting of an initial cost $\lambda_{\boldsymbol{\theta}}(\cdot)$, a discounted stage cost $L(\cdot)$ over a horizon N and a terminal cost $V_{\boldsymbol{\theta}}^{f}(\cdot)$, subject to vessel dynamics (14b), force (14c) and azimuth (14d) bounds and vessel initial condition (14e). In this formulation we can note that the initial cost $\lambda_{\boldsymbol{\theta}}(\boldsymbol{x}_0, \boldsymbol{x}_{d,0})$ does not effect the solution of the OCP, but is used when performing the reinforcement learning. We should also note that discounting the stage cost with a discount factor $\gamma < 1$, ensures that the cumulative cost converges to a finite value over the infinite horizon, allowing us to approximate it with the terminal cost.

2.2.1. Cost function

In order to perform trajectory tracking, we need to formulate a cost function (14a) which ensures that the NMPC performs the desired trajectory tracking behaviour. For the stage cost the following cost function was chosen:

$$\begin{aligned}
L(\boldsymbol{x}, \boldsymbol{x}_d, \boldsymbol{u}) = & \; q_{x,y} \cdot c_{x,y}(\boldsymbol{\eta}, \boldsymbol{\eta}_d) \\
& + q_{\psi} \cdot c_{\psi}(\boldsymbol{\eta}, \boldsymbol{\eta}_d) \\
& + \left(\boldsymbol{\nu} - \boldsymbol{\nu}_d\right)^{\top} \boldsymbol{Q} \left(\boldsymbol{\nu} - \boldsymbol{\nu}_d\right) \\
& + \boldsymbol{\alpha}^{\top} \boldsymbol{R}_{\alpha} \boldsymbol{\alpha} + \boldsymbol{f}^{\top} \boldsymbol{R}_{f} \boldsymbol{f}
\end{aligned} \tag{15}$$

The stage cost in (15) uses a quadratic penalty on velocity and control actions with weight matrices $\boldsymbol{Q}$, $\boldsymbol{R}_{\alpha}$ and $\boldsymbol{R}_{f}$. The position cost $c_{x,y}(\boldsymbol{\eta}, \boldsymbol{\eta}_d)$, weighted by $q_{x,y}$, is chosen as a pseudo-Huber function, penalizing the difference between the vessel pose $\boldsymbol{\eta}$ and the desired pose $\boldsymbol{\eta}_d$, and is given as follows:

$$c_{x,y}(\boldsymbol{\eta}, \boldsymbol{\eta}_d) = \delta^2 \left(\sqrt{1 + \frac{(x - x_d)^2 + (y - y_d)^2}{\delta^2}} - 1 \right). \tag{16}$$

Using a pseudo-Huber cost, provides a quadratic penalty when the quadrature position error is small and linear when the position error is large. This helps with numerical stability, as well as performance when large position errors are observed Gros and Zanon (2017); Gros and Diehl (2013). For the heading cost function $c_{\psi}(\boldsymbol{\eta}, \boldsymbol{\eta}_d)$, weighted by q_{ψ}, the following was chosen:

$$c_{\psi}(\boldsymbol{\eta}, \boldsymbol{\eta}_d) = \frac{1 - \cos(\psi - \psi_d)}{2}, \tag{17}$$

as it avoids the problem of heading wraparound.

For the parametric initial cost $\lambda_\theta(\boldsymbol{x}_0, \boldsymbol{x}_{d,0})$, a simple bias term was chosen, giving the following:

$$\lambda_\theta(\boldsymbol{x}_0, \boldsymbol{x}_{d,0}) = \theta_\lambda.$$

Similarly, the parametric terminal cost approximation $V_\theta^f(\cdot)$ was chosen as a quadratic cost as follows:

$$V_\theta^f(\boldsymbol{x}_N, \boldsymbol{x}_{d,N}) = (\boldsymbol{x}_N - \boldsymbol{x}_{d,N})^\top \text{diag}(\boldsymbol{\theta}_V)(\boldsymbol{x}_N - \boldsymbol{x}_{d,N}),$$

with the parameters θ_λ and $\boldsymbol{\theta}_V$ being learned through reinforcement learning.

2.3. Reinforcement Learning-based NMPC

Given the model parameters $\boldsymbol{\theta}_{\text{model}}$ and cost function parameters θ_λ and $\boldsymbol{\theta}_V$ the goal is to learn the the parameters $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} = (\boldsymbol{\theta}_{\text{model}}, \theta_\lambda, \boldsymbol{\theta}_V),$$

based on data gathered on line, in a way that optimizes the closed loop performance of the NMPC given by (14). In recent works, such as Gros and Zanon (2019); Zanon and Gros (2020); Martinsen et al. (2020c), this has been done by allowing RL to use a NMPC as a function approximator. This combines the benefits of data-driven optimization from RL with the tools available for analysing and certifying the closed loop performance of NMPC. For our implementation, we will use the approach in Martinsen et al. (2020c), where the RL-based NMPC is combined with system identification (SYSID) in a way that minimizes plant model mismatch while optimizing the closed loop performance of the NMPC. In the next subsections we will show how this approach can be applied to the trajectory tracking problem in (14).

2.3.1. Value functions and policy

Given the parametric optimization problem (14), we define the parametric action-value function as:

$$\begin{aligned} Q_\theta(\boldsymbol{s}, \boldsymbol{a}) = \min_{\boldsymbol{x}, \boldsymbol{u}} \quad & \text{(14a)} && \text{(18a)} \\ \text{s.t.} \quad & \text{(14b) - (14d)}, && \text{(18b)} \\ & \boldsymbol{x}_0 = \boldsymbol{s}, && \text{(18c)} \\ & \boldsymbol{u}_0 = \boldsymbol{a}. && \text{(18d)} \end{aligned}$$

This action-value function $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$ approximates the expected cumulative discounted cost when taking an action $\boldsymbol{a}$ in a state $\boldsymbol{s}$. Using the action-value function $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$, we can express the state-value function $V_\theta(\boldsymbol{s})$ and policy $\pi_\theta(\boldsymbol{s})$ as follows:

$$V_\theta(\boldsymbol{s}) = \min_{\boldsymbol{a}} Q_\theta(\boldsymbol{s}, \boldsymbol{a}), \tag{19a}$$

$$\pi_\theta(\boldsymbol{s}) = \arg\min_{\boldsymbol{a}} Q_\theta(\boldsymbol{s}, \boldsymbol{a}), \tag{19b}$$

where the the policy $\pi_\theta(\boldsymbol{s})$ approximates the optimal action for a state $\boldsymbol{s}$, and the state-value function $V_\theta(\boldsymbol{s})$ approximates the expected cumulative discounted cost under the policy.

2.3.2. Q-Learning

The goal of RL is to find the parameters $\boldsymbol{\theta}$ that maximize the closed loop performance under the policy $\pi_\theta(\boldsymbol{s})$. While a number of different approaches exist, we will focus on the classical Q-Learning method Watkins (1989). In Q-Learning the goal is to find the parameterization which best fits the action-value function to the observed data. Given an observed transition $(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{x}_{t+1})$ Q-Learning can be performed by minimizing the temporal-difference error:

$$\delta_t = y_t - Q_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t), \tag{20}$$

where $y_t = L(\boldsymbol{x}_t, \boldsymbol{x}_{d,t}, \boldsymbol{u}_t) + \gamma V_\theta(\boldsymbol{x}_{t+1})$ is the fixed target value. Defining the squared temporal-difference error as the minimization objective, and assuming that the target value is independent of the parameterization $\boldsymbol{\theta}$, we get the semi-gradient update Sutton and Barto (2018):

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta_Q \delta \nabla_\theta Q_\theta(\boldsymbol{x}_t, \boldsymbol{u}_t), \tag{21}$$

where $\beta_Q > 0$ is the step-size or learning rate. For the classical semi-gradient Q-learning scheme given in (21), a second order method can be implemented by using quasi-Newton steps instead of gradient steps. This results in the following update law:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta_Q \underbrace{\delta \boldsymbol{H}_Q^{-1} \nabla_\theta Q_\theta(\boldsymbol{x}_t, \boldsymbol{u}_t)}_{:=\boldsymbol{\Delta}\theta_Q}, \tag{22}$$

where $\boldsymbol{H}_Q = \nabla_\theta^2 (y_t - Q_\theta(\boldsymbol{x}_t, \boldsymbol{u}_t))^2$ is the Hessian of the error between the targets and the action-value function. It is also possible to further generalize this method for a batch of transitions, resulting in a nonlinear least squares problem Martinsen et al. (2020c).

2.3.3. System Identification

In addition to learning the parameters $\boldsymbol{\theta}$ from Q-Learning, it is also possible to learn the parameters associated to the MPC model using SYSID. One such approach is the Prediction Error Method (PEM) where the objective is to minimize the difference between the observed state and the predicted state given the observed transition $(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{x}_{t+1})$. For a parametric model approximation of the form:

$$f_\theta(\hat{\boldsymbol{x}}_{t+1}, \boldsymbol{x}_t, \boldsymbol{u}_t) = 0$$

the prediction error $\boldsymbol{e}_t$ between the parametric model and the observed state can then be expressed as follows:

$$\boldsymbol{e}_t = f_\theta(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t, \boldsymbol{u}_t).$$

In the simplest case, where the state vector $\boldsymbol{x}$ is fully observable, PEM can be performed by minimizing the squared error $\|\boldsymbol{e}_t\|^2$ between the observed state, and the predicted state. This optimization problem can be tackled via gradient descent, giving the following update law:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta_f \nabla_\theta \boldsymbol{e}_t^\top \boldsymbol{e}_t,$$

where β_f is the learning rate. Similarly to the RL objective, we can use Quasi newton steps:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta_f \underbrace{\boldsymbol{H}_f^{-1} \nabla_{\boldsymbol{\theta}} \boldsymbol{e}_t^\top \boldsymbol{e}_t}_{:=\boldsymbol{\Delta\theta}_f}, \tag{23}$$

where $\boldsymbol{H}_f$ is the hessian of the squared prediction error. This can be further generalized for a batch of transitions Martinsen et al. (2020c). We can also note that if the model is linear in the parameters, which is the case for the vessel model in (13), this becomes a linear least squares problem, for which the global minimum can be found by taking the a full newton step i.e. choosing $\beta_f = 1$.

2.3.4. Parameter update law

When updating the parameters $\boldsymbol{\theta}$ it is possible to combine both Q-Learnig and SYSID. The simplest approach is to directly combine the steps from both the Q-Learning and SYSID. Using the second order update laws in (22) and (23), with the parameter updates $\Delta\boldsymbol{\theta}_Q$ and $\Delta\boldsymbol{\theta}_f$ respectively, we get the following:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta_Q \boldsymbol{\Delta\theta}_Q + \beta_f \boldsymbol{\Delta\theta}_f. \tag{24}$$

Here the step-lengths β_Q and β_f can be thought of as the weighting between the Q-Learning and SYSID respectively. However, the end goal is arguably to maximize the closed-loop performance of the MPC scheme rather than minimizing the prediction error of the model, hence if the two objectives are competing, the RL objective should be prioritized. In order to prioritize the RL objective we use instead the following update law:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta_Q \boldsymbol{\Delta\theta}_Q + \beta_f \boldsymbol{P} \boldsymbol{\Delta\theta}_f, \tag{25}$$

where $\boldsymbol{P}$ is a projection matrix. As proposed in Martinsen et al. (2020c), we can choose the projection matrix as the direction in the parameter space for which the RL objective is the least sensitive, i.e. direction of the smallest eigenvalues of $\boldsymbol{H}_Q$. This allows for minimizing the prediction error of the model with SYSID, while prioritizing the RL objective of optimizing the closed loop performance of the MPC.

3. Implementation

In order to test the proposed RL base trajectory tracking NMPC, we performed tests on two different vessels. In this section we will introduce the two platforms, and discuss how we implemented the proposed RL based trajectory tracking control method in a way that allowed for it to be used in real time.

3.1. Advanced-step Nonlinear Model Predictive Control

While a number of NMPC schemes exist, that guarantee closed loop stability Mayne et al. (2000), the necessary on-line computation time is typically not taken into account. Even though recent hardware and software developments have lead to faster and more efficient numerical solution methods for open-loop optimal control, the solution time is often significant in the context of closed-loop control. The resulting delay caused by solving the NMPC problem online can often lead to degraded performance, or in some cases even instability of the closed loop system. In order to account for the computational delay, a number of methods have been proposed Chen et al. (2000); Findeisen and Allgöwer (2004); Diehl et al. (2005); Zavala and Biegler (2009). One such approach is the advanced-step NMPC (asNMPC) Zavala and Biegler (2009), where the idea is to first use the current state measurement and control action to predict the state one step into the future and then solve the corresponding NMPC problem in advance. Transcribing the OCP in (14) into standard form nonlinear programming problem (NLP):

$$\begin{aligned} &\min_{\boldsymbol{w}} \ \phi(\boldsymbol{w}, \boldsymbol{p}) \\ &\text{s.t. } \boldsymbol{g}(\boldsymbol{w}, \boldsymbol{p}) = 0 \\ &\quad \boldsymbol{h}(\boldsymbol{w}, \boldsymbol{p}) \leq 0, \end{aligned} \tag{26}$$

where $\boldsymbol{w}$ are the decision variables, and $\boldsymbol{p}$ are the parameters, chosen to be the initial state $\boldsymbol{s}$. This gives the First-Order Necessary Conditions of a primal dual interior point method as follows:

$$\boldsymbol{r}(\boldsymbol{z}, \boldsymbol{p}) = \begin{bmatrix} \nabla_{\boldsymbol{w}}\phi + \nabla_{\boldsymbol{w}}\boldsymbol{g}\lambda + \nabla_{\boldsymbol{w}}\boldsymbol{h}\boldsymbol{\mu} \\ \boldsymbol{g} \\ \mathrm{diag}(\boldsymbol{\mu})\boldsymbol{h} + \tau \end{bmatrix} = 0 \tag{27}$$

where $\boldsymbol{z} = [\boldsymbol{w}, \lambda, \boldsymbol{\mu}]$ are the primal-dual variables, and τ is a constraint relaxation parameter. If the Linear Independence Constraint Qualification (LICQ) and Second Order Sufficient Conditions (SOSC) hold Nocedal and Wright (2006) for the NLP in (26), then the Implicit Function Theorem (IFT) guarantees that:

$$\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{p}} = 0. \tag{28}$$

Solving the NLP in (26) for a parameterization $\boldsymbol{p}_0$, with the solution of the primal-dual variables $\boldsymbol{z}_0$, we can construct the following linear predictor.

$$\boldsymbol{z} = \boldsymbol{z}_0 + \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{p}}(\boldsymbol{p} - \boldsymbol{p}_0) \tag{29}$$

Using the first order predictor on the NLP resulting from an NMPC problem, with the initial state $\boldsymbol{x}_0$ chosen as the parameter vector $\boldsymbol{p} = \boldsymbol{x}_0$, the asNMPC can be summarized as follows:

- In the background between time step t and $t+1$:
  - Predict the state $\hat{\boldsymbol{x}}_{t+1}$ using forward simulation.
  - Solve the NMPC problem with $\boldsymbol{p}_0 = \hat{\boldsymbol{x}}_{t+1}$, to get the primal dual variables $\boldsymbol{z}_0$.
  - Compute the parameter sensitivity $\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{p}}$ using the IFT (28).
- On-line at time step $t+1$:

Figure 2: The Revolt test platform is a 1 : 20 scale model of a autonomous concept vessel with two fully rotatable azimuth thrusters in the stern and one fully rotatable azimuth thruster in the bow.

Figure 3: The MilliAmpere test platform has two fully rotatable azimuth thrusters along the cenerline of the vessel.

- Obtain the true state of the system $\boldsymbol{x}_{t+1}$ from sensor measurements.
- Use the linear predictor (29) to find the approximate solution $\boldsymbol{z}$ of the NLP.
- Extract the first control input $\boldsymbol{u}_{t+1}$ from the approximate solution $\boldsymbol{z}$.
- Apply the control input $\boldsymbol{u}_{t+1}$ to the plant, and return to the background step.

The above asNMPC algorithm will then yield an approximate control law with a minimal delay between the measurement of the state and the application of the control input. This allows us to approximately solve the NMPC problem in (14) in real time, making the the proposed control scheme feasible for use on physical platforms. It should however be noted that this is still a computationally demanding control architecture, and requires that the OCP can be solved within one time step. In general, this requirement will limit the length of the prediction horizon of the asNMPC, and the complexity of the parameterized model.

3.2. Experimental Platforms

In order to test the proposed method, simulations studies were performed on two different platforms, with additional full scale experiments carried out on one of them. In the next sections we will introduce the two platforms, namely *ReVolt* and *milliAmpere*, and discuss how the proposed RL based trajectory tracking NMPC was implemented.

3.2.1. ReVolt platform

The *ReVolt*, shown in Figure 2, is a 1 : 20 scale model of a autonomous concept vessel developed and buildt by DNV GL in collaboration with NTNU. The 3 meter long and 0.72 meter wide model, weighs approximately 257 kg, and has three fully rotatable azimuth thrusters for propulsion. The thrust configuration seen in Figure 2, consists of two identical stern thrusters, and one slightly less powerful bow thruster, giving the vessel a total combined engine power of 360 W and a top speed of 2 knots (approximatly 1 m/s).

For simulating the *ReVolt* an accurate Digital Twin, developed by DNV GL, was used. The Digital Twin is based on a full 6DOF model, with parameters identified through towtank experiments, as well as frequency domain analysis of a 3D model of the vessel hull. The Digital Twin allowed for rapidly testing how the proposed control scheme performed under ideal conditions, as well as under different environmental conditions, including disturbances from wind, waves and ocean currents.

The trajectory tracking controller for the *ReVolt* was implemented as an asNMPC, solving the OCP in (14) in each sampling interval. Due to the vessel having three fullty rotatable azimuth thrusters with relatively fast dynamics, the thruster dynamics were not modeled, and an additional singularity avoidance penalty (30) was added to the cost function, in order to encourage nonsingular thrust configurations Johansen et al. (2004).

$$\frac{\rho}{\epsilon + \det\left(\boldsymbol{T}(\boldsymbol{\alpha})\boldsymbol{T}^{\top}(\boldsymbol{\alpha})\right)} \tag{30}$$

Using the solution of the advanced step prediction, the thrust and azimuth angle commands were directly applied to the vessel thrusters. The sampling time of the controller was chosen as $T_s = 0.2$s (5Hz) giving enough time to solve the NMPC, while still being fast enough to stabilize the system. In order to learn the parameters $\boldsymbol{\theta}$ on-line, the update law in (25) was used on a batch of $M = 1$ samples which were recorded on-line. A list of parameter values used for the NMPC implementation is given in Table 3.

3.2.2. milliAmpere platform

The *milliAmpere*, shown in Figure 3, is an experimental autonomous urban passenger ferry which has been in development at the Norwegian University of Science and Technology (NTNU) since 2017. *milliAmpere* has served as a platform for testing and developing autonomous technology, including software, sensor arrays, as well as hardware solutions. The platform is 5 meters long and 2.8 meters wide, with a symetric footprint. It has two fully rotatable azimuth thrusters mounted along the center line of the vessel, giving it a top speed of 5 knots (approximatly 2.5 m/s).

For simulating the *milliAmpere*, a nonlinear 3DOF model of the vessel was used together with models of the thruster and azimuth dynamics, with the model parameters being identified through experiments.

The trajectory tracking controller for the *milliAmpere* was implemented as an asNMPC, solving the OCP in (14). Due to slow azimuth thruster rotation, the azimuth dynmaics were also included in the vessel model, allowing for the NMPC to account for the the dynamics when planning the control actions. Similar to the *ReVolt* implementation, the sampling

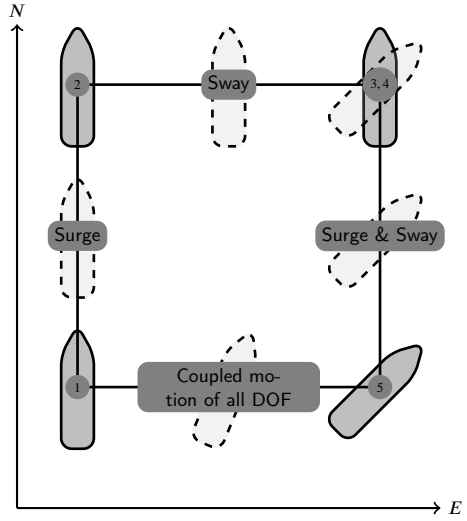

Figure 4: Illustration of the four corner DP test, used for testing trajectory tracking performance in individual as well as coupled degrees of freedom.

time for the *milliAmpere* controller was chosen as $T_s = 0.2$s (5Hz) giving enough time to solve the NMPC, while still being fast enough to stabilize the system. The parameter update law (25) was also used on the *milliAmpere*, on a batch of $M = 10$ samples recorded on-line. This allowed the controller to continuously adapt to the changing conditions. A list of parameter values used for the NMPC implementation is given in Table 4.

4. Results

In this section we present the results from simulations on the *ReVolt* test platform (Figure 2), as well as simulations and sea trials on the *milliAmpere* test platform (Figure 3).

4.1. Four corner DP test

In order to evaluate the trajectory tracking capabilities of the proposed control method, the four corner test seen in Figure 4 is used. This test is used in order to evaluate the trajectory tracking capabilities of the vessel for individual degrees of freedom, as well as the coupled motion of all degrees of freedom. The four corner test starts with the vessel pointing north 0°, then performs the following commands:

1. Move l meters due north, this tests the the surge motion of the vessel.
2. Move l meters due east, this tests the the sway motion of the vessel.
3. Rotate to a heading of 45° while keeping the same position, this tests the the yaw motion of the vessel.
4. Move l meters due south while keeping the same heading, this tests the the coupled surge and sway motion of the vessel.
5. Move l meters due west while rotating to a heading of 0°, this tests coupled motion of all degrees of freedom.

Table 1
List test scenarios for *ReVolt* (only simulations).

Scenario	Description
$R1$	Simulation of baseline Adaptive dynamic programming (ADP) method from Martinsen et al. (2020b)
$R2$	Simulation of RL-based NMPC without online learning
$R3$	Simulation of RL-based NMPC with online learning
$R4$	Simulation of RL-based NMPC with online learning and 3m/s wind from the north
$R5$	Simulation of RL-based NMPC with online learning and 0.1m/s current from the west

For the four corner test we chose the side length l to be 5 meters, for the *ReVolt*, and l to be 10 meters for the *milliAmpere*. Each maneuver is given 60 seconds to complete, and reference filter is used to generate a continuous trajectory between the commanded maneuvers.

In order to evaluate the transient performance of the dynamic positioning, we use the Integral Absolute Error (IAE) given in (31).

$$IAE(t) = \int_0^t \sqrt{(\boldsymbol{\eta} - \boldsymbol{\eta}_d)^\top \boldsymbol{W}_{\text{IAE}}^{-1} (\boldsymbol{\eta} - \boldsymbol{\eta}_d)} dt \tag{31}$$

Where $\boldsymbol{W}_{\text{IAE}}$ is a weighting factor, which is chosen to normalize the pose between ± 5 meters in north and east direction, and $\pm 50°$ in heading, giving the following.

$$\boldsymbol{W}_{\text{IAE}} = \begin{bmatrix} 5^2 & 0 & 0 \\ 0 & 5^2 & 0 \\ 0 & 0 & 50^2 \end{bmatrix}$$

4.2. Results ReVolt (Simulations Only)

For the *ReVolt* platform, validation of the proposed trajectory tracking controller was performed in simulations for the five different scenarios given in Table 1. Performing the four corner test, we got the results seen in Figure 5, with the IAE performance seen in Figure 6.

From the trajectory and tracking error for our method ($R3$) seen in Figure 5, we can observe that the positioning error is less the 10cm in both the North and east direction, while the heading is within 5° of the desired heading. Compared to the baseline Adaptive Dynamic Programming (ADP) method ($R1$), our method does not have the same spikes when transitioning between maneuvers. This is likely due to the NMPC planning ahead for the maneuver changes, while the the baseline approach is having to react to them as they happen. The added prediction horizon of the NMPC is a definite advantage of the proposed approach, however it does come at the cost of computational complexity. While the proposed control scheme is limited to a 5Hz update rate due to the time it takes to solve the OCP, the ADP based solution is easily able to run at 10Hz.

From the IEA performance in Figure 6, we see the how the proposed controller performs in different conditions, with

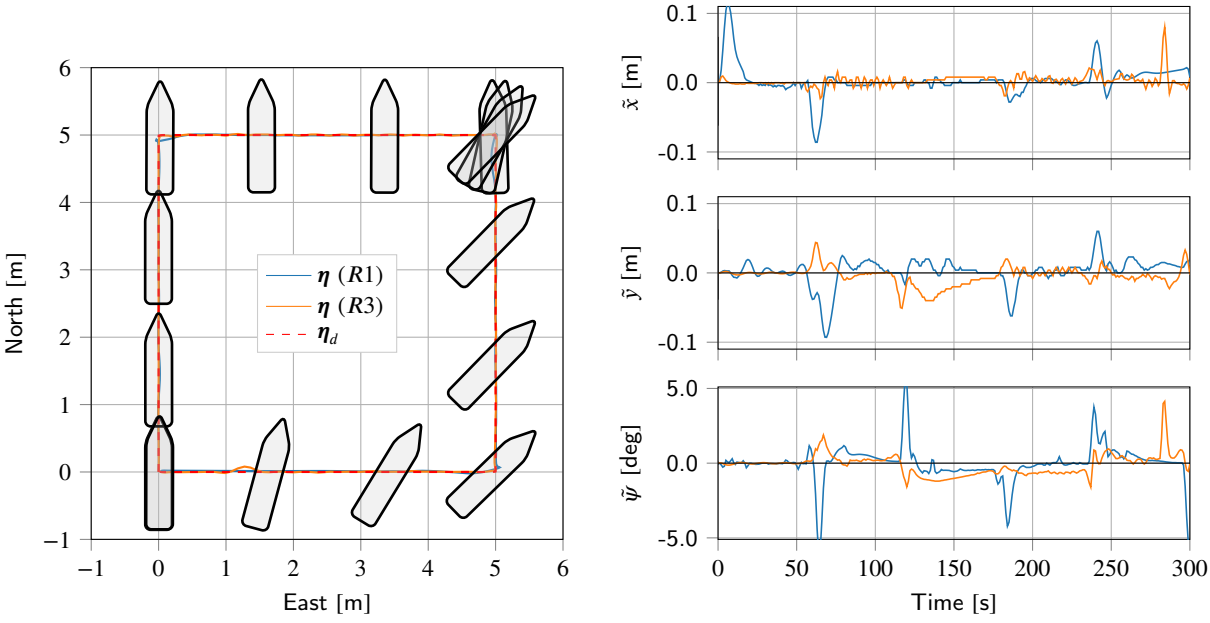

Figure 5: Simulation results for *ReVolt* with online learning ($R3$), and baseline ($R1$)

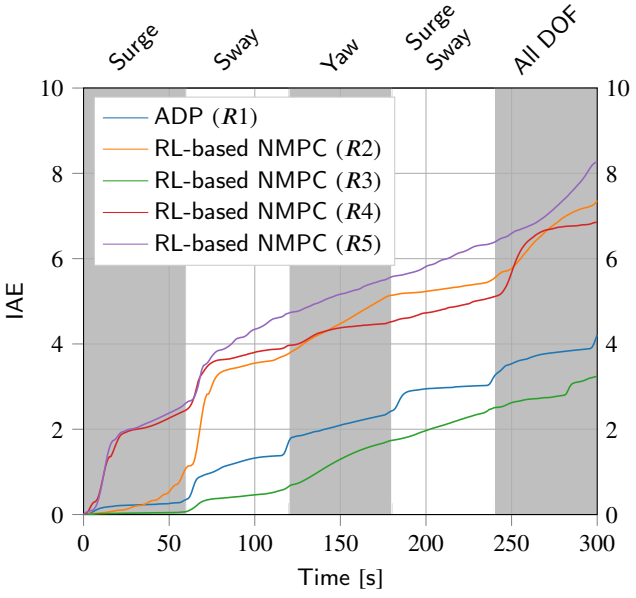

Figure 6: Integral Absolute Error (IAE) for *ReVolt*, the white and gray bands show the different phases of the four corner test.

and without parameter updates, as well as the performance compared to a baseline ADP approach. Looking at the performance of our method without online learning ($R2$), compared to our method with online learing ($R3$), we see a significant difference in performance. This is due to model mismatch between the initial model used in the OCP and the Simulator. Using online parameter updates, allows the model and performance of the NMPC to be improved based on gathered data, and results in a significant performance boost. It is also worth noting the performance difference between the baseline approach ($R1$), and our method with online learning ($R3$). For the baseline approach ($R1$), we see large increases in IAE when transitioning between the different maneuvers, while for our approach ($R3$), these increases are less prevalent. This is due to the RL-based NMPC being able to take into account the trajectory over future time horizon, as well as including the thrust allocation in the OCP, allowing for more accurately planning and performing the maneuvers. Looking at our method when subject to external disturbances in terms of wind ($R4$) and current ($R5$), we see an initial increase in the IAE before the performance starts to stabilize, with a slope similar to that of trained RL-based NMPC. This behaviour is expected, as the initial increase happens since the controller is not aware of the disturbance, and flattens out as the controller learns how to compensate for the disturbance as a constant force and torque in the NED frame (11).

4.3. Results milliAmpere (Simulations and Sea Trials)

For the *milliAmpere* platform, validation of the proposed trajectory tracking controller was performed for five different scenarios, Table 2, including both simulations as well as sea trials. Performing the four corner test, we got the simulation results seen in Figure 7, the experimental results seen in Figure 8 and 11. The performance in terms of the IAE is given in Figure 9, and the performance in terms of power consumption is shown in Figure 10.

Based on the results from our method in Figure 7 and 8, we see good tracking performance, with similar results for both the simulations ($M4$) and the sea trials ($M5$). From

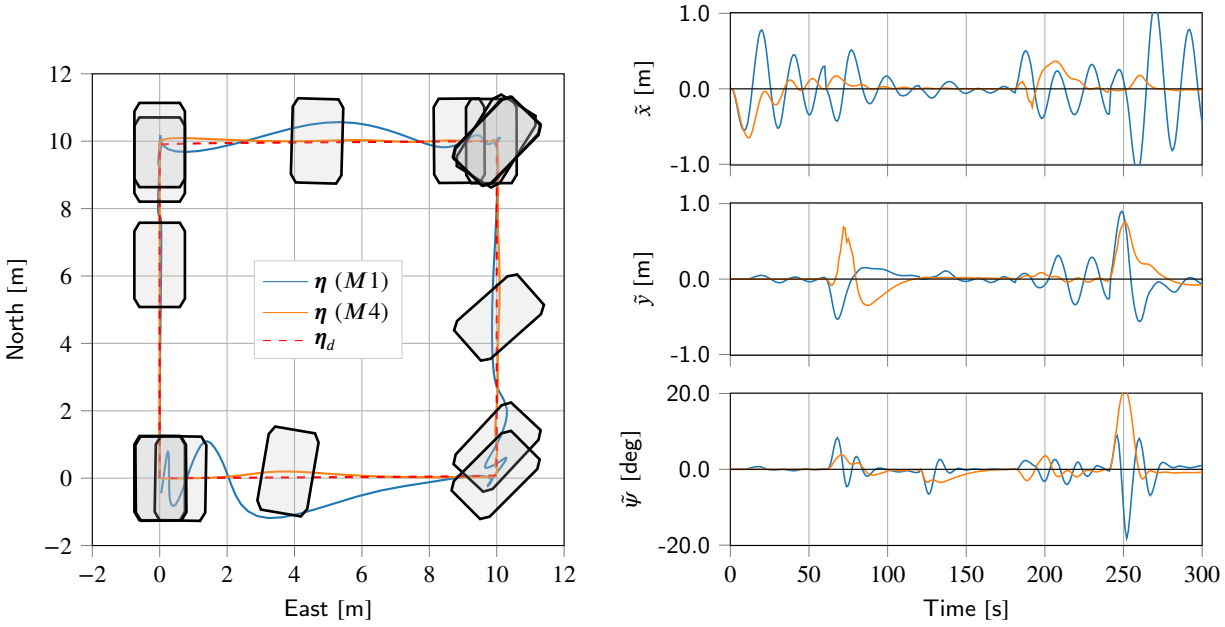

Figure 7: Simulation results for *milliAmpere* with online learning ($M4$), and baseline ($M1$).

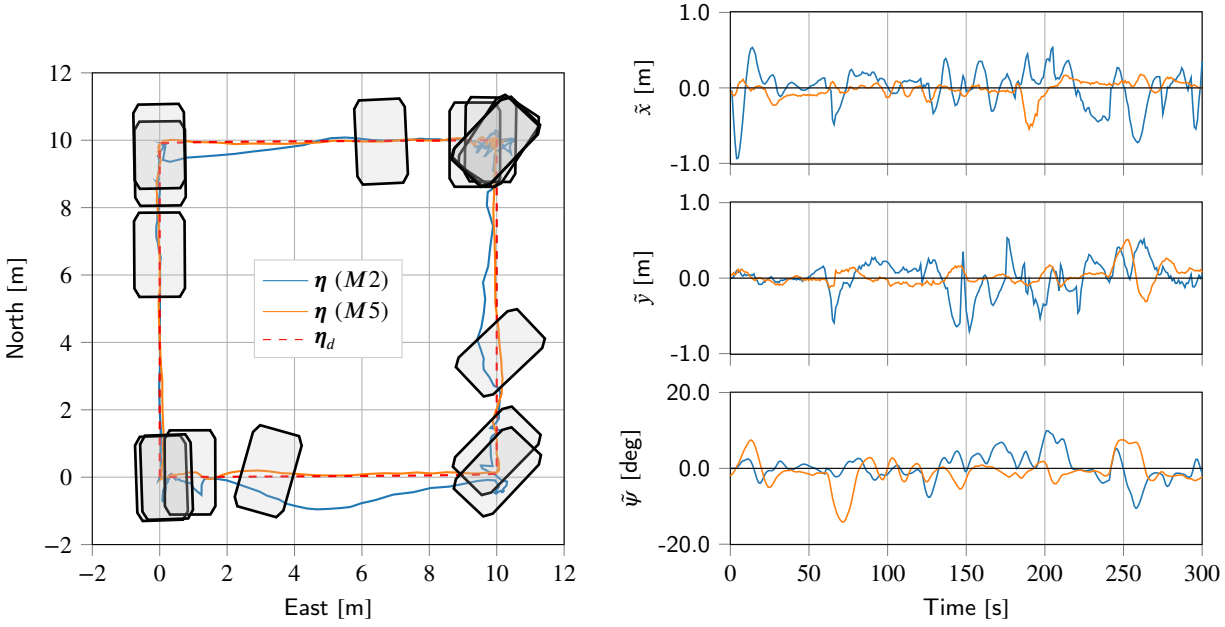

Figure 8: Sea trial results for *milliAmpere* with online learning ($M5$), and baseline ($M2$).

the tracking error we see that the trajectory is well within one meter, with most of the major tracking errors happening after command changes. This is mostly due to the vessel not being able to accelerate fast enough to follow the reference trajectory when switching between the different poses. For the heading error, we see a maximum error of about 20°. This is a relatively large error, and is the largest contributor to the IAE as can be seen in Figure 9. By choosing the weighting between the position error $q_{x,y}$ and the heading error q_ψ it is possible to change the priority between heading and position error, with our main focus being on the position error when choosing the parameters.

In order to evaluate our proposed control scheme, we performed the same simulations ($M1$) and sea trials ($M2$) using a standard Proportional Integral Derivative (PID) based DP controller with an optimization based control allocation scheme Torben et al. (2019). It should be noted that the PID controller was not tuned to optimize any performance measure, however it still provides a good benchmark. Compared to our approach, the PID based method has slightly

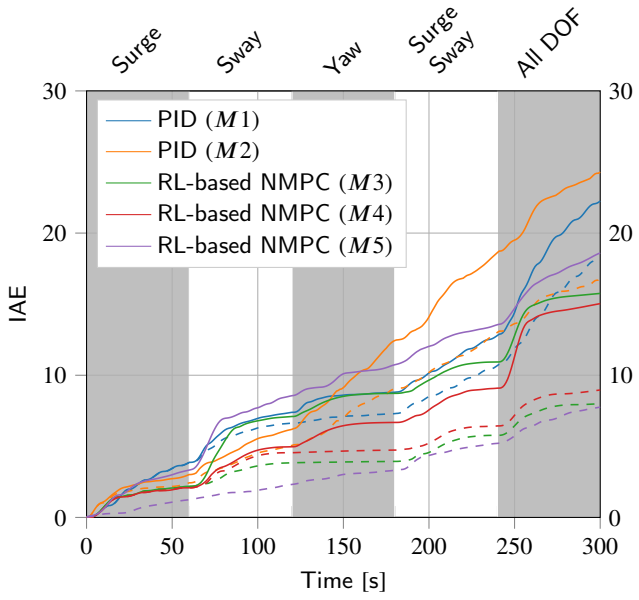

Figure 9: Integral Absolute Error (IAE) for *milliAmpere*, the white and gray bands show the different phases of the four corner test. The solid lines show the IAE with the heading error, while the dashed lines show the IAE without the heading error.

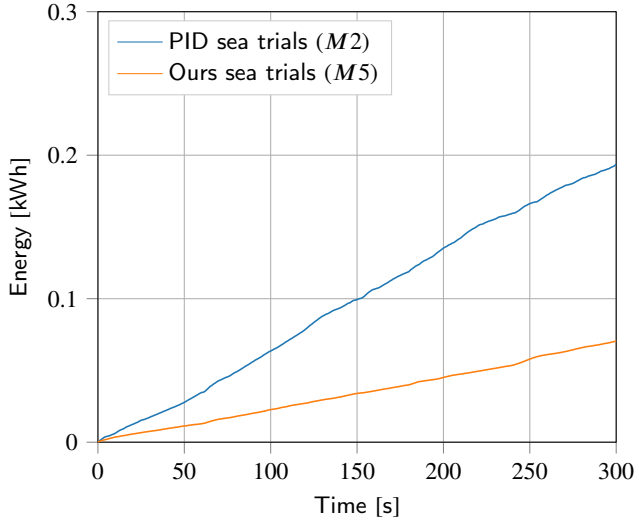

Figure 10: Power consumption for the Dynamic Positioning performed on *milliAmpere*.

less heading error, while our approach has significantly lower position error as can be seen in Figure 7 and 8, as well as in the IAE in Figure 9. Our approach also has about half the power consumption when performing the maneuver compared with the PID based DP controller, as seen in Figure 10. This is likely due to the PID based DP controller relying on an aggressive control allocation method, while our approach integrates the control allocation into the optimization problem, allowing it to consume less power while still being able

Table 2
List test scenarios for *milliAmpere* (simulations and sea trials).

Scenario	Description
$M1$	Simulation of baseline PID based DP method from Torben et al. (2019)
$M2$	Sea trials of baseline PID based DP method from Torben et al. (2019)
$M3$	Simulation of RL-based NMPC without online learning
$M4$	Simulation of RL-based NMPC with online learning
$M5$	Sea trials of RL-based NMPC with online learning

to perform the desired maneuvers. The integration of control allocation and thruster dynamics into OCP is a definite advantage of the proposed control scheme, as it significantly reduces the mismatch between desired and actual thrust, allowing for more accurate maneuvering. It should however be noted that the addition of control allocation and thruster dynamics does increase the model complexity, and hence the computational complexity when solving the OCP.

During the experiments, an unexpected azimuth failure occurred on one of the thrusters, but the proposed method was able to compensate and still perform the desired maneuver by learning a new model of the vessel which compensated for the azimuth failure. These additional results are shown in Appendix B.

5. Conclusion

We have presented a method for trajectory tracking control of surface vessels using a RL-based NMPC. The proposed method performs optimal tracking control, as well as control allocation by considering both the dynamics and kinematics of the vessel and the actuators. In order to account for model mismatch, and external disturbances, the proposed method uses RL and SYSID in order to update the model and NMPC on line and improve the closed loop performance. In order to run the method in real time, asNMPC was used. This reduced the computational delay of solving the optimization problem in each sampling interval, making the method real time feasible. It should however be noted that the proposed control architecture is still more computationally demanding then more traditional methods, which can be considered a drawback of the proposed method. This added complexity does however come with a trade off in terms of optimality and the ability to include complex truster dynamics and physical constraints into the OCP, resulting in better tracking performance then other traditional methods. This is further improved by using RL and SYSID to learn and identify disturbances and modelling errors in order to optimize the closed loop performance.

Based on both simulations and sea trials on both the *ReVolt* and *milliAmpere* platforms we have shown the flexibility of the proposed method. The experimental results also show how using a NMPC for handling both tracking and control al-

location, allows the controller to account for the performance of the vessel over a prediction horizon. This makes the controller preemptive, leading to better tracking performance and less power consumption compared with other methods. The addition of an RL and SYSID update law allows for the control scheme to adapt to the environmental disturbances, and model mismatch in a way that optimizes the closed loop performance of the proposed control scheme. These benefits do however come with some drawbacks, including the computational complexity of the solving OCP, and robustness of the RL and SYSID update laws with respect to disturbances and measurement noise.

For future work, we would like to look more into how to perform more robust and safe RL and SYSID parameter updates, as care must be taken in order to avoid problematic parameter updates caused by for example noisy and inaccurate measurements. This is mostly a problem when running on a physical platform, and for us it was solved using batches of transitions, and sufficiently small learning rate. For future research it would also be interesting to look at different vessel models, including under-actuated vessels. It would also be interesting to look at other problems than tracking, such as for example planning and docking, where the problems have economic cost functions and additional constraints that need to be taken into account. An additional area of potential research is to optimize the implementation of the control scheme in order to improve the computation time. With dedicated hardware, and a dedicated real time NMPC implementations it should be possible to make to significantly improve computation time, allowing for the proposed method to be used on systems requiring faster update rates, more complex models and with a longer prediction horizon.

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.

A. Controller parameters

The parameter values for the *ReVolt* and *milliAmpere* the MPC implementation are given in Table 3 and 4 respectively.

Table 3
List of parameter values for *ReVolt*.

Symbol	Value	Description
$q_{x,y}$	30	Huber penalty weight
q_ψ	50	Heading penalty weight
$\boldsymbol{Q}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix}$	Velocity weight matrix
$\boldsymbol{R}_\alpha$	$\begin{bmatrix} 10^{-2} & 0 & 0 \\ 0 & 10^{-2} & 0 \\ 0 & 0 & 10^{-2} \end{bmatrix}$	Azimuth angle weight matrix
$\boldsymbol{R}_f$	$\begin{bmatrix} 10^{-1} & 0 & 0 \\ 0 & 10^{-1} & 0 \\ 0 & 0 & 10^{-1} \end{bmatrix}$	Thruster force weight matrix
ρ	10^{-5}	Singular value penalty weight
ϵ	10^{-3}	Singular value penalty offset
δ	10	Huber penalty slope
T_s	0.2s	Time step
N	50	OCP shooting intervals
M	1	Batch size

Table 4
List of parameter values for *milliAmpere*.

Symbol	Value	Description
$q_{x,y}$	20	Huber penalty weight
q_ψ	50	Heading penalty weight
$\boldsymbol{Q}$	$\begin{bmatrix} 5 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 50 \end{bmatrix}$	Velocity weight matrix
$\boldsymbol{R}_\alpha$	$\begin{bmatrix} 10^{-1} & 0 \\ 0 & 10^{-1} \end{bmatrix}$	Azimuth angle weight matrix
$\boldsymbol{R}_f$	$\begin{bmatrix} 10^{-5} & 0 \\ 0 & 10^{-5} \end{bmatrix}$	Thruster force weight matrix
δ	10	Huber penalty slope
T_s	0.2s	Time step
N	50	OCP shooting intervals
M	10	Batch size

B. Thruster failure results

During the *milliAmpere* sea trials, an unexpected azimuth failure occurred during one of the tests, see Figure 11. The proposed controller was however still able to perform the desired maneuver, and learn how to compensate for the failure.

References

Aguiar, A.P., Hespanha, J.P., 2007. Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty. IEEE transactions on automatic control 52, 1362–1379. doi:10.1109/TAC.2007.902731.

Aguiar, A.P., Pascoal, A.M., 2007. Dynamic positioning and waypoint tracking of underactuated auvs in the presence of ocean currents. International Journal of Control 80, 1092–1108. doi:10.1080/00207170701268882.

Ahmed, Y.A., Hasegawa, K., 2016. Fuzzy reasoned waypoint controller for automatic ship guidance. IFAC-PapersOnLine 49, 604–609. doi:10.1016/j.ifacol.2016.10.501.

Bertsekas, D.P., 2019. Reinforcement learning and optimal control. Athena Scientific.

Bibuli, M., Bruzzone, G., Caccia, M., Lapierre, L., 2009. Path-following algorithms and experiments for an unmanned surface vehicle. Journal of Field Robotics 26, 669–688. doi:10.1002/rob.20303.

Bitar, G., Martinsen, A.B., Lekkas, A.M., Breivik, M., 2020. Two-stage optimized trajectory planning for asvs under polygonal obstacle constraints: Theory and experiments. IEEE Access 8, 199953–199969. doi:10.1109/ACCESS.2020.3035256.

Burns, R.S., 1995. The use of artificial neural networks for the intelligent optimal control of surface ships. IEEE Journal of Oceanic Engineering 20, 65–72. doi:10.1109/48.380245.

Caccia, M., Bibuli, M., Bono, R., Bruzzone, G., 2008. Basic navigation, guidance and control of an unmanned surface vehicle. Autonomous

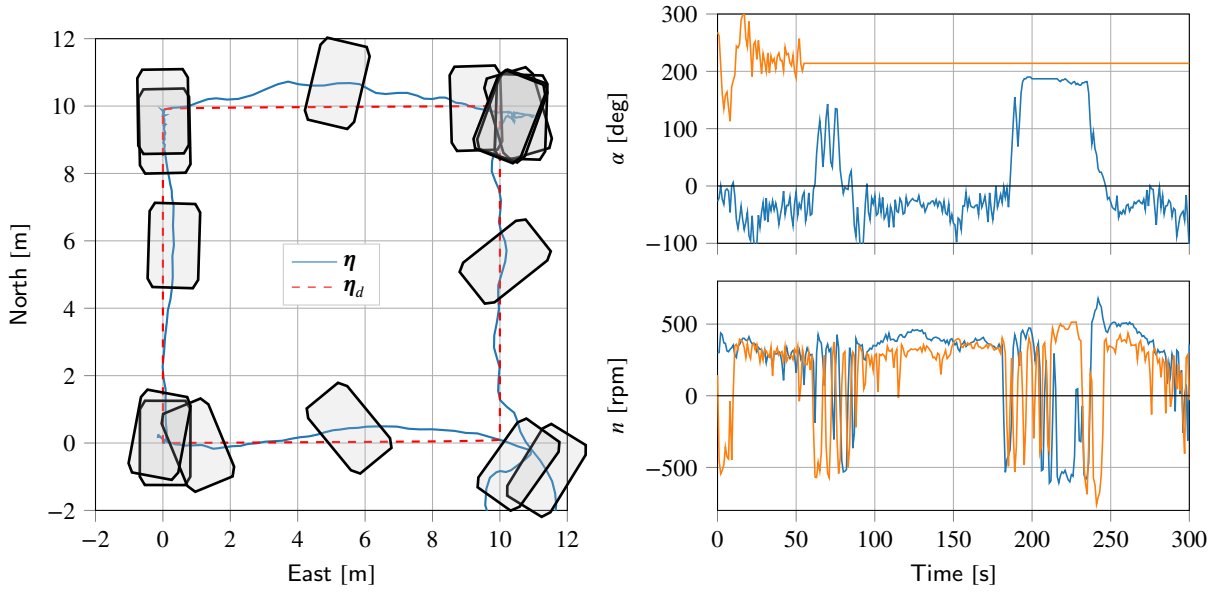

Figure 11: Sea trial results for *milliAmpere*, with azimuth failure at 50 seconds

Robots 25, 349–365. doi:10.1007/s10514-008-9100-0.

Chen, W.H., Ballance, D.J., O'Reilly, J., 2000. Model predictive control of nonlinear systems: computational burden and stability. IEE Proceedings-Control Theory and Applications 147, 387–394. doi:10.1049/ip-cta:20000379.

Diehl, M., Bock, H.G., Schlöder, J.P., 2005. A real-time iteration scheme for nonlinear optimization in optimal feedback control. SIAM Journal on control and optimization 43, 1714–1736. doi:10.1137/S0363012902400713.

Do, K.D., Jiang, Z.P., Pan, J., 2004. Robust adaptive path following of underactuated ships. Automatica 40, 929–944. doi:10.1016/j.automatica.2004.01.021.

Eriksen, B.O.H., 2019. Collision Avoidance and Motion Control for Autonomous Surface Vehicles. Ph.D. thesis. Norwegian University of Science and Technology (NTNU). Trondheim. URL: http://hdl.handle.net/11250/2616394.

Findeisen, R., Allgöwer, F., 2004. Computational delay in nonlinear model predictive control. IFAC Proceedings Volumes 37, 427–432. doi:10.1016/S1474-6670(17)38769-4.

Fossen, T.I., 2011. Handbook of marine craft hydrodynamics and motion control. John Wiley & Sons.

Fossen, T.I., Grovlen, A., 1998. Nonlinear output feedback control of dynamically positioned ships using vectorial observer backstepping. IEEE transactions on control systems technology 6, 121–128. doi:10.1109/87.654882.

Gros, S., Diehl, M., 2013. Nmpc based on huber penalty functions to handle large deviations of quadrature states, in: 2013 American Control Conference, IEEE. pp. 3159–3164. doi:10.1109/ACC.2013.6580317.

Gros, S., Zanon, M., 2017. Penalty functions for handling large deviation of quadrature states in nmpc. IEEE Transactions on Automatic Control 62, 3848–3860. doi:10.1109/TAC.2017.2649043.

Gros, S., Zanon, M., 2019. Data-driven economic nmpc using reinforcement learning. IEEE Transactions on Automatic Control 65, 636–648. doi:10.1109/TAC.2019.2913768.

Hwang, W.Y., 1980. Application of system identification to ship maneuvering. Ph.D. thesis. Massachusetts Institute of Technology.

Johansen, T.A., Fossen, T.I., 2013. Control allocation—a survey. Automatica 49, 1087–1103. doi: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 Transactions on Control Systems Technology 12, 211–216. doi:10.1109/TCST.2003.821952.

Kamalapurkar, R., Walters, P., Rosenfeld, J., Dixon, W., 2018. Reinforcement Learning for Optimal Feedback Control: A Lyapunov-Based Approach. Springer. doi:10.1007/978-3-319-78384-0.

Lefeber, E., Pettersen, K.Y., Nijmeijer, H., 2003. Tracking control of an underactuated ship. IEEE transactions on control systems technology 11, 52–61. doi:10.1109/TCST.2002.806465.

Li, Z., Sun, J., 2011. Disturbance compensating model predictive control with application to ship heading control. IEEE transactions on control systems technology 20, 257–265. doi:10.1109/TCST.2011.2106212.

Liu, C., Zheng, H., Negenborn, R.R., Chu, X., Wang, L., 2015. Trajectory tracking control for underactuated surface vessels based on nonlinear model predictive control, in: International Conference on Computational Logistics, Springer. pp. 166–180. doi:10.1007/978-3-319-24264-4_12.

Martinsen, A.B., Bitar, G., Lekkas, A.M., Gros, S., 2020a. Optimization-based automatic docking and berthing of asvs using exteroceptive sensors: Theory and experiments. IEEE Access 8, 204974–204986. doi:10.1109/ACCESS.2020.3037171.

Martinsen, A.B., Lekkas, A., Gros, S., Glomsrud, J.A., Pedersen, T.A., 2020b. Reinforcement learning-based tracking control of usvs in varying operational conditions. Frontiers in Robotics and AI 7, 32. doi:10.3389/frobt.2020.00032.

Martinsen, A.B., Lekkas, A.M., 2018a. Curved path following with deep reinforcement learning: Results from three vessel models, in: OCEANS 2018 MTS/IEEE Charleston, IEEE. pp. 1–8. doi:10.1109/OCEANS.2018.8604829.

Martinsen, A.B., Lekkas, A.M., 2018b. Straight-path following for underactuated marine vessels using deep reinforcement learning. IFAC-PapersOnLine 51, 329–334. doi:10.1016/j.ifacol.2018.09.502.

Martinsen, A.B., Lekkas, A.M., Gros, S., 2020c. Combining system identification with reinforcement learning-based mpc. arXiv preprint arXiv:2004.03265 .

Mayne, D.Q., Rawlings, J.B., Rao, C.V., Scokaert, P.O., 2000. Constrained model predictive control: Stability and optimality. Automatica 36, 789–814. doi:10.1016/S0005-1098(99)00214-9.

Meyer, E., Heiberg, A., Rasheed, A., San, O., 2020. Colreg-compliant collision avoidance for unmanned surface vehicle using deep reinforcement learning. IEEE Access 8, 165344–165364. doi:10.1109/ACCESS.2020.3022600.

Nocedal, J., Wright, S., 2006. Numerical optimization. Springer Science

& Business Media. doi:10.1007/978-0-387-40065-5.

Pettersen, K.Y., Nijmeijer, H., 2001. Underactuated ship tracking control: theory and experiments. International Journal of Control 74, 1435–1446. doi:10.1080/00207170110072039.

Rajesh, G., Bhattacharyya, S.K., 2008. System identification for nonlinear maneuvering of large tankers using artificial neural network. Applied Ocean Research 30, 256–263. doi:10.1016/j.apor.2008.10.003.

Ramirez, W.A., Leong, Z.Q., Nguyen, H., Jayasinghe, S.G., 2018. Non-parametric dynamic system identification of ships using multi-output gaussian processes. Ocean Engineering 166, 26–36. doi:10.1016/j.oceaneng.2018.07.056.

Rawlings, J.B., Amrit, R., 2009. Optimizing Process Economic Performance Using Model Predictive Control, in: Nonlinear model predictive control. Springer, pp. 119–138. doi:10.1007/978-3-642-01094-1_10.

Skjetne, R., Fossen, T.I., 2001. Nonlinear maneuvering and control of ships, in: MTS/IEEE Oceans 2001. An Ocean Odyssey. Conference Proceedings (IEEE Cat. No. 01CH37295), IEEE. pp. 1808–1815. doi:10.1109/OCEANS.2001.968121.

Sutton, R.S., Barto, A.G., 2018. Reinforcement learning: An introduction. MIT press.

Torben, T.R., Brodtkorb, A.H., Sørensen, A.J., 2019. Control allocation for double-ended ferries with full-scale experimental results, in: Proceedings of the 12th IFAC CAMS, Daejeon, South Korea, pp. 45–50. doi:10.1016/j.ifacol.2019.12.281.

Veksler, A., Johansen, T.A., Borrelli, F., Realfsen, B., 2016. Dynamic positioning with model predictive control. IEEE Transactions on Control Systems Technology 24, 1340–1353. doi:10.1109/TCST.2015.2497280.

Wang, N., Er, M.J., Sun, J.C., Liu, Y.C., 2015. Adaptive robust online constructive fuzzy control of a complex surface vehicle system. IEEE Transactions on Cybernetics 46, 1511–1523. doi:10.1109/TCYB.2015.2451116.

Wang, N., Gao, Y., Zhao, H., Ahn, C.K., 2020. Reinforcement learning-based optimal tracking control of an unknown unmanned surface vehicle. IEEE Transactions on Neural Networks and Learning Systems doi:10.1109/TNNLS.2020.3009214.

Watkins, C.J.C.H., 1989. Learning from delayed rewards. Ph.D. thesis. King's College, Cambridge.

Zanon, M., Gros, S., 2020. Safe reinforcement learning using robust mpc. IEEE Transactions on Automatic Control doi:10.1109/TAC.2020.3024161.

Zavala, V.M., Biegler, L.T., 2009. The advanced-step nmpc controller: Optimality, stability and robustness. Automatica 45, 86–93. doi:10.1016/j.automatica.2008.06.011.

Zheng, H., Negenborn, R.R., Lodewijks, G., 2014. Trajectory tracking of autonomous vessels using model predictive control. IFAC Proceedings Volumes 47, 8812–8818. doi:10.3182/20140824-6-ZA-1003.00767.