




Reinforcement Learning-Based Tracking Control of USVs in Varying Operational Conditions

Andreas B. Martinsen^{1*}, Anastasios M. Lekkas^{1,2}, Sébastien Gros¹, Jon Arne Glomsrud³ and Tom Arne Pedersen³

¹ Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway, ² Centre for Autonomous Marine Operations and Systems, Norwegian University of Science and Technology, Trondheim, Norway, ³ Digital Assurance Program, Group Technology and Research, DNV GL, Trondheim, Norway

 We present a reinforcement learning-based (RL) control scheme for trajectory tracking of fully-actuated surface vessels. The proposed method learns online both a model-based feedforward controller, as well as an optimizing feedback policy in order to follow a desired trajectory under the influence of environmental forces. The method's efficiency is evaluated via simulations and sea trials, with the unmanned surface vehicle (USV) *ReVolt* performing three different tracking tasks: The four corner DP test, straight-path tracking and curved-path tracking. The results demonstrate the method's ability to accomplish the control objectives and a good agreement between the performance achieved in the *ReVolt* Digital Twin and the sea trials. Finally, we include an section with considerations about assurance for RL-based methods and where our approach stands in terms of the main challenges.

Keywords: reinforcement learning, trajectory tracking, optimal control, model-based adaptive control, approximate dynamic programming (ADP), dynamic positioning (DP), autonomous ships, system identification

OPEN ACCESS

Edited by:

Marco Bibuli,
Italian National Research Council, Italy

Reviewed by:

Ning Wang,
Dalian Maritime University, China
Farah Bouakrif,
University of Jijel, Algeria

*Correspondence:

Andreas B. Martinsen
andreas.b.martinsen@ntnu.no

Specialty section:

This article was submitted to
Robotic Control Systems,
a section of the journal
Frontiers in Robotics and AI

Received: 06 November 2019

Accepted: 20 February 2020

Published: 20 March 2020

Citation:

Martinsen AB, Lekkas AM, Gros S,
Glomsrud JA and Pedersen TA (2020)
Reinforcement Learning-Based
Tracking Control of USVs in Varying
Operational Conditions.
Front. Robot. AI 7:32.
doi: 10.3389/frobt.2020.00032

1. INTRODUCTION

Control of marine vehicles is a challenging problem, mostly due to the unpredictable nature of the sea and the difficulty in developing accurate mathematical models to represent the varying marine vehicle dynamics. As a result, considerable research effort has been dedicated to the topic since the early 90's (Fossen, 1994), resulting in a vast literature utilizing ideas from virtually every branch of control engineering: Linear, non-linear, adaptive, intelligent, optimal, fuzzy, and stochastic control approaches, to name a few, have been developed and tested over the years, and many of their properties are well-understood (Hasegawa et al., 1989; Pettersen and Egeland, 1996; Katebi et al., 1997; Fossen, 2000; McGookin et al., 2000; Soetanto et al., 2003; Wang et al., 2015; Do, 2016). Due to the fact that the hydrodynamic coefficients, and consequently the behavior, of a marine vehicle can vary significantly in different speed regimes, a common approach has been to design controllers for specific motion control scenarios. This approach simplifies the vessel modeling process and has led to dynamic positioning (DP) and station keeping controllers for speeds close to zero, and trajectory tracking or path following (depending on whether temporal constraints are considered) controllers when a vessel is in transit mode. Naturally, the main drawback is that, when moving from one speed regime to another, controllers and/or models with different properties are needed. Two well-researched

ways to achieve such performance diversity with conventional methods are to design numerous controllers and switch among them when needed, or to use adaptive approaches. To this end, research effort has been dedicated to developing flexible methods for updating the model parameters by, for instance, using system identification methods or parameter estimation via neural networks (Källström and Åström, 1981; Kallstrom, 1982; Fossen et al., 1996; Sutton et al., 1997; Mišković et al., 2011; Dai et al., 2012; Wang et al., 2017). In the majority of the aforementioned works, model-based approaches exploiting human knowledge on hydrodynamics and the laws of motion were considered.

Reinforcement learning (RL), also known as neuro-dynamic programming or approximate dynamic programming, is a field of research developed by the Artificial Intelligence (AI) community for achieving optimal sequential decision making under system and environment uncertainty. The roots of RL can be traced back to the 60's and a thorough overview of its evolution can be found in Sutton and Barto (2018) and Bertsekas (2019). Contrary to optimal control theory, RL is based on *evaluative*, rather than *instructive*, feedback and comes in different forms, which may or may not include partial knowledge of the environment or the system. The process typically involves hand-engineering a reward function, which assigns a reward, or penalty, to the actions that induce desired, or undesired, outcomes, respectively. An RL algorithm is then assigned to find a policy (or controller, in control engineering terminology) that solves the control objective optimally, given the problem constraints and uncertainties. To sum up, RL algorithms use the reward function as a guide, and through trial and error, learn to model the system and its environment, which then leads to a policy that provides an optimal solution to the assigned problem.

Despite a number of successes for RL on simple problems, including algorithms, such as *Q-learning* and *REINFORCE*, the field has seen limited interest. In recent years there has however been a resurgence of interest due to the development of Deep Reinforcement Learning (DRL), starting with Deep Mind developing the *Deep Q-Network* (DQN) algorithm that achieved superhuman performance in several Atari games (Mnih et al., 2013), followed by Deep Mind's *AlphaGo* algorithm becoming the first computer program to beat a human champion in the game of *Go* (Silver et al., 2016). Since then, DRL has been successful in surpassing all previous computer programs in chess and learning how to accomplish complex robotic tasks (Silver et al., 2017; Andrychowicz et al., 2018). Given DRL's ability to tackle problems with high uncertainty, implementations to motion control scenarios involving marine vessels have been presented recently (Shen and Guo, 2016; Zhang et al., 2016; Pham Tuyen et al., 2017; Yu et al., 2017; Cheng and Zhang, 2018; Martinsen and Lekkas, 2018a,b). In most of these works the authors implemented algorithms pertaining to the class of *actor-critic* RL methods, which involves two parts (Konda and Tsitsiklis, 2000): The *actor*, where the gradient of the performance is estimated and the policy parameters are directly updated in a direction of improvement. The main drawbacks of the actor are that it is prone to variance and the new

gradient is estimated independently of past estimates. The *critic*, learns an approximation of the value function, leading to an approximate solution to the Bellman or Hamilton-Jacobi-Bellman equation, which then is expected to prescribe a near-optimal policy. The critic's main drawback is that it lacks reliable guarantees in terms of near-optimality of the resulting policy. The actor-critic approach involves the actor improving the policy parameters' estimation based on the approximations learned by the critic. In the case of DRL, one main novelty was the use of two DNNs as function approximators of the policy and the value function, which resulted in considerably improved performance compared to previous approaches. However, DNNs have drawbacks, with some of the most important being lack of transparency and interpretability, lack of robustness, and inability to generalize to situations beyond their past experiences.

In this paper, we follow and extend the work by Kamalapurkar et al. (2018) and Walters et al. (2018) in order to **build a trajectory tracking control system for a fully-actuated unmanned surface vehicle (USV)**. Conceptually, the approach is quite similar to dynamic positioning (DP) (Sørensen, 2011), but extends to higher velocity operational domains, while also trying to **optimize tracking performance and compensate for environmental forces** (Lekkas and Fossen, 2014). The method combines elements from reinforcement learning, Lyapunov stability theory and **system identification**. We assume the **structure of the vessel model is known** but all of its **parameters** are unknown and have to be estimated online, as well as **updated** accordingly when the operational **conditions** change. Then we derive the tracking error dynamics for a generic reference trajectory and a stabilizing parametric control law (the *actor*), whose parameters are estimated **during operation**.

In order to validate the control scheme, the proposed method was tested in both in simulations, and on a physical model of DNV GL's ReVolt platform.

2. REINFORCEMENT LEARNING-BASED TRAJECTORY TRACKING

In this section we will derive a trajectory tracking control system for fully-actuated USVs. Since the approach is a model based reinforcement learning approach, we will start by looking at how ASVs can be modeled, and how the models can be approximated online using system identification. We will **derive a feedforward control law for tracking the desired trajectory, and a feedback control law based on reinforcement learning**, for controlling the drift of the vessel in a way that **minimizes a given cost function**.

2.1. Vessel Model



The mathematical model used to describe the system can then be kept reasonably simple by limiting it to the planar position and orientation of the vessel. The motion of a surface vessel can be represented by the pose vector $\eta = [x, y, \psi]^T \in \mathbb{R}^2 \times \mathbb{S}$, and velocity vector $v = [u, v, r]^T \in \mathbb{R}^3$. Here, (x, y) 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 in Fossen (2011) we can describe a 3-DOF vessel model as follows

$$\begin{aligned} \dot{\eta} &= J(\eta)v, \\ M\dot{v} + D(v)v + C(v)v &= \tau_{\text{Thrust}} + \tau_{\text{Environment}} \end{aligned} \quad (1)$$

where $M \in \mathbb{R}^{3 \times 3}$, $D(v) \in \mathbb{R}^{3 \times 3}$, $C(v) \in \mathbb{R}^{3 \times 3}$, $\tau_{\text{Thrust}}, \tau_{\text{Environment}} \in \mathbb{R}^3$ and $J(\eta) \in SO(3)$ are the inertia matrix, damping matrix, coriolis matrix, control input vector, environmental forces, and rotation matrix, respectively. The rotational matrix $J(\eta) \in SO(3)$ is given by

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

and is the rotation from the body frame to the earth-fixed North East Down (NED) reference frame.

$$D(v) = \begin{bmatrix} -X_u - X_{|u|u} \cdot |u| & 0 & 0 \\ 0 & -Y_v - Y_{|v|v} \cdot |v| - Y_{|r|v} \cdot |r| & -Y_r - Y_{|v|r} \cdot |v| - Y_{|r|r} \cdot |r| \\ 0 & -N_v - N_{|v|v} \cdot |v| - N_{|r|v} \cdot |r| & -N_r - N_{|v|r} \cdot |v| - N_{|r|r} \cdot |r| \end{bmatrix} \quad (5)$$

2.2. Model Approximation

While the structure of a vessel model, as given above, is well-known, the model parameters are often difficult to find. For our approach we wish to make as few assumptions on the parameters of the vessel model as possible, and use online system identification in order to model the vessel based on gathered data. For this we assume that we know the model structure as given in (1), but that the model parameters are unknown. Splitting the model into a known and unknown part, we get the following:

$$\dot{x} = f_{\theta}(x) + f_1(x) + g(x)u \quad (3)$$

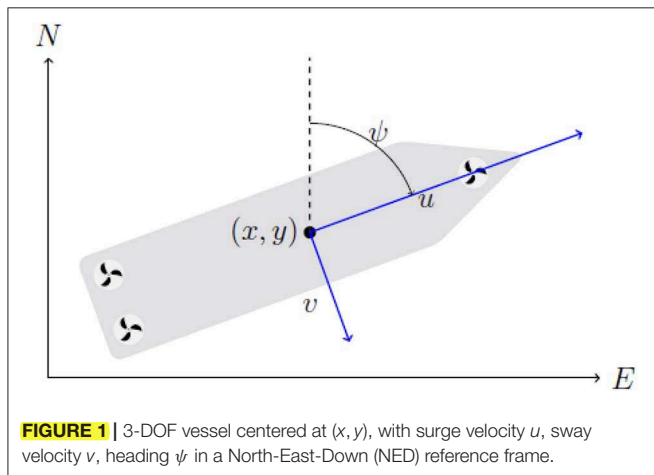


FIGURE 1 | 3-DOF vessel centered at (x, y) , with surge velocity u , sway velocity v , heading ψ in a North-East-Down (NED) reference frame.

where $f_1(x)$ and $g(x)$ are known, and $f_{\theta}(x)$ is unknown. For the vessel model in (1), with the state vector $x = [\eta, v]^T$ and the control vector $u = \tau_{\text{Thrust}}$. We have the following:

$$\begin{aligned} f_{\theta}(x) &= \begin{bmatrix} 0_{3 \times 1} \\ -M^{-1}(D(v)v + C(v)v - \tau_{\text{Environment}}) \end{bmatrix} \\ f_1(x) &= \begin{bmatrix} J(\eta)v \\ 0_{3 \times 1} \end{bmatrix} \\ g(x) &= \begin{bmatrix} 0_{3 \times 3} \\ M^{-1} \end{bmatrix} \end{aligned}$$

hence we assume the mass matrix is known, but the damping and coriolis matrix are unknown. For the damping and coriolis matrices we assume the vessel has port starboard symmetry, from Fossen (2011) this gives the following structure.

$$C(v) = \begin{bmatrix} 0 & 0 & Y_{\dot{v}} \cdot v + Y_{\dot{r}} \cdot r \\ 0 & 0 & -X_{\dot{u}} \cdot u \\ -Y_{\dot{v}} \cdot v + Y_{\dot{r}} \cdot r & X_{\dot{u}} \cdot u & 0 \end{bmatrix} \quad (4)$$

For the damping matrix $D(v)$, both linear and non-linear terms are included. The linear terms are important for low speed maneuvering and station keeping, while ensuring the velocity converges exponentially to zero. The non-linear terms are required as they dominate at higher velocities. This ensures that the model is able to handle a large range of velocities, i.e., it can be used for both high speed trajectory tracking and low speed station keeping and dynamic positioning. For the coriolis matrix, we use only the added mass terms. Since the structure of the rigid body, and added mass is the same for the coriolis matrix, the coriolis matrix given above will be able to capture both the added mass and rigid body dynamics.

In addition to learning the vessel dynamics, we also wanted to be able to compensate for environmental forces. In order to allow for the environmental forces to be learned, they are modeled as an additional unknown pressure vector $p_{\text{env}}^{\text{NED}} = [p_{\text{North}}, p_{\text{East}}, 0]^T$ assumed constant in the NED frame. The resulting force in the body frame is then assumed to be proportional to the cross sectional area of the vessel times the pressure in the body frame, giving the following relationship.

$$\tau_{\text{Environment}}^{\text{body}} = \text{diag}([w, l, 0])J^T(v)p_{\text{Environment}}^{\text{NED}} \quad (6)$$

where w and l are the width and length of the vessel, respectively, note that for better accuracy calculated pressure coefficients based on the design of the hull may be used instead of the width and length. The unknown parameters are

$$\begin{aligned} \theta &= [X_{\dot{u}}, Y_{\dot{v}}, Y_{\dot{r}}, X_{\dot{u}}, Y_{\dot{v}}, Y_{\dot{r}}, N_{\dot{v}}, N_{\dot{r}}, X_{|u|u}, Y_{|v|v}, Y_{|v|r}, Y_{|r|r}, \\ &N_{|v|v}, N_{|v|r}, N_{|r|r}, N_{|r|r}, p_{\text{North}}, p_{\text{East}}]^T \end{aligned} \quad (7)$$

and the function $f_\theta(\mathbf{x})$ can be written as a linear function in θ :

$$f_\theta(\mathbf{x}) = Y(\mathbf{x})\theta \quad (8)$$

where $Y(\mathbf{x})$ is:

$$Y(\mathbf{x}) = \begin{bmatrix} \mathbf{0}_{3 \times 3} \\ -\mathbf{M}^{-1} \end{bmatrix} \begin{bmatrix} 0 & v \cdot r & r^2 & -u & 0 & 0 & 0 & 0 & -|u|u & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & w \cos \psi & w \sin \psi \\ -u \cdot r & 0 & 0 & 0 & -v & -r & 0 & 0 & 0 & -|v|v & -|v|r & -|r|v & -|r|r & 0 & 0 & 0 & 0 & -l \sin \psi & l \cos \psi \\ u \cdot v & -v \cdot u & -r \cdot u & 0 & 0 & 0 & -v & -r & 0 & 0 & 0 & 0 & -|v|v & -|v|r & -|r|v & -|r|r & 0 & 0 \end{bmatrix} \quad (9)$$

We therefore obtain the following parametric model:

$$\dot{\mathbf{x}} = Y(\mathbf{x})\theta + f_1(\mathbf{x}) + g(\mathbf{x})\mathbf{u}, \quad (10)$$

which is linear in the parameters θ .

2.2.1. Model Assumptions

- The vessel is port starboard symmetric, with a structure as given as in (Figure 1).
- The vessel dampening is linear and quadratic with respect to the linear and angular velocity.
- Environmental forces are constant in the NED frame, and proportional to vessel cross section.
- The vessel is fully actuated.

2.3. Trajectory Tracking

In this section we will develop an adaptive feedforward control law which given a time-varying trajectory, finds the control inputs required to follow the trajectory, given the model approximation found in the previous section.

When the control objective is to track a bounded continuously differentiable signal \mathbf{x}_d , the dynamics of the tracking error $\mathbf{e} = \mathbf{x} - \mathbf{x}_d$ can be written as

$$\dot{\mathbf{e}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u} - \dot{\mathbf{x}}_d \quad (11)$$

Assuming $g(\mathbf{x})$ is bounded and has full column rank for all \mathbf{x} (Kamalapurkar et al., 2018), then the system is controllable, which in this case holds as the vessel is fully actuated. This gives the feedforward control for the reference trajectory as:

$$\mathbf{u}_d(\mathbf{x}_d, \dot{\mathbf{x}}_d) = g^+(\mathbf{x}_d)(\dot{\mathbf{x}}_d - f(\mathbf{x}_d)) \quad (12)$$

where g^+ is the left Moore–Penrose pseudo-inverse, given as $g^+ = (g^\top g)^{-1} g^\top$. Using a reference model $\dot{\mathbf{x}}_d = h_d(\mathbf{x}_d)$, the feedforward control for the reference trajectory can be written as:

$$\mathbf{u}_d(\mathbf{x}_d) = g^+(\mathbf{x}_d)(h_d(\mathbf{x}_d) - f(\mathbf{x}_d)) \quad (13)$$

We can then formulate the tracking problem as the following time-invariant optimal control problem.

$$\underbrace{\begin{bmatrix} \dot{\mathbf{e}} \\ \dot{\mathbf{x}}_d \end{bmatrix}}_{\xi} = \underbrace{\begin{bmatrix} f(\mathbf{e} + \mathbf{x}_d) + g(\mathbf{e} + \mathbf{x}_d)\mathbf{u}_d(\mathbf{x}_d) \\ h_d(\mathbf{x}_d) \end{bmatrix}}_{F(\xi)} + \underbrace{\begin{bmatrix} g(\mathbf{e} + \mathbf{x}_d) \\ 0 \end{bmatrix}}_{G(\xi)} \pi \quad (14)$$

Where π is an input correction for the drift dynamics, which we will define in the next section. Given the parametric model in (10), the parametric version of the tracking problem is given as:

$$\underbrace{\begin{bmatrix} \dot{\mathbf{e}} \\ \dot{\mathbf{x}}_d \end{bmatrix}}_{\xi} = \underbrace{\begin{bmatrix} Y(\mathbf{e} + \mathbf{x}_d)\theta + f_1(\mathbf{e} + \mathbf{x}_d) + g(\mathbf{e} + \mathbf{x}_d)\mathbf{u}_d(\mathbf{x}_d; \theta) \\ h_d(\mathbf{x}_d) \end{bmatrix}}_{F(\xi; \theta)} + \underbrace{\begin{bmatrix} g(\mathbf{e} + \mathbf{x}_d) \\ 0 \end{bmatrix}}_{G(\xi)} \pi \quad (15)$$

where the parametric feedforward control for the reference trajectory $\mathbf{u}_d(\mathbf{x}_d; \theta)$ is given as:

$$\mathbf{u}_d(\mathbf{x}_d; \theta) = g^+(\mathbf{x}_d)(h_d(\mathbf{x}_d) - Y(\mathbf{x}_d)\theta - f_1(\mathbf{x}_d)) \quad (16)$$

Given the formulation above, with the feedforward control for the reference trajectory $\mathbf{u}_d(\mathbf{x}_d)$, and the optimal model parameters θ^* , the exact feedforward control for the reference trajectory is possible to compute. The dynamics above guarantee trajectory tracking when $\dot{\mathbf{e}} = 0$, i.e., when the tracking error is zero. When the tracking error is not zero however, we need to control the drift dynamics in order to ensure convergence to the desired trajectory by designing the feedback control $\pi(t)$ such that $\lim_{t \rightarrow \infty} e(t) = 0$. The objective of the optimal control problem is to design the feedback control law $\pi(t)$ such that it minimizes a given cost function.

2.4. Approximate Optimal Control of Drift Dynamics

In the previous section we developed a feedforward control law $\mathbf{u}_d(\mathbf{x}_d; \theta)$ for tracking a desired trajectory. Due to inaccuracies in model approximation and disturbances, using only the feedforward control law, the vessel will experience drift. In order to compensate for the inevitable drift, we will in this section develop a feedback control law $\pi(\cdot)$, which controls the drift dynamics in a way that optimizes a given cost function. We will additionally show **how the parameters of the feedback control law can be learned by using reinforcement learning**.

The optimal control problem we wish to solve is that of minimizing the cost function:

$$J(\xi, \pi) = \int_{t_0}^{\infty} r(\xi(\tau), \pi(\tau)) d\tau \quad (17)$$

Where $r(\cdot)$ is scalar function defining the local cost, and should not be confused with the yaw rate. The cost function is defined as:

$$r(\xi, \pi) = Q(\xi) + \pi^\top R \pi \quad (18)$$

where $\mathbf{R} > 0$ is a positive definite symmetric matrix. And $Q(\boldsymbol{\zeta})$ is a positive definite function. Assuming that a minimizing control policy $\boldsymbol{\pi}(\cdot)$ exists, the optimal value function is given as:

$$V^*(\boldsymbol{\zeta}) = \min_{\boldsymbol{\pi}(\tau), \tau \in [t_0, \infty)} \int_{t_0}^{\infty} r(\boldsymbol{\zeta}(\tau), \boldsymbol{\pi}(\tau)) d\tau \quad (19)$$

We can now note that for a small time step Δt , the above expression can be formulated as:

$$V^*(\boldsymbol{\zeta}(t)) = \min_{\boldsymbol{\pi}(\tau), \tau \in [t, t+\Delta t)} \int_t^{t+\Delta t} r(\boldsymbol{\zeta}(\tau), \boldsymbol{\pi}(\tau)) d\tau + V^*(\boldsymbol{\zeta}(t+\Delta t))$$

Taking the limit of this as $\Delta t \rightarrow 0$, for the optimal value function under the optimal policy, we get (Doya, 2000):

$$V^*(\boldsymbol{\zeta}(t)) = \min_{\boldsymbol{\pi}(t)} r(\boldsymbol{\zeta}(t), \boldsymbol{\pi}(t)) + V^*(\boldsymbol{\zeta}(t)) + \dot{V}^*(\boldsymbol{\zeta}(t))$$

Simplifying this we get the Hamilton-Jacobi-Bellman (HJB) equation for the optimal control problem as follows:

$$\begin{aligned} H^* &= \dot{V}^*(\boldsymbol{\zeta}) + r(\boldsymbol{\zeta}, \boldsymbol{\pi}^*(\boldsymbol{\zeta})) \\ &= \nabla_{\boldsymbol{\zeta}} V^*(\boldsymbol{\zeta})^\top \dot{\boldsymbol{\zeta}} + r(\boldsymbol{\zeta}, \boldsymbol{\pi}^*(\boldsymbol{\zeta})) \\ &= \nabla_{\boldsymbol{\zeta}} V^*(\boldsymbol{\zeta})^\top (F(\boldsymbol{\zeta}) + G(\boldsymbol{\zeta})\boldsymbol{\pi}^*(\boldsymbol{\zeta})) + r(\boldsymbol{\zeta}, \boldsymbol{\pi}^*(\boldsymbol{\zeta})) = 0 \end{aligned} \quad (20)$$

Where H^* , $\boldsymbol{\pi}^*$ and V^* is the optimal hamiltonian, policy and value function, respectively. From calculus of variation (Liberzon, 2011) we have the Hamiltonian minimization condition, which states that a value function V is the optimal Value function if and only if there exists a controller $\boldsymbol{\pi}(\cdot)$ and trajectory $\boldsymbol{\zeta}(\cdot)$ under $\boldsymbol{\pi}(\cdot)$ satisfy the equation:

$$\begin{aligned} &\nabla_{\boldsymbol{\zeta}} V(\boldsymbol{\zeta})^\top (F(\boldsymbol{\zeta}) + G(\boldsymbol{\zeta})\boldsymbol{\pi}(\boldsymbol{\zeta})) + r(\boldsymbol{\zeta}, \boldsymbol{\pi}(\boldsymbol{\zeta})) \\ &= \min_{\hat{\boldsymbol{\pi}} \in U} \{ \nabla_{\boldsymbol{\zeta}} V(\boldsymbol{\zeta})^\top (F(\boldsymbol{\zeta}) + G(\boldsymbol{\zeta})\hat{\boldsymbol{\pi}}(\boldsymbol{\zeta})) + r(\boldsymbol{\zeta}, \hat{\boldsymbol{\pi}}(\boldsymbol{\zeta})) \} \end{aligned} \quad (21)$$

The necessary conditions for this to hold are:

$$\nabla_{\boldsymbol{\pi}} \left(\nabla_{\boldsymbol{\zeta}} V(\boldsymbol{\zeta})^\top (F(\boldsymbol{\zeta}) + G(\boldsymbol{\zeta})\boldsymbol{\pi}(\boldsymbol{\zeta})) + r(\boldsymbol{\zeta}, \boldsymbol{\pi}(\boldsymbol{\zeta})) \right) = 0 \quad (22)$$

which gives the closed form solution of the optimal controller as:

$$\begin{aligned} G^\top(\boldsymbol{\zeta}) (\nabla_{\boldsymbol{\zeta}} V(\boldsymbol{\zeta})) + \nabla_{\boldsymbol{\pi}} r(\boldsymbol{\zeta}, \boldsymbol{\pi}) &= 0 \Leftrightarrow \\ 2\mathbf{R}\boldsymbol{\pi} &= -G^\top(\boldsymbol{\zeta}) (\nabla_{\boldsymbol{\zeta}} V(\boldsymbol{\zeta})) \Leftrightarrow \\ \boldsymbol{\pi}^*(\boldsymbol{\zeta}) &= -\frac{1}{2}\mathbf{R}^{-1}G^\top(\boldsymbol{\zeta}) (\nabla_{\boldsymbol{\zeta}} V(\boldsymbol{\zeta})) \end{aligned} \quad (23)$$

Hence assuming that an optimal controller exists, the closed form solution given by the HJB equation is given by (23). Note that the value function is assumed time independent, and hence we are looking for a stationary solution of the HJB equation. This holds true, as the reformulation into a trajectory tracking problem (15) gives a time independent system.

The Universal Approximation theorem (Kamalapurkar et al., 2018, Property 2.3) states that a single layer neural network can

simultaneously approximate a function and its derivative given a sufficiently large number of basis functions. Using this, we can approximate any continuous function as:

$$V(\mathbf{x}) = \mathbf{W}^\top \boldsymbol{\sigma}(\mathbf{x}) + \epsilon(\mathbf{x}) \quad (24)$$

where \mathbf{W} is the weighting matrix, $\boldsymbol{\sigma}(\mathbf{x})$ is the vector of basis functions, and $\epsilon(\mathbf{x})$ is the approximation error, which can be made arbitrarily small by increasing the number of basis functions. Note that the basis functions can here be chosen to be any parameterization, such as Radial-Basis functions, polynomials or even a Fourier series. Using this we can represent the value function as a neural network which is linear in the parameters, giving the optimal value function:

$$V^*(\boldsymbol{\zeta}) = \mathbf{W}^\top \boldsymbol{\sigma}(\boldsymbol{\zeta}) + \epsilon(\boldsymbol{\zeta}) \quad (25)$$

and the optimal policy as a feedback control law on the form:

$$\boldsymbol{\pi}^*(\boldsymbol{\zeta}) = -\frac{1}{2}\mathbf{R}^{-1}G^\top(\boldsymbol{\zeta}) \left(\nabla_{\boldsymbol{\zeta}} \boldsymbol{\sigma}(\boldsymbol{\zeta})^\top \mathbf{W} + \nabla_{\boldsymbol{\zeta}} \epsilon(\boldsymbol{\zeta}) \right) \quad (26)$$

By making the parameterizations sufficiently rich, we make the approximation error small. We can then use the approximations given below, for the value function and control policy, respectively.

$$\hat{V}(\boldsymbol{\zeta}; \hat{\mathbf{W}}_c) = \hat{\mathbf{W}}_c^\top \boldsymbol{\sigma}(\boldsymbol{\zeta}) \quad (27)$$

$$\hat{\boldsymbol{\pi}}(\boldsymbol{\zeta}; \hat{\mathbf{W}}_a) = -\frac{1}{2}\mathbf{R}^{-1}G^\top(\boldsymbol{\zeta}) \nabla_{\boldsymbol{\zeta}} \boldsymbol{\sigma}(\boldsymbol{\zeta})^\top \hat{\mathbf{W}}_a \quad (28)$$

In order to find the parameters $\hat{\mathbf{W}}_c$ and $\hat{\mathbf{W}}_a$, we will in the next section find update laws, based on reinforcement learning, to be able to optimize performance online.

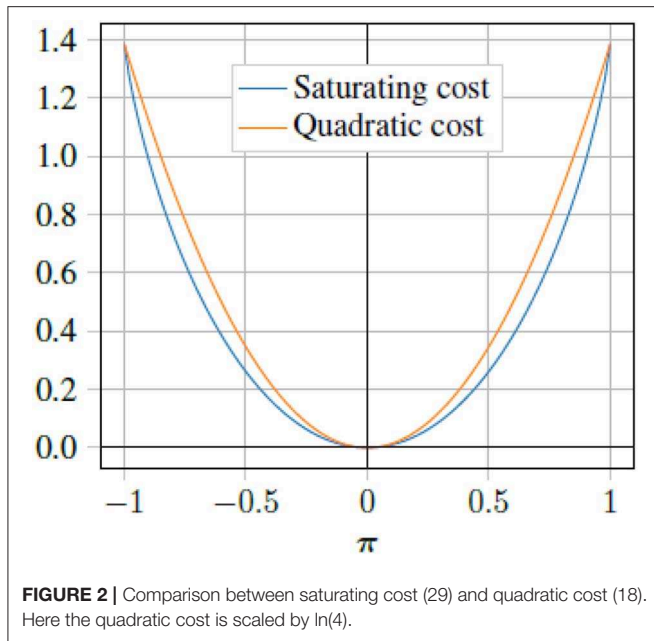
Unfortunately, **policy (28) does not account for the saturating constraints**, such as the maximum force the actuators of the physical vessel can produce. In order to account for the actuator limitations, we propose a different control policy which uses a **saturating function** (Doya, 2000) in order to avoid this problem. Using the following cost function:

$$r(\boldsymbol{\zeta}, \boldsymbol{\pi}) = Q(\boldsymbol{\zeta}) + 2 \sum_{i=1}^m r_i \int_0^{\pi_i} \tanh^{-1}(\xi) d\xi \quad (29)$$

where r_i is the i th entry of the diagonal of \mathbf{R} , i.e., $\mathbf{R} = \text{diag}([r_1, r_2 \dots r_m])$. **Figure 2** shows a comparison of the saturating input cost, and a pure quadratic cost. Performing the same analysis as for the quadratic penalty, we can get the following saturating control law:

$$\boldsymbol{\pi}^*(\boldsymbol{\zeta}) = -\tanh \left(\frac{1}{2}\mathbf{R}^{-1}G^\top(\boldsymbol{\zeta}) (\nabla_{\boldsymbol{\zeta}} V(\boldsymbol{\zeta})) \right) \quad (30)$$

Since $\tanh(\cdot)$ saturates at ± 1 , this means that the feedback control law $\boldsymbol{\pi}$ will saturate at ± 1 , the outputs can then be easily scaled to fit other bounds. It can be shown that since $\tanh(\cdot)$ is a monotonically increasing continuously differentiable function,



the control law satisfies the first order necessary conditions, and the second order sufficient conditions of the Hamiltonian minimization condition. This means that if an optimal controller exists the closed form solution is given by (30). Using an approximation we get the following approximate optimal policy

$$\hat{\pi}(\zeta; \hat{W}_a) = -\tanh\left(\frac{1}{2}R^{-1}G^T(\zeta)\nabla_{\zeta}\sigma(\zeta)^T\hat{W}_a\right) \quad (31)$$

It should be noted, that while the policy in (31) uses a value function approximation in order to approximate the optimal policy, the parameters \hat{W}_a are not the same as the parameters \hat{W}_c in the value function approximation in (27). In this way we can separate the learning of the policy and value function, this is known as an actor critic method, where the value function is known as the critic, and the policy is known as an actor. The intuitive reason for doing this, is that it allows the critic to learn the value function resulting from the behavior of the policy, and in this way it can critique the policy. Similarly, the policy or actor, can learn to improve its performance based on the criticism of the critic. How the learning is performed is further discussed in the next section.

2.5. Update Laws

Now that we have expressed the control laws $u_d(x_d; \hat{\theta})$, $\hat{\pi}(\zeta; \hat{W}_a)$ and value function $\hat{V}(\zeta; \hat{W}_c)$, the challenge becomes finding update laws for the parameters of the system identification $\hat{\theta}$, the critic \hat{W}_c and the actor \hat{W}_a . For the model parameters $\hat{\theta}$, we will use methods from system identification and adaptive control, to try to optimize the fit between the parameterized model, and the observed vessel states. For the actor and critic parameters \hat{W}_a and \hat{W}_c , we will use model based reinforcement learning to find the parameters that gives the optimal value function, and consequently the optimal feedback control policy.

For the system identification parameters $\hat{\theta}$, the goal is to find the parameters for which the model behaves as similarly as possible to the observed behavior. Running our physical system, and collecting observations (\dot{x}_i, x_i, u_i) $i \in 1, 2, \dots, N$, we can formulate a least squares optimization problem for finding the parameters that minimize the difference between the observed state derivative \dot{x}_i and the parametric model (3) as follows.

$$\theta^* = \arg \min_{\hat{\theta}} \sum_{i=1}^N \frac{1}{2} \underbrace{\|\dot{x}_i - Y(x_i)\hat{\theta} - f_1(x_i) - g(x_i)u_i\|_2^2}_{L(\hat{\theta})}$$

This is a linear least squares optimization problem for which there exists a closed form solution, however we can also solve the problem by performing stochastic gradient decent on the parameters $\hat{\theta}$, as follows:

$$\hat{\theta} \leftarrow \hat{\theta} - \nabla_{\hat{\theta}} L(\hat{\theta})$$

The gradient decent law above, works in discrete iteration, however we can reformulate it as an ordinary differential equation (ODE). Doing some further changes motivated by the stability analysis of the convergence of the parameter estimates, we get the concurrent learning based approach proposed in Chowdhary and Johnson (2011b) as:

$$\begin{aligned} \dot{\hat{\theta}}(t) = & \Gamma_{\theta} Y^T(x(t))\tilde{x}(t) + \frac{k_{\theta}}{N} \Gamma_{\theta} \sum_{i=1}^N Y^T(x_i) (\dot{x}_i - f_1(x_i) \\ & - g(x_i)u_i - Y(x_i)\hat{\theta}) \end{aligned} \quad (32)$$

where Γ_{θ} is a parameter weight matrix, and k_{θ} is a scalar weight factor. Assuming that the prerecorded data is sufficiently rich such that the matrix $\sum_{i=1}^N Y^T(x_i)Y(x_i)$ is full rank, the parameter error can be shown to converge. As the convergence rate of the system identifier is proportional to the minimum singular value of $\sum_{i=1}^N Y^T(x_i)Y(x_i)$, replacing data in the data stack can be done by using a singular value maximizing algorithm (Chowdhary and Johnson, 2011a) in order to get faster convergence. Note, that since we are assuming a sufficiently rich prerecorded data set, we no longer need persistence of excitation (PE), in order to guarantee parameter convergence.

In order to find the update laws for the critic or value function parameters \hat{W}_c , we need a way of evaluating the optimality of the value function given the current parameters. For this we look back at the HJB Equation (20) given as:

$$0 = r(\zeta, \pi^*(\zeta)) + \nabla_{\zeta} V^*(\zeta)^T (F(\zeta) + G(\zeta)\pi^*(\zeta))$$

Substituting the estimates \hat{V} and $\hat{\pi}$ for the optimal value function V^* and optimal policy π , we can formulate the Bellman error as the error in the HJB equation as follows:

$$\begin{aligned} \delta(\zeta; \hat{\theta}, \hat{W}_c, \hat{W}_a) = & \underbrace{Q(\zeta) + \hat{\pi}^T(\zeta; \hat{W}_a)R\hat{\pi}(\zeta; \hat{W}_a)}_{r(\zeta, \hat{\pi}(\zeta; \hat{W}_a))} \\ & + \nabla_{\zeta} \hat{V}(\zeta; \hat{W}_c)^T (F(\zeta; \hat{\theta}) + G(\zeta)\hat{\pi}(\zeta; \hat{W}_a)) \end{aligned} \quad (33)$$

The Bellman error can intuitively be thought of as the error between the optimal value function under the policy, and the estimates. Since the goal for the value function or critic is to find the parameters W_c that best approximates the value function, a natural choice becomes to find the parameters that minimize the bellman error. With reinforcement learning we can use a data stack of prerecorded state transitions $\xi_i(t) = [x_i - x_{d,i}, x_{d,i}]^T$ $i \in 1, 2, \dots, N$, to formulate the following optimization problem:

$$\min_{\hat{W}_c} \sum_{i=1}^N \frac{1}{2} \delta(\xi_i; \hat{\theta}, \hat{W}_c, \hat{W}_a)^2$$

This is a non-linear optimization problem, but we may again use a methods like gradient decent in order to iteratively learn parameters that improve the optimization problem given above. Writing the gradient decent in terms of an ODE, and making some changes motivated by a stability analysis (Kamalapurkar et al., 2018). A least-squares update law with forgetting factor (Ioannou and Sun, 2012) can be formulated for the critic as follows:

$$\begin{aligned} \dot{\hat{W}}_c(t) &= -k_{c,1} \Gamma(t) \frac{\omega(\xi(t), t)}{\rho(\xi(t), t)} \hat{\delta}(\xi(t), t) - \frac{k_{c,2}}{N} \Gamma(t) \\ &\quad \sum_{i=1}^N \frac{\omega(\xi_i(t), t)}{\rho_i(\xi_i(t), t)} \hat{\delta}(\xi_i(t), t) \end{aligned} \quad (34)$$

$$\dot{\Gamma}(t) = \begin{cases} \beta \Gamma(t) - k_{c,1} \Gamma(t) \frac{\omega(\xi(t), t) \omega^T(\xi(t), t)}{\rho^2(\xi(t), t)} \Gamma(t) & \text{If } \|\Gamma\| \leq \bar{\Gamma} \\ 0 & \text{Otherwise} \end{cases} \quad (35)$$

In critic update law above $k_{c,1}$ and $k_{c,2}$ are scalar learning rates, while Γ is an adaptive weight matrix, and β is a scalar forgetting factor, which controls how previous data samples are discounted. For brevity of notation we used the functions $\omega(\cdot)$, $\rho(\cdot)$, and $\hat{\delta}(\cdot)$ defined as:

$$\begin{aligned} \omega(\xi, t) &= \nabla_{\xi} \sigma(\xi) \left(F(\xi; \hat{\theta}(t)) + G(\xi) \hat{\pi}(\xi; \hat{W}_a(t)) \right) \\ \rho(\xi, t) &= 1 + \omega^T(\xi, t) \Gamma(t) \omega(\xi, t) \\ \hat{\delta}(\xi, t) &= \delta(\xi; \hat{\theta}(t), \hat{W}_c(t), \hat{W}_a(t)) \end{aligned}$$

Here, ω can be considered a regressor vector, while ρ is a normalization factor, and $\hat{\delta}$ the Bellman error.

The actor update law (36) is chosen such that it learns from the critic, while at the same time trying to stay close to the initial control law.

$$\dot{\hat{W}}_a(t) = \text{proj} \left(-k_{a,1} (\hat{W}_a(t) - \hat{W}_c(t)) - k_{a,2} (\hat{W}_a(t) - W_0) \right) \quad (36)$$

In the actor update law above, the first term will make the actor parameters follow the critic parameters, while the second term will try to keep the actor parameters close to the initial parameters W_0 . $k_{a,1}$ and $k_{a,2}$ are scalar learning rates for the two terms. A smooth projection (Ioannou and Sun, 2012) is added such

that the actor weights are within a predefined region, for which the control law is stable. Any smooth projection can be chosen, however we chose a projection ensuring the actor weights were bounded within a region of the initial weights W_0 .

2.6. Stability Analysis

For the system identification parameters θ , we consider the candidate Lyapunov function:

$$V_p(x) = \tilde{\theta}^T \Gamma_{\theta}^{-1} \tilde{\theta}, \quad (37)$$

where $\tilde{\theta} = \hat{\theta} - \theta^*$ is the difference between the predicted and optimal model parameters. Assuming the system is time invariant (including time invariant environmental forces in the NED frame), and given a positive definite weighting matrix Γ_{θ} . The time derivative of the candidate Lyapunov function is:

$$\begin{aligned} \dot{V}_p(x) &= 2\tilde{\theta}^T \Gamma_{\theta}^{-1} \dot{\tilde{\theta}} \\ &= 2\tilde{\theta}^T \Gamma_{\theta}^{-1} \Gamma_{\theta} Y(x)^T \tilde{x} + \frac{2k_{\theta}}{N} \tilde{\theta}^T \Gamma_{\theta}^{-1} \Gamma_{\theta} \sum_{i=1}^N Y^T(x_i) \tilde{x}_i \end{aligned} \quad (38)$$

Using the fact that: $\tilde{x} = \dot{x} - f_1(x) - Y(x)\hat{\theta} - g(x)\tau = -Y(x)\tilde{\theta}$ we get:

$$\dot{V}_p(x) = -2\tilde{x}^T \tilde{x} - \frac{2k_{\theta}}{N} \tilde{\theta}^T \sum_{i=1}^N (Y^T(x_i) Y(x_i)) \tilde{\theta} \leq 0, \quad (39)$$

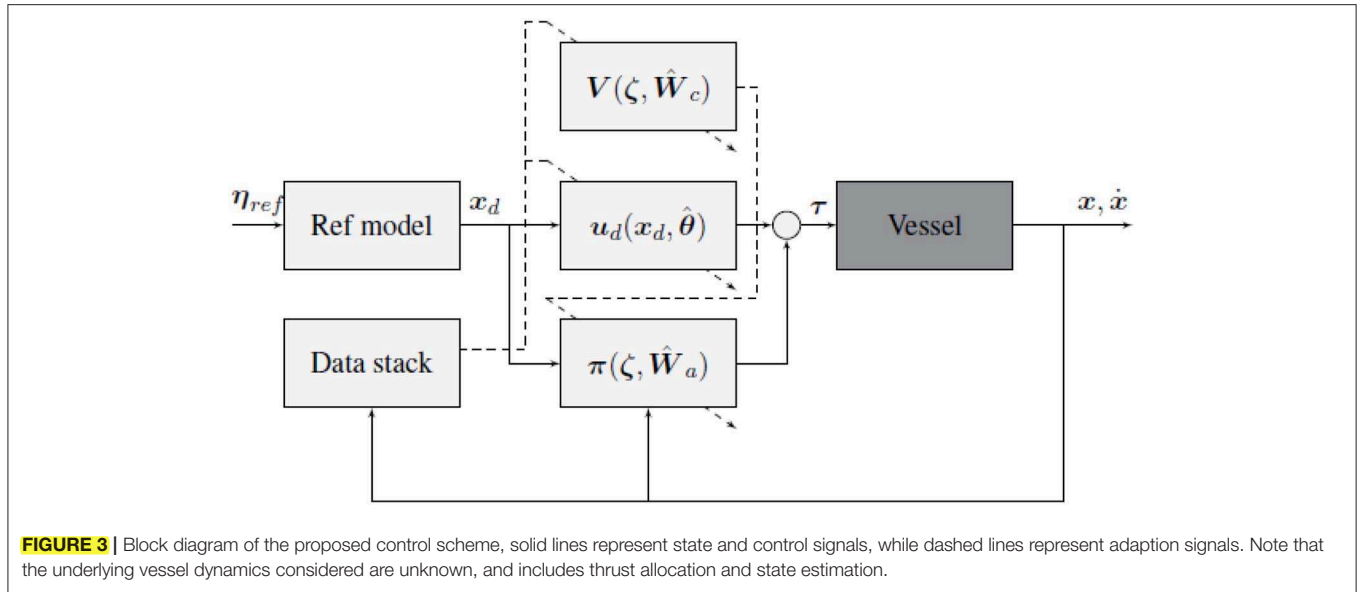
hence the model error \tilde{x} and parameter error $\tilde{\theta}$ converge exponentially to zero as $t \rightarrow \infty$. We can also note that the rate of the parameter convergence is given by the singular values of $\sum_{i=1}^N (Y^T(x_i) Y(x_i))$.

For the RL update laws in (34)–(36), it can be shown that under a number of strict assumptions, a system on the form given in (15), with an unconstrained policy, is uniformly ultimately bounded in terms of the error dynamics e , as well as the weights and parameters W_a , W_c , and θ . The stability analysis can be found in Kamalapurkar et al. (2018). For our purposes, we further constrain the parameters W_a of the feedback control law by projecting them into a region close to a known stable initial parameterization. Closed loop stability is important for assurance of the control system, this is further discussed in section 4.

2.7. Reference Model

When generating a reference path, we must ensure that it is sufficiently smooth in order to be able to say something about the convergence to the path. In practice however, we may have a signal which is discrete, defining the desired pose only at certain times. In order to smooth the trajectory we therefore use a reference model, which tracks the discrete reference pose, and generates a continuous reference trajectory pose $\eta_d = [x_d, y_d, \psi_d]^T$ and velocity vector $v_d = [u_d, v_d, r_d]^T$. For the pose we can make a reference model on the following form:

$$\begin{bmatrix} \dot{\eta}_d \\ \ddot{\eta}_d \\ \ddot{\eta}_d \end{bmatrix} = \begin{bmatrix} 0 & I & 0 \\ 0 & 0 & I \\ -\Omega^3 & -(2\Delta + I)\Omega^2 & -(2\Delta + I)\Omega \end{bmatrix} \begin{bmatrix} \eta_d \\ \dot{\eta}_d \\ \ddot{\eta}_d \end{bmatrix}$$



$$+ \begin{bmatrix} 0 \\ 0 \\ \Omega^3 \end{bmatrix} \eta_{ref} \quad (40)$$

Where $\Omega = \text{diag}([\omega_1, \dots, \omega_n])$ and $\Delta = \text{diag}([\delta_1, \dots, \delta_n])$. Choosing $\Delta = I$ ensures the reference model is critically damped, while Ω controls the rate of convergence of the states. We must also generate the velocity vector, however based on the pose, the velocity can be calculated as:

$$\begin{aligned} v_d &= J^T(\eta_d) \dot{\eta}_d \\ \dot{v}_d &= -S([0, 0, r_d]^T) J^T(\eta_d) \dot{\eta}_d + J^T(\eta_d) \ddot{\eta}_d \end{aligned} \quad (41)$$

where $-S([0, 0, r_d]^T) J^T(\eta_d) = \dot{J}^T(\eta_d)$, and $S(\omega)$ is the skew symmetric matrix:

$$S([\omega_1, \omega_2, \omega_3]^T) = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$$

The reason we here use a third order filter for the reference model, is to ensure a smooth pose, velocity, and acceleration, even when a step in the reference is observed. This ensures that the feedforward control for the reference trajectory (16) can track the reference.

A block diagram of the final control structure is given in **Figure 3**. The diagram shows how the controller is split into a feedback control law π , and a feedforward control law u_d . Where the Reference filter is used to generate the pose and velocity reference x_d , and the data stack collected from observing vessel transitions, is used to update the parameters of the control laws.

3. EXPERIMENTS

In this section we present the results from simulations, and sea trials on the *ReVolt* test platform (see **Figure 4**), when using

the control scheme proposed in the previous section. We will first present the implementation details for the for the control algorithm. After that we will briefly present the experimental platform, before finally presenting the simulation, and sea trial results for varying operational conditions. The experiments include both low speed dynamic positioning, and high speed trajectory tracking.

3.1. Implementation Details

For the implementation the parameter update laws (32), (36), (35), and (34) were implemented with a 4th order Runge-Kutta integration scheme, with a **timestep of 0.1 s**. Additionally the reference model in (40) and (41) were implemented, also using a 4th order Runge-Kutta scheme, in order to generate the reference trajectory $x_d = [\eta_d, v_d]^T$ and its derivative $\dot{x}_d = h_d = [\dot{\eta}_d, \dot{v}_d]^T$.

For the parameterization of the system identifier, the θ and $Y(x)$ were chosen as in (7) and (9), while for the actor and critic, the parameterization $\sigma(\zeta)$ was chosen as the vector of all the second order cross terms of the position and velocity error in the body frame $e^{\text{body}} = [\tilde{\eta}^{\text{body}}, \tilde{v}]$ where $\tilde{\eta}^{\text{body}} = J^T(\eta) \tilde{\eta}$, giving the following expression:

$$W\sigma(\zeta) = \sum_{x_i \in e^{\text{body}}} \sum_{x_j \in e^{\text{body}}} w_{i,j} x_i x_j \quad (42)$$

The reason that we use the error in the body frame, is the assumption that the cost is invariant to rotations when in the body frame, as this is the same frame the dynamics of the system are given in. The initial conditions for the actor and critic weights were chosen such that they matched the continuous time algebraic Riccati equation for a simplified linear model of the vessel.

For the control law, the constrained closed form controller (30) was used. And the output was scaled to fit the max thrust and torque $\bar{\tau} = \frac{1}{\sqrt{3}}[50.0, 20.0, 32.0]^T$ the vessel is able to



FIGURE 4 | *ReVolt* test platform courtesy of DNV GL.

produce. While $[50.0, 20.0, 32.0]^T$ is the **max force** the vessel is able to produce in each direction individually, due to the coupling between thrusters, we assume the maximum thrust in any given coupled direction can be approximated by the an ellipse with axis lengths 50.0, 20.0, and 32.0. **Since the proposed method only allows us to constrain thrust in each individual direction, we use the largest inner approximation of the ellipse as our thrust bound, giving the max thrust and torque as $\bar{\tau}$ given above.** It should be noted, that while this constrains the thrust, we can still not guarantee that the vessel is able to produce the desired amount of thrust as $\bar{\tau}$ is only an inner approximation of the elliptic approximation, whereas the true thrust bound may be much more complex. It should also be noted that using the inner approximation $\bar{\tau}$ as a bound, means we are not able to fully utilize the full thrust that the vessel has to offer. One way of solving these issues would be to **include the thrust allocation as part of the problem formulation**, however this is beyond the scope of this paper. It should also be noted that the desired thrust vector includes both the path tracking control law and drift correction $\tau_{\text{Thrust}} = u_d(x_d; \hat{\theta}) + \hat{\pi}(\xi; \hat{W}_a)$ where the saturation is only considered in the drift controller and not the path tracking control law. This means the desired path should be generated in a way that satisfies the thrust constraints.

For the state cost function $Q(\xi)$ a quadratic cost on the form $Q(\xi) = [\tilde{\eta}^{\text{body}}, \tilde{v}]^T Q [\tilde{\eta}^{\text{body}}, \tilde{v}]$ was chosen, where Q is a positive definite weight matrix. Given as:

$$Q = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 10.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 10.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 10.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 10.0 \end{bmatrix}$$

The weight matrix is given as a mostly diagonal matrix, with a small cross term between the position error in sway direction, and heading error. The cross term is added in order to encourage the vessel to travel in the surge direction when there is a large position error, as this is the most efficient direction of travel.

The data stack that was used consisted of 100 samples, and a singular value maximization scheme was implemented in order to increase the convergence rate. Using a purely singular value maximization based data selection scheme, while giving good performance on a stationary system, does not work for time

TABLE 1 | *ReVolt* hardware and software specifications.

Onboard computer:	Tank-720
Sensors:	Xsens MTI-G-710 IMU
	Vector VS330 GNSS Receiver
Software:	Linux Ubuntu LTS 16.04
	ROS Kinetic Kame

varying system, and hence does not allow for estimating the slowly varying environmental forces. In order to account for this, weighting of the singular value maximization, and data sample age was used in order to save recent samples with high singular values.

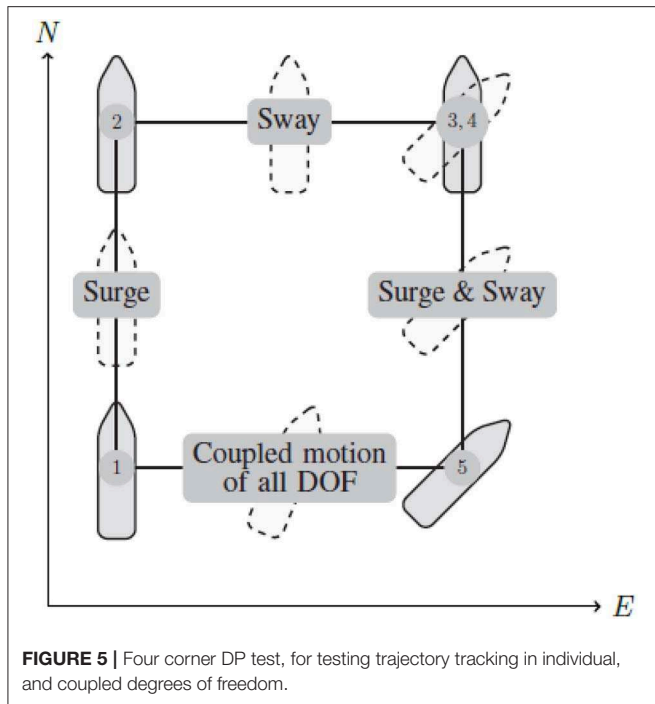
3.2. Experimental Platform

The *ReVolt*, shown in **Figure 4**, is a 1:20 scale model of a autonomous concept vessel developed by DNV GL in collaboration with NTNU. The model is 3 m long, 0.72 m wide, and weighs 257 kg. *ReVolt* has a **top speed** of 2 knots (~ 1 m/s) with a total combined engine power of 360 W. The thrust configuration is given as in **Figure 1**, with two identical stern thrusters, and one slightly less powerful bow thruster, all of which are fully rotatable azimuth thrusters, and are controlled by an optimization based thrust allocation (TA) algorithm. The vessel state is estimated using a non-linear observer consisting of an Extended Kalman Filter (EKF), and combines measurements from a Global Navigation Satellite System (GNSS) with Real-Time Kinematic (RTK) correction data, on board accelerometer, gyroscope, and compass. This provides accurate heading and position down to $\pm 0.2^\circ$ and ± 1 cm. A description of the *ReVolt* hardware and software is given in **Table 1**.

While the physical vessel was used for the sea trials, a high fidelity Digital Twin of *ReVolt*, developed by DNV GL, was used for simulation. The Digital Twin is based on a full 6DOF model, with parameters identified through tow-tank 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 sea states, ocean currents and wind conditions.**

3.3. Simulations and Sea Trials

In order to test the proposed control scheme, a number of experiments were devised. As the control scheme was build to



be able to handle both high speed and low speed maneuvering, we wanted to test both, by doing low speed Dynamic Positioning (DP), as well as higher speed path tracking.

3.3.1. Dynamic Positioning (DP)

In order to test the dynamic positioning capabilities of the control method, the four corner test seen in **Figures 5, 6** is used. The four corner DP test is used, as it shows the tracking capabilities of the vessel for individual degrees of freedom, as well as the coupled motion of all degrees of freedom, it is also worth noting that the vessel returns to the initial pose, meaning the test can easily be repeated. The four corner test starts with the vessel pointing north 0° , then performs the following commands:

1. Change position l meters due north, and come to a complete stop. This tests the surge motion of the vessel.
2. Change position l meters due east, and come to a complete stop. This tests the sway motion of the vessel.
3. Change heading 45° , and come to a complete stop. This tests the yaw motion of the vessel.
4. Change position l meters due south, and come to a complete stop. This tests the coupled surge and sway motion of the vessel.
5. Change position l meters due west, and heading to 0° and come to a complete stop. This tests coupled motion of all degrees of freedom.

For the box test we chose the box side length l to be 5 m, and the reference path was generated by linearly interpolating the pose between the commands, with 55 s to execute each command and a 5 s pause between commands in order for the reference filter to catch up to the reference, and ensure that the vessel comes to a

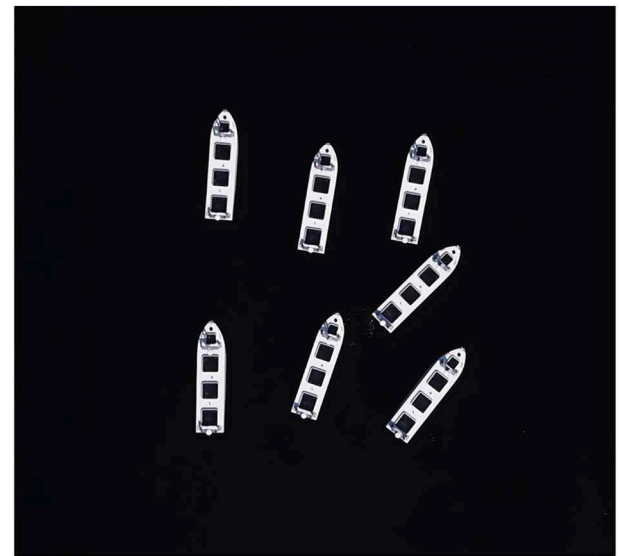


FIGURE 6 | Time-lapse drone photo of four corner DP test. It should be noted that the time-lapse above is of an early test, where errors in the navigation system resulted in poor performance.

TABLE 2 | Reference pose for four corner DP test, note that the reference that was used was a linear interpolation of the poses in the table.

Time [s]	0	55	60	115	120	175	180	235	240	295	300
x_r [m]	0	5	5	5	5	5	5	0	0	0	0
y_r [m]	0	0	0	5	5	5	5	5	5	0	0
ψ_r [deg]	0	0	0	0	0	45	45	45	45	0	0

stop. The reference poses used for the experiments are given in **Table 2**.

In order to evaluate the performance of the dynamic positioning, The **Integral Absolute Error (IAE)** given in (43) was used.

$$IAE(t) = \int_0^t \sqrt{(\bar{\eta} - \bar{\eta}_d)^T (\bar{\eta} - \bar{\eta}_d)} dt \quad (43)$$

Where $\bar{\eta}$ and $\bar{\eta}_d$ are the **normalized pose vectors**, normalized between ± 5 m in north and east direction, and $\pm 50^\circ$ in heading, giving the following.

$$\bar{\eta} = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix}^T, \quad \bar{\eta}_d = \begin{bmatrix} x_d \\ y_d \\ \psi_d \end{bmatrix}^T$$

Running the proposed control scheme in simulations on the Digital Twin of the *ReVolt* vessel, we got the trajectory and errors seen in **Figure 7**. For the same test performed on the physical vessel during the sea trials, we got the trajectory and errors seen in **Figure 8**. The IAE for the tests are shown in **Figure 9**.

3.3.2. Path Tracking

For both straight line path tracking and curved path tracking, the way-points in **Table 3** were used to generate a linearly, and

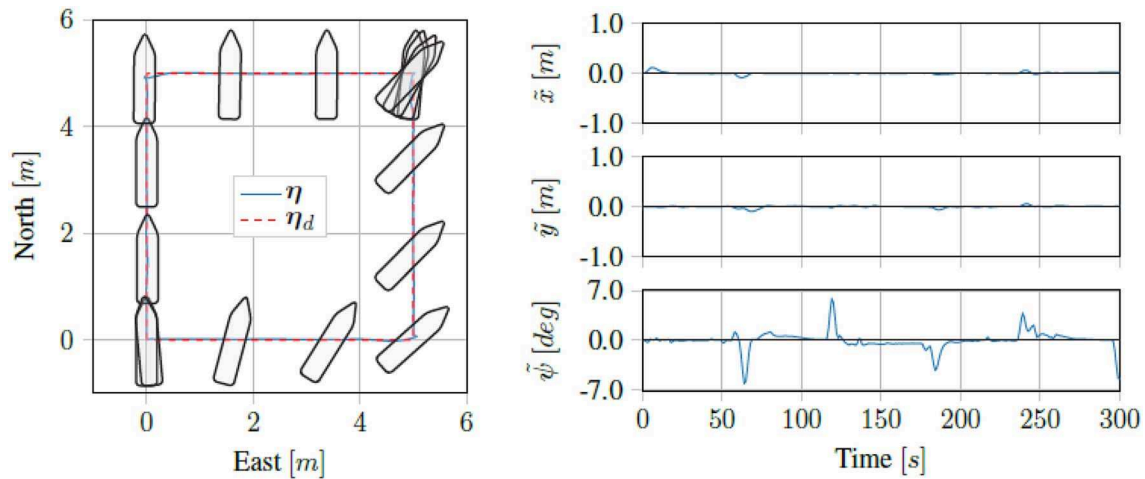


FIGURE 7 | Simulation results for four corner DP tests.

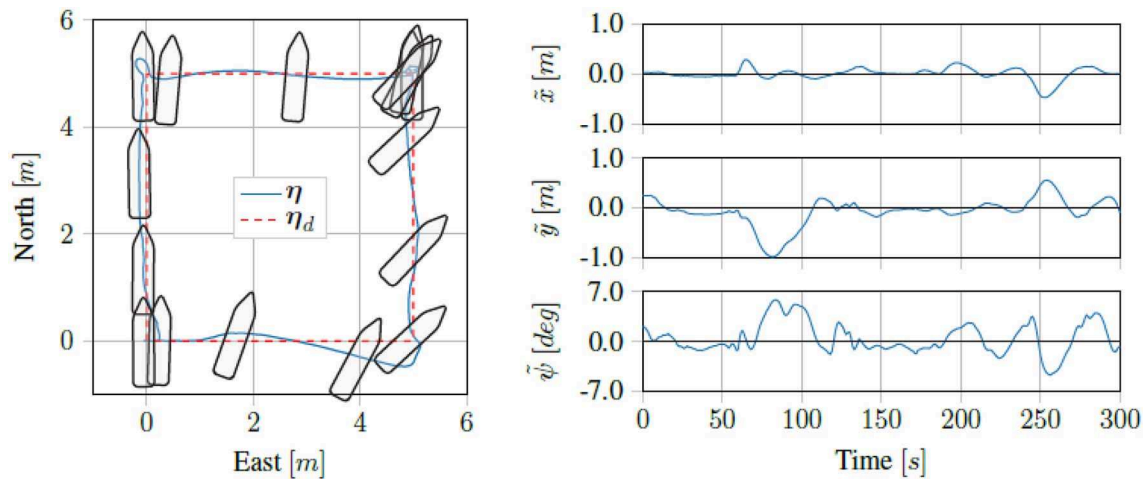


FIGURE 8 | Sea trial results for four corner DP tests.

quadratically interpolated path, respectively. For the heading, the path direction was used to generate the desired heading, giving the following reference heading.

$$\psi_r = \text{atan2}(\dot{y}_r, \dot{x}_r) \quad (44)$$

In order to encourage the vessel to converge to path in the surge direction, a small cross term was added in the state cost function $Q(\xi)$ between the heading error, and the position error in the surge direction of the body frame. The key insight here, is that the for large errors in surge, this term will encourage the vessel to turn the bow toward the desired position, meaning the vessel is encouraged to travel in the surge direction, which is the most efficient direction of travel, due to the design of the hull. For our implementation, where pose error is given in the body frame of the vessel, and the state penalty is given as a quadratic function $Q(\xi) = \xi^T Q \xi$, this penalty is added by simply adding a term to

the off-diagonals of Q corresponding to the cross terms between position error in the y direction, and the heading error.

Running the straight line path tracking on the Digital Twin of the *ReVolt* vessel we got the results seen in **Figure 10**. Running the same tests on the physical vessel, we got the results seen in **Figure 11**. As we can see, the proposed control scheme is able to follow the path quite well.

3.4. Results

Based on the results, the proposed method seems to work very well, in both simulations and the physical platform. While the simulator has been designed to perform as closely as possible to the physical platform, there are slight discrepancies that may explain the performance drop. The main factors of the **performance drop** is however most likely **due to the measurement and observation noise** that is present on the physical vessel. While the RTK GNSS is able to give a good

measurement for the pose of the vessel, the estimated vessel velocities that the algorithm is dependent on become very inaccurate, especially at low speeds when the signal to noise ratio becomes small. Another error source is likely the thruster dynamics. While the algorithm above assumes the desired thrust is produced immediately, in reality producing the desired thrust vector takes time, as the thrust allocation involves rotating the thrusters to a given angle, as well as spinning up to a desired motor RPM. An additional source of error may also have been a vertical stabilizer, which had recently been added to the vessel between the two rear thrusters, but had not been taken into account in the thrust allocation algorithm. Overall, the results are quite good, especially considering the size of the vessel, the relatively low thrust capability, and the precision to which the maneuvers are performed, even under the uncertainty created by the sensor noise, and environmental forces.

4. ASSURANCE OF RL-BASED CONTROLLERS

Assurance is the structured collection of evidence supporting claims and arguments that a system is safe or fit for its intended purpose. Assurance is required to develop trustworthy systems and solutions for use in real-world applications. Principles of assurance can be found in any certification or verification framework, where claims and arguments most often can be considered requirements of verification, while evidence is the result from this verification. Two types of verification are used: (1) *Product verification*, which performs direct verification of the developed product or system and produces *primary evidence*; (2) *process verification*, which performs verification of some part of the development process and produces *circumstantial evidence*. Using established verification frameworks applied to conventional marine control systems, experience has shown what requirements and evidence are most important when verifying these conventional control systems. With novel technology, such as data-driven methods, the verification requirements and evidence that is needed for assurance are still unknown, as they pose a new set of challenges when assuring the system.

Data-driven approaches are not new, but with increasing computational power and abundance of data there has been an increasing interest in these methods. Within control theory, the field of system identification has been a key part of control engineering for many years (Åström et al., 1965; Ho and Kálmán, 1966), and data-driven modeling for control purposes has been practiced since. Such models are typically based on the physical properties that govern the system, and hence the parameters estimated by such methods may reflect measurable properties of the system, thus providing benchmarks for verification. However, most models represent the physical system only within certain operational limits, e.g., weather or sea states, or for certain vessel speed ranges, which restricts the validity of the models accordingly.

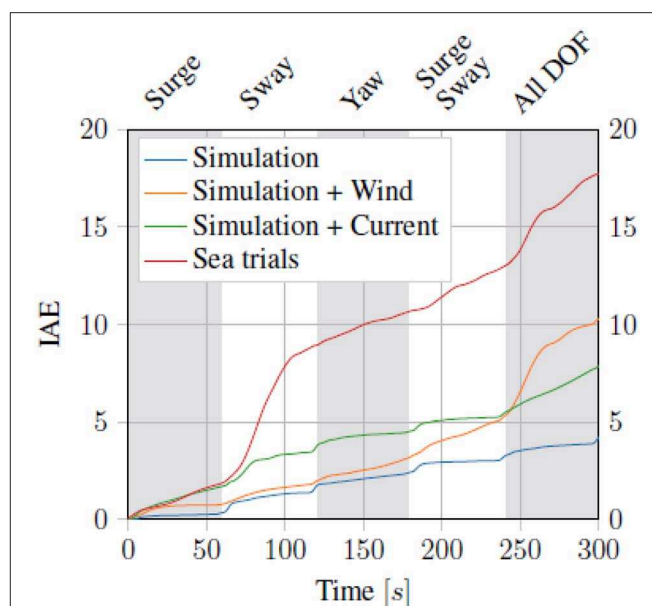


FIGURE 9 | Integral absolute error (IAE) for the dynamic positioning task, the gray and white bands mark the different commands/phases of the four corner test.

TABLE 3 | Reference pose for the straight line path and curved path, note that the reference that was used for straight line path tracking was a linear interpolation of the poses, and the reference pose for the curved path, was a quadratic interpolation of the poses.

Time [s]	0	100	200	300
x_r [m]	0	50	100	150
y_r [m]	0	0	50	50

Contrary to the more static nature of classical data-driven approaches, where tuning the control parameters relies on offline estimation of the model parameters, in this paper the key difference is that model based-RL is used for online tuning of both the vessel model (including an estimation of unknown disturbances) and the control policy parameters. The vessel model and the control policy are based on proven methods used in the maritime industry for vessel station keeping and guidance, but there are still some key issues that must be considered. For instance, the control policy is highly dependent on an instantaneously valid vessel model, which in turn means the behavior of the vessel is highly dependant on the validity of the learned model. Both the vessel model and the control policy parameters are all continuously learned, but it is critical that all allowed parameter combinations give a sufficiently safe behavior. The proposed control scheme in this paper continuously learns and updates the parameters in order to optimize the tracking behavior. In terms of safety, the main concern is whether the learned model and policy parameters lead to a safe and acceptable behavior. Verifying this in a setting where the parameters are learned online is still an open problem.

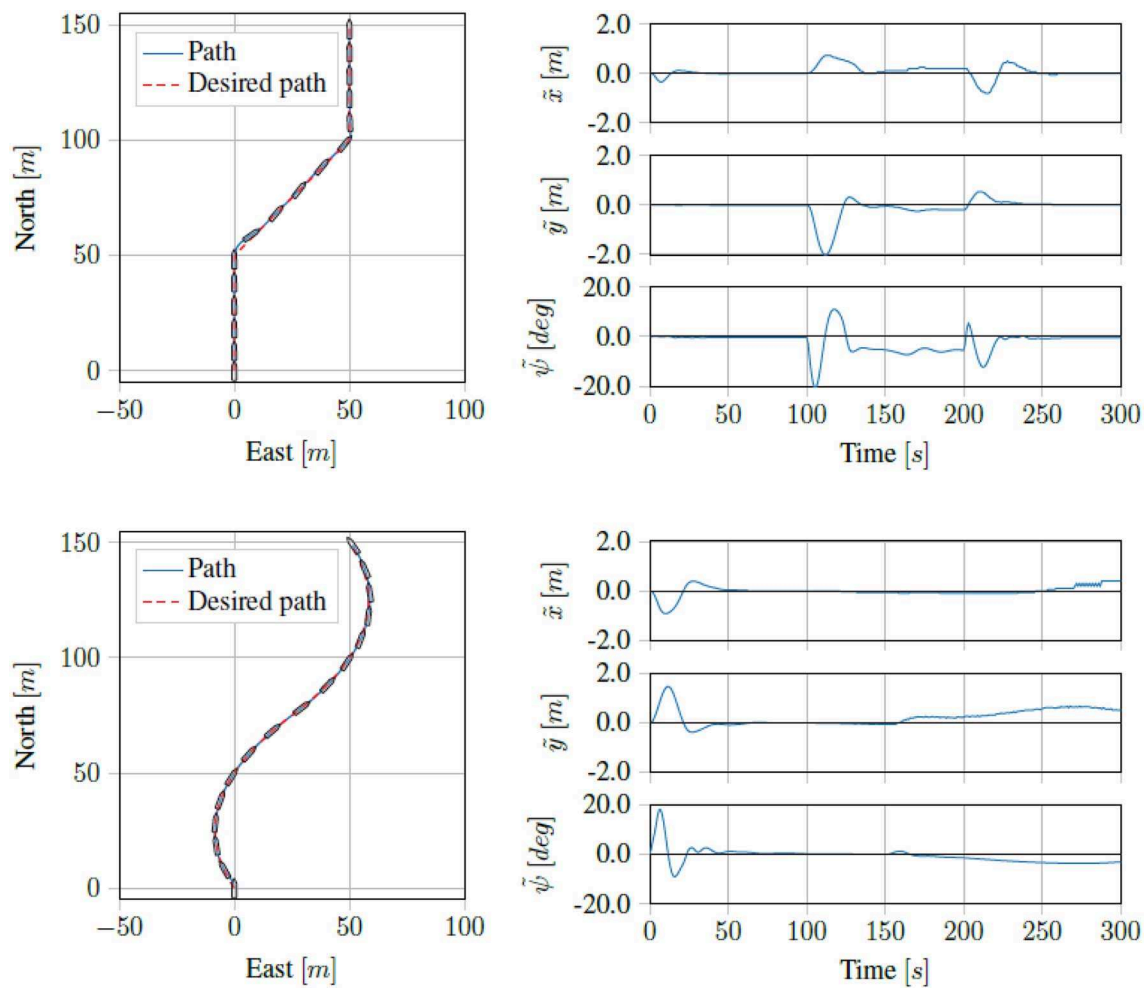


FIGURE 10 | Simulation results for straight line and curved path tracking.

Amodei et al. (2016) discusses five basic concrete problem areas related to RL and safety, which must be taken into account for any application of RL.

1. **Avoiding reward hacking:** the first problem, is that of hacking or gaming the cost function. For the tracking problem in this paper, a positive definite quadratic penalty on the error dynamics is used. From control theory these methods are known to converge to the origin, i.e., where the error is zero. This means the intended behavior is guaranteed when the policy converges to the optimal policy.
2. **Avoiding negative side effects:** the second problem of avoiding negative side effects, is similar to the first, but addresses the issue of choosing the cost function such that the optimal policy does not give bad or unintended behavior. For the method proposed in this paper, making such guarantees is quite difficult, as tuning the parameters of the quadratic cost function will still have an effect on the vessel behavior when converging to the origin. One example of this is that we typically want the vessel to approach the path head on if we
3. **Scalable oversight:** this pertains to how we can ensure that the RL agent respects aspects of the objective that are encountered infrequently. In terms of the trajectory tracking problem, the environment is quite limited, and the objective is clearly defined, hence the problem of scalable oversight is of limited relevance to the work presented in this paper.
4. **Safe exploration:** exploration is necessary in order to improve performance, but bears risk, and thus performing exploration in a safe manner is not trivial. Safe exploration also encompasses the evaluation of the quality of the training data that is gathered. For a real world application, this means accounting for faulty hardware, and noisy measurements, which may lead to problematic training data. For the method proposed in this paper, where the system is learning continuously online, the problem of safe exploration and learning is highly relevant. Some measures are taken, such

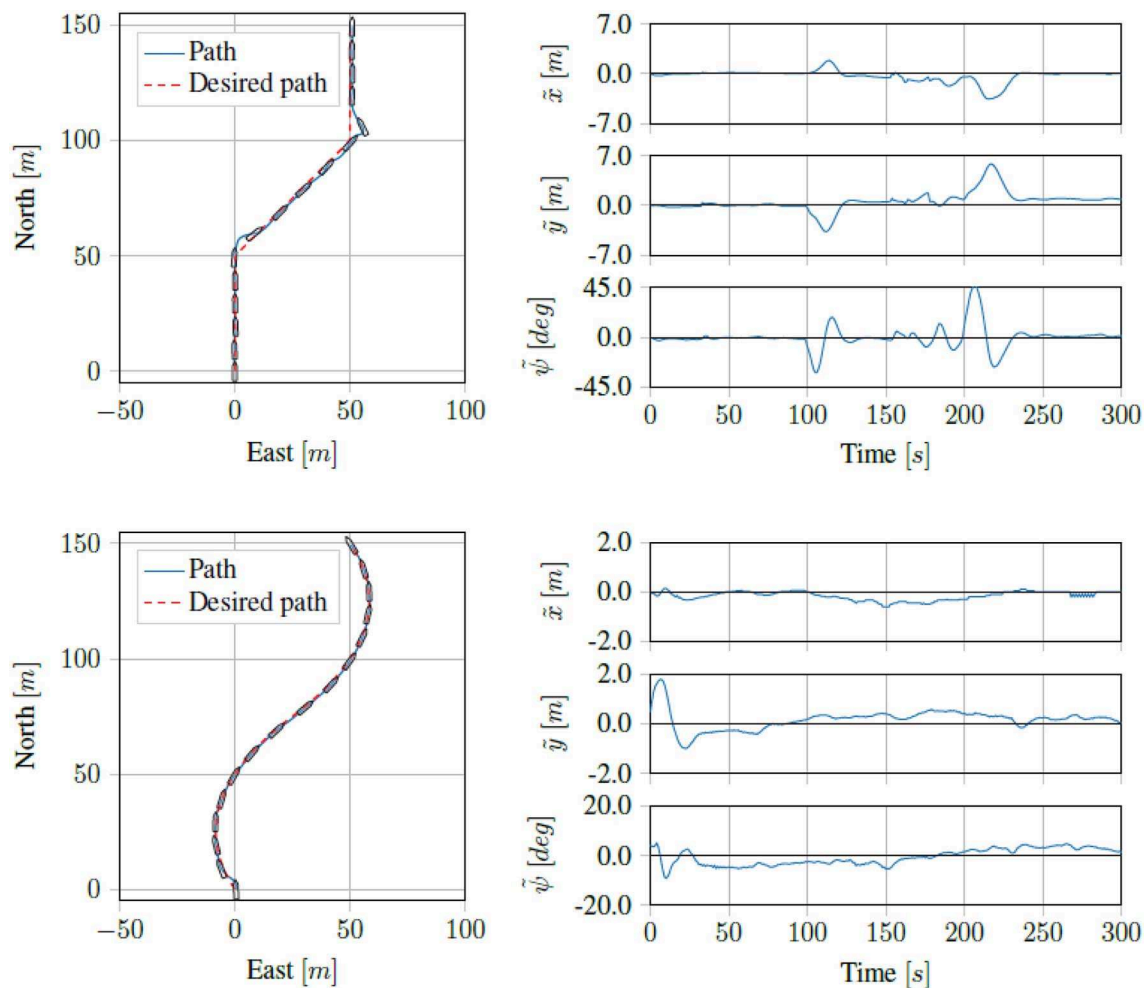


FIGURE 11 | Sea trial results for straight line and curved path tracking.

as using batches of training data and restricting the values that the policy parameterization may take. However, these measures only serve to mitigate potential problems, hence safe exploration and learning is still an open problem.

5. **Robustness to distributional shift:** this refers to how we can ensure the agent is robust to changes in the operating environment. For the proposed method, this is mostly solved by continuously learning online, which ensures that the agent learns the distributional shift when the environment changes. However, as discussed in the fourth problem of safe exploration, learning online complicates the matter of ensuring data quality.

In order to produce the evidence needed for verification of data driven methods, there are two main approaches, namely *scenario based verification*, and *theoretical verification*. *Scenario based verification* would be to conduct extensive testing in representative scenarios, which in practice would mean simulation-based testing as this would be the only feasible online solution. More limited real testing should also be used,

but targeted toward validating the simulation accuracy. Many RL solutions are in practice not viable without simulation-based training or development and this would mean the same tool can be utilized both for testing, and offline training. The challenge in this case would be to induce a representative set of scenarios to prove the safety or validity of the solution, and such scenario selection is an open question for testing AI or systems operating in complex environments in general. The second approach *theoretical verification*, would be to impose constraints on the RL algorithm in order to avoid unwanted behavior. This would entail combining methods from control theory, a physical or mathematical understanding of the system, and experience or insights of the control scheme, in order to express and implement various constraints on the learnable model and policy parameters. This may not conceptually be a new approach since similar methods already exist, but this approach is difficult to use in practice, as finding parameter constraints that ensure safe operations is non-trivial.

In conclusion, an assurance framework for technologies, such as the one presented in this paper is an open research question.

However, one can with confidence state that it should include both process and product verification, i.e., considering not only what is developed, but also how it is developed. This would mean that adequate development and assurance processes should be developed, including verification methods that can produce the required evidence both for efficient development, as well as assurance. In addition, novel data-driven methods should be combined with prior knowledge, verified solutions and proven physical or mathematical relations (Eldevik, 2018). This, in order to be able to explain the behavior, and in turn guarantee that the methods are safe and fit for the intended purpose.

5. CONCLUSION

The proposed method performed very well in all three tested tracking scenarios both in simulations and in sea trials. The method is also versatile, as using it on different vessels only requires knowledge of the inertia matrix, with the update laws providing a tool for **learning the other model parameters, and a control policy**. For **future work**, it may be interesting to improve and update the **thrust allocation algorithm** to get a smaller error between produced and desired thrust, and investigate whether this results in **better accuracy**. Alternatively, feedback from the thrusters can be used to get better estimates of the thrust vector for use in the data stack, and model estimation.

REFERENCES

- Åström, K. J., Bohlin, T., and Wensmark, S. (1965). *Automatic Construction of Linear Stochastic Dynamic Models for Stationary Industrial Processes With Random Disturbances Using Operating Records*. Technical Paper TP 18, IBM Nordic Laboratory, Stockholm.
- Amodi, D., Olah, C., Steinhart, J., Christiano, P., Schulman, J., and Mané, D. (2016). Concrete problems in AI safety. *arXiv [Preprint] arXiv:1606.06565*.
- Andrychowicz, M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., et al. (2018). Learning dexterous in-hand manipulation. *arXiv 1808.00177*. doi: 10.1177/0278364919887447
- Bertsekas, D. P. (2019). *Reinforcement Learning and Optimal Control*. Belmont, MA: Athena Scientific.
- Cheng, Y., and Zhang, W. (2018). Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels. *Neurocomputing* 272, 63–73. doi: 10.1016/j.neucom.2017.06.066
- Chowdhary, G., and Johnson, E. (2011a). “A singular value maximizing data recording algorithm for concurrent learning,” in *Proceedings of the 2011 American Control Conference* (San Francisco, CA: IEEE), 3547–3552. doi: 10.1109/ACC.2011.5991481
- Chowdhary, G. V., and Johnson, E. N. (2011b). Theory and flight-test validation of a concurrent-learning adaptive controller. *J. Guid. Control Dyn.* 34, 592–607. doi: 10.2514/1.46866
- Dai, S.-L., Wang, C., and Luo, F. (2012). Identification and learning control of ocean surface ship using neural networks. *IEEE Trans. Ind. Inform.* 8, 801–810. doi: 10.1109/TII.2012.2205584
- Do, K. D. (2016). Global path-following control of underactuated ships under deterministic and stochastic sea loads. *Robotica* 34, 2566–2591. doi: 10.1017/S0263574715000211
- Doya, K. (2000). Reinforcement learning in continuous time and space. *Neural Comput.* 12, 219–245. doi: 10.1162/089976600300015961
- Eldevik, S. (2018). *AI + Safety*. Available online at: <https://ai-and-safety.dnvgi.com/>
- Fossen, T. I. (1994). *Guidance and Control of Ocean Vehicles*. New York, NY: John Wiley & Sons Inc.
- Fossen, T. I. (2000). A survey on nonlinear ship control: from theory to practice. *IFAC Proc. Vol.* 33, 1–16. doi: 10.1016/S1474-6670(17)37044-1
- Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons.
- Fossen, T. I., Sagatun, S. I., and Sørensen, A. J. (1996). Identification of dynamically positioned ships. *Control Eng. Pract.* 4, 369–376. doi: 10.1016/0967-0661(96)00014-7
- Hasegawa, K., Kouzuki, A., Muramatsu, T., Komine, H., and Watabe, Y. (1989). Ship auto-navigation fuzzy expert system (safes). *J. Soc. Naval Archit. Jpn.* 1989, 445–452. doi: 10.2534/jjasnaoe1968.1989.166_445
- Ho, B., and Kálmán, R. E. (1966). Effective construction of linear state-variable models from input/output functions. *AT-Automatisierungstechnik* 14, 545–548. doi: 10.1524/auto.1966.14.112.545
- Ioannou, P. A., and Sun, J. (2012). *Robust Adaptive Control*. Mineola, NY: Courier Corporation.
- Kallstrom, C. G. (1982). *Identification and Adaptive Control Applied to Ship Steering*. Technical report, Department of Automatic Control, Lund Institute of Technology (LTH), Lund.
- Källström, C. G., and Åström, K. J. (1981). Experiences of system identification applied to ship steering. *Automatica* 17, 187–198. doi: 10.1016/0005-1098(81)90094-7
- Kamalapurkar, R., Walters, P., Rosenfeld, J., and Dixon, W. (2018). *Reinforcement Learning for Optimal Feedback Control: A Lyapunov-Based Approach*. Springer.
- Katebi, M., Grimble, M., and Zhang, Y. (1997). H-∞ robust control design for dynamic ship positioning. *IEEE Proc. Control Theory Appl.* 144, 110–120. doi: 10.1049/ip-cta:19971030
- Konda, V. R., and Tsitsiklis, J. N. (2000). “Actor-critic algorithms,” in *Advances in Neural Information Processing Systems*, eds S. A. Solla, T. K. Leen, and K. Müller (Denver, CO: MIT Press), 1008–1014.
- Lekkas, A. M., and Fossen, T. I. (2014). “Trajectory tracking and ocean current estimation for marine underactuated vehicles,” in *2014 IEEE Conference on Control Applications (CCA)* (Juan Les Antibes: IEEE), 905–910. doi: 10.1109/CCA.2014.6981451

DATA AVAILABILITY STATEMENT

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation, to any qualified researcher.

AUTHOR CONTRIBUTIONS

AM and AL conceived the presented idea. AM developed the theory and performed the simulations, while AM, TP, and AL carried out the experiments. JG and TP contributed with the assurance section from a classification society perspective. AM, AL, and SG discussed the results and methods. All authors contributed to the final manuscript.

- Liberzon, D. (2011). *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton, NJ; Oxford: Princeton University Press.
- Martinsen, A. B., and Lekkas, A. M. (2018a). "Curved path following with deep reinforcement learning: results from three vessel models," in *OCEANS 2018 MTS/IEEE* (Charleston, SC: IEEE), 1–8. doi: 10.1109/OCEANS.2018.8604829
- Martinsen, A. B., and Lekkas, A. M. (2018b). "Straight-path following for underactuated marine vessels using deep reinforcement learning," in *11th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles (CAMS)* (Opatija). doi: 10.1016/j.ifacol.2018.09.502
- McGookin, E. W., Murray-Smith, D. J., Li, Y., and Fossen, T. I. (2000). Ship steering control system optimisation using genetic algorithms. *Control Eng. Pract.* 8, 429–443. doi: 10.1016/S0967-0661(99)00159-8
- Mišković, N., Vukić, Z., Bibuli, M., Bruzzone, G., and Caccia, M. (2011). Fast in-field identification of unmanned marine vehicles. *J. Field Robot.* 28, 101–120. doi: 10.1002/rob.20374
- Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., et al. (2013). Playing Atari with deep reinforcement learning. *arXiv [Preprint] arXiv:1312.5602*.
- Pettersen, K. Y., and Egeland, O. (1996). "Exponential stabilization of an underactuated surface vessel," in *Proceedings of 35th IEEE Conference on Decision and Control*, Vol. 1 (Kobe: IEEE), 967–972. doi: 10.1109/CDC.1996.574602
- Pham Tuyen, L., Layek, A., Vien, N., and Chung, T. (2017). "Deep reinforcement learning algorithms for steering an underactuated ship," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)* (Daegu: IEEE), 602–607. doi: 10.1109/MFI.2017.8170388
- Shen, H., and Guo, C. (2016). "Path-following control of underactuated ships using actor-critic reinforcement learning with mlp neural networks," in *Sixth International Conference on Information Science and Technology (ICIST)* (Dalian: IEEE), 317–321. doi: 10.1109/ICIST.2016.7483431
- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature* 529:484. doi: 10.1038/nature16961
- Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., et al. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv [Preprint] arXiv:1712.01815*.
- Soetanto, D., Lapierre, L., and Pascoal, A. (2003). "Adaptive, non-singular path-following control of dynamic wheeled robots," in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, Vol. 2 (IEEE), 1765–1770. doi: 10.1109/CDC.2003.1272868
- Sørensen, A. J. (2011). A survey of dynamic positioning control systems. *Annu. Rev. Control* 35, 123–136. doi: 10.1016/j.arcontrol.2011.03.008
- Sutton, R., Roberts, G., and Taylor, S. (1997). Tuning fuzzy ship autopilots using artificial neural networks. *Trans. Inst. Meas. Control* 19, 94–106. doi: 10.1177/014233129701900204
- Sutton, R. S., and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.
- Walters, P., Kamalapurkar, R., Voight, F., Schwartz, E. M., and Dixon, W. E. (2018). Online approximate optimal station keeping of a marine craft in the presence of an irrotational current. *IEEE Trans. Robot.* 34, 486–496. doi: 10.1109/TRO.2018.2791600
- Wang, N., Qian, C., Sun, J.-C., and Liu, Y.-C. (2015). Adaptive robust finite-time trajectory tracking control of fully actuated marine surface vehicles. *IEEE Trans. Control Syst. Technol.* 24, 1454–1462. doi: 10.1109/TCST.2015.2496585
- Wang, N., Su, S.-F., Yin, J., Zheng, Z., and Er, M. J. (2017). Global asymptotic model-free trajectory-independent tracking control of an uncertain marine vehicle: an adaptive universe-based fuzzy control approach. *IEEE Trans Fuzzy Syst.* 26, 1613–1625. doi: 10.1109/TFUZZ.2017.2737405
- Yu, R., Shi, Z., Huang, C., Li, T., and Ma, Q. (2017). "Deep reinforcement learning based optimal trajectory tracking control of autonomous underwater vehicle," in *36th Chinese Control Conference (CCC)* (Dalian: IEEE), 4958–4965. doi: 10.23919/ChiCC.2017.8028138
- Zhang, L., Qiao, L., Chen, J., and Zhang, W. (2016). "Neural-network-based reinforcement learning control for path following of underactuated ships," in *35th Chinese Control Conference (CCC)* (Chengdu: IEEE), 5786–5791. doi: 10.1109/ChiCC.2016.7554262

Conflict of Interest: JG and TP were employed by the company DNV GL.

The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2020 Martinsen, Lekkas, Gros, Glomsrud and Pedersen. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.