Curved Path Following with Deep Reinforcement Learning: Results from Three Vessel Models

Andreas B. Martinsen
Department of Engineering Cybernetics
Norwegian University of Science and Technology
Trondheim, Norway
andreabm@stud.ntnu.no

Anastasios M. Lekkas
Department of Engineering Cybernetics
Norwegian University of Science and Technology
Trondheim, Norway
anastasios.lekkas@ntnu.no

***Abstract*—This paper proposes a methodology for solving the curved path following problem for underactuated vehicles under unknown ocean current influence using deep reinforcement learning. Three dynamic models of high complexity are employed to simulate the motions of a mariner vessel, a container vessel and a tanker. The policy search algorithm is tasked to find suitable steering policies, without any prior info about the vessels or their environment. First, we train the algorithm to find a policy for tackling the straight line following problem for each of the simulated vessels and then perform *transfer learning* to extend the policies to the curved-path case. This turns out to be a much faster process compared to training directly for curved paths, while achieving indistinguishable performance.**

***Index Terms*—Deep reinforcement learning, path following, transfer learning, marine control systems, unknown disturbances**

I. Introduction

The objective of a path-following controller is to generate and execute heading angles that enable a constant-speed vehicle to follow a predefined path with minimum cross-track error and without temporal constraints. The literature pertaining to solving the path following problem for marine vehicles is rich and diverse. Compared to straight lines, curved path following presents additional challenges, the most obvious one being the time-varying effect of (even constant) disturbances while the vehicle follows the path. In previous works, a well-studied framework has been to utilize models representing the vessel dynamics and kinematics before employing methods from linear/nonlinear control theory for devising suitable kinematic (i.e. guidance) and dynamic (i.e. control) laws to achieve the control objective.

The guidance and control systems are often viewed as a cascaded system, with guidance being the perturbing system, and vehicle control the perturbed system. Vehicle control is significantly slower compared to guidance, since it involves stabilizing the dynamics. The cascade structure helps simplify the stability analysis and has been used extensively in the past, see for instance [1–3]. The guidance system often involves the line-of-sight (LOS) guidance law (see [4, 5]), which can be extended to account for external disturbances and optimize performance at the kinematic level [6–10].

Reinforcement learning (RL), also known as *neuro-dynamic programming* and *approximate dynamic programming*, is a theory developed by the Artificial Intelligence (AI) community for reaching optimal performance under system and environment uncertainty [11–13]. Contrary to optimal control theory, RL is based on evaluative, rather than instructive, feedback and comes in different forms, which might, or not, include partial knowledge of the environment or the system. The process typically involves hand-engineering a *reward function*, which determines the good actions that receive a reward, and the undesired actions that receive a penalty. The RL algorithm is then assigned to find a *policy* (series of control actions) that solves the control objective in the best possible way.

After the recent breakthrough by Deep Mind, where Deep Neural Networks (DNNs) were fused with RL and resulted in the *AlphaGo* program, the field of Deep Reinforcement Learning (DRL) started to receive significant attention. Notable results include [14, 15] for problems with discrete state and action spaces, and for continuous state and action spaces, algorithms such as [16–19] have been shown to work for a number of complex problems such as robotic manipulation, bipedal locomotion, and game play. The fact that DRL algorithms can discover efficient policies with limited prior information about the system and its environment make them good candidates for tackling marine control problems. A few recent works have already moved toward this direction and the brief overview given here focuses on using DRL for path following of underactuated surface vessels.

In [20, 21], the authors combined an actor-critic method with DNNs to find a policy for controlling the course of an underactuated ship. The guidance system, however, was designed in the traditional LOS-based manner. In [22], the authors implemented the Deep Deterministic Policy Gradients (DDPG) algorithm for trajectory tracking of underwater vehicles, but since only horizontal motion is considered, the problem is identical to the 3-DOF surface case. A simple model was used to simulate the vehicle and the authors used Lyapunov theory to justify the reward function they chose and compared the efficiency of their method with that of a PID controller. In [23], the authors developed two DRL-based controllers (DDPG and Normalized Advantage Function) for steering an underactuated ship through a specified gate, instead of following a desired path. In [24], the present authors developed a DRL methodology, based on DDPG, for solving the straight-

978-1-5386-4814-8/18/$31.00 ©2018 IEEE

path following problem for underactuated marine vessels under the influence of unknown ocean currents. The vessel model used was much more complex compared to earlier works and the reward function was designed to be simple, but at the same time, able to generate smooth rudder commands.

This work extends the results of [24]. More specifically, we implement a technique called *transfer learning* to allow the DRL controller to learn to follow curved paths under the influence of unknown currents using minimum training. We start with the neural network (policy) that solves the straight path following problem and then perform additional training to produce a policy which solves the more complex curved path following problem. Similarly to [24], the reward function is very simple and avoids bang-bang rudder behavior. Finally, we implement this process to three complex vessel models, one of which is a 4-DOF model that simulates the roll angle.

II. Preliminaries

A. Path following

Path following is a motion control scenario where the vehicle must converge to a predefined trajectory without temporal constraints. The vehicle can be assumed to have a constant total speed and, consequently, suitable heading angle actions must be taken in order to achieve the control objective.

For curved paths we describe the path in terms of a parameterized curve $\boldsymbol{P_d}(\omega) = [x_d(\omega), y_d(\omega)]^\top$, defined in the the North-East-Down (NED) coordinate system, where ω is the parameterization variable. The angle of the path defined by the parameterized curve can then be computed by:

$$\gamma_p = \text{atan2}(y_d'(\omega), x_d'(\omega)) \tag{1}$$

where $\text{atan2}(y, x)$ is the four quadrant version of $\arctan(y/x)$. Using the path angle γ_p we can find the cross-track error normal to the path using the following equation.

$$y_e = -\sin(\gamma_p)(x - x_d(\omega)) + \cos(\gamma_p)(y - y_p(\omega)) \tag{2}$$

The path reference point $x_d(\omega), y_d(\omega)$, when the vessel is located at x, y, is found by computing the parameterization variable ω, which minimizes the Euclidean distance between the path and the vessel.

$$\min_{\omega} f(\omega) = (x - x_d(\omega))^2 + (y - y_d(\omega))^2 \tag{3}$$

The path and vessel geometry is visualized in Figure 1. For the path following task, the objective is for the vessel to converge to the desired path, this means that we want the cross-track error to go to zero, i.e. $\lim_{t\to\infty} y_e(t) = 0$.

B. Vessel models

The three vessel models used to perform the simulations were taken from the Marine Systems Simulator (MSS) toolbox [25], and include the 3-DOF Mariner vessel [26], 3-DOF Tanker vessel [27], and a 4-DOF Container vessel [28]. While the vessels all have similar structures, they exhibit very different performance. The Tanker is large with slow dynamics, the mariner vessel is smaller and faster, with the Container vessel being in between. The container vessel also takes into consideration the roll dynamics which is strongly coupled to the sway and yaw, making it a more complex model. It should be noted that the vessel models were treated as a black boxes, and only used for simulations during training and verification of the control algorithms. The detailed models are given in Appendix A.

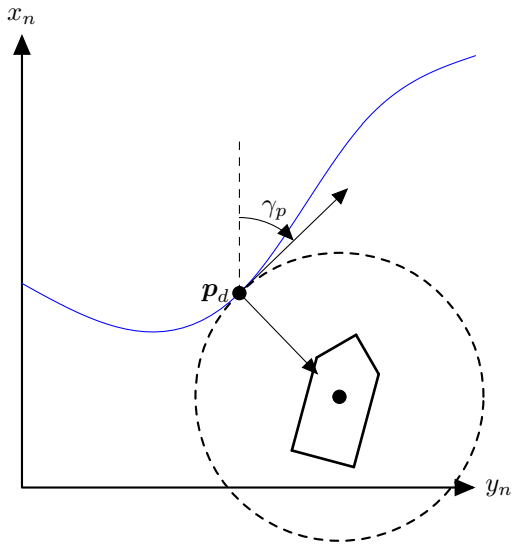


Fig. 1: The origin of the path centered coordinate frame $\boldsymbol{p}_d$ is the point on the path the shortest Euclidean distance from the vessel, and the orientation γ_p is given by the directional derivative of the path at $\boldsymbol{p}_d$

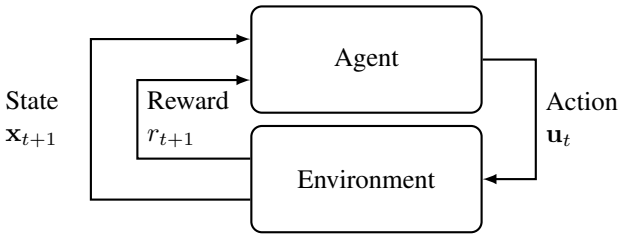


Fig. 2: Visualization of the agent environment interactions in reinforcement learning

C. Reinforcement learning

We assume the environment can be modeled by a Markov Decision Process (MDP) as presented by Bertsekas [12]. An MDP consists of a set of states $\mathcal{X}$, a set of actions $\mathcal{U}$, a discrete time transition model, which describes the probability of transitioning from one state $\mathbf{x}_t$ to another state $\mathbf{x}_{t+1}$ when taking an action $\mathbf{u}_t$, and a reward function $r_t = R(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{u}_{t-1})$. The reinforcement learning agent acts on the environment by taking actions and observing the resulting state and reward, as can be seen in Fig. 2.

The goal of the reinforcement learning algorithm is to find the optimal policy for selecting control actions $\mathbf{u_t}^* = \pi^*(\mathbf{x_t})$

which maximizes the reward gathered over time, that is, we wish to maximize the expected discounted return:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (4)$$

where $0 \leq \gamma \leq 1$ is called the discount rate. This can be expressed in terms of an action-value function

$$Q(x, u) = \mathbb{E}\left[G_t | x_t = x, u_t = u\right] \quad (5)$$

describing the expected discounted reward when taking action u in the state x, while following a given policy. The goal of reinforcement learning is to find the optimal policy:

$$\pi^*(\mathbf{x}_t) = \arg\max_u Q^*(\mathbf{x}_t, \mathbf{u}) \quad (6)$$

where Q^* is the action-value function when following the optimal policy π^*. The RL Agent does this by interacting with the environment, learning from the feedback, and using this experience to improve the policy.

DRL is a field of machine learning that fuses the ideas of deep learning and reinforcement learning. DRL has two main classes of algorithms for control problems with continuous action and state spaces, namely *actor-only* algorithms, and *actor-critic* algorithms. In actor-only algorithms, only the control policy, called the actor, is approximated by a function approximator, in this case the DNN. In actor-critic algorithms, both the policy as well as a value function, called the critic, are approximated.

To find the optimal policy, we implement an actor-critic method called Deep Deterministic Policy Gradients (DDPG) by [16], where both the policy and action-value functions are learned using DNNs. We denote the parameterized policy and action-value functions as $\pi(\mathbf{x}; \theta_a)$, and $Q(\mathbf{x}, \mathbf{u}; \theta_c)$, respectively, where θ_a and θ_c are the vectors of the DNN parameters. Learning takes place by implementing gradient descent on parameter vectors, θ_a and θ_c, as follows:

$$\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta), \quad (7)$$

where α is the learning rate, and $J(\theta)$ is the loss function which we want to minimize. The update rules for the policy and action-value function parameterizations are given in the next section, Eqs. 10–11 respectively.

D. *Transfer Learning*

In the context of machine learning, transfer learning refers to the the situation where what has been learned in one setting, is used in order to improve generalization in an other, usually similar setting. In the context of DRL, models trained in one domain are used as initial models for training the agent in new, similar domains [29]. It has also been used in transferring knowledge from simulated environments, to physical environments [30]. This has proven to be a useful tool as it speeds up training, and increases robustness, however the extent to which it works is highly dependant on the task similarity. In this work, we first train the agent to perform straight path following and then perform additional training to achieve curved path following. In both cases, unknown external disturbances act on the vehicles. Our experience so far shows that in this way shorter overall training time is needed and it is possible to keep increasing the complexity of the environment.

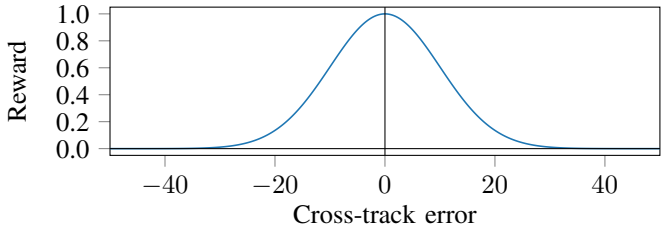


Fig. 3: Gaussian reward functions. The algorithm gives a reward dependant on distance from the path.

III. Implementation

For the path following control problem of marine vessels, the objective is to control the vessel in such a way that it converges to the path. The main indicator of convergence is the cross-track error y_e, which tells us how far away the vessel is from the path. Using the cross-track error we propose the following Gaussian reward function:

$$R(y_e, \tilde{\psi}, \dot{\delta}) = -c_{\dot{\delta}\dot{\delta}} \dot{\delta}^2 + \begin{cases} a e^{-\frac{y_e^2}{2\sigma}} & \text{if } |\tilde{\psi}| < \frac{\pi}{2}, \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

The first term assigns a penalty to solutions that involve fast rudder changes, see [24] for more details. The Gaussian reward function is given as a Gaussian curve with an amplitude $a = 1$ and a standard deviation $\sigma = 10m$, illustrated in Figure 3. The Gaussian reward function is chosen, as it is strictly increasing, as the path converges, while having properties that give it a time minimizing characteristic. The reward function additionally includes a bound on the yaw error $\tilde{\psi}$, which is the angle between the yaw angle of the vessel and the path heading. This is included in order to ensure the vessel travels along the path in the forward direction.

The state representation for the learning algorithm is given in terms of the error dynamics in a path relative coordinate frame, and chosen as:

$$\mathbf{x} = [y_e, \dot{y}_e, \chi - \gamma_p, \dot{\chi} - \dot{\gamma}_p, \psi - \gamma_p, r - \dot{\gamma}_p, u, v, r]^\top \quad (9)$$

where y_e is the cross-track error, $\chi - \gamma_p$ is the course relative to the path and $\psi - \gamma_p$ is the heading relative to the path. We also include the surge u, sway v, and yaw rate r, as well the derivatives of the path relative course and heading, this, in order to give the learning algorithm the necessary information to be able to observe the system states, and hence be able to learn how to perform the path following task. The action available to the algorithm is controlling the desired rudder angle δ_c which is saturated according to the vessel model.

We implement an actor-critic method, based on [16], in where the policy $\pi(\mathbf{x}; \theta_\mathbf{a})$ and the action-value function

$Q(\mathbf{x}, \mathbf{u}; \theta_\mathbf{c})$, are neural networks, which each consist of two fully connected layers with 400 and 300 units respectively, and rectified linear unit activation functions.

Training was performed by simulating the environment in episodes of 1000 seconds. Each episode was randomly initialized in where the position of the vessel was chosen within a certain bound of the path, this was done in order to generate new training data for better generalization. During training the state x_t, action u_t, reward r_t and successive state x_{t+1} was recorded and saved in a replay buffer at each time-step t. In addition to training the models from scratch, transfer learning was used in where models trained on straight line paths were used to initialize training for curved paths.

In order to improve the policy and action-value function estimates, $\pi(\mathbf{x}; \theta_\mathbf{a})$ and $Q(\mathbf{x}, \mathbf{u}; \theta_\mathbf{c})$, stochastic gradient decent is performed on batches $\mathbb{B}$ of transitions (x_i, u_i, r_i, x_{i+1}), using the following update rules:

$$\theta_c \leftarrow \theta_c - \alpha_c \frac{1}{N} \sum_{i \in \mathbb{B}} \nabla_{\theta_c} \left(y_i - Q(x_i, u_i; \theta_c)\right)^2 \tag{10}$$

$$\theta_a \leftarrow \theta_a + \alpha_a \frac{1}{N} \sum_{i \in \mathbb{B}} \nabla_{u_i} Q_(x_i, u_i; \theta_a) \nabla_{\theta_a} \pi(x_i; \theta_a) \tag{11}$$

where y_i is the action-value estimate of the state action (x_i, u_i), given as:

$$y_i = r_i + \gamma Q(x_{i+1}, \pi(x_{i+1}; \theta_{a'}); \theta_{c'}) \tag{12}$$

Additionally soft target updates were used in order to stabilize training, where the target networks given by the parameterizations $\theta_{a'}$ and $\theta_{c'}$ slowly tracked the learned parameterizations using the following update rule at each time step.

$$\theta_{a'} = (1 - \tau)\theta_{a'} + \tau\theta_a \tag{13}$$

$$\theta_{c'} = (1 - \tau)\theta_{c'} + \tau\theta_c \tag{14}$$

The training was performed by randomly drawing batches of 64 transitions from a buffer of previously observed transitions, with learning rates $\alpha_a = 1e-4$ and $\alpha_c = 1e-3$, target network update rate $\tau = 1e{-}3$, and discounting rate $\gamma = 0.99$. An overview of the architecture is given in Figure 4, and the DDPG algorithm is summarized in Algorithm 1.

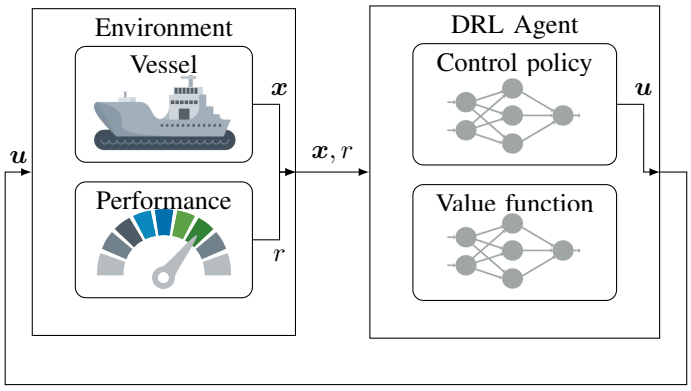


Fig. 4: DDPG architecture used in this paper. Note that the vessel model is unknown to the RL Agent.

Algorithm 1 Deep Deterministic Policy Gradients

Randomly initialize critic and actor weights $\boldsymbol{\theta}_c$ and $\boldsymbol{\theta}_a$.
Initialize target networks with weights $\boldsymbol{\theta}_{c'} \leftarrow \boldsymbol{\theta}_c$, $\boldsymbol{\theta}_{a'} \leftarrow \boldsymbol{\theta}_a$
for episode $= 1, ..., M$ **do**
  Receive initial observation state x_1
  for time step $t = 1, ..., T$ **do**
    Take action u_t based on current policy $\pi(x_t; \boldsymbol{\theta}_a)$, observe reward r_t and new state x_{t+1}, and store transition (x_t, u_t, r_t, x_{t+1}) in replay buffer
    Sample N transitions from replay buffer, and update critic and actor using Equations 10 and 11
    Update target networks using Equations 13 and 14
  end for
end for

	Mariner		Container		Tanker	
Current	$0m/s$	$0.9m/s$	$0m/s$	$0.9m/s$	$0m/s$	$0.9m/s$
LOS	161	499	807	829	65	62
DRL	3264	3134	2430	2763	1010	819

TABLE I: Cumulative Gaussian reward for the curved path following problem

IV. Simulations

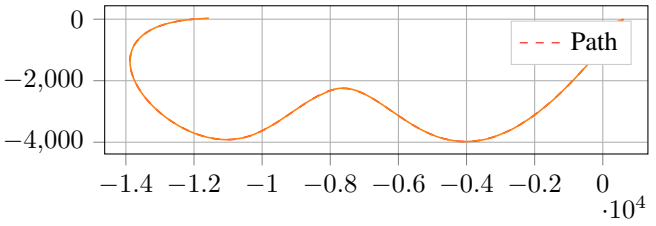
Simulating the policies found when training for the curved path following problem with current, we got the results seen in Figure 5, 6 and 7, for the Mariner, Container and Tanker vessel respectively. We observe that the learned policies are able to control the vessels such that they follow the desired path with very small cross-track errors, especially given the size and complexity of the simulated vessels.

When recording the cumulative reward for the curved path following task we get the results seen in Table I. From the results we see a significant increase in performance for the DRL guidance in comparison to standard line-of-sight (LOS) guidance. While LOS results may vary depending on the tuning of the control parameters, they still give a indication of the performance of the proposed DRL guidance method.

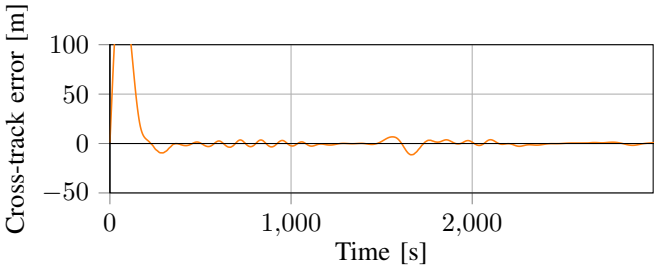
In addition to training the models from scratch, transfer learning was performed using models trained on straight line paths. This resulted in similar performance, however gave faster training. In particular, for a transfer learning approach where 30% of the training data pertained to straight lines, there was a reduction of 40-50% in training time compared to training with curved path data only. This is likely due to the intermediate straight line path following problem being easier to learn, and refine when training for curved path following.

V. Conclusion

In this paper we extended our previous results and implemented a framework based on deep reinforcement learning for curved path following of marine vessels in the presence of ocean currents. The DDPG algorithm was tasked to find suitable steering policies without having any prior info about either the vessel or its environment. In addition to attempting

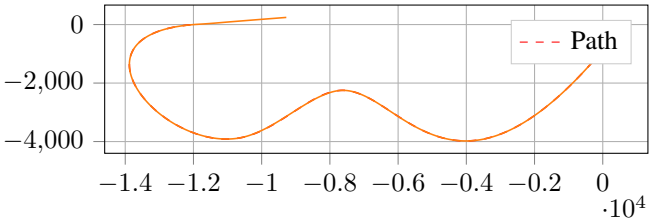


(a) Path.

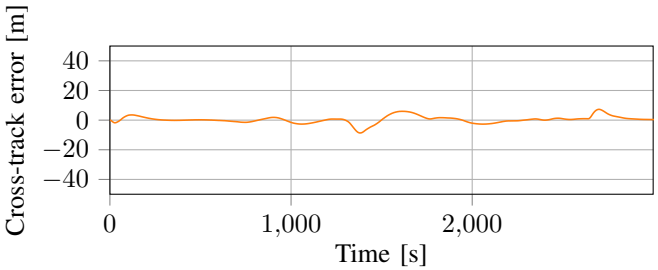


(b) Cross-track error.

Fig. 5: Mariner curved path following when exposed to ocean currents.



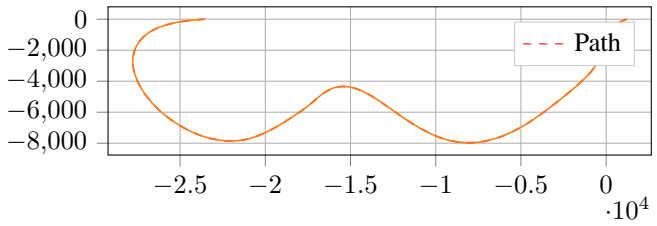
(a) Path.



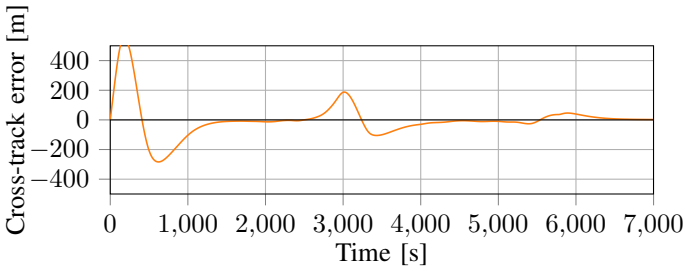
(b) Cross-track error.

Fig. 6: Container curved path following when exposed to ocean currents.

to solve the problem from scratch (i.e. train the algorithm to follow curved paths directly), we used transfer learning to extend a previously trained policy (originally for straight paths) for the same purpose. It turned our that the transfer learning approach was much faster while achieving indistinguishable performance. Three vessel models, with different characteristics and increased complexity compared to previous works, were employed to demonstrate the practicality of the approach and its ability to generalize. This is an indication that



(a) Path.



(b) Cross-track error.

Fig. 7: Tanker curved path following when exposed to ocean currents.

transfer learning might be a reasonable path when attempting to transition DRL-based controllers from simulated environments to the real world.

References

[1] L. Lapierre, D. Soetanto, and A. Pascoal, "Nonlinear path following with applications to the control of autonomous underwater vehicles," in *42nd IEEE Conference on Decision and Control*, pp. 1256–1261, 2003.

[2] E. Fredriksen and K. Y. Pettersen, "Global κ-exponential way-point maneuvering of ships: Theory and experiments," *Automatica*, vol. 42, no. 4, pp. 677–687, 2006.

[3] A. M. Lekkas and T. I. Fossen, "Integral LOS path following for curved paths based on a monotone cubic Hermite spline parametrization," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 6, pp. 2287–2301, 2014.

[4] T. I. Fossen, M. Breivik, and R. Skjetne, "Line-of-sight path following of underactuated marine craft," *IFAC Proceedings Volumes*, vol. 36, no. 21, pp. 211–216, 2003.

[5] R. Skjetne, U. Jørgensen, and A. R. Teel, "Line-of-sight path-following along regularly parametrized curves solved as a generic maneuvering problem," in *50th IEEE Conference on Decision and Control and European Control Conference*, pp. 2467–2474, 2011.

[6] A. P. Aguiar and J. P. Hespanha, "Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty," *IEEE Transactions on Automatic Control*, vol. 52, no. 8, pp. 1362–1379, 2007.

[7] A. M. Lekkas and T. I. Fossen, "Trajectory tracking and ocean current estimation for marine underactuated

vehicles," in *IEEE Conference on Control Applications (CCA)*, pp. 905–910, 2014.

[8] S. Moe, K. Y. Pettersen, T. I. Fossen, and J. T. Gravdahl, "Line-of-sight curved path following for underactuated USVs and AUVs in the horizontal plane under the influence of ocean currents," in *24th IEEE Mediterranean Conference on Control and Automation (MED)*, pp. 38–45, 2016.

[9] L. Liu, D. Wang, and Z. Peng, "ESO-based line-of-sight guidance law for path following of underactuated marine surface vehicles with exact sideslip compensation," *IEEE Journal of Oceanic Engineering*, vol. 42, no. 2, pp. 477–487, 2017.

[10] S.-R. Oh and J. Sun, "Path following of underactuated marine surface vessels using line-of-sight based model predictive control," *Ocean Engineering*, vol. 37, no. 2-3, pp. 289–295, 2010.

[11] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT Press, 1998.

[12] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 2. Athena Scientific, 4th ed., 2012.

[13] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-dynamic programming," 1996.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[15] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[16] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[17] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of The 33rd International Conference on Machine Learning*, pp. 1928–1937, 2016.

[18] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[19] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3389–3396, IEEE, 2017.

[20] H. Shen and C. Guo, "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)*, pp. 317–321, IEEE, 2016.

[21] L. Zhang, L. Qiao, J. Chen, and W. Zhang, "Neural-network-based reinforcement learning control for path following of underactuated ships," in *35th Chinese Control Conference (CCC)*, pp. 5786–5791, IEEE, 2016.

[22] R. Yu, Z. Shi, C. Huang, T. Li, and Q. Ma, "Deep reinforcement learning based optimal trajectory tracking control of autonomous underwater vehicle," in *36th Chinese Control Conference (CCC)*, pp. 4958–4965, IEEE, 2017.

[23] L. Pham Tuyen, L. Abu, N. Vien, and T. Chung, "Deep reinforcement learning algorithms for steering an underactuated ship," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pp. 602–607, IEEE, 2017.

[24] A. B. Martinsen and A. M. Lekkas, "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)*, 2018.

[25] T. I. Fossen and T. Perez, "Marine systems simulator (mss)," 2004.

[26] M. Chislett and J. Strøm-Tejsen, *Planar Motion Mechanism Tests and Full-scale Steering and Manoeuvring Predictions for a Mariner Class Vessel*. Report (Hydro- og earodynamisk laboratorium. Hydrodynamics Department), Hydro- and Aerodynamics Laboratory, 1965.

[27] W. B. Van Berlekom and T. A. Goddard, "Maneuvering of large tankers," 1972.

[28] K.-H. Son and K. Nomoto, "5. on the coupled motion of steering and rolling of a high-speed container ship," *Naval Architecture and Ocean Engineering*, vol. 20, pp. 73–83, 1982.

[29] R. Glatt, F. L. d. Silva, and A. H. R. Costa, "Towards knowledge transfer in deep reinforcement learning," in *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, pp. 91–96, Oct 2016.

[30] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," *CoRR*, vol. abs/1703.06907, 2017.

APPENDIX

Mariner-class vessel: The mariner model is a 3-DOF surface vessel based on the work of M.S. Chislett and J. Stroem-Tejsen [26], and released as a MATLAB model in the MSS Toolbox [25]. The vessel takes in the control rudder angle δ_c, and has the following vessel dynamics:

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{\nu} \tag{15}$$

$$\dot{\boldsymbol{\nu}} = \boldsymbol{M}^{-1}\boldsymbol{N}(\boldsymbol{\nu}, \delta) \tag{16}$$

$$\dot{\delta} = \begin{cases} \delta_c - \delta \text{ If } \delta <= \delta_{\max} \\ 0 \text{ otherwise} \end{cases} \tag{17}$$

The mass matrix is given as

$$\mathbf{M} = \begin{bmatrix} \frac{L}{U^2} & 0 & 0 \\ 0 & \frac{L}{U^2} & 0 \\ 0 & 0 & \frac{L^2}{U^2} \end{bmatrix} \left(\begin{bmatrix} m & 0 & 0 \\ 0 & m & mx_g \\ 0 & mx_g & I_z \end{bmatrix} + \begin{bmatrix} -X_{\dot{u}} & 0 & 0 \\ 0 & -Y_{\dot{v}} & -Y_{\dot{r}} \\ 0 & -Y_{\dot{r}} & -N_{\dot{r}} \end{bmatrix} \right) \tag{18}$$

where $U = \sqrt{u^2 + v^2}$ is the absolute velocity of the vessel. The nonlinear terms are based on the Abkovitz model described above, giving the following polynomial vector:

$$\boldsymbol{N}(\boldsymbol{\nu}, \delta) = \begin{bmatrix} X(\boldsymbol{\nu}, \delta) \\ Y(\boldsymbol{\nu}, \delta) \\ N(\boldsymbol{\nu}, \delta) \end{bmatrix} \tag{19}$$

where the polynomials describing the nonlinear components, are given as:

$$\begin{aligned} X(\boldsymbol{\nu}, \delta) =& X_u u + X_{uu} u_r^2 + X_{uuu} u_r^3 + X_{vv} v_r^2 + X_{rr} r^2 \\ &+ X_{rv} r v_r X_{\delta\delta} \delta^2 + X_{u\delta\delta} u_r \delta^2 X_{v\delta} v_r \delta + X_{uv\delta} u_r v_r \delta \\ Y(\boldsymbol{\nu}, \delta) =& Y_v v_r + Y_r r + Y_{vvv} v_r^3 + Y_{vvr} v_r^2 r + Y_{vu} v_r u_r \\ &+ Y_{ru} r u_r + Y_\delta \delta + Y_{\delta\delta\delta} \delta^3 + Y_{u\delta} u_r \delta + Y_{uu\delta} u_r^2 \delta \\ &+ Y_{v\delta\delta} v \delta^2 + Y_{vv\delta} v^2 \delta + \\ &(Y_0 + Y_{0u} u_r + Y 0_{uu} u_r^2) \\ N(\boldsymbol{\nu}, \delta) =& N_v v_r + N_r r + N_{vvv} v_r^3 + N_{vvr} v_r^2 r + N_{vu} v_r u_r \\ &+ N_{ru} r u_r + N_\delta \delta + N_{\delta\delta\delta} \delta^3 + N_{u\delta} u_r \delta + N_{uu\delta} u_r^2 \delta \\ &+ N_{v\delta\delta} v_r \delta^2 + N_{vv\delta} v_r^2 \delta \\ &+ (N_0 + N_{0u} u_r + N_{0uu} u_r^2) \end{aligned}$$

For the mariner vessel, the parameters are given in Table II, it should also be noted that the velocities $u_r = u/U$ and $v_r = v/U$ used in the nonlinear terms, are relative velocities, with respect to the absolute velocity.

Term	Value	Term	Value	Term	Value
$X_{\dot{u}}$	$-42e-5$	$Y_{\dot{v}}$	$-748e-5$	$N_{\dot{v}}$	$4.646e-5$
X_u	$-184e-5$	$Y_{\dot{r}}$	$-9.354e-5$	$N_{\dot{r}}$	$-43.8e-5$
X_{uu}	$-110e-5$	Y_v	$-1160e-5$	N_v	$-264e-5$
X_{uuu}	$-215e-5$	Y_r	$-499e-5$	N_r	$-166e-5$
X_{vv}	$-899e-5$	Y_{vvv}	$-8078e-5$	N_{vvv}	$1636e-5$
X_{rr}	$18e-5$	Y_{vvr}	$15356e-5$	N_{vvr}	$-5483e-5$
$X_{\delta\delta}$	$-95e-5$	Y_{vu}	$-1160e-5$	N_{vu}	$-264e-5$
$X_{u\delta\delta}$	$-190e-5$	Y_{ru}	$-499e-5$	N_{ru}	$-166e-5$
X_{rv}	$798e-5$	Y_δ	$278e-5$	N_δ	$-139e-5$
$X_{v\delta}$	$93e-5$	$Y_{\delta\delta\delta}$	$-90e-5$	$N_{\delta\delta\delta}$	$45e-5$
$X_{uv\delta}$	$93e-5$	$Y_{u\delta}$	$556e-5$	$N_{u\delta}$	$-278e-5$
		$Y_{uu\delta}$	$278e-5$	$N_{uu\delta}$	$-139e-5$
		$Y_{v\delta\delta}$	$-4e-5$	$N_{v\delta\delta}$	$13e-5$
L	160.93	$Y_{vv\delta}$	$1190e-5$	$N_{vv\delta}$	$-489e-5$
m	$798e-5$	Y_0	$-4e-5$	N_0	$3e-5$
I_z	$39.2e-5$	Y_{0u}	$-8e-5$	N_{0u}	$6e-5$
x_g	-0.023	Y_{0uu}	$-4e-5$	N_{0uu}	$3e-5$

TABLE II: Mariner parameter value table

Tanker Model: The tanker model is a 3-DOF surface vessel based on the work of Van Berlekom, W.B. and Goddard, T.A. [27], and released as a MATLAB model in the MSS Toolbox [25]. The vessel takes in the control vector $\boldsymbol{u}$ consisting of the desired rudder angle δ_c, and desired shaft velocity n_c. The vessel dynamics of the tanker are given as:

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{\nu} \tag{20}$$

$$\dot{\boldsymbol{\nu}} = \boldsymbol{M}^{-1}\boldsymbol{N}(\boldsymbol{\nu}, \delta) \tag{21}$$

$$\dot{\delta} = \begin{cases} \delta_c - \delta \text{ If } \delta <= \delta_{\max} \\ 0 \text{ otherwise} \end{cases} \tag{22}$$

$$\dot{n} = \frac{60}{T_m}(n_c - n) \tag{23}$$

The mass matrix is given as

$$\mathbf{M} = \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & 0 \\ 0 & 0 & m_{33} \end{bmatrix} \tag{24}$$

The nonlinear terms are based on the Abkovitz model described above, where the the nonlinear terms are given by the following polynomial vector:

$$\boldsymbol{N}(\boldsymbol{\nu}, \delta) = \begin{bmatrix} X(\boldsymbol{\nu}, \delta) \\ Y(\boldsymbol{\nu}, \delta) \\ N(\boldsymbol{\nu}, \delta) \end{bmatrix} \tag{25}$$

where the polynomials describing the nonlinear components, are given as:

$$\begin{aligned} X(\boldsymbol{\nu}, \boldsymbol{u}) =& 1/L(X_{uu} u^2 + L d_{11} v r + X_{vv} v^2 + X_{cc\delta\delta} |c| c \delta^2 \\ &+ X_{cc\beta\delta} |c| c \beta\delta + L g_T (1-t) + X_{uuz} u^2 z \\ &+ L X_{vrz} v r z + X_{vvzz} v^2 z^2) \\ Y(\boldsymbol{\nu}, \boldsymbol{u}) =& 1/L(Y_{uv} uv + Y_{vv} |v| v + Y_{cc\delta} |c| c \delta + L d_{22} u r \\ &+ Y_{cc\beta\beta\delta} |c| c \beta\beta\delta + Y_T g_T L \\ &+ L Y_{urz} u r z + Y_{uvz} u v z + Y_{vvz} |v| v z \\ &+ Y_{cc\beta\beta\delta z} |c| c |\beta| \beta |\delta| z) \\ N(\boldsymbol{\nu}, \boldsymbol{u}) =& N_{uv} uv + L N_{vr} |v| r + N_{cc\delta} |c| c \delta + L d_{33} u r \\ &+ N_{cc\beta\beta\delta} |c| c |\beta| \beta |\delta| + L N_T g_T \\ &+ L N_{urz} u r z + N_{uvz} u v z + L N_{vrz} |v| r z \\ &+ N_{cc\beta\beta\delta z} |c| c |\beta| \beta |\delta| z \end{aligned}$$

where β, g_T and c are computed as the following

$$\begin{aligned} \beta &= \frac{v}{u} \\ g_T &= 1/L T_{uu} u^2 + T_{un} u n + L T_{nn} |n| n \\ c &= \sqrt{c_{un} u n + c_{nn} n^2}; \end{aligned}$$

and the parameters for the tanker are given in Table III.

Term	Value	Term	Value	Term	Value
d_{11}	2.020	m_{11}	1.050	T_{uu}	-0.00695
d_{22}	-0.752	m_{22}	2.020	T_{un}	-0.00063
d_{33}	-0.231	m_{33}	0.1232	T_{nn}	0.0000354
X_{uuz}	-0.0061	Y_T	0.04	N_T	-0.02
X_{uu}	-0.0377	Y_{vv}	-2.400	N_{vr}	-0.300
X_{vv}	0.3	Y_{uv}	-1.205	N_{uv}	-0.451
X_{udotz}	-0.05	Y_{vdotz}	-0.387	N_{rdotz}	-0.0045
X_{uuz}	-0.0061	Y_{urz}	0.182	N_{urz}	-0.047
X_{vrz}	0.387	Y_{vvz}	-1.5	N_{vrz}	-0.120
X_{ccdd}	-0.093	Y_{uvz}	0	N_{uvz}	-0.241
X_{ccbd}	0.152	Y_{ccd}	0.208	N_{ccd}	-0.098
X_{vvzz}	0.0125	Y_{ccbbd}	-2.16	N_{ccbbd}	0.688
		Y_{ccbbdz}	-0.191	N_{ccbbdz}	0.344
t	0.22	c_{un}	0.605	L	304.8
T_m	50	c_{nn}	38.2	g	9.8
T	18.46				

TABLE III: Tanker parameter value table

Container Model: The tanker model is a 4-DOF surface vessel based on the work of K.-H Son and K. Nomoto [28]. The model includes the additional roll dynamics, and can be written on vectorial form as:

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}(\nu)\boldsymbol{\nu} \tag{26}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau} \tag{27}$$

The major differences from the 3-DOF model is the addition of the pitch angle ϕ, giving the state vector:

$$\boldsymbol{\nu} = [u, v, \psi, \phi]^\top \tag{28}$$

and the addition of pitch in the rotation matrix.

$$\boldsymbol{J}(\boldsymbol{\eta}) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{29}$$

A. Container model

The Container model is a 4-DOF surface vessel based on the work of Son og Nomoto [28], and released as a MATLAB model in the MSS Toolbox [25]. The vessel model takes in the control vector $\boldsymbol{u}$ consisting of the desired rudder angle δ_c, and desired shaft velocity n_c, and returns the state derivatives $\dot{\boldsymbol{\nu}}$ and $\dot{\boldsymbol{\eta}}$. The vessel dynamics of the tanker are given as:

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{30}$$

$$\dot{\boldsymbol{\nu}} = \boldsymbol{M}^{-1}\boldsymbol{N}(\boldsymbol{\nu}, \delta) \tag{31}$$

$$\dot{\delta} = \begin{cases} \delta_c - \delta \text{ If } \delta <= \delta_{\max} \\ 0 \text{ otherwise} \end{cases} \tag{32}$$

$$\dot{n} = \frac{60}{T_m}(n_c - n) \tag{33}$$

The mass matrix is given as

$$\mathbf{M} = \frac{L}{U^2} \begin{bmatrix} m + m_x & 0 & 0 & 0 \\ 0 & m + m_y & -m_y l_y L & m_y \alpha_y L \\ 0 & -m_y l_y & (I_x + J_x)L & 0 \\ 0 & m_y \alpha_y & 0 & (I_z + J_z)L \end{bmatrix} \tag{34}$$

where $U = \sqrt{u^2 + v^2}$ is the absolute velocity. The nonlinear terms are based on the Abkovitz model described above, where the the nonlinear terms are given by the following polynomial vector:

$$\boldsymbol{N}(\boldsymbol{\nu}, \delta) = \begin{bmatrix} X(\boldsymbol{\nu}, \delta) \\ Y(\boldsymbol{\nu}, \delta) \\ K(\boldsymbol{\nu}, \delta) \\ N(\boldsymbol{\nu}, \delta) \end{bmatrix} \tag{35}$$

where the polynomials describing the nonlinear components, are given as:

$$\begin{aligned} X(\boldsymbol{\nu}, \boldsymbol{u}) =& X_{uu}u^2 + (1-t)T + X_{vr}vr + X_{vv}v^2 + X_{rr}r^2 \\ &+ X_{\phi\phi}\phi^2 + c_{RX}F_N \sin(\delta) + (m + m_y)vr \\ Y(\boldsymbol{\nu}, \boldsymbol{u}) =& Y_v v + Y_r r + Y_p p + Y_\phi \phi + Y_{vvv}v^3 + Y_{rrr}r^3 \\ &+ Y_{vvr}v^2 r + Y_{vrr}vr^2 + Y_{vv\phi}v^2\phi + Y_{v\phi\phi}v\phi^2 \\ &+ Y_{rr\phi}r^2\phi + Y_{r\phi\phi}r\phi^2 + (1 + a_H)F_N \cos(\delta) \\ &- (m + m_x)ur \\ K(\boldsymbol{\nu}, \boldsymbol{u}) =& K_v v + K_r r + K_p p + K_\phi \phi + K_{vvv}v^3 + K_{rrr}r^3 \\ &+ K_{vvr}v^2 r + K_{vrr}vr^2 + K_{vv\phi}v^2\phi + K_{v\phi\phi}v\phi^2 \\ &+ K_{rrphi}r^2\phi + K_{r\phi\phi}rphi^2 - (1 + a_H)z_R F_N \cos(\delta) \\ &+ m_x l_x ur - WG_M\phi \\ N(\boldsymbol{\nu}, \boldsymbol{u}) =& N_v v + N_r r + N_p p + N_\phi \phi + N_{vvv}v^3 + N_{rrr}r^3 \\ &+ N_{vvr}v^2 r + N_{vrr}vr^2 + N_{vv\phi}v^2\phi + N_{v\phi\phi}v\phi^2 \\ &+ N_{rr\phi}r^2\phi + N_{r\phi\phi}r\phi^2 + (x_R + a_H x_H)F_N \cos(\delta) \end{aligned}$$

utilizing the following calculations

$$\begin{aligned} v_R &= g_a v + c_{Rr}r + c_{Rrrr}r^3 + c_{Rrrv}r^2 v \\ u_P &= \cos(v)((1 - w_p) + \tau((v + x_p r)^2 + c_{pv}v + c_{pr}r)) \\ J &= u_P U/(nD) \\ K_T &= 0.527 - 0.455J \\ u_R &= u_P \epsilon\sqrt{1 + 8k_k K_T/(\pi J^2)} \\ \alpha_R &= \delta + \text{atan}(v_R/u_R) \\ F_N &= -((6.13\Delta)/(\Delta + 2.25))(A_R/L^2)(u_R^2 + v_R^2)\sin(\alpha_R) \\ T &= 2\rho D^4/(U^2 L^2 \rho)K_T n|n| \\ W &= \rho g \nabla/(\rho L^2 U^2/2) \end{aligned}$$

where the parameters are given in Table IV.

Term	Value	Term	Value	Term	Value
m	0.00792	m_x	0.000238	m_y	0.007049
I_x	0.0000176	α_y	0.05	l_x	0.0313
l_y	0.0313	I_x	0.0000176	I_z	0.000456
J_x	0.0000034	J_z	0.000419	x_G	0
B	25.40	d_F	8.00	g	9.81
d_A	9.00	d	8.50	∇	21222
K_M	10.39	K_B	4.6154	A_R	33.0376
Δ	1.8219	D	6.533	G_M	$0.3/L$
ρ	1025	t	0.175	T	0.0005
X_{uu}	−0.0004226	X_{vr}	−0.00311	X_{rr}	0.00020
$X_{\phi\phi}$	−0.00020	X_{vv}	−0.00386		
K_v	0.0003026	K_r	−0.000063	K_p	−0.0000075
K_ϕ	−0.000021	K_{vvv}	0.002843	K_{rrr}	−0.0000462
K_{vvr}	−0.000588	K_{vrr}	0.0010565	$K_{vv\phi}$	−0.0012012
$K_{v\phi\phi}$	−0.0000793	$K_{rr\phi}$	−0.000243	$K_{r\phi\phi}$	0.00003569
Y_v	−0.0116	Y_r	0.00242	Y_p	0
Y_ϕ	−0.000063	Y_{vvv}	−0.109	Y_{rrr}	0.00177
Y_{vvr}	0.0214	Y_{vrr}	−0.0405	$Y_{vv\phi}$	0.04605
$Y_{v\phi\phi}$	0.00304	$Y_{rr\phi}$	0.009325	$Y_{r\phi\phi}$	−0.001368
N_v	−0.0038545	N_r	−0.00222	N_p	0.000213
N_ϕ	−0.0001424	N_{vvv}	0.001492	N_{rrr}	−0.00229
N_{vvr}	−0.0424	N_{vrr}	0.00156	$N_{vv\phi}$	−0.019058
$N_{v\phi\phi}$	−0.0053766	$N_{rr\phi}$	−0.0038592	$N_{r\phi\phi}$	0.0024195
k_k	0.631	ϵ	0.921	x_R	−0.5
w_p	0.184	τ	1.09	x_p	−0.526
c_{pv}	0.0	c_{pr}	0.0	g_a	0.088
c_{Rr}	−0.156	c_{Rrrr}	−0.275	c_{Rrrv}	1.96
c_{RX}	0.71	a_H	0.237	z_R	0.033
x_H	−0.48				

TABLE IV: Container parameter value table