A red lion and black text

Description automatically generated

**TRAJECTORY TRACKING CONTROL SYSTEM FOR AUTONOMOUS SURFACE VESSELS USING DEEP REINFORCEMENT LEARNING**

Student: Nam Tran (675973)

Lecturer: Hung D. Nguyen

Unit Code: JEE506

Unit Title: Modelling and Simulation of Marine Systems

Date Due: 11 November 2024

Table of Contents

[**INTRODUCTION** 3](#_Toc181586627)

[**METHODOLOGY** 3](#_Toc181586628)

[**1.** **Container Vessel** 3](#_Toc181586629)

[**2.** **PID Controller** 5](#_Toc181586630)

[**3.** **Deep Reinforcement Learning Approach** 6](#_Toc181586631)

[**SIMULATION SETUP** 6](#_Toc181586632)

[**1.** **PID Controller** 6](#_Toc181586633)

[**2.** **DDPG Agent** 6](#_Toc181586634)

[**RESULTS AND DISCUSSION** 6](#_Toc181586635)

[**1.** **Autopilot** 6](#_Toc181586636)

[**2.** **Speed Control** 6](#_Toc181586637)

[**3.** **Trajectory Tracking Control** 6](#_Toc181586638)

[**CONCLUSION** 7](#_Toc181586639)

[**REFERENCES** 7](#_Toc181586640)

[Figure 1. Turning Circle Test 4](#_Toc181587042)

[Figure 2. 20o/20o Zigzag Test 5](#_Toc181587043)

[Table 1. Container Ship's Parameters 3](#_Toc181587052)

# **INTRODUCTION**

Traditionally, maritime navigation relies on classical control methods, especially the Proportional-Integral-Derivative (PID) controllers, for autopilot and trajectory tracking systems. While effective in stable and predictable conditions, PID controllers are limited in adapting to complexities of real-world environments, where disturbances such as wind, waves and ocean currents can affect a vessel’s path (Fossen 2011). These limitations are particularly evident in tasks requiring precise path-following, such as survey missions or area coverage operations, where complex patterns are used to maximise coverage (Galceran et al. 2015). With recent advancements in machine learning, Deep Reinforcement Learning (DRL) has emerged as a promising alternative to traditional control systems (Buşoniu et al. 2018). Unlike PID controllers, DRL-based controllers can learn directly from interacting with the environment, enabling them to adaptively handle a wide range of scenarios without explicit configurations. This adaptability makes DRL a suitable candidate for the trajectory tracking control of marine vessels, which often encounter unpredictable environmental dynamics. This paper aims to design a DRL-based trajectory tracking controller for a container vessel, leveraging the vessel’s model to enable efficient tracking of patterns, therefore, advancing the application of adaptive control in maritime navigation.

In the past, several studies have implemented reinforcement learning for controlling marine systems. Rohit et al. (2023) conduct simulations on the Krisco container ship using a 3-DOF dynamic model, focusing on yaw, surge, and sway. The authors develop a Deep Q-Learning (DQN) based controller for path following, where the speed of the vessel is constant, and the controllable actions are three discrete values of rudder angle to change directions. Four observation states: course-track error, course angle error, distance to destination and yaw rate, are monitored to keep track of the path. Along with simulation works, the authors deploy to field experiments with a model scale vessel to validate their work, proving that with simple discrete changes in rudder angle, the DRL agent is able to control the vessel for navigating along the predefined path.

Zhang et al. (2020) propose a DRL-based design for controlling a four-thruster ASV using Deep Deterministic Policy Gradient (DDPG), which is specialised to train agents with continuous state space and action space. In the paper, the DDPG agent is compared against a nonlinear model predictive control (NMPC) in fixed-point control and trajectory tracking. The DDPG controller shows higher performance than NMPC in the fixed-point control environment and a good trajectory tracking effect.

Similarly, Yu, Shi, Huang, Li, et al. (2017) implement DDPG to control an AUV in a horizontal plane. The authors compare with the traditional PID controller in two path tracking scenarios: straight line and curve line. In the simulation work, the DDPG controller results in a higher accuracy and faster response than the PID controller in both cases.

In this paper, a reinforcement learning environment is constructed with MATLAB, and a DDPG agent is created to learn the control law of a nonlinear 3-DOF container vessel model. The machine learning model is tested against the PID controller in three scenarios: autopilot, speed control, and trajectory tracking control.

# **METHODOLOGY**

1. **Container Vessel**

In this study, the container ship’s mathematical model presented by Son and Nomoto (1981) is chosen for running simulation and designing control systems. The motion of the container vessel is described through a 3 DOF (degree of freedom) model, which focuses on motion in x-direction (surge), motion in y-direction (sway), and rotational motion about z-axis (yaw). Figure 1 shows the kinematics of a vessel in the global coordinate system, with yaw angle , rudder angle , surge velocity and sway velocity .

The general equations to calculate the yaw angle, surge and sway velocities are expressed by Equation (1).

|  |  |  |  |
| --- | --- | --- | --- |
| Surge | |  | (1) |
| Sway | |  |
| Yaw | |  |
| where | | m : mass of ship [kg.s2/m] | | |
|  | | : acceleration in x direction [m/s2] | | |
|  | | : acceleration in y direction [m/s2] | | |
|  | | : angular acceleration [rad/s2] | | |
|  | | IZZ : inertia moment with respect to z-axis [kgs2m] | | |
|  | | XH, YH : Hydrodynamic forces acting on ship’s hull [kg] | | |
|  | | NH : Hydrodynamic moment acting on the ship’s hull [kg.m] | | |
|  | | XP : Propulsive force of propeller [kg] | | |
|  | | XR, YR : Hydrodynamic forces acting on ship’s rudder [kg] | | |
|  | | NR : Hydrodynamic moment acting on ship’s rudder [kg.m] | | |
|  | | YT : Hydrodynamic force induced by thruster [kg] | | |
|  | | NT : Hydrodynamic moment induced by thruster [kg.m] | | |

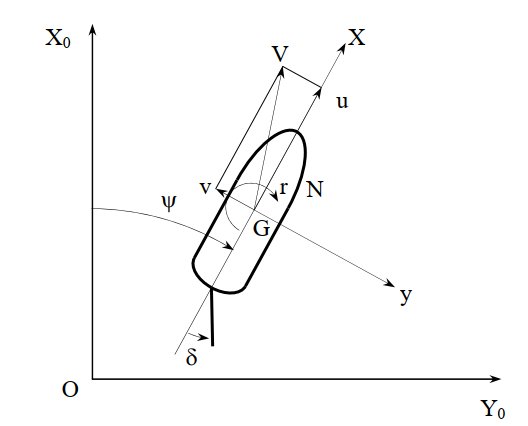


Figure 1. 3-DOF Kinematics of a Vessel

The state equations of ship’s heading and position on the earth-fixed coordinate system can be derived as:

|  |  |  |
| --- | --- | --- |
| Yaw rate |  | (2) |
| Position in x-axis |  |
| Position in y-axis |  |

The container ship’s parameters are listed in Table 1.

Table 1. Container Ship's Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | | **Value** | |
| Length (L) | | 175.00 | [m] |
| Breadth (B) | | 25.40 | [m] |
| Draft | fore (dF) | 8.00 | [m] |
| aft (dA) | 9.00 | [m] |
| mean (d) | 8.50 | [m] |
| Displacement volume () | | 21,222.00 | [m3] |
| Height from keel to transverse metacentre (KM) | | 10.39 | [m] |
| Height from keel to centre of buoyancy (KB) | | 4.6154 | [m] |
| Block coefficient (CB) | | 0.559 | [-] |
| Rudder area (AR) | | 33.0376 | [m2] |
| Aspect ratio () | | 1.8219 | [-] |
| Propeller diameter | | 6.533 | [m] |

The nonlinear equations of motion, including surge, sway, roll and yaw, are presented by Equation (3).

|  |  |
| --- | --- |
|  | (3) |
|  |
|  |
|  |

In this set of equations, and are the added mass and added moment of inertia in the x and y directions and about the z and x axes, respectively. denotes the x-coordinates of the centre of . and are the z-coordinates of the centres of and . The hydrodynamic forces and moment are derived in Equation (4).

From these equations, the dynamic model of the container vessel is programmed in MATLAB and expressed in the form of a state vector:

|  |  |
| --- | --- |
|  | (4) |
|  |
|  |
|  |

**Turning circle**

Turning circle is one of the common methods to test the manoeuvrability of a vessel (IMO 2002). In this test, the ship’s heading is initially set as and the rudder angle is kept at with the commanded shaft velocity of .

A graph with a blue circle

Description automatically generated

Figure . Turning Circle Test

After running the open-loop simulation for 600 seconds, the container ship’s trajectory is plotted in Figure 2. The smooth and circular path shows that the vessel achieved a steady-state turn, maintaining a nearly constant turning radius once it fully entered the circular path. This consistent radius is important in evaluating the vessel’s response to control commands for a stable turning circle.

**Zigzag test**

Another standard test for manoeuvring check is the 20o/20o zigzag test. Initially, the vessel starts at a steady speed on a straight course. The rudder angle is turned to 20o on one side and held until the vessel reaches a heading deviation from its original course. The rudder is then shifted to 20o on the opposite side and the vessel turns to the opposite direction. The process is repeated each time the vessel deviates 20o from its new heading, creating a zigzag pattern. The 20o/20o zigzag test is a valuable manoeuvring trial for measuring how well a vessel’s control system can manage repeated course corrections.

In this open-loop system, the simulation is executed for 600 seconds, and the zigzag trajectory is plotted in Figure 3.

A graph with a line graph

Description automatically generated

Figure . 20o/20o Zigzag Test

The trajectory displays a clear zigzag pattern, with symmetrical deviations to the left and right of the centreline, suggesting a consistent response to rudder inputs. This symmetry indicates that the model is stable, maintaining a balanced response in each direction.

1. **PID Controller**

The Proportional-Integral-Derivative (PID) controller is a widely used control algorithm in engineering due to its simplicity and effectiveness in a variety of applications, including maritime navigation and trajectory tracking. The PID controller operates by calculating an error value as the difference between a desired setpoint and a measured process variable (Ogata 2020). The controller then adjusts the control output to minimize this error by applying three distinct control actions: proportional, integral, and derivative.

* Proportional control (P): The proportional term produces an output value that is proportional to the current error. It is calculated as:

|  |  |
| --- | --- |
|  | (5) |

where *Kp* is the proportional gain, and *e(t)* is the error at time *t*. The proportional control action provides immediate response to the current error, making it effective for reducing the magnitude of the error.

* Integral control (I): The integral term is concerned with the accumulation of past errors, addressing the residual steady-state error that often occurs with proportional control alone. It is calculated as:

|  |  |
| --- | --- |
|  | (6) |

where *Ki* is the integral gain. The integral action integrates the error over time, allowing the controller to eliminate steady-state error by adjusting the control output based on the cumulative error.

* Derivative control (D): The derivative term predicts future error based on its rate of change, providing a damping effect on the system. It is calculated as:

|  |  |
| --- | --- |
|  | (7) |

where *Kd* is the derivative gain. This term helps to counteract the overshoot and improve system stability by considering how quickly the error is changing.

The overall control output of the PID controller is given by the sum of these three terms:

|  |  |
| --- | --- |
|  | (8) |

where *u(t)* is the control signal sent to the system.

In this study, the PID controller serves as a benchmark for evaluating the performance of the proposed Deep Reinforcement Learning (DRL) controller, providing a baseline against which the adaptive capabilities of the DRL approach can be measured.

1. **Deep Reinforcement Learning Approach**

The Deep Deterministic Policy Gradient (DDPG) algorithm is a model-free, off-policy reinforcement learning method designed for continuous action spaces (Lillicrap 2015). It is an extension of the deterministic policy gradient (DPG) algorithm that leverages deep neural networks to approximate both the policy and value functions. DDPG is particularly suitable for environments with high-dimensional continuous states and actions, such as autonomous vehicle navigation and control tasks (Sutton and Barto 2018).

DDPG utilises an actor-critic architecture, where two neural networks are used. The actor network learns the policy, mapping states to deterministic actions, while the critic network estimates the value function to evaluate the quality of the chosen actions, given the current state. To break the correlation between consecutive experiences, DDPG utilises an experience replay buffer. The agent stores transitions in the buffer and sample mini batches for training. This improves the stability of the learning process by allowing the model to learn from past experiences in a random order (Mnih et al. 2015).

DDPG uses target networks to stabilize training. These are copies of the actor and critic networks that are updated slowly using soft updates. The target networks provide stable targets for the Q-value updates during training, reducing the variance and instability in the learning process. The target critic network is updated by a soft update rule:

|  |  |
| --- | --- |
|  | (9) |

where and are the parameters of the main and target networks, respectively, and is a small factor used to slowly update the target network.

The critic network is trained by minimising the loss between the predicted Q-values and the target Q-values, which are computed using the Bellman equation:

|  |  |
| --- | --- |
|  | (10) |

where is the discount factor, is the action chosen by the target actor network, and is the target critic network.

The actor network is updated by performing a gradient ascent on the deterministic policy, which maximises the expected return. The gradient is computed as follows:

|  |  |
| --- | --- |
|  | (11) |

where is the action taken by the actor network for state , and is the Q-value predicted by the critic network.

To balance exploration and exploitation during training, DDPG adds noise to the action chosen by the actor. This is done by adding noise to the deterministic actions to encourage exploration of the action space.

# **SIMULATION SETUP**

1. **PID Controller**

For all test scenarios, the following parameters are chosen for the PID controller (Table 2).

Table . PID Parameters

|  |  |  |
| --- | --- | --- |
| Autopilot |  | 2.5 |
|  | 0.0002 |
|  | 3 |
| Speed Control |  |  |
|  |  |
|  |  |
| Trajectory Tracking |  |  |
|  |  |
|  |  |

The first simulation scenario is autopilot, where the rudder angle needs to be controlled to keep a desired heading angle value. The surge velocity is set to a constant value of throughout the experiment.

The second test scenario is speed control. In this simulation, the rudder angle stays constant while the shaft velocity is controlled to reach a desired speed.

The final simulation is trajectory tracking control.

1. **DDPG Agent**

**Autopilot**

To train the DDPG agent for the autopilot scenario, the observation space, action space and reward function need to be defined. In each state, the agent observes the yaw angle, which is limited to its physical range of . For the action space, the agent can choose a continuous value between [-10\*pi/180 10\*pi/180]. The set up of observation space and action space is similar to the PID controller design approach.

For the reward function, a simple logic is applied using the error between the desired heading and actual heading:

|  |  |
| --- | --- |
|  | (11) |

By applying this reward function, the agent will try to maximise the reward by minimising the penalty, which is reducing the error.

**Speed control**

Similar to the previous scenario, since the controlled target is the speed of the vessel, the observation space is the surge velocity value. In turn, the agent can take action by adjusting the shaft velocity from 1 to 200 RPM.

# **RESULTS AND DISCUSSION**

1. **Autopilot**
2. **Speed Control**
3. **Trajectory Tracking Control**

# **CONCLUSION**

# **REFERENCES**

1. Buşoniu, L., De Bruin, T., Tolić, D., Kober, J. and Palunko, I., 2018. Reinforcement learning for control: Performance, stability, and deep approximators. *Annual Reviews in Control*, *46*, pp.8-28.
2. Deraj, R., Kumar, R.S., Alam, M.S. and Somayajula, A., 2023. Deep reinforcement learning based controller for ship navigation. *Ocean Engineering*, *273*, p.113937.
3. Fossen, T.I., 2011. Handbook of marine craft hydrodynamics and motion control. *John Willy & Sons Ltd*.
4. Galceran, E., Campos, R., Palomeras, N., Ribas, D., Carreras, M. and Ridao, P., 2015. Coverage path planning with real‐time replanning and surface reconstruction for inspection of three‐dimensional underwater structures using autonomous underwater vehicles. *Journal of Field Robotics*, *32*(7), pp.952-983.
5. IMO, E., 2002. Standards for ship manoeuvrability. *Imo Resolut. Msc*, *137*(76), p.4.
6. Lillicrap, T.P., 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.
7. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S., 2015. Human-level control through deep reinforcement learning. *nature*, *518*(7540), pp.529-533.
8. Ogata, K., 2020. Modern control engineering.
9. Son, K. and Nomoto, K., 1981. On the coupled motion of steering and rolling of a high speed container ship. *Journal of the Society of Naval Architects of Japan*, *1981*(150), pp.232-244.
10. Sutton, R.S. and Barto, A.G., 2018. *Reinforcement learning: An introduction*. MIT press.
11. Yu, R., Shi, Z., Huang, C., Li, T. and Ma, Q., 2017, July. Deep reinforcement learning based optimal trajectory tracking control of autonomous underwater vehicle. In *2017 36th Chinese control conference (CCC)* (pp. 4958-4965). IEEE.
12. Zhang, S., Yang, R., Chen, Z. and Li, M., 2020, November. Autonomous Surface Vehicle Control Method Using Deep Reinforcement Learning. In *2020 Chinese Automation Congress (CAC)* (pp. 1664-1668). IEEE.