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

Abstract

In maritime engineering, control for guidance and navigation of vessels is a crucial problem. Traditionally, this can be done by designing and applying a Proportional-Integral-Derivative (PID) controller for a stable closed-loop system. With recent advancement in machine learning, reinforcement learning proves to be an excellent replacement for the classical control method, due to its sequential decision-making ability. In this study, a Deep Deterministic Policy Gradient (DDPG) model is developed and validated against the PID approach, to control a container vessel in three simulation scenarios: autopilot, speed control and trajectory tracking control. The results show that the PID controller outperforms the DDPG agent. However, the reinforcement learning-based controller can be improved in the future with better configuration and design.

Table of Contents

INTRODUCTION 3
METHODOLOGY 4
1. **Container Vessel** 4
2. **PID Controller** 7
3. **Deep Reinforcement Learning Approach** 8
4. **Test Scenarios** 9
SIMULATION SETUP 10
1. **PID Controller** 10
2. **DDPG Agent** 10
RESULTS AND DISCUSSION 11
1. **Autopilot** 11
2. **Speed Control** 12
3. **Trajectory Tracking Control** 13
CONCLUSION 14
REFERENCES 15
APPENDICES 16
APPENDIX A. App Designer 16

Figure 1. 3-DOF Kinematics of a Vessel 4
Figure 2. Turning Circle Test 6
Figure 3. 20°/20° Zigzag Test 7
Figure 4. Autopilot Training Reward 11
Figure 5. Autopilot Scenario 12
Figure 6. Speed Control Training Reward 12
Figure 7. Speed Control Scenario 13
Figure 8. Trajectory Tracking Control Parameters 14
Figure 9. Trajectory Tracking Control Scenario 14
Figure 10. Proposed Idea of the App 16

Table 1. Container Ship's Parameters 5
Table 2. PID Parameters 10
Table 3. DDPG Agent Configuration 11

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 ψ, rudder angle δ, surge velocity u and sway velocity v.

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

$$\begin{aligned} &\text{Surge} && m(\dot{u} - vr) = X_H + X_P + X_R \\ &\text{Sway} && m(\dot{u} - vr) = Y_H + Y_T + Y_R \\ &\text{Yaw} && I_{ZZ}\dot{r} = N_H + N_T + N_R \end{aligned} \tag{1}$$

where
- m : mass of ship [kg.s^2/m]
- $\dot{u}$: acceleration in x direction [m/s^2]
- $\dot{v}$: acceleration in y direction [m/s^2]
- $\dot{r}$: angular acceleration [rad/s^2]
- I_{ZZ} : inertia moment with respect to z-axis [kgs^{2}m]
- X_H, Y_H : Hydrodynamic forces acting on ship's hull [kg]
- N_H : Hydrodynamic moment acting on the ship's hull [kg.m]
- X_P : Propulsive force of propeller [kg]
- X_R, Y_R : Hydrodynamic forces acting on ship's rudder [kg]
- N_R : Hydrodynamic moment acting on ship's rudder [kg.m]
- Y_T : Hydrodynamic force induced by thruster [kg]
- N_T : Hydrodynamic moment induced by thruster [kg.m]

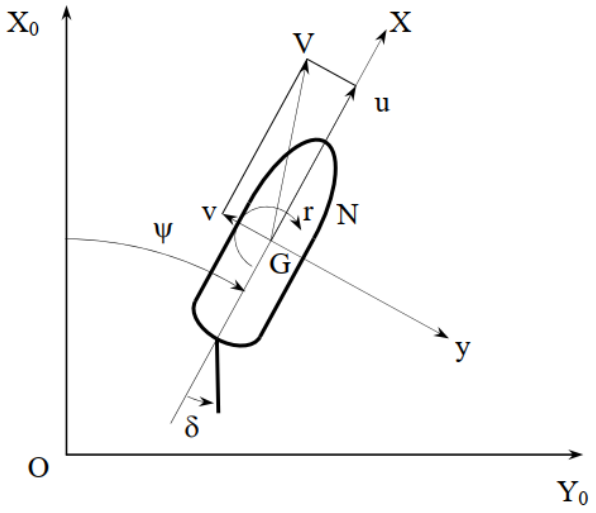

Figure 1. 3-DOF Kinematics of a Vessel

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

Yaw rate	$\dot{\psi} = r$	
Position in x-axis	$\dot{X} = u\cos\psi - v\sin\psi$	(2)
Position in y-axis	$\dot{Y} = u\sin\psi + v\cos\psi$	

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 (d_F)	8.00	[m]
	aft (d_A)	9.00	[m]
	mean (d)	8.50	[m]
Displacement volume (∇)		21,222.00	[m^3]
Height from keel to transverse metacentre (KM)		10.39	[m]
Height from keel to centre of buoyancy (KB)		4.6154	[m]
Block coefficient (C_B)		0.559	[-]
Rudder area (A_R)		33.0376	[m^2]
Aspect ratio (Λ)		1.8219	[-]
Propeller diameter		6.533	[m]

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

$$\begin{gathered}(m' + m'_x)\dot{u}' - (m' + m'_y)v'r' = X' \\ (m' + m'_y)\dot{v}' + (m' + m'_x)u'r' + m'_y\alpha'_y\dot{r}' - m'_y l'_y\dot{p}' = Y' \\ (I'_x + J'_x)\dot{p}' - m'_y l'_y\dot{v}' - m'_x l'_x u'r' + W'\overline{GM}'\phi' = K' \\ (I'_z + J'_z)\dot{r}' - m'_y\alpha'_y\dot{v}' = N' - Y'x'_G\end{gathered} \tag{3}$$

In this set of equations, m'_x, m'_y, J'_z and J'_x are the added mass and added moment of inertia in the x and y directions and about the z and x axes, respectively. α'_y denotes the x-coordinates of the centre of m'_y. I'_x and I'_y are the z-coordinates of the centres of m'_x and m'_y. 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:

$$\dot{x} = \begin{bmatrix} u - \text{surge velocity [m/s]} \\ v - \text{sway velocity [m/s]} \\ r - \text{yaw velocity [rad/s]} \\ x - \text{position in x [m]} \\ y - \text{position in y [m]} \\ \psi - \text{yaw angle [rad]} \\ p - \text{roll velocity [rad/s]} \\ \phi - \text{roll angle [rad]} \\ \delta - \text{rudder angle [rad]} \\ n - \text{shaft velocity [rpm]} \end{bmatrix}$$

$$X' = X'(u') + (1-t)T'(J) + X'_{vr}v'r' + X'_{vv}v'^2 + X'_{rr}r'^2 + X''_{\phi\phi}\phi'^2 + c_{RX}F'_N \sin\delta'$$

$$\begin{aligned} K' = {} & K'_v v' + K'_r r' + K'_p p' + K'_\phi \phi' + K'_{vvv}v'^3 + K'_{rrr}r'^3 + K'_{vvr}v'^2 r' + K'_{vrr}v'r'^2 \\ & + K'_{vv\phi}v'^2\phi' + K'_{v\phi\phi}v'\phi'^2 + K'_{rr\phi}r'^2\phi' + K'_{r\phi\phi}r'\phi'^2 \\ & - (1+a_H)z'_R F'_N \cos\delta' \end{aligned}$$

$$\begin{aligned} Y' = {} & Y'_v v' + Y'_r r' + Y'_p p' + Y'_\phi \phi' + Y'_{vvv}v'^3 + Y'_{rrr}r'^3 + Y'_{vvr}v'^2 r' + Y'_{vrr}v'r'^2 \\ & + Y'_{vv\phi}v'^2\phi' + Y'_{v\phi\phi}v'\phi'^2 + Y'_{rr\phi}r'^2\phi' + Y'_{r\phi\phi}r'\phi'^2 \\ & + (1+a_H)F'_N \cos\delta' \end{aligned} \quad (4)$$

$$\begin{aligned} N' = {} & N'_v v' + N'_r r' + N'_p p' + N'_\phi \phi' + N'_{vvv}v'^3 + N'_{rrr}r'^3 + N'_{vvr}v'^2 r' + N'_{vrr}v'r'^2 \\ & + N'_{vv\phi}v'^2\phi' + N'_{v\phi\phi}v'\phi'^2 + N'_{rr\phi}r'^2\phi' + N'_{r\phi\phi}r'\phi'^2 \\ & + (x'_R + a_H x'_H)F'_N \cos\delta' \end{aligned}$$

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 $\psi = 0^o$ and the rudder angle is kept at $\delta = 20^o$ with the commanded shaft velocity of $n = 80\ m/s$.

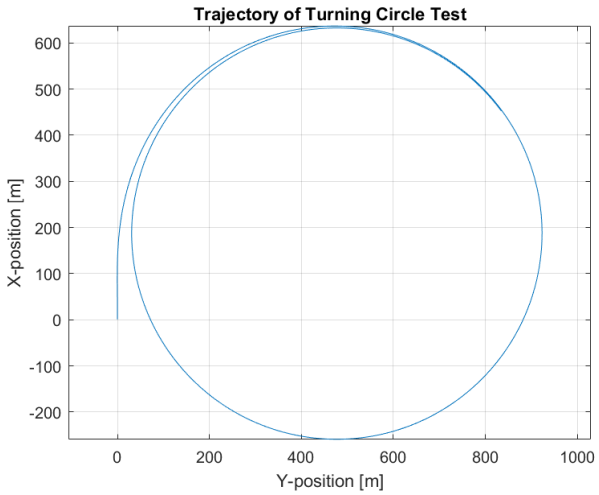

Figure 2. Turning Circle Test

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 20°/20° zigzag test. Initially, the vessel starts at a steady speed on a straight course. The rudder angle is turned to 20° on one side and held until the vessel reaches a heading deviation from its original course. The rudder is then shifted to 20° on the opposite side and the vessel turns to the opposite direction. The process is repeated each time the vessel deviates 20° from its new heading, creating a zigzag pattern. The

20°/20° 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.

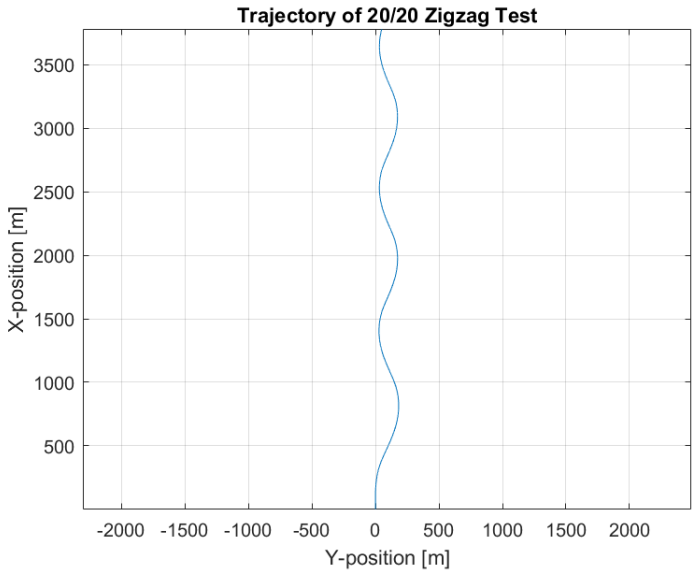

Figure 3. 20°/20° Zigzag Test

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.

2. 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:

$$\mathrm{P} = \mathrm{K_p} \times \mathrm{e(t)} \tag{5}$$

where K_p 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:

$$I = K_i \times \int_0^t e(\tau)\, d\tau \tag{6}$$

where K_i 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:

$$D = K_d \times \frac{de(t)}{dt} \tag{7}$$

where K_d 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:

$$u(t) = P + I + D \tag{8}$$

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.

3. 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 (s_t, a_t, r_t, s_{t+1}) 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 Q' is updated by a soft update rule:

$$\theta' = \tau\theta + (1 - \tau)\theta' \tag{9}$$

where θ and θ' are the parameters of the main and target networks, respectively, and τ 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:

$$y_t = r_t + \gamma Q'\left(s_{t+1}, \pi'(s_{t+1})\right) \tag{10}$$

where γ is the discount factor, $\pi'(s_{t+1})$ is the action chosen by the target actor network, and Q' 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:

$$\nabla_{\theta^\mu} J \approx E_{s_t \sim \rho^\beta}[\nabla_a Q(s_t, a)|_{a=\mu(s_t)} \nabla_{\theta^\mu} \mu(s_t) \tag{11}$$

where $\mu(s_t)$ is the action taken by the actor network for state s_t, and $Q(s_t, a)$ 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.

4. Test Scenarios

In this paper, there are three test scenarios to assess the performances between the PID and DDPG controllers, namely autopilot, speed control and trajectory tracking control.

Autopilot

The first simulation scenario is autopilot, where the rudder angle needs to be controlled to keep a desired heading angle value. Initially, the surge velocity and shaft velocity are set as $u = 8\ m/s$ and $n = 70$ rpm, respectively. The desired yaw angle is $\psi = 20^o$ throughout the experiment.

Speed Control

The second test scenario is speed control. Similar to the autopilot scenario, the initial value of surge velocity is 8 m/s and the shaft velocity is set to 80 rpm. The goal of this test is to control both rudder angle and shaft velocity to keep the desired heading of $\psi = 20^o$ and speed, or surge velocity, of $u = 10\ m/s$.

Trajectory Tracking Control

For the final simulation scenario, a set of waypoints are selected to create a path for the container vessel to follow, which are:

$$waypoints = (x, y) = [(0,0), (1000, 1000), (1000, 3000), (2000, 5000), (4000, 5000)]$$

For the navigation task, a simple waypoint guidance algorithm is applied to calculate the desired heading angle in each timestep, expressed by Equation 12 (Fossen 2002).

$$\psi_d(t) = \text{atan2}\left(y_k - y(t), x_k - x(t)\right) \tag{12}$$

where (x_k, y_k) is the current waypoint, and $\left(x(t), y(t)\right)$ is the vessel position.

When navigating between waypoints, a switching mechanism for selecting the next waypoint is needed. In this case, the radius of circle of acceptance is defined as $R_0 = L_{pp}/2$, where L_{pp} is the length of the vessel. Follow by Equation 13, it can be checked if the vessel has reached the waypoint, and if so, a new waypoint with index $k = k + 1$ is selected.

$$\psi_d(t) = \text{atan2}\big(y_k - y(t), x_k - x(t)\big) \tag{13}$$

In this simulation, to achieve a smooth path along the waypoints, both speed and heading are controlled, where the initial and desired surge velocities are $u_0 = 2\ m/s$ and $u_d = 5\ m/s$, and the yaw angle is updated in each timestep.

SIMULATION SETUP

1. PID Controller

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

Table 2. PID Parameters

Rudder Angle	K_p	2.5
	K_i	0.0002
	K_d	3
Shaft Velocity	K_p	196
	K_i	5.9
	K_d	414.38

These parameters are tuned using the built-in PID tuner in Simulink. The first set of PID gains are used to control the rudder angle to keep the desired heading angle for the autopilot scenario. For the second and third scenarios, both rudder angle and shaft velocity have different PID gains to control the heading (yaw) and speed (surge) of the container vessel for speed control and trajectory tracking control.

2. 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 $\psi = [-pi\ \ pi]$. 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 square of error e between the desired heading and actual heading (Equation 14).

$$\text{reward} = -|e^2| = -|(\psi_{\text{desired}} - \psi_{\text{actual}})^2| \tag{14}$$

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 targets are speed and heading of the vessel, the observation space consists of surge velocity and yaw angle. In turn, the agent can take action by adjusting the shaft velocity from 1 to 200 RPM and selecting a rudder angle of [-10*pi/180 10*pi/180].

To create a consistency among the experiments, the reward function for speed control has the same working mechanism as autopilot, with the addition of speed error (Equation 15).

$$\text{reward} = -\left|e^2_{\text{heading}} + e^2_{\text{speed}}\right| = -\left|(\psi_{\text{desired}} - \psi_{\text{actual}})^2 + (u_{\text{desired}} - u_{\text{actual}})^2\right| \quad (14)$$

The configuration of DDPG agent and hyperparameters are summarised in Table 3.

Table 3. DDPG Agent Configuration

Parameter	Value
Number of hidden units	64
Learning rate	0.001
Discount factor	0.99
Standard deviation	0.3
Mean	0
Number of episodes	1000
Episode length	600

RESULTS AND DISCUSSION

1. Autopilot

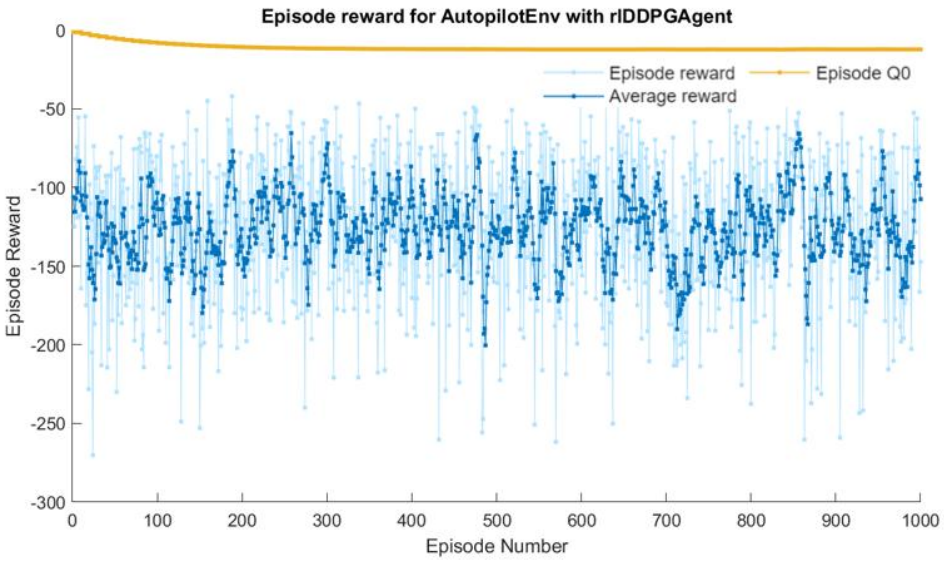

Figure 4. Autopilot Training Reward

Figure 4 shows the plot of episode reward and average reward after 1000 episodes. Based on the chart, it is noticeable that the reward does not improve much over time, thus, leads to unsuccessful decision-making and control process.

To test the performance of the machine learning model, Figure 5 compares between the PID controller and the DDPG controller. From the first plot, it is evident that the DDPG agent failed to learn the control laws of the heading angle, results in a significant lower performance in comparison to the PID controller, where it reaches the desired course after approximately 100 seconds.

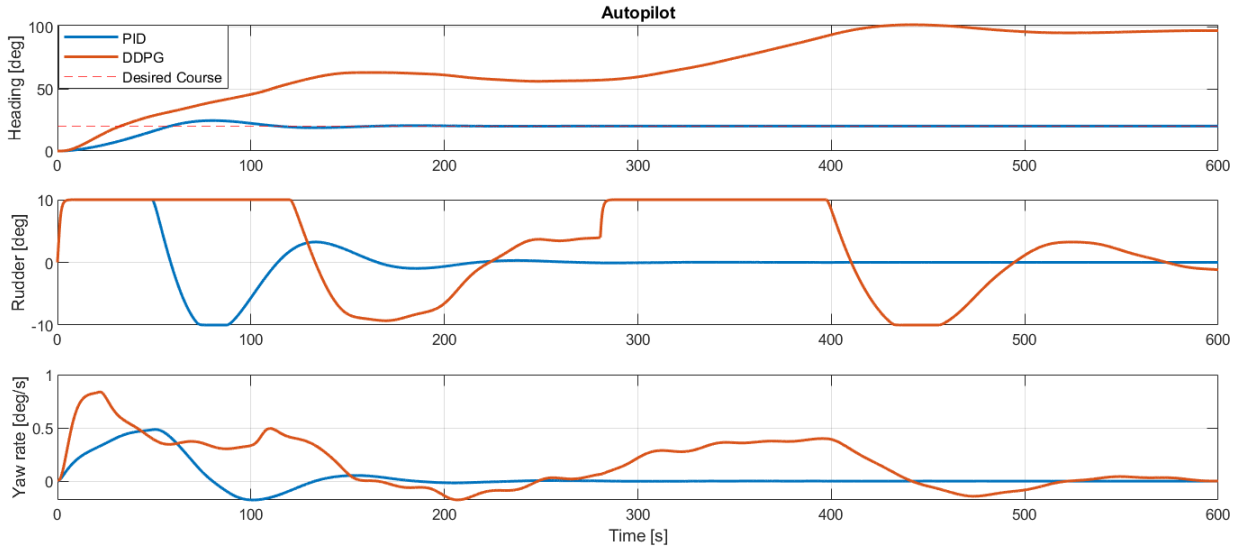

Figure 5. Autopilot Scenario

2. Speed Control

For the speed control scenario, the agent is also trained for 1000 episodes, where the training result is shown in Figure 6.

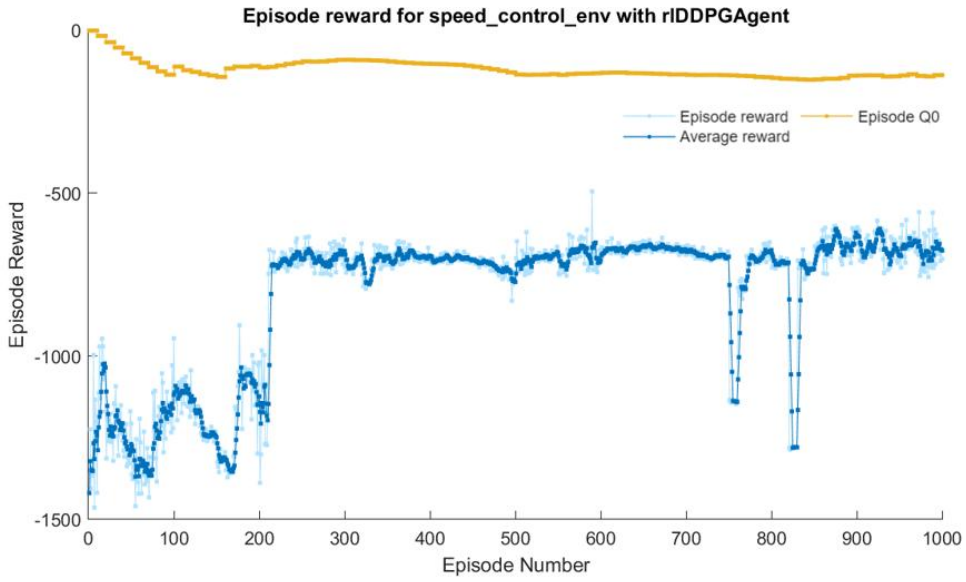

Figure 6. Speed Control Training Reward

The training of speed control shows a more promising result than the previous scenario, where the reward is improved by more than half. However, it is not close enough to a good performance, where the reward is still relatively low.

When compared to the PID controller (Figure 7), the surge velocity of DDPG agent reaches a steady-state earlier than the previous case, however, there is an error of 1m/s in the reinforcement learning-based controller. As for the vessel heading control, the DDPG agent still fails to keep a steady yaw angle, while the PID controller has the better performance.

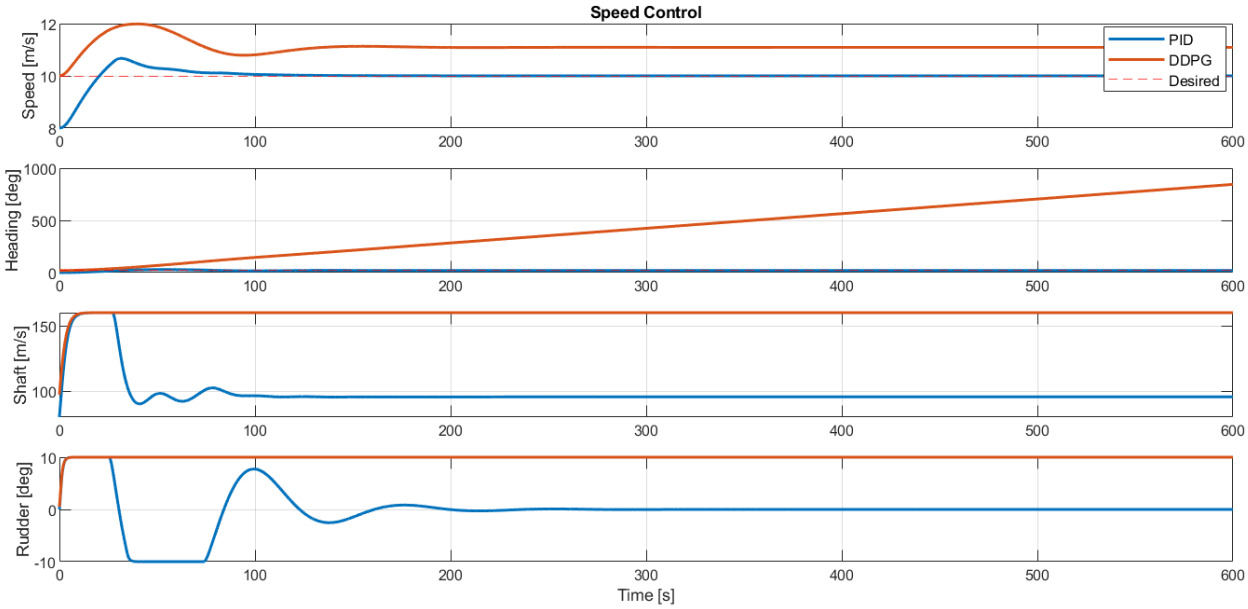

Figure 7. Speed Control Scenario

From the two scenarios, it is clear that the DDPG agent cannot control the heading and speed of the vessel accurately. There are several reasons why this happens. Firstly, the environment in MATLAB may be set up incorrectly, leads to errors in observation or action space. The second reason is that the reward function is too simple and does not help the agent learn the optimal policy. By improving these two points, the DDPG agent may be able to outperform the PID controller, as proven by related studies.

3. Trajectory Tracking Control

Since the DDPG agent is not good enough to control the yaw angle and surge velocity, only PID controller is implemented for the trajectory tracking control scenario.

Figure 8 shows the plot of speed and heading over time, along with the shaft velocity and rudder angle. Overall, the PID controller is able to keep the desired steady-state value of speed, and the yaw angle is stable after reaching each waypoint.

To investigate further, Figure 9 maps the trajectory of the vessel over 1600 seconds, the red dots indicate the waypoints, and the dashed line represent the reference trajectory. The container vessel is able to reach each checkpoint with relatively smooth trajectories.

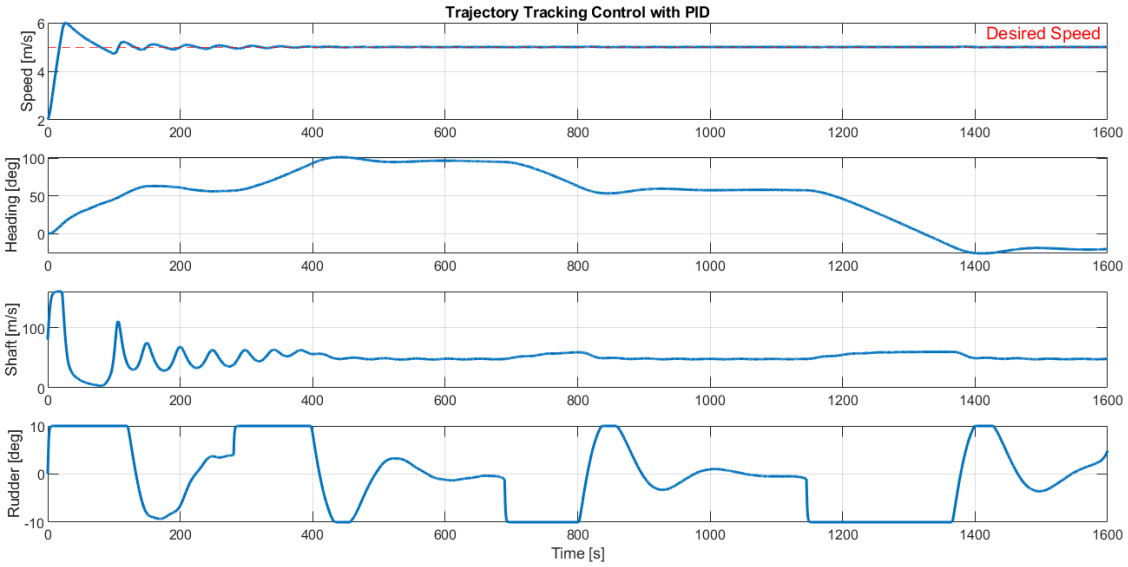

Figure 8. Trajectory Tracking Control Parameters

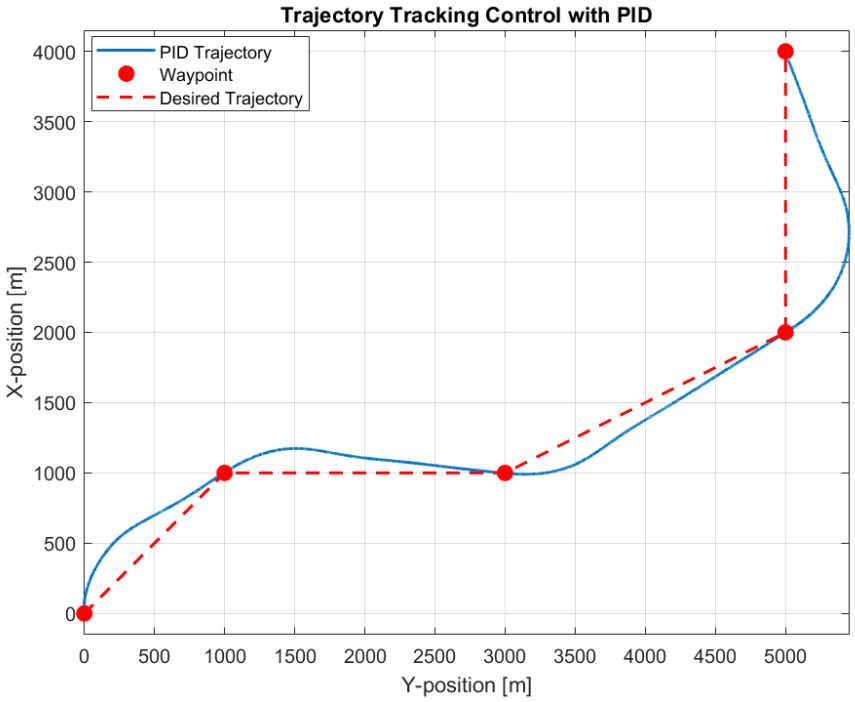

Figure 9. Trajectory Tracking Control Scenario

CONCLUSION

In this study, two controllers, PID and DDPG, are designed to control a container vessel for autopilot, speed control, and trajectory tracking control. The results show that the DDPG agent fail to learn the control laws and unable to control the vessel accurately, in comparison to the classical PID controller, which contradicts the hypothesis made by related studies. The reason for this is concluded as lack of knowledge in reinforcement learning in general, which leads to inaccuracy in defining observation and action space, along with poor reward function design. For future work, the reinforcement learning-based model will be revised for a better training results and overall performance.

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., 2002. Marine control systems–guidance. navigation, and control of ships, rigs and underwater vehicles. *Marine Cybernetics, Trondheim, Norway, Org. Number NO 985 195 005 MVA, www. marinecybernetics. com, ISBN: 82 92356 00 2*.

[4] Fossen, T.I., 2011. Handbook of marine craft hydrodynamics and motion control. *John Willy & Sons Ltd*.

[5] 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.

[6] IMO, E., 2002. Standards for ship manoeuvrability. *Imo Resolut. Msc*, *137*(76), p.4.

[7] Lillicrap, T.P., 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

[8] 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.

[9] Ogata, K., 2020. Modern control engineering.

[10] 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.

[11] Sutton, R.S. and Barto, A.G., 2018. *Reinforcement learning: An introduction*. MIT press.

[12] 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.

[13] 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.

APPENDICES

APPENDIX A. App Designer

As part of the requirement for this assignment, an app needs to be designed to show the results in each section conveniently, along with an instruction manual. However, due to time constraints, I was unable to finish the app in time. In the submission file, the folder "App Designer" includes three main scripts:

- AutopilotPlot.m: Upon running this script, the results between the two controllers are displayed with their trajectory.
- SpeedControlPlot.m: The vessel's parameters and trajectory from both controllers are also displayed upon running the file, simulating the speed and heading control scenario.
- TrajectoryTrackingPID.m: This script only displays the results of the PID controller, since the DDPG agent was not developed for this scenario.

The "main.mlapp" file shows the proposed idea of the app, which I will explain shortly.

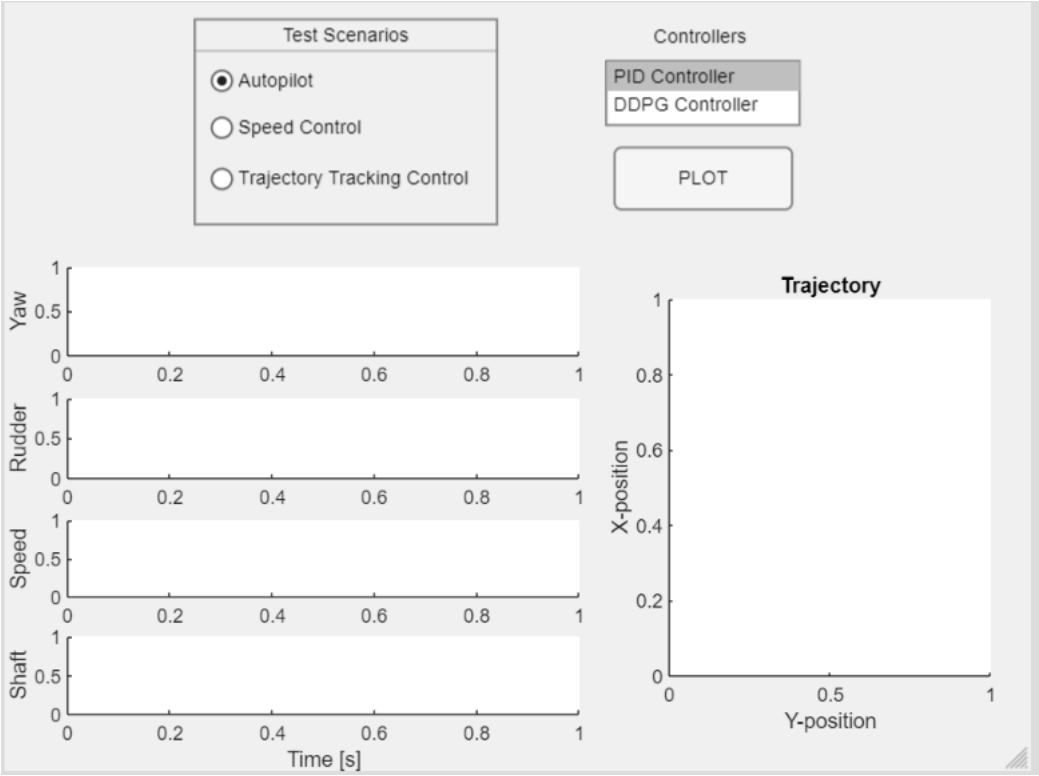

Figure 10. Proposed Idea of the App

Initially, the user is required to select one of the three scenarios: autopilot, speed control, or trajectory tracking control. Then, the user can select either PID or DDPG controller, or both controllers. By clicking PLOT, the yaw and rudder angle, speed and shaft velocity will be displayed on the left side, while the trajectory of the vessel is mapped on the right-hand side. By implementing this, the user can observe the PID or DDPG controller separately, or both of them at the same time to compare the performance.