International Journal of Applied Resilience and Sustainability, Volume 2, Issue 2, April 2026, pp. 1130-1145

Published online at Deep Science

International Journal of Applied Resilience and Sustainability

Journal homepage: https://deepscipub.com/ijars

Multi-agent cooperative control of a three-tank liquid system using reinforcement learning algorithms

Lincoln Nobert Munhenzwa [1], Adlen Kerboua [2], Godfrey Murairidzi Gotora [3]

[1] *Agriculture and Biosystems Engineering, University of Zimbabwe, Harare, Zimbabwe*
[2] *Department of Petrochemistry LGMM Laboratory, University of Skikda, Skikda, Algeria*
[3] *Electrical and Electronics Engineering*, University of Zimbabwe, Harare, Zimbabwe

Article Info:
Received 19 February 2026
Revised 25 March 2026
Accepted 28 March 2026
Published 06 April 2026

Corresponding Author:
Godfrey Murairidzi Gotora
E-mail: godfrey.gotora@ieee.org

Abstract

This research paper aims to present a multi-agent cooperative reinforcement learning approach for controlling the highly nonlinear dynamics of a three-tank liquid system. Meaning to say the system's complexity arises from the interdependence of the tanks, where precise control is required to maintain stable fluid levels. Initially, we deploy two twin-delayed deep deterministic policy Gradient agents to manage the inflow valves, then followed by two proximal policy optimization agents tasked with controlling the valves. The main goal is to compare the performance of these two models of agents against traditional proportional-integral-derivative controllers. In addition to promote effective collaboration between agents, a cooperative reward structure is implemented, encouraging agents to work together to maintain balanced fluid levels within all three tanks. The reward function penalizes deviations from target levels, accounting for both local performance and system-wide stability. The proposed method also addresses key challenges in multi-agent systems, such as non-stationarity and coordination in decentralized control, by integrating a centralized critic during training with decentralized execution. Experimental results reveal that the twin-delayed deep deterministic policy agents outperform the proportional integral derivative control system in terms of settling time, rise time, and robustness, showcasing their ability to handle the nonlinear nature of the system with minimal tuning.

Keywords: Reinforcement learning, Multi-agent, Control system, Artificial intelligence, Deep reinforcement learning, Machine learning.

1. Introduction

The reinforcement learning has emerged as a promising alternative to traditional control methods for complex, non- linear systems. A canonical example of a nonlinear, nonstationary, multi-variable process with strong inter-tank coupling, presents limitations for classical PID controllers due to their susceptibility to performance degradation from parametric uncertainty and unmodeled dynamics [1]. This study proposes a multi-agent cooperative reinforcement learning framework employing a hybrid architecture: twin-delayed deep deterministic policy gradient agents for continuous inflow valve actuation and proximal policy optimization agents for stable policy iteration [1,2]. The framework utilizes a centralized training with decentralized execution paradigm to mitigate non-stationarity, where a centralized critic coordinates collaboration via a bespoke altruistic reward function that minimizes a global cost functional penalizing state deviation [2]. Empirical evaluation demonstrates that this TD3-PPO framework achieves superior closed-loop performance metrics specifically improved settling time, rise time, and robustness compared to PID baselines by effectively learning a coordinated control policy that manages the system's nonlinear dynamics through a cooperative reward structure which aligns local actuator objectives with system-wide stability [2,3].

https://doi.org/10.70593/deepsci.0202046

2. Methodology

While significant progress has been made in using MARL for multi-agent control systems, particularly with algorithms like MADDPG, PPO, and TD3, there remain considerable challenges in terms of scalability, coordination, and non-stationarity. The application of MARL to complex, nonlinear systems, such as the three-tank liquid system, demonstrates the potential of RL-based approaches to outperform traditional control methods like PID. However, the development of more robust, scalable, and generalizable MARL algorithms is necessary to fully realize the potential of multi-agent systems in real-world applications.

Overview of the proposed method

Before delving into the proposed control system based on cooperative MARL frame- work where TD3 and in some cases the PPO agents work in tandem to control the inflow and outflow valves of the three-tank, we proceed by presenting the detailed mathematical description of the highly nonlinear Three-Tank Liquid system, and then we describe the structure of the proposed control system. The proposed system is a cooperative multi-agent control framework designed to manage the nonlinear dynamics of a three-tank liquid system (Fig. 1), where two types of reinforcement learning agents, twin-delayed deep deterministic policy gradient) and proximal policy optimization, are deployed to control the inflow valves of each tank. The system leverages a cooperative reward structure to encourage agents to collaborate in maintaining stable fluid levels across the interdependent tanks, where each agent observes real-time tank levels (and potentially flow rates) and takes actions to adjust valve positions accordingly. A centralized critic is used during training to coordinate learning between agents, but in real-time, agents operate independently with decentralized execution, allowing for scalable and flexible control.

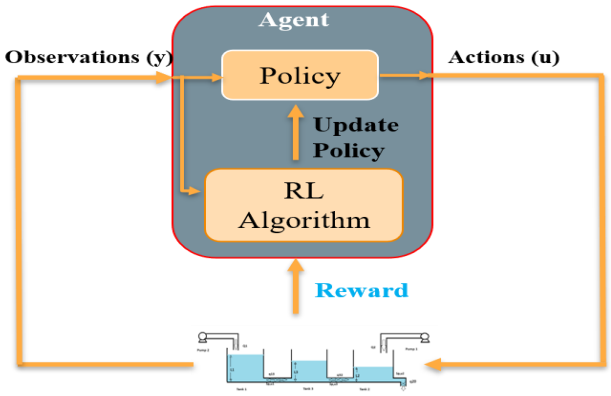

Fig. 1 Overview of the proposed agent-based control system.

Mathematical model of the Three-Tank Liquid (TTL) system

The Three-Tank Liquid (TTL) system is widely used in process control to model complex, nonlinear fluid dynamics. It consists of three interconnected tanks with two inflows and three outputs, representing a multivessel flow system. This system is non- linear due to the interactions between the tanks, making it a popular benchmark for testing control algorithms

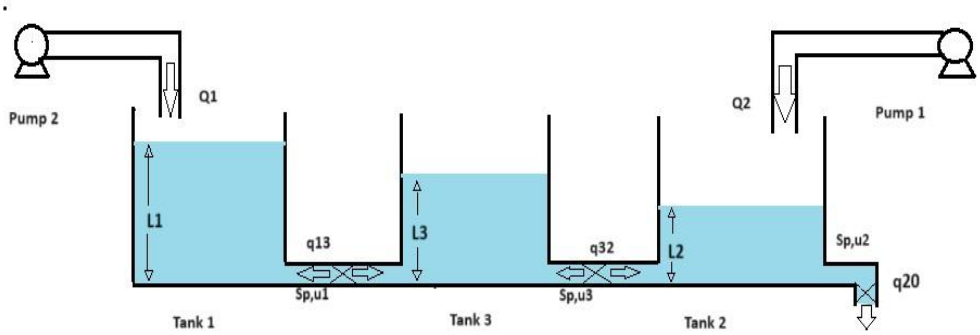

Fig. 2 Schematic of three coupled tank system.

The Three-Tank Liquid (TTL) system described here consists of two inputs, Q1 and Q2, which represent the inflow rates from Pump1 and Pump2 into Tank1 and Tank2 respectively. The liquid from Tank3 is supplied by the coupling effect between Tank1 and Tank2. The liquid that enters the tanks flows from a reservoir, which is also the source of liquid for the pumps. The tanks are interconnected through bidirectional valves, which allow liquid to flow between them based on pressure differences, and this results in a closed-loop system.

Inputs: Q1 and Q2 (inflow rates from the two pumps).

Outputs: L1, L2, and L3 (liquid levels in Tank1, Tank2, and Tank3 respectively).

In this closed system, Tank3 receives liquid as a result of the interactions between Tank1 and Tank2, and this configuration ensures a complex coupling between the tanks. The main control challenge is reducing the response time and achieving a faster settling of liquid levels to the desired values.

To model the system mathematically, we can apply a volume balance on each tank. The liquid level in each tank can be described by differential equations, derived from the continuity equation or volume balance law, which relates the rate of change of liquid level to the difference between inflows and outflows.

Let S_1, S_2, and S_3 represent the cross-sectional areas of the three tanks. The height (liquid level) in each tank is the three outputs L_1, L_2, and L_3 for Tank1, Tank2, and Tank3, respectively. The nonlinear model, which reflects the dynamic behavior of the system, is derived from Torricelli's law.

$$S_i \frac{dL_i(t)}{dt} = \sum inflow\ rate\ - \sum outflow\ rate \tag{1}$$

The differences between the inflow and outflow rates changes the level inside each tank. The nonlinear state model reflecting the dynamic behavior of the process is written as:

$$ForTank1: S\frac{dL_1(t)}{dt} = Q_1 - Q_{13}$$

$$ForTank2: S\frac{dL_2(t)}{dt} = Q_2 - Q_{32} - Q_{20}$$

$$ForTank3: S\frac{dL_3(t)}{dt} = Q_{13} - Q_{32} \tag{2}$$

Where Q_{ij} represents the flow rates between two adjacent tanks i and j. According to Torricelli's relation. Each of these equations governs the rate of change of the liquid level in the tanks, depending on the inflow from the pumps and the coupling flows between the tanks [3].

$$Q_{ij} = \mu_{ij} S_p \sqrt{2g}\, sign(L_i - L_j) \sqrt{|L_i - L_j|} \tag{3}$$

The output flow $Q_{20} = \mu_{20} S_p \sqrt{2gL_2}$, then:

$$\begin{bmatrix} \dot{L_1} \\ \dot{L_2} \\ \dot{L_3} \end{bmatrix} = \frac{1}{S}[A]\begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} + \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = I_{3\times 3}\begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} \quad (4)$$

With the Matrice A given as below;

$$\begin{bmatrix} -\mu_{13}S_p\sqrt{2g}sign(L_1\_L_3)\sqrt{|L_1 - L_3|} \\ S_p\sqrt{2g}sign(L_3\_L_2)\sqrt{|L_3 - L_2|} - \mu_{20}\sqrt{L_2} \\ S_p\sqrt{2g}((sign(L_1\_L_3)\sqrt{|L_1 - L_3|}) - \mu_{32}sign(L_3\_L_2)\sqrt{|L_3 - L_2|} \end{bmatrix}$$

We obtain three nonlinear differential equations because of the square roots on the heights L_1, L_2, and L_3 [4,5]. Under the hypothesis $L_1 > L_3 > L_2$, above nonlinear model becomes:

$$\begin{bmatrix} \dot{L_1} \\ \dot{L_2} \\ \dot{L_3} \end{bmatrix} = \frac{1}{S}[A]\begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} + \frac{1}{S}\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} \quad (6)$$

With A being

$$A = \begin{bmatrix} -\mu_{ij}S_p\sqrt{2g}sign(L_i\_L_j) \\ S_p\sqrt{2g}\mu_{32}\sqrt{|L_3 - L_2|} - \mu_{20}\sqrt{L_2} \\ S_p\sqrt{2g}\mu_{13}(\sqrt{|L_1 - L_3|}) - \mu_{32}\sqrt{|L_3 - L_2|} \end{bmatrix}$$

Where μ_{ij} denotes the pipe flow coefficients between T_i and T_j, and g represents acceleration due to gravity [5,6]. The dynamics of the system can be represented in State Space form as shown in Eq (9) below:

$$\begin{aligned} x(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \quad (7)$$

The linearized state space model was obtained using nominal process parameters and it is mentioned in Eq (2.5)

$$A = \begin{bmatrix} -0.0136 & 0 & 0.0136 \\ 0 & -0.0278 & 0.096 \\ 0.0136 & 0.0096 & -0.0233 \end{bmatrix}, \quad B = \begin{bmatrix} 64.9351 & 0 \\ 0 & 64.9351 \\ 0 & 0 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In this study, the state space model of the system is considered as the agent environment. the primary challenge is to design a control system that efficiently regulates $Q_1$and Q_2 to achieve the desired liquid levels L_1, L_2, and L_3.

1.Reinforcement Learning Strategies

Despite major advances in control engineering, the PID controller remains the most often used controller in many real-world control applications because of its simple design and reliable performance, as noted by PI/PID controllers [6]. The PI form is typically used for level control applications [7]. That's why PID tuning is the focus of this study. In order to modify the control action, the PID controller calculates the error, e(t), between the measured process variable and the intended set point (SP) at each instant. The controller output, u(t), is provided in Eq (11). The performance of this tuned PID is compared to those of trained RL algorithms.

$$u(t) = K_p e(t) + K_i\left(\int_0^t e(\tau)d\tau\right) + K_d\frac{de(t)}{dt} \quad (8)$$

The K_i, K_p and K_d controller parameters are precisely determined using the MAT- LAB internal PID tuner using a linearized model of the plant. These settings might be adjusted using both online and offline methods. On the other hand, when the control aims or the system dynamics change, it becomes crucial to modify the PID parameters online. In this study, two popular RL algorithms TD3 and Proximal Policy Optimization (PPO) are proposed to control the liquid levels in the 3 tanks. These algorithms are

modelled in a mimic the performance of a PID controller with one fully connected layer, with integral, derivative and proportional error as in the Eq. (12).

$$u = \left[\int edte \frac{de}{dt}\right] * \left[K_p K_i K_d\right]^T \tag{9}$$

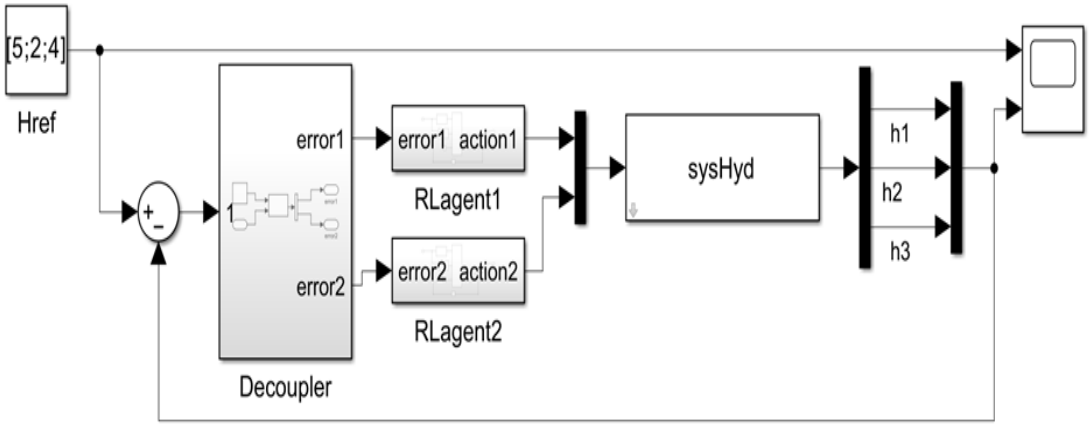

Fig. 3 MATLAB Block diagram of the three-tank system and the controller.

u is the output of the actor neural network, k_p, k_i and k_d are the absolute values of the neural network weights $e = H_{ref} - h$, h is the height of the tank, and H_{ref} is the reference height. This method's low learning efficiency and challenges in training the networks make it challenging to apply directly. In order to reduce training time and prevent undesirable (bad) states during exploration, this work suggests applying the pre-training and fine-tuning strategy to deep learning. Pre-training, a term used in deep learning, is the process of teaching networks how to complete a task before they begin, imitating how people absorb new information. The weights saved from the previously trained network will serve as the starting weight for the current experiment [8]. The pre-training and fine-tuning strategy are employed in this work to enable new models in effectively completing new tasks based on prior experience rather than starting from blank. The PID controller parameters evaluated using IMC tuning rules shown in Table 1 are used as initial weights for the TD3 algorithm to avoid the bad states and improve the learning efficiency.

Table 1. Parameters of PID controllers

Parameter	PID 1	PID 2
K_p	0.0001	0.0007
K_i	0	0
K_d	0.0008	6.4221
N (Filter Coefficient)	0.0089	6.4221

2. Reward function

The main element of the RL framework is the reward. The agent will receive rewards for each action it does in the real-world scenario. An agent is rewarded positively if they take the initiative to behave well; if they conduct poorly, they are penalised or given a negative reward. The convergence, speed, and stability of reinforcement learning (RL) are determined by the reward function that agents use to learn optimal policies. In this work, the RL agent's reward function is defined as the negative of the LQG cost; by maximising this reward, the RL agent minimises the LQG cost.

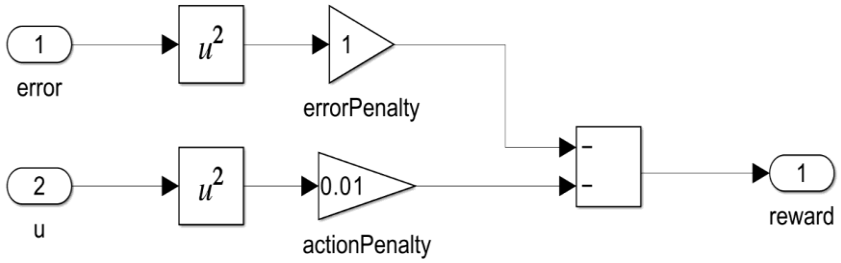

Fig. 4 The reward function.

$$\text{Reward} = -(H_{ref} - h(t))^2 + 0.001u^2(t)\,dt \tag{11}$$

$e = H_{ref} - h$, h is the current height of the tank, and H_{ref} is the reference height.

Observation vector

The observation vector, sometimes referred to as the status vector, is an essential element in reinforcement learning (RL) that captures the agent's perception of the current condition of the environment. The three observation states examined by the RL algorithms in this work are the error, integral error, and derivative error.

$$Observation\ vector = \left[\int e dt\ e\ \frac{de}{dt}\right]^T \tag{12}$$

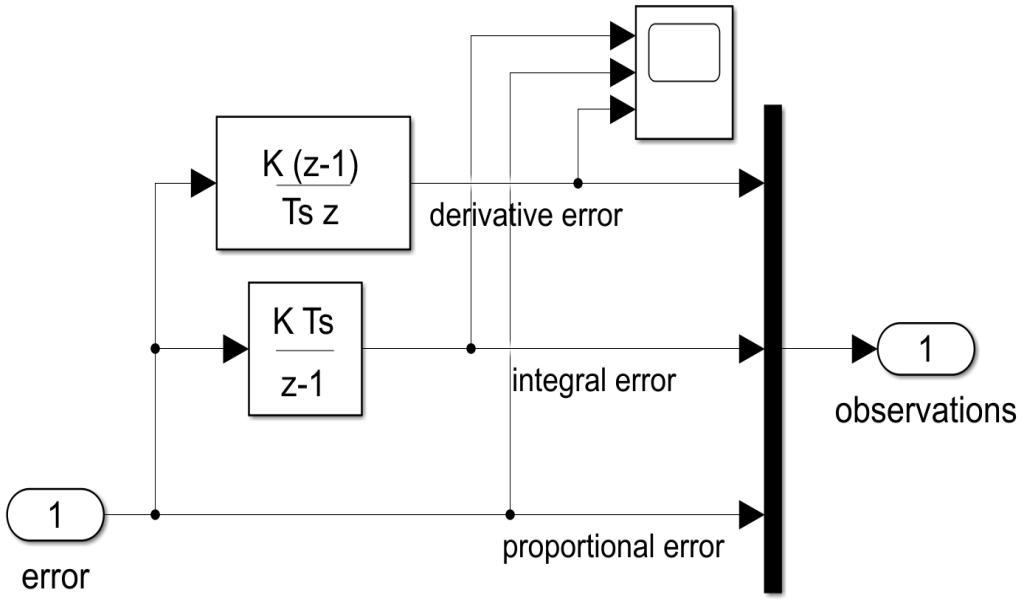

Fig. 5 The observation vector.

$e = H_{ref} - h$, h is the height of the tank and H_{ref} is the reference height. The observation vector is used to update the agent's policy through the learning algorithm [9]. The agent learns to associate states (observations) with actions that maximize some notion of cumulative reward.

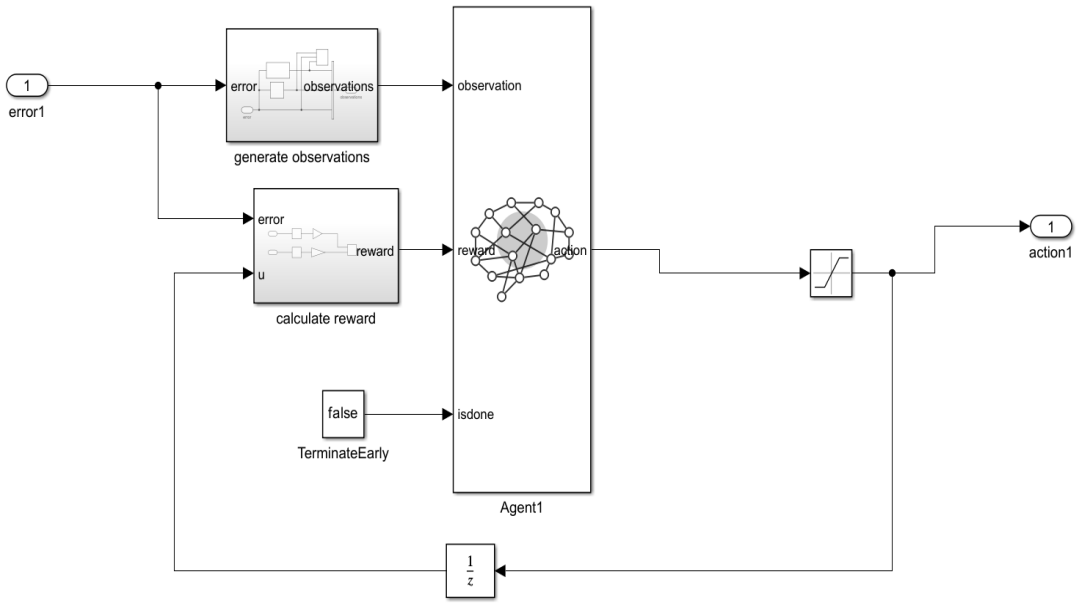

Fig. 9 The internal structure of the RL Agent

TD3

A traditional RL framework comprises five fundamental elements: the agent, environment, reward, states, and actions [10]. The primary goal of the RL is to maximize the expected cumulative reward, as shown in Eq. (3.5).

$$\arg\max E[R_t = \sum_{m=0}^{\infty} \gamma^m r_{t+m}] \tag{13}$$

Where γ is the discount factor: $\gamma \epsilon [0, 1]$, R is the accumulated reward, r is the instant reward and t is the time instant.

As demonstrated by Eq. (3.5), the agent or controller in the RL constantly learns how to interact with the plant or environment in order to maximise the expected cumulative reward [11]. At every time step t the environments and agents interact as follows: The agent selects an action a_t from a state space s and an action space A based on its observation of a representation of the state s_t of the environment. The agent and environment interact when it finds the new state, s_{t+1}, and receives a reward r_t, from the environment at $t + 1$ instant.

The most important advancement in reinforcement learning is Q-learning, which offers the best course of action throughout iterations [12]. The Q-learning method, given a policy TT (TT: $S \rightarrow A$), solves the RL problem for a finite set of states and actions, and, after executing an action at and a state s_t, yields the predicted maximum return Q_π (s_t, a_t), as demonstrated in Eq. (13).

$$Q_\pi(S_t, a_t) = E_\pi[\sum_{m=0}^{\infty} \gamma^m r_{t+m} | S_t = s, a_t = a] \tag{14}$$

The primary goal of Q-learning is to obtain the optimum policy π^* by identifying the optimum Q-value, which is the highest expected return that any policy can achieve for a state-action combination and is shown in Eq. (14) [13].

$$Q_\pi^*(S_t, a_t) = E_\pi[r_t + \gamma_\omega] \tag{15}$$

$\omega \epsilon (Q_\pi^*)$: $\omega = \max Q_\pi^*(S_{t+1} + a_{t+1} | S_t = s, a_t = a)$

For each state-action pair, the Q-learning algorithm iteratively updates the Q-value till the Q-function reaches optimal or sub-optimal Q-value on completion of maximum iterations and as shown in Eq. (15).
$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\xi$

$$\xi = [r_t + \gamma max Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{16}$$

Q-learning is restricted to discrete state and action space, whereas chemical process control has continuous state-action space. The COD problem is addressed by deep Q-learning, a hybrid of Q learning and deep neural networks [13], which uses neural networks to approximate value function and policy. The states are utilised as input to the value network, which returns the Q-value corresponding to every action that is attainable in the action space. The deep Q networks train the value network by choosing appropriate mini-batches from the replay buffer [14]. The loss is determined using the mean square error (MSE) between the goal and current Q-values, as stated in Eq. (16).

$$Loss = E[(Q_{\pi}^{*}(s_t, a_t) - Q(s_t + a_t))^2] \tag{17}$$

An alternative approach to solving the RL problem is the policy gradient (PG) method, which determines the optimal policy by analysing the policy gradients rather than the value functions [15]. The PG method solves for continuous stochastic situations using the gradient of the objective function with respect to the policy parameters θ, as seen in Eq. (17) [16,17]. However, noisy gradients are produced by the large variance in PG techniques, which slows convergence.

$$\nabla_{\theta}(J(\pi_{\theta})) = \nabla_{\theta}(E[R(\tau])$$

And

$$\nabla_{\theta}(E[R(\tau]) = \left[\left(\sum_{t=0}^{T} \nabla_{\theta} \log \pi_{\theta}(a_t|s_t)\right) R(\tau)\right]$$

Where π_{θ} is the policy, R (τ) is the return and θ is the parameter.

Since the actor-critic networks lower the variance gradient estimates in actor-critic techniques, they have gained popularity as a framework for working with continuous action spaces [18,19]. The deterministic policy gradient (DPG) is an actor-critic off-policy method for continuous action spaces that was first presented [20,21]. It uses Eq. (18) to assess the gradient of the objective function with respect to parameters θ in order to determine the optimal strategy [22,23].

$$\nabla_{\theta}(J(\mu_{\theta})) = E\left[\nabla_{\theta} Q_u(s, a)|_{a=\mu\theta(s)}\right]) \tag{19}$$

Where μ is the deterministic policy 2015 saw the release of the DDPG, which combined the DPG with deep Q networks [24,25]. The functional approximation error in DDPG overestimates value estimations, which leads to policy stability problems and the possibility of local optima convergence. An improved version of the DDPG algorithm, called TD3, is suggested as a countermeasure [26]. The TD3 algorithm uses the following three key features to address the shortcomings of the DDPG: Prevent overestimating the target Q-value: The clipped double Q learning computes the target Q value using two main target networks and two critic networks in order to prevent overestimating the target Q-value. Next, the minimum of these two networks is computed to determine the loss [27-29]. Updates to policies are delayed: In TD3, the critic network is updated at each stage of the show, while the actor network is updated and delayed. and policy networks, respectively [30-33]. The roles and updating guidelines for every network are as follows:

Actor-network: The purpose of this network is to search the environment for s and r, and then update the actor parameters and select the current action based on that state. By using Polyak averaging, these network parameters are updated. Actor target-network: is in charge of selecting the subsequent action (a) based on the experience replay buffer's sampled next state (s) [34-36]. The actor-network parameters$\emptyset c1$, $\emptyset c2$ are regularly replicated into the actor target network parameters $\emptyset c1$.Tar,

Critic network: This network is in charge of calculating each critic network's current Q value and changing the Q network settings. The Polyak averaging approach is used to update these network properties. Critic Target network: This network is in charge of figuring out the Target Q-value [37-39]. Every episode, the parameters for the critic target network are changed and replicated from the critic network [40-41].

PPO

A model-free, ϕnline, on-policy, policy gradient reinforcement learning technique is called proximal policy optimisation (PPO). This approach is a kind of policy gradient training that alternates between utilising stochastic gradient descent to optimise a clipped surrogate objective function and sampling data through environmental inter- action [42,43]. Through the limitation of the amount of the policy change at each step, the clipped surrogate objective function enhances training stability [44,45]. PPO is a condensed form of TRPO. Although TRPO requires more computing power than PPO, in low-dimensional observations with deterministic environment dynamics, TRPO typically outperforms PPO in terms of robustness [46,47]. Discrete or continuous action spaces and discrete or continuous observation spaces can be used to train PPO agents. In training, one of the PPO agents: Estimates probabilities of taking each action in the action space and randomly selects actions based on the probability distribution [48]. Interacts with the environment for multiple steps using the current policy before using mini-batches to update the actor and critic properties over multiple epochs. A PPO agent keeps track of two function approximators in order to estimate the policy and value function. Performer $\pi(A\ S; \theta)$: Using parameters θ, the actor produces one of the following as the conditional probability of doing each action A when in state S: Discrete action space: The probability of taking each discrete action. The sum of these probabilities across all actions is 1.

Continuous action space: The mean and standard deviation of the Gaussian probability distribution for each continuous action.

Critic $V(S; \phi)$: Given observation S, the critic yields the equivalent expectation of the discounted long-term reward, given parameters u.

Goal of smoothing policy: Different target Q values are produced for the same action by the DDPG method. Consequently, for the same action, the variance of the goal value is high [49,50]. Adding noise to the desired action helps lower the variance. The TD3 algorithm, as shown in Fig 3.5 is constituted of one actor-network, one actor target network, two critic networks, and two critic target networks. From Fig 3.3 and Fig 3.4, the TD3 algorithm, the state (s), action (a), reward (r), and next state (s) are linked together. By linking the state (s), action (a), reward (r), and next state (s') together, TD3 can learn from experience and improve the policy over time [51,52]. The agent can store and learn from a wide range of transitions thanks to the replay buffer, and the deterministic policy gradient and mistakes are used to update the Q-function.

3. Results and discussions

This section discusses simulation results for the control of the TTL system. The TTL system is controlled in the decentralized configuration as shown in Fig.3.26, and the agent's hyperparameters to control the TTL system are tabulated in Table 4.1, and the optimizer options for the actor and critic networks for both agents are tabulated in Table 4.2. The performance of the TD3 agent, PPO agent and loop tune tuned PID controller are compared.

Table 2. Hyperparameters used to train the agents

Hyperparameter	Value
Actor learning rate	1e-3
Author	1e-3
Batch size	256
Iteration	1000
Algorithm	ADAM
Device	CPU

TD3 agent average training time 17 hrs. PPO agent average training time 3 hrs.

All the training processes are performed with a personal computer (PC) (AMD RYZEN 5 5500U processor and 8 Gb of RAM). The MATLAB Simulink environment, whose schematic is displayed in Fig. 5, is used for conducting the tests. The environment's sampling interval is set at 0.1 s. The TTL

model is used as a state space model for training, with the model equations found in Eq (2.4). The initial weights and hyperparameters for both TD3 and PPO agents were the same, and the reward function Reward was always employed. Table 4.3 lists the state parameters that were applied in each scenario.

Table 3. Initial parameters of the system

Parameter	Value
Initial liquid flowrates Q1	125.7 m3/s
Initial liquid flowrates Q2	120.7 m3/s
Initial height h1 (Tank 1)	4 m
Initial height h2 (Tank 2)	1 m
Initial height h3 (Tank 3)	3 m
Desired tank1 height (Href 1)	5 m
Desired tank2 height (Href 2)	2 m
Desired tank3 height (Href 3)	4 m

Neural networks for the actor and critic make up the agent. Fig. 4.1 displays the structures of the actor and critic. The actor is made up of an output layer (1) that provides the PI parameters for each agent, an input layer (2), and a fully linked hidden layer 64. In a similar vein, the critic network consists of an output layer, a ReLU layer, a fully connected layer (64), a concatenation layer that combines state and actions, and the state (2) and action (1) layers at the top of the neural network design.

Comparison of results

For a more thorough investigation of the system's performance, we applied different weights to the reward function Eq 20. This allowed us to better observe and compare the performance of the two algorithms.

This reward equation was split into 2, Reward1 with lower penalty weights and Reward2 with higher penalty weights.

$$\text{Reward1} = -(\text{Href} - h(t))^2 + 0.001u^2(t) \tag{20}$$

$$\text{Reward2} = -10(\text{Href} - h(t)) + 1u(t) \tag{21}$$

Then the error value is computed as fellow:

$e = -h$, h is the height of the tank, and *Href* is the reference height.

Reward1

Below are the compared results of the performance of the TD3 decentralised agents and the tuned PID controller. The PPO agent failed to give good results under the same conditions

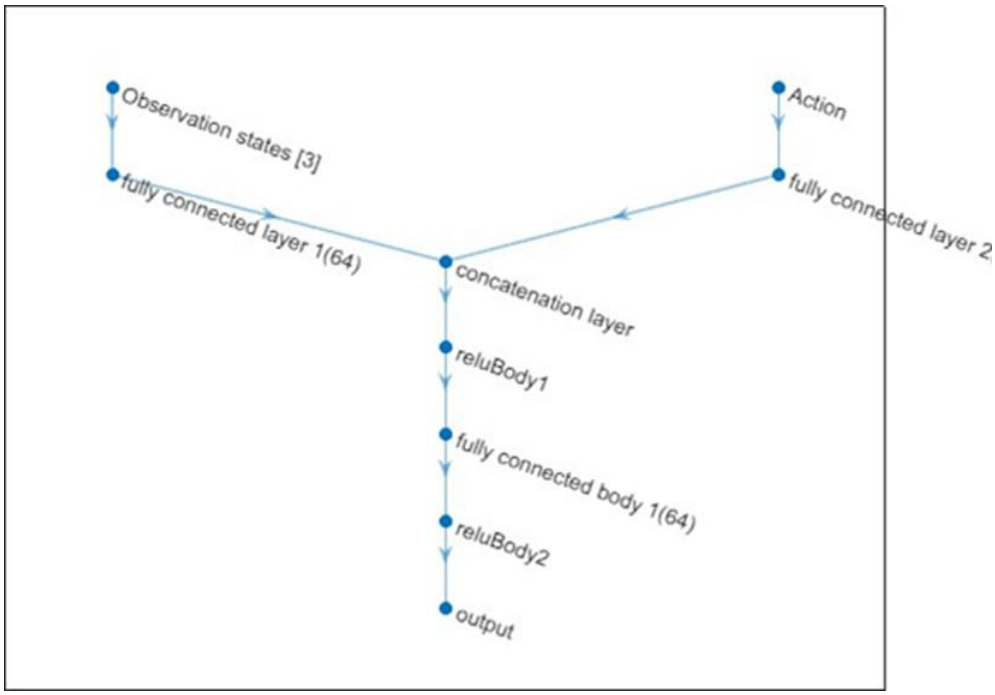

Fig. 6. Neural network architecture

Based on the Table 4.5, the TD3 algorithm performs well in terms of settling time, robustness, and rise time. Although the agents did produce overshoots, they are within an acceptable range. In contrast, the PPO agent did not produce any MATLAB tools obtainable results using the same strategy.

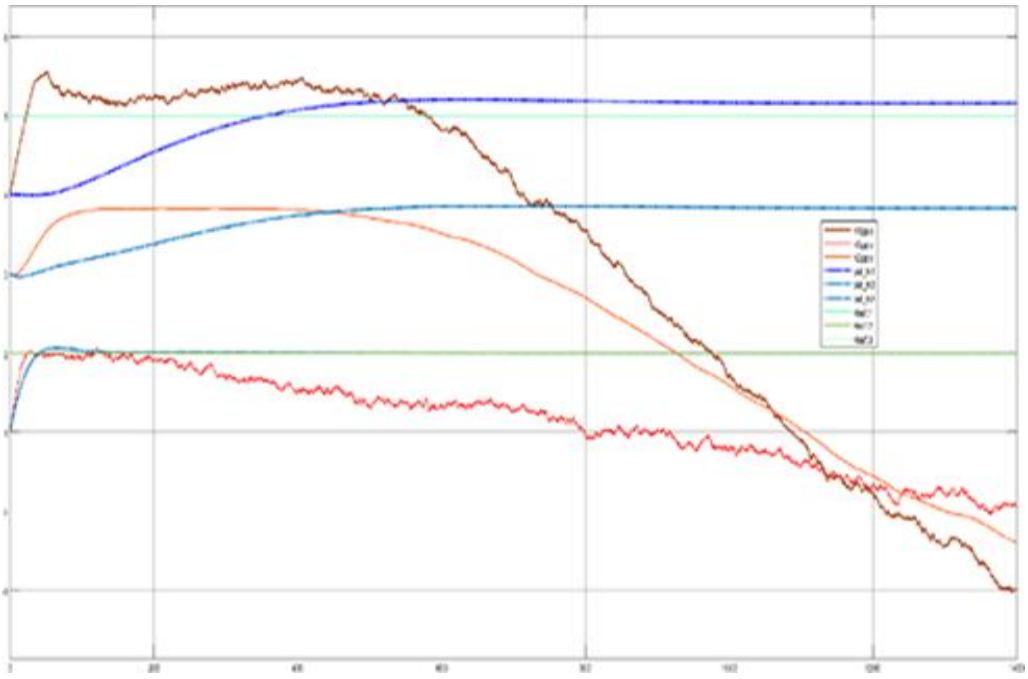

Fig. 7 PPO agent performance on Reward 1.

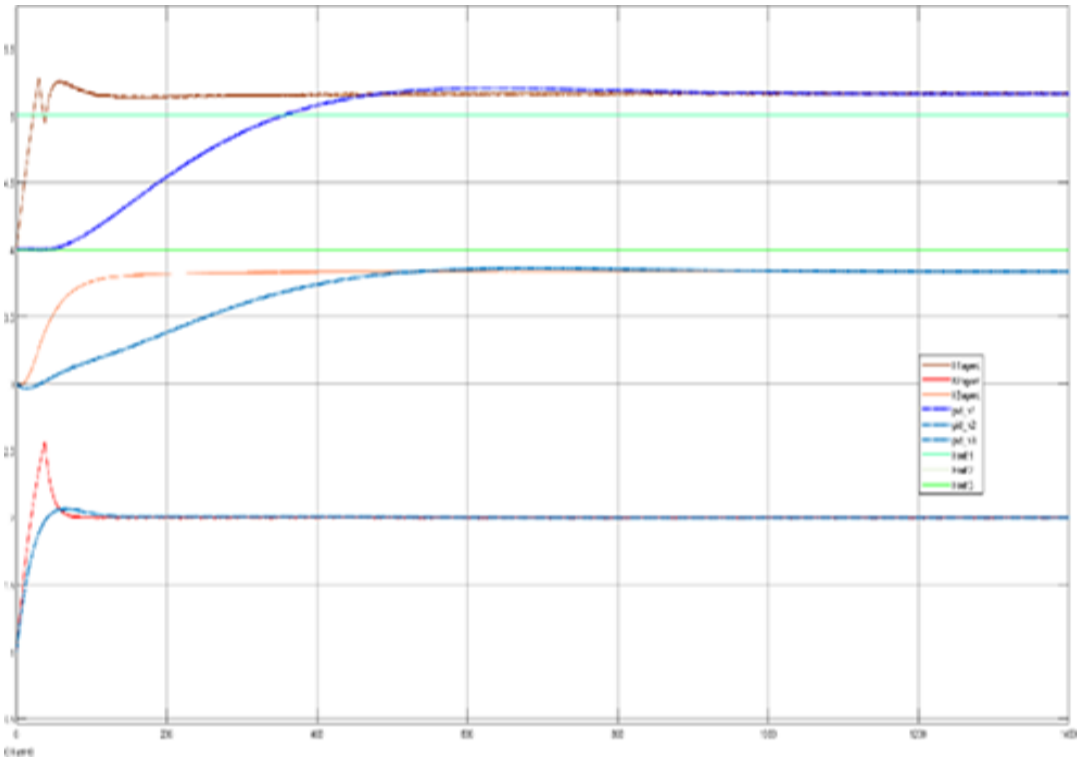

Fig. 8 TD3 agent performance on Reward 1.

Table 4. Comparison of results for Reward 1

	Tank 1			**Tank 2**			**Tank 3**		
Type of controller	TD3	PPO	PID	TD3	PPO	PID	TD3	PPO	PID
Overshoot (%)	10.5	n/a	3.6	0.1	n/a	0.3	58	n/a	7
Rise time (s)	21.1	n/a	286.1	76.2	n/a	298.4	15.3	n/a	28.2
Settling time (s)	190	n/a	1190	396	n/a	1200	110	n/a	701
Static error	Yes	n/a	yes	Yes	n/a	yes	Non	non	non

Reward 2

Based on the results in the Table 5 below, the TD3 agents outperformed both the PPO agents and the PID controller. As shown in the figure, an overshoot occurred only in tank 1. The PPO agents produced a relatively good response but failed to reach a steady state due to continuous, acceptable oscillations. Although the PPO agents responded faster than the PID controller, the PID provided more stable and reliable results between the two. The TD3 agents proved to be the best controllers in this research using the Reward2 strategy. It's also important to note that static errors appeared in both tank 1 and tank 2, likely due to uncertainties in modelling such a highly non-linear MIMO system.

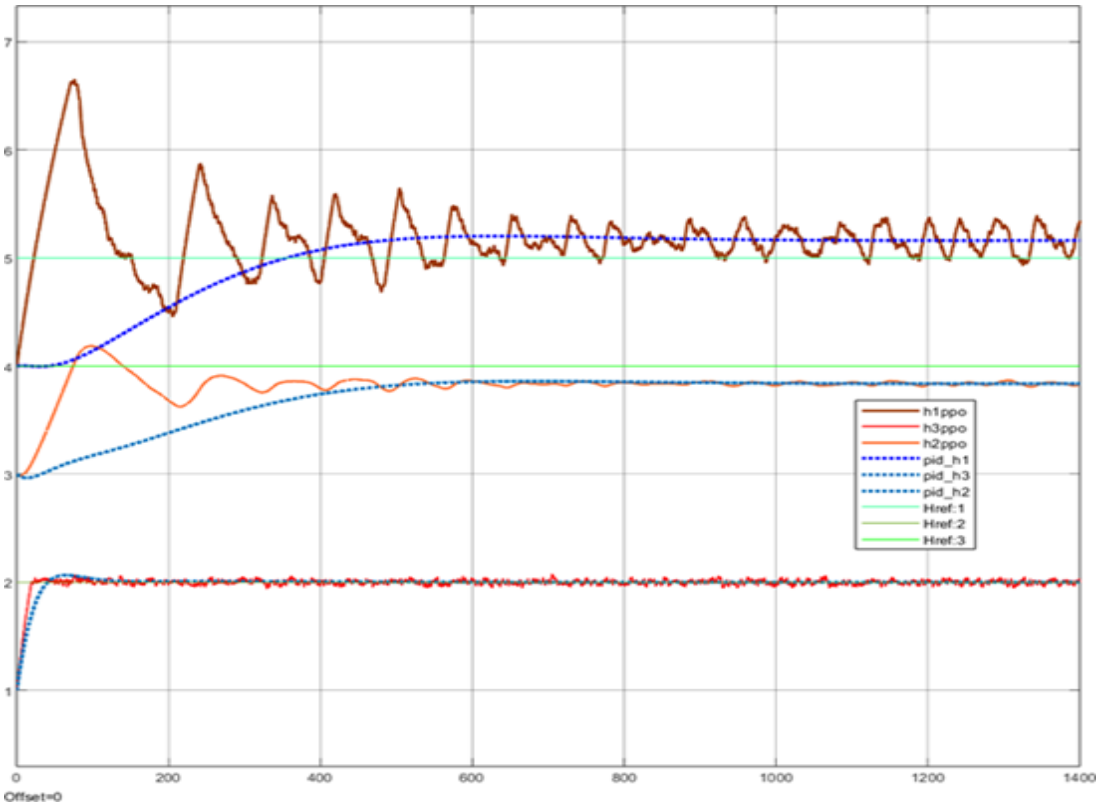

Fig. 10 PPO and PID performance on Reward 2

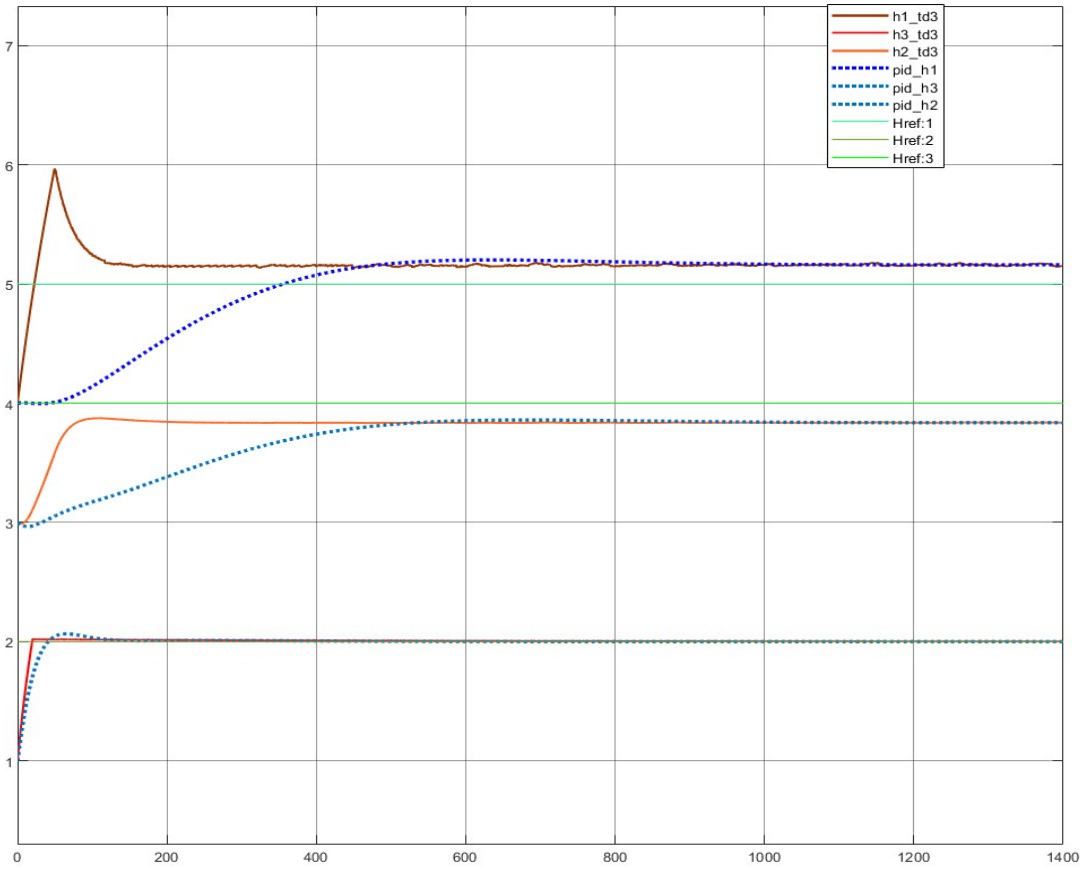

Fig. 10 TD3 and PID performance on Reward 2

Table 5. Comparison of results for Reward 2

	Tank 1			Tank 2			Tank 3		
Type of controller	TD3	PPO	PID	TD3	PPO	PID	TD3	PPO	PID
Overshoot (%)	94.3	86.2	3.6	4.7	38.2	0.3	1.5	9.5	7
Rise time (s)	16.3	3.3	286.1	45.77	42.2	298.4	15.3	9.7	28.2
Settling time (s)	190	n/a	1190	396	n/a	1200	110	n/a	701
Static error	Yes	yes	yes	Yes	yes	yes	Non	non	non

4. Conclusions

In conclusion this study demonstrated the superiority of Twin-Delayed Deep Deterministic Policy Gradient agents in controlling the nonlinear dynamics of a three-tank liquid system, outperforming both Proximal Policy Optimization agents and traditional Proportional-Integral-Derivative controllers. While the PID con- troller provided more stable and reliable results, the TD3 agents achieved faster response times and better overall performance with the Reward2 strategy, despite a slight overshoot in tank 1. The PPO agents, although faster than PID, failed to achieve a steady state due to continuous oscillations. Finally, this study opens the door to explore the performance of newer or less commonly used reinforcement learning algorithms, such as Soft Actor-Critic (SAC), to see if they provide better stability or faster convergence.

Author Contributions

All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by LMN, AK and GMG.

Conflict of interest

The authors declare no conflicts of interest.

References

[1] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347. 2017 Jul 20

[2] Fujimoto S, Hoof H, Meger D. Addressing function approximation error in actor-critic methods. International conference on machine learning 2018 Jul 3 (pp. 1587-1596). PMLR.

[3] Gupta JK, Egorov M, Kochenderfer M. Cooperative multi-agent control using deep reinforcement learning. International conference on autonomous agents and multiagent systems 2017 May 8 (pp. 66-83). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-71682-4_5

[4] Suzuki A, Kawahara R, Harada S. Cooperative multi-agent deep reinforcement learning for dynamic virtual network allocation with traffic fluctuations. IEEE Transactions on Network and Service Management. 2022 Feb 7;19(3):1982-2000. https://doi.org/10.1109/TNSM.2022.3149243

[5] Yun WJ, Park S, Kim J, Shin M, Jung S, Mohaisen DA, Kim JH. Cooperative multiagent deep reinforcement learning for reliable surveillance via autonomous multi-UAV control. IEEE Transactions on Industrial Informatics. 2022 Jan 14;18(10):7086-96. https://doi.org/10.1109/TII.2022.3143175

[6] Dittrich MA, Fohlmeister S. Cooperative multi-agent system for production control using reinforcement learning. CIRP Annals. 2020 Jan 1;69(1):389-92. https://doi.org/10.1016/j.cirp.2020.04.005

[7] Nguyen ND, Nguyen T, Nahavandi S. Multi-agent behavioral control system using deep reinforcement learning. Neurocomputing. 2019 Sep 24;359:58-68. https://doi.org/10.1016/j.neucom.2019.05.062

[8] Zhang Y, Quinones-Grueiro M, Barbour W, Zhang Z, Scherer J, Biswas G, Work D. Cooperative multi-agent reinforcement learning for large scale variable speed limit control. In2023 IEEE International Conference on Smart Computing (SMARTCOMP) 2023 Jun 26 (pp. 149-156). IEEE. https://doi.org/10.1109/SMARTCOMP58114.2023.00036

[9] Rivera C, Staley E, Llorens A. Learning multi-agent cooperation. Frontiers in Neurorobotics. 2022 Oct 14;16:932671. https://doi.org/10.3389/fnbot.2022.932671

[10] Lillicrap TP, Hunt JJ, Pritzel A, Heess NM, Erez T, Tassa Y, Silver D, Wierstra DP. Continuous control with deep reinforcement learning. Google Patents. US Patent. 2020;10.

[11] Lowe R, Wu YI, Tamar A, Harb J, Pieter Abbeel O, Mordatch I. Multi-agent actor-critic for mixed cooperative-competitive environments. Advances in neural information processing systems. 2017;30.

[12] Amato C. An introduction to centralized training for decentralized execution in cooperative multi-agent reinforcement learning. arXiv preprint arXiv:2409.03052. 2024 Sep 4.

[13] Hernandez-Leal P, Kartal B, Taylor ME. A survey and critique of multiagent deep reinforcement learning. Autonomous Agents and Multi-Agent Systems. 2019 Nov;33(6):750-97. https://doi.org/10.1007/s10458-019-09421-1

[14] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347. 2017 Jul 20.

[15] Fujimoto S, Hoof H, Meger D. Addressing function approximation error in actor-critic methods. InInternational conference on machine learning 2018 Jul 3 (pp. 1587-1596). PMLR.

[16] Gupta JK, Egorov M, Kochenderfer M. Cooperative multi-agent control using deep reinforcement learning. InInternational conference on autonomous agents and multiagent systems 2017 May 8 (pp. 66-83). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-71682-4_5

[17] Suzuki A, Kawahara R, Harada S. Cooperative multi-agent deep reinforcement learning for dynamic virtual network allocation with traffic fluctuations. IEEE Transactions on Network and Service Management. 2022 Feb 7;19(3):1982-2000. https://doi.org/10.1109/TNSM.2022.3149243

[18] Yun WJ, Park S, Kim J, Shin M, Jung S, Mohaisen DA, Kim JH. Cooperative multiagent deep reinforcement learning for reliable surveillance via autonomous multi-UAV control. IEEE Transactions on Industrial Informatics. 2022 Jan 14;18(10):7086-96. https://doi.org/10.1109/TII.2022.3143175

[19] Dittrich MA, Fohlmeister S. Cooperative multi-agent system for production control using reinforcement learning. CIRP Annals. 2020 Jan 1;69(1):389-92. https://doi.org/10.1016/j.cirp.2020.04.005

[20] Nguyen ND, Nguyen T, Nahavandi S. Multi-agent behavioral control system using deep reinforcement learning. Neurocomputing. 2019 Sep 24;359:58-68. https://doi.org/10.1016/j.neucom.2019.05.062

[21] Zhang Y, Quinones-Grueiro M, Barbour W, Zhang Z, Scherer J, Biswas G, Work D. Cooperative multi-agent reinforcement learning for large scale variable speed limit control. In2023 IEEE International Conference on Smart Computing (SMARTCOMP) 2023 Jun 26 (pp. 149-156). IEEE. https://doi.org/10.1109/SMARTCOMP58114.2023.00036

[22] Lillicrap TP, Hunt JJ, Pritzel A, Heess NM, Erez T, Tassa Y, Silver D, Wierstra DP, inventors; DeepMind Technologies Ltd, assignee. Continuous control with deep reinforcement learning. United States patent US 10,776,692. 2020 Sep 15.

[23] Lowe R, Wu YI, Tamar A, Harb J, Pieter Abbeel O, Mordatch I. Multi-agent actor-critic for mixed cooperative-competitive environments. Advances in neural information processing systems. 2017;30.

[24] Zhang K, Yang Z, Başar T. Multi-agent reinforcement learning: A selective overview of theories and algorithms. Handbook of reinforcement learning and control. 2021 Jun 24:321-84. https://doi.org/10.1007/978-3-030-60990-0_12

[25] Amato C. An introduction to centralized training for decentralized execution in cooperative multi-agent reinforcement learning. arXiv preprint arXiv:2409.03052. 2024 Sep 4.

[26] Rajasekhar N, Radhakrishnan, T K, Samsudeen N, Decentralized multi-agent control of a three-tank hybrid system based on twin delayed deep deterministic policy gradient reinforcement learning algorithm. Int. J. Dynam. Control 12, 1098-1115 (2024). https://doi.org/10.1007/s40435-023-01227-0

[27] Zhang Y, Zhao Y, Zhang C, Feng C, Multi-agent reinforcement learning-based method for demand response of building HVAC systems, Journal of Building Engineering, Volume 108, 2025, 112734, ISSN 2352-7102. https://doi.org/10.1016/j.jobe.2025.112734

[28] Sheikh A, Machine Learning for Optimization and Decision Support in Complex System-of-Systems: Applications in Microelectronics, Healthcare, Smart Cities, and the Circular Economy, Colorado State University ProQuest Dissertations & Theses, 2025. 32236453.

[29] Mallik S, Mathivanan SK, Shivahare BD, S.K.B, S., Jayagopal, P., & Chakraborty, S. (Eds.). (2026). Robotics in Weaponry using Machine Learning and Engineering (1st ed.). CRC Press. https://doi.org/10.1201/9781003663461

[30] Abdel A W, (2009). Pathophysiology. In: Passing the USMLE. Springer, New York, NY. https://doi.org/10.1007/978-0-387-68980-7_10

[31] Hoseini S A, Hassan J, Bokani A, Kanhere SS. In Situ MIMO-WPT Recharging of UAVs Using Intelligent Flying Energy Sources. Drones. 2021; 5(3):89. https://doi.org/10.3390/drones5030089

[32] Xiaoqiang C, Haobo Y, Jianjun M et al. Research on multi-section energy-saving operation control methods of urban rail train based on deep reinforcement learning, 18 October 2024, PREPRINT (Version 1) available at Research Square
[33] Cheng Z. Towards Trustworthy Deep Reinforcement Learning, Northwestern University ProQuest Dissertations & Theses, 2025. 32122366.
[34] Lu Q, Fang H, Yin Z, Zhu G. HAPS-PPO: A Multi-Agent Reinforcement Learning Architecture for Coordinated Regional Control of Traffic Signals in Heterogeneous Road Networks. Applied Sciences. 2025; 15(20):10945. https://doi.org/10.3390/app152010945
[35] El-Nabulsi AR, Non-Linear Dynamics with Non-Standard Lagrangians. Qual. Theory Dyn. Syst. 12, 273-291 (2013). https://doi.org/10.1007/s12346-012-0074-0
[36] Ma T, Xu C, Yang S, et al. An intelligent proactive defense against the client-side DNS cache poisoning attack via self-checking deep reinforcement learning. Int J Intell Syst. 2022;37:8170-8197. https://doi.org/10.1002/int.22934
[37] Prabu A, Integrating Data-Driven Control Methods with Motion Planning: A Deep Reinforcement Learning-Based Approach, Purdue University ProQuest Dissertations & Theses, 2023. 32363572.
[38] Wu M, Eldele E, Chen Z, Pan S, Wen Q, & Li, X. (Eds.). (2026). AI for Time Series: Volume 1: Unlocking Patterns with Deep Learning (1st ed.). CRC Press. https://doi.org/10.1201/9781003612742
[39] Misbaudeen AA, Hammed O, Anis R, Wook-Ho N, Qazeem OO, Timothy Denen Akpenpuun, Min-Hwi Kim, Hyeon-Tae Kim, Bo-Yeong Kang, Hyun-Woo Lee, Deep reinforcement learning for PID parameter tuning in greenhouse HVAC system energy Optimization: A TRNSYS-Python cosimulation approach, Expert Systems with Applications, Volume 252, Part A, 2024, 124126, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2024.124126
[40] Tao L, Zhang J and Zhang, X. Multi-Phase Multi-Objective Dexterous Manipulation with Adaptive Hierarchical Curriculum. J Intell Robot Syst 106, 1 (2022). https://doi.org/10.1007/s10846-022-01680-7
[41] Le BG, Ta VC, Low variance trust region optimization with independent actors and sequential updates in cooperative multi-agent reinforcement learning. Auton Agent Multi-Agent Syst 39, 12 (2025). https://doi.org/10.1007/s10458-025-09695-8
[42] Abdel AW (2009). Pathophysiology. In: Passing the USMLE. Springer, New York, NY. https://doi.org/10.1007/978-0-387-68980-7_10
[43] Hoseini SA, Hassan J, Bokani A, Kanhere SS. In Situ MIMO-WPT Recharging of UAVs Using Intelligent Flying Energy Sources. Drones. 2021; 5(3):89. https://doi.org/10.3390/drones5030089
[44] Bujgoi G, Sendrescu D. Tuning of PID Controllers Using Reinforcement Learning for Nonlinear System Control. Processes. 2025; 13(3):735. https://doi.org/10.3390/pr13030735
[45] Mahapatra AGKRS, and Mahapatro SR, Design of a decentralized control law for variable area coupled tank systems using H∞ complimentary sensitivity function, Asian J. Control. 26 (2024), 1540-1552. https://doi.org/10.1002/asjc.3281
[46] Sawant HH, Gujar R, Mandhare N, Sable MJ, Ambadekar, P.K. and Gawande, S.H. (2025), Comparative Analysis of Reinforcement Learning Agents for Optimizing Airfoil Shapes. Int J Numer Meth Fluids, 97: 1142-1156. https://doi.org/10.1002/fld.5395
[47] Rao A & Jelvis T, (2022). Foundations of Reinforcement Learning with Applications in Finance (1st ed.). Chapman and Hall/CRC. https://doi.org/10.1201/9781003229193-1
[48] Roberto CR, Orbital Maneuvers and Interplanetary Trajectory Design via Reinforcement Learning, Embry-Riddle Aeronautical University ProQuest Dissertations & Theses, 2025. 32237533.
[49] Ho TM, Nguyen KK and Cheriet M, "Converging Game Theory and Reinforcement Learning For Industrial Internet of Things," in IEEE Transactions on Network and Service Management, vol. 20, no. 2, pp. 890-903, June 2023, https://doi.org/10.1109/TNSM.2022.3202168
[50] Gautam M, Deep Reinforcement Learning for Resilient Power and Energy Systems: Progress, Prospects, and Future Avenues. Electricity. 2023; 4(4):336-380. https://doi.org/10.3390/electricity4040020
[51] Kim N, Ecological and Predictive Indicators of Social and Emotional Skills among Korean Adolescents: A Person-Centered Analysis within the OECD SSES Framework. Child Ind Res (2026). https://doi.org/10.1007/s12187-026-10355-w
[52] Dodda, A. (2025). Artificial Intelligence and Financial Transformation: Unlocking the Power of Fintech, Predictive Analytics, and Public Governance in the Next Era of Economic Intelligence. Deep Science Publishing. Chapter 5, 1-20. https://doi.org/10.70593/978-81-988918-1-5