2024 Tenth Indian Control Conference (ICC)
9 - 11 December, 2024
IISER Bhopal, India

# Reinforcement Learning-Based Control of Multivariable Processes with Time Delays

Dinesh Patil*, Tanuja Joshi*, Vivek Pinnamaraju*, Nandkishor Kubal*, Sudhakar Munusamy*

*Abstract*— **The dynamics of industrial processes are often characterized by significant and varied time delays. Direct application of conventional reinforcement learning architectures using feed-forward neural networks is inadequate, often resulting in deteriorated control performance. In this work, we embed a recurrent neural network (RNN) architecture in the reinforcement learning framework for the actor and critic networks to effectively handle multivariate process dynamics with large and varied delays. This approach enables the resulting trained control policy to understand that current actions affect the output(s) with certain delay(s). The efficacy of the proposed approach has been tested on simulation examples comprising single-input single-output (SISO) and multi-input multi-output (MIMO) systems with delays for set-point tracking and disturbance rejection scenarios. The results demonstrate the efficacy of the approach in effectively managing multivariate systems, even in the presence of inherent delays.**

***Keywords:*** Reinforcement learning, Time Delays, Multivariate, Process Control, Disturbance Rejection

## I. INTRODUCTION

Reinforcement Learning (RL) has emerged as a revolutionary method in artificial intelligence, offering robust solutions to complex decision-making problems across various fields [1]. The application of RL has achieved significant success in gaming [2]–[4], robotics [5]–[7], and autonomous driving [8]. Recent advancements, particularly in continuous action spaces [9], [10], have unlocked new possibilities for industrial process control applications, enabling finer control and more nuanced responses crucial in dynamic environments [11]. Algorithms like Deep Deterministic Policy Gradient (DDPG) [9], [12] and Proximal Policy Optimization (PPO) [13] have paved the way for RL to effectively tackle challenges in continuous-time domains.

Most literature on RL relevant to process control assumes the underlying process adheres to the Markovian property, where a future state depends solely on the current state and current action only. These methods have demonstrated promising results in simple Single Input Single Output (SISO) systems and small-scale Multiple Input Multiple Output (MIMO) systems [11], [14] involving minimal to no delays. However, many industrial processes are characterized by long and variable time delays in input-output relationships, caused by transportation delays or inherent process sluggishness.

Such processes do not strictly conform to the Markovian assumption, as future states depend on past states up to a certain delay. This implies that actions guided by the trained RL agent at the current time interact with the process or environment and affect state/output measurements only after a delay. Therefore, it is essential to use a neural network model structure that can efficiently learn a control policy capable of handling these delays.

The conventional reinforcement learning techniques, assuming the Markovian property and utilizing feed forward neural networks for training the control policy, lack memory capabilities due to the absence of feedback in the model structure. Additionally, they fail to include sufficient past state or measurement information as additional input features to the network. As a result, these techniques are inadequate in handling delays and can lead to degraded control performance and instability, especially with significant delays.

To address these challenges, this work proposes an RL-based control method tailored specifically for multi-variable systems with inherent delays. Our approach integrates the state-of-the-art TD3 algorithm [15] with a recurrent neural network architecture. This integration enables the RL agent to learn a control policy that effectively maps system states to appropriate control actions over time, accommodating process delays.

Furthermore, to enhance the RL agent's performance, we augment the state representation by incorporating error and integral error as additional measurement or state variables. This inclusion helps in eliminating offsets, a common issue in process control. Additionally, we introduce action smoothness [16] as an additional regularization term in the loss function. This term encourages the RL agent to generate smooth action policies, thereby reducing fluctuations and ensuring stable system operation. Overall, the proposed approach efficiently handles delays, ensures smoothness in agent actions, and eliminates offsets in system responses.

The paper is structured as follows: Section II provides necessary preliminaries, including an introduction to reinforcement learning concepts and recurrent neural networks. Section III explains the proposed RL-based control approach for time-delay processes and the associated problem formulation. Section IV presents simulation case studies to demonstrate the efficacy of the proposed approach. Finally, Section V concludes the paper with final remarks.

## II. BACKGROUND

### A. Reinforcement Learning

Reinforcement Learning is a type of machine learning where feedback from the environment is given in the form of rewards based on the desired behavior. The RL framework

* All authors are affiliated with ABB, India Corporate Research Center, Bangalore [dinesh.patil, tanuja.joshi, vivek.pinnamaraju, nandkishor.kubal, sudhakar.munusamy]@in.abb.com

includes three main components: an agent, an environment, and a reward function. Typically, an RL algorithm can be used to solve problems modeled as Markov Decision Processes (MDPs) [1]. The agent interacts with the environment in the sequence of discrete-time steps $t = 0, 1, 2, 3, \ldots$. The RL agent selects an action $a_t$ upon receiving the state $s_t$ from the environment, which results in a reward $r_t := r(s_t, a_t)$. The environment then transitions to the next state $s_{t+1}$ based on the transition probability $p(s', r|s, a)$, forming a sequence of tuples $(s_t, a_t, r_t, s_{t+1})$ that represent the state, action, reward, and next state. This function $p(s', r|s, a) = \Pr\{s_{t+1} = s', r_{t+1} = r|s_t = s, a_t = a\}$ defines the dynamics of the MDP. Owing to the assumed Markov property the subsequent state $s_{t+1}$ depends exclusively on the current state $s_t$ and the action applied (1). In other words, the Markov property indicates that the system's evolution is determined only by its latest state and the action taken:

$$\Pr[s_{t+1}|s_t, a_t] = \Pr[s_{t+1}|a_0, s_0, ..., a_t, s_t] \tag{1}$$

In this context, policy is a function that maps a state, $s$ to an action $a$. For a deterministic policy, this mapping is direct, expressed as $a = \pi_{\phi_A}(s)$. For a stochastic policy, the mapping involves a distribution of actions, represented as $a \sim \pi_{\phi_A}(\cdot|s)$ [17]. The objective in RL is to find an optimal policy, $\pi^*$ that maximizes the expected discounted reward, as described below:

$$\pi^* = \arg\max_{\pi} E\ [G_t := \sum_{k=0}^{\infty} \gamma^k r_{t+k}] \tag{2}$$

Policy Gradient is an RL approach that aims to maximise the expected return by computing the gradient w.r.t the stochastic parameterized policy. Traditionally, Policy Gradient (PG) algorithms have been exclusively used with a stochastic policy function $\pi_{\phi_A}$. However, Silver et al. demonstrated that a deterministic policy can also be utilized in the policy gradient algorithm by providing a deterministic analogue to PG, leading to the idea of Deterministic Policy Gradient (DPG) [18].

In process control applications as the state and action spaces are continuous, we have employed the deterministic Actor-Critic (AC) RL approach to develop the DRL controller. Here, the actor is the agent's policy (which outputs the action), while the critic evaluates this policy by estimating the Q-value, indicating how advantageous it is for the agent to be in a given state-action pair. Both the actor and the critic are usually parameterised functions e.g., function approximators denoted as $\pi_{\phi_A}$ and $Q_{\phi_C}(s, a)$. This allows both functions to operate effectively on a continuous domain. The actor learns by penalising the approximate policy gradient as:

$$\nabla_{\phi_\mathbf{A}}(J(\pi_{\phi_\mathbf{A}})) \approx \underset{a\sim\pi_{\phi_\mathbf{A}}}{E} [\nabla_{\phi_\mathbf{A}} \log \pi_{\phi_\mathbf{A}}(a_t|s_t) Q_{\phi_\mathbf{C}}(s, a)] \tag{3}$$

The actor parameters $\phi_\mathbf{A}$ are updated using the following update equation:

$$\Delta\phi_\mathbf{A} = \beta_{\phi_\mathbf{A}} \nabla_{\phi_\mathbf{A}} \log \pi_{\phi_\mathbf{A}}(a_t|s_t) * Q_{\phi_\mathbf{C}}(s_t, a_t) \tag{4}$$

The critic parameter $\phi_\mathbf{C}$ is updated by minimising the temporal difference (TD) error using the following update equation:

$$\Delta\phi_\mathbf{C} = \beta_{\phi_\mathbf{C}} \big(r(s_t, a_t) + \gamma Q_{\phi_\mathbf{C}}(s_{t+1}, a_{t+1}) - Q_{\phi_\mathbf{C}}(s_t, a_t)\big) * \nabla_{\phi_\mathbf{C}}(Q(s_t, a_t)) \tag{5}$$

where $\delta = \big[r(s_t, a_t) + \gamma Q_{\phi_\mathbf{C}}(s_{t+1}, a_{t+1})\big] - Q_{\phi_\mathbf{C}}(s_t, a_t)$ is the TD error

Fujimoto et al. introduced the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm [15], an advanced actor-critic RL method that addresses limitations found in other algorithms like the overestimation in DDPG. This feature makes TD3 particularly suitable for the current study. In control applications, oscillatory control response is a significant challenge, leading to poor control response if left unattended. Mysore et al. proposed a solution by introducing a smoothness term directly into the actor's loss function, ensuring that states are smoothly mapped to actions:

$$J(\pi_{\phi_\mathbf{A}})_{new} = J(\pi_{\phi_\mathbf{A}}) - \lambda_T L_T - \lambda_S L_S \tag{6}$$

Refer to [16] for further details. Here, $L_T$ (temporal loss) imposes a penalty on policies when actions taken in the next state significantly differ under the state transition probabilities compared to those in the current state. $L_S$ spatial loss ensures that policies behave similarly in similar states, which lowers noise in the system dynamics. The hyperparameters $\lambda_T$ and $\lambda_S$ control the temporal and spatial smoothing factors.

### *B. Recurrent Neural Network (RNN)*

Recurrent Neural Networks excel at recognizing patterns in sequential data like time series and text, where understanding the temporal sequence is critical [19]. They are particularly effective for dynamic systems that require sequencing of inputs and outputs.

In the context of Actor-Critic networks, RNNs play a crucial role in mitigating process delays, which often violate the Markov decision property. Failing to account for the delays lead to the violation of the Markovian property and results in partially observable MDPs (POMDPs). Unlike in MDPs, where the environmental state $s_t$ is fully observable by the agent, in POMDPs, the state $x_t$ is only partially observable by the agent, depicted by the expression $x_t \subset \mathrm{s}_t$, where $x_t$ is not Markovian. RNNs tackle this problem by maintaining a sort of memory, using information from past outputs and actions to reveal hidden states. This ability allows RNNs to piece together the complete state of the system over time, thereby providing capability to handle POMDP introduced by time delays. This facilitates smarter control decisions, boosting the performance of Actor-Critic networks in scenarios involving time delays or partial observability. In our study, we choose the Gated Recurrent Unit (GRU) [20], a type of RNN, due to its superiority over other RNN methods. These include better handling of the vanishing gradient problem, simpler architecture with fewer parameters, and faster training times.

## III. Proposed Methodology

In this section, we discuss the proposed workflow, namely, the Smooth Recurrent Actor-Critic architecture (SRAC) for efficiently handling processes with delays. This comprises of TD3 algorithm utilizing RNN architecture as opposed to the conventional feed-forward neural networks for actor and critic networks, while the smooth actions are ensured by introducing regularization term for the actions in the loss function. The details of the overall workflow are discussed here.

The challenge of delayed observations are handled by incorporating past states $(s_{t-l}...s_{t-1})$, past actions $(a_{t_l}...a_{t-1})$, and two Gated Recurrent Units (GRUs) layers into the actor-critic network. Here, $l$ represents the number of past lags, which provides the system a coherent understanding of the environment over time despite the delays. The proposed SRAC architecture is pictorially illustrated in Figure 1 and the underlying elements are discussed below:

*1) Critic Network:* For the critic network, we merge the past states and actions using a concatenation (CAT) layer. This combined information is then inputted into the first GRU layer, which processes it to generate a hidden state capturing the historical context.

Thereafter, this hidden state, representing the past, combines with the current observation and propagates to the second GRU layer. The output from the final time step of this layer encapsulates both past and current states. Subsequently, we concatenate this state with the current action, enriching the representation that passes through two fully connected (FC) layers. Each layer employs Relu activation functions, enhancing the network's ability to learn intricate underlying patterns. Finally, the output from the last FC layer, using a linear activation function, provides the Q Value: a crucial metric responsible for guiding the agents decisions.

*2) Actor Network:* The actor network mirrors the critic network's structure but the objective is tailored to optimize the action selection. Here, the output from the final time step of the second GRU layer, encompassing past and current states, directly connects to the FC layer, bypassing the combination with the current action found in the critic network. This streamlined connection allows the actor-network to focus solely on generating actions based on the state representation. Additionally, the activation function in the final FC layer of the actor-network shifts from linear to hyperbolic tangent (tanh), ensuring that generated actions adhere to specified limits.

Through integration of recurrent layers, our architecture effectively captures the temporal relationships in delayed feedback, enabling it to provide better decisions. The GRUs efficiently manage sequences of data, allowing the network to maintain an updated internal state that reflects the environment's history.

Furthermore, incorporating a smoothness term directly into the actor's loss function, as shown in Equation 6, significantly reduces undesirable fluctuations in the actions. These collective enhancements, integrated into both network architecture and loss function, constitute the foundation of our proposed novel algorithm, namely, Smooth Recurrent TD3 (SRTD3). This approach effectively handles the complexities introduced by process delays, ensuring the delivery of smooth control policies.

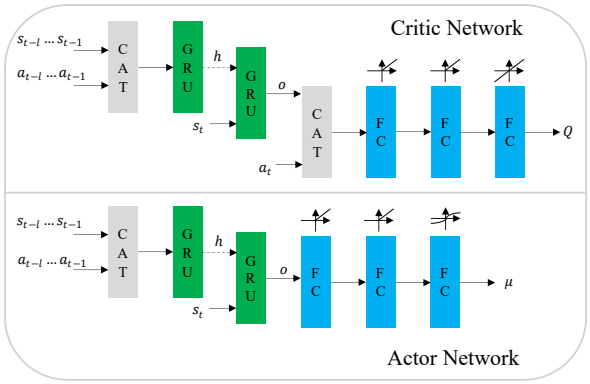



Fig. 1. Proposed Smooth Recurrent Actor-Critic (SRAC) network architecture

## IV. Simulation Results

In this section, we present simulation results demonstrating the effectiveness of our proposed algorithm for controlling industrial processes with delays. The following case studies are considered to demonstrate the efficacy:

1) A single-input single-output (SISO) system with considerable delay.
2) *Woodberry distillation* column benchmark, which is essentially a two-input two-output (TITO) system with reasonable delays.
3) A subsystem of *shell oil fractionator* (SOF) benchmark, representing a fairly complex multi-input multi-output (MIMO) systems with long and varied delays among the input-output relationships.

In the first case study, the efficacy of the proposed algorithm SRTD3 in handling time delays is demonstrated by comparing the closed loop performance with the traditional TD3 actor-critic network featuring fully connected layers. Upon establishing that SRTD3 outperforms TD3 through the first case study, we only present our proposed SRTD3 algorithm performance for the remaining case studies, namely, the *wood berry distillation* column and the *shell oil fractionator*. The hyper-parameters used for training the TD3 and SRTD3 agents for the considered case studies are summarized in Table II. The parameter values reported with an *asterisk* in the superscript specifically correspond to the SOF case study.

Further, all three simulation processes were modelled using MATLAB Simulink, with the models converted into Functional Mock-up Units (FMUs) acting as the process environments during RL agent training. The algorithm was implemented in Python, leveraging the PyTorch package for training deep neural networks. The FMPy library facilitated

TABLE I

SISO SYSTEM RL FORMULATION

| | |
|---|---|
| Observations (3) | Output: $y$<br>Error: $y_{ref} - y$<br>Integral Error: $\sum(y_{ref} - y)t_s$ |
| Actions (1) | u |
| Reward | $\lvert y_{ref} - y \rvert$ |

TABLE II

SRTD3 AND TD3 AGENT PARAMETER DETAILS

| Hyperparameters | SRTD3 | TD3 |
|---|---|---|
| Number of GRU layers | 1 | N/A |
| Number of hidden states | 10, 50* | N/A |
| Number of history samples | 20, 30* | N/A |
| Critic and Actor FC layers size | $[400, 300]$ | $[400, 300]$ |
| Action Noise | 0.1 | 0.1 |
| Target Noise | 0.2 | 0.2 |
| Actor and Critic learning rate | 0.0003 | 0.0003 |
| Replay memory size | 1000000 | 1000000 |
| Reward discount factor | 0.99 | 0.99 |
| Target smoothing factor | 0.005 | 0.005 |
| Batch Size | 100 | 100 |
| Action Range | $[-1, 1]$ | $[-1, 1]$ |
| Temporal Smoothing Factor ($\lambda_T$) | 1 | N/A |
| Spatial Smoothing Factor ($\lambda_S$) | 1 | N/A |

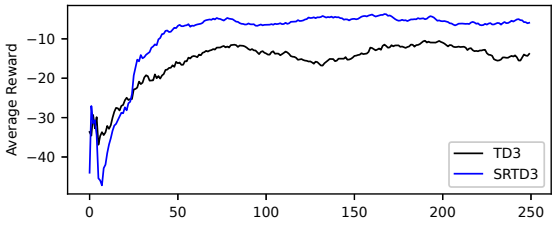



Fig. 2. Average Reward Plot for SISO System

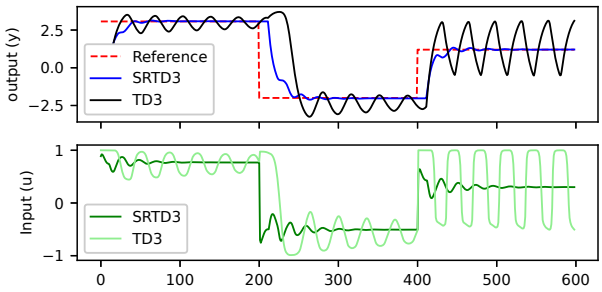



Fig. 3. Validation of Setpoint Tracking in SISO System

FMU simulation in Python. In all experiments, a uniform sampling interval ($t_s$) of 1 second was maintained, and all observations were standardized to fit within the [-1,1] range.

### A. SISO with Delay

We considered a simple single-input single-out (SISO) system with a reasonable time delay for evaluating the proposed approach. For simulation purposes, we assume that $u$ and $y$ are related according to the following transfer function:

$$G(s) = \frac{4e^{-10s}}{20s + 1}$$

*1) RL agent Formulation:* The goal of the RL agent is to maintain $y$ at specified setpoint $y_{ref}$ by adjusting the $u$. The details of the considered states/observations, actions and formulated reward are summarized in Table I.

*2) Training Process:* We trained two RL agents for our study: one using the TD3 algorithm with a traditional actor-critic network based on fully connected layers, and another, the proposed agent SRTD3.

During the training process for both agents, each episode began with a randomly selected feasible $y$ value within the range of [-4, 4] as the reference setpoint $y_{ref}$. Each episode consisted of 200 time steps. Both agents were trained over 250 episodes. Figure 2 illustrates the average reward for both agents using a moving window size of 20. From Figure 2, it can be inferred that the performance of the SRTD3 agent is superior to that of the TD3 agent.

*3) Validation Process:* To validate the performance, we ran simulations for 600 samples, where the trained agents took actions to maintain the output close to the requested setpoint. The reference setpoint was varied every 200 samples. Figure 3 illustrates the setpoint tracking capabilities of both the agents. It can be inferred from Figure 3 that the SRTD3 agent yields superior setpoint tracking performance compared to the TD3 agent due to imbibing delay-handling capabilities in the proposed architecture.

### B. Wood and Berry Distillation Column

We demonstrate our proposed approach on the Wood-Berry distillation column, a recognized benchmark in process control applications. This column posses two inputs and two outputs with considerable inherent process delays. The underlying input-output relationships are shown below [21]:

$$\begin{bmatrix} x_D \\ x_B \end{bmatrix} = \begin{bmatrix} \frac{12.8e^{-s}}{16.7s+1} & \frac{-18.9e^{-3s}}{21s+1} \\ \frac{6.6e^{-7s}}{10.9s+1} & \frac{-19.4e^{-3s}}{14.4s+1} \end{bmatrix} \begin{bmatrix} R \\ S \end{bmatrix}$$

*1) RL agent Formulation:* The goal of the RL agent is to maintain the desired distillate composition ($x_D$) and bottom composition ($x_B$) at specified setpoints by adjusting the reflux flow ($R$) and steam flow ($S$). The details of the considered observations, actions and the reward formulation are provided in Table III.

*2) Training Process:* During training, each episode started with randomly chosen feasible compositions as reference setpoints. Each episode lasted 200 time steps.

TABLE III

WOOD AND BERRY RL FORMULATION

| | |
|---|---|
| Observations (6) | Concentrations: $x_D, x_B$<br>Errors: $x_{D_{ref}} - x_D, x_{B_{ref}} - x_B$<br>Integral Errors: $\sum(x_{D_{ref}} - x_D)t_s$, $\sum(x_{B_{ref}} - x_B)t_s$ |
| Actions (2) | Reflux Flow(R), Steam Flow(S) |
| Reward | $\lvert x_{D_{ref}} - x_D \rvert + \lvert x_{B_{ref}} - x_B \rvert$ |

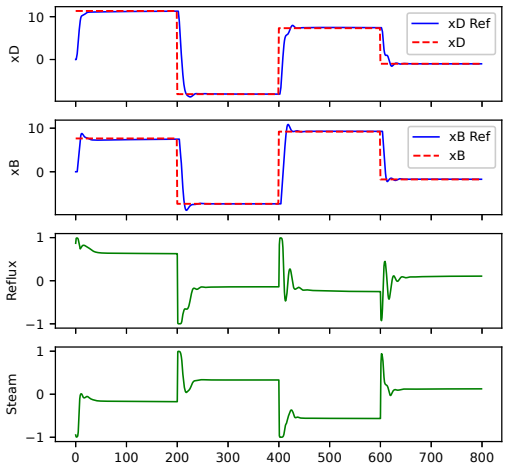



Fig. 4. Validation of setpoints tracking in the Wood and Berry system

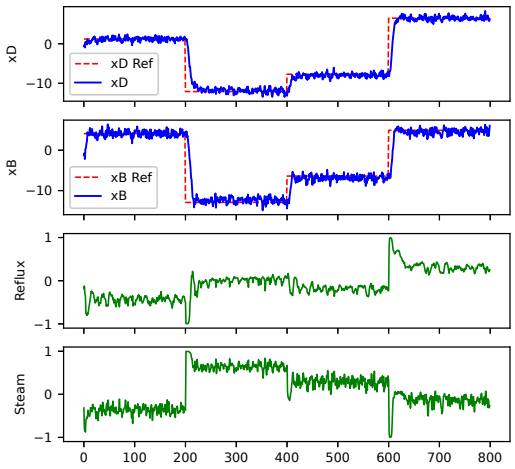



Fig. 5. Validation of Setpoint Tracking in the Wood and Berry System with Noise

*3) Validation Process:* To validate the performance, we conducted simulations for 800 samples, during which the trained agents took actions to maintain the outputs to the requested setpoints. The reference setpoints are varied every 200 samples. As can be inferred from Figure 4, the proposed SRTD3 RL agent is able to effectively track the setpoints for the considered MIMO system with delays. Further, to evaluate the noise handling capability, we corrupted the distillate and bottom composition measurements with colored noise, to replicate the effect of unmeasured disturbances. The colored noise was generated with a mean of 0 and a variance of 0.5, and then passed through the following discrete-time filter:

$$H(z) = \frac{1}{1 - 0.95z^{-1}}$$

As shown in Figure 5, the SRTD3 agent is able to effectively handle process noise along with process delays.

### C. Shell Heavy Oil Fractionator

We consider the subsystem of the shell oil fractionator (SOF) benchmark to test the efficacy of our proposed algorithm on considerably complex multivariate systems with varied delays among the input output relationships. The considered subsystem comprises of three outputs [Top End Point ($y_1$), Side End Point ($y_2$), and Top Temperature ($y_3$)], three inputs [Top Draw ($u_1$), Side Draw ($u_2$), and Bottom Reflux Duty ($u_3$)], and two disturbances [Inter Reflux Duty ($d_1$) and Upper Reflux Duty ($d_2$)]. The multivariate input-output relationship is summarised below:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} \frac{4.05e^{-27s}}{50s+1} & \frac{1.77e^{-28s}}{60s+1} & \frac{5.88e^{-27s}}{50s+1} \\ \frac{5.39e^{-18s}}{50s+1} & \frac{5.72e^{-14s}}{60s+1} & \frac{6.90e^{-15s}}{40s+1} \\ \frac{3.66e^{-2s}}{9s+1} & \frac{1.65e^{-20s}}{30s+1} & \frac{5.53e^{-2s}}{40s+1} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \begin{bmatrix} \frac{1.2e^{-27s}}{45s+1} & \frac{1.44e^{-27s}}{40s+1} \\ \frac{1.52e^{-15s}}{25s+1} & \frac{1.83e^{-15s}}{20s+1} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$$

TABLE IV
SHELL HEAVY OIL FRACTIONATOR RL FORMULATION

| | |
|---|---|
| Observations (9) | Observations: $y_1, y_2, y_3$<br>Errors: $y_{1,ref} - y_1, y_{2,ref} - y_2, y_{3,ref} - y_3$<br>Integral Errors: $\sum(y_{1,ref} - y_1)t_s$,<br>$\sum(y_{2,ref} - y_2)t_s, \sum(y_{3,ref} - y_3)t_s$ |
| Actions (3) | Top Draw($u_1$), Side Draw($u_2$),<br>Bottom Reflux Duty($u_3$) |
| Reward | $\lvert y_{1,ref} - y_1 \rvert + \lvert y_{2,ref} - y_2 \rvert + \lvert y_{3,ref} - y_3 \rvert$ |

*1) RL agent Formulation:* Here, the RL agent aims to maintain the top end point concentration, side end point concentration, and top temperature to the desired setpoint profiles by adjusting the manipulated variables, namely, top draw, side draw, and the bottom reflux duty. Additionally, the agent must simultaneously reject the effect of unmeasured disturbances, namely, inter reflux duty and upper reflux duty. The details of the considered observations, actions and the reward formulation are provided in Table IV:

*2) Training Process:* During training, each episode began with randomly selected feasible top temperatures and concentrations for the top-end and side-end points as the reference setpoints. Each episode spanned 300 time steps, during which disturbances $d_1$ and $d_2$ were randomly introduced.

*3) Validation Process:* To evaluate the performance, we conducted simulations for 800 samples, during which the trained agents took actions to maintain outputs close to the requested setpoints. The setpoints are varied for every 200 samples. Figure 6 illustrates the setpoint tracking capabilities of the SRTD3 agent for the MIMO system with variable delays across the input-output relationships.

To evaluate the disturbance rejection capability, we applied variable step disturbance profiles to the inter reflux duty $d_1$ and upper reflux duty $d_2$ variables as shown in Figure 7. The results shown in the figure reveal the efficacy of the proposed SRTD3 agent in effectively rejecting the introduced

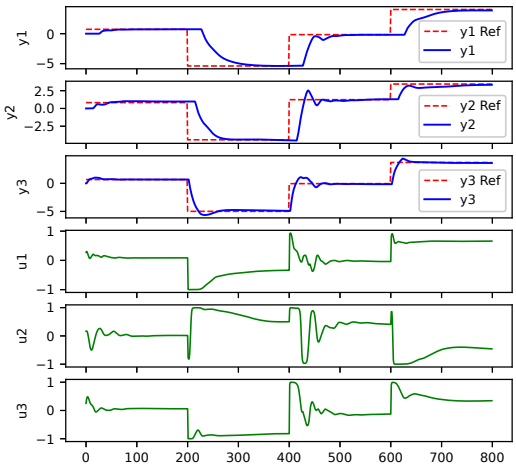



Fig. 6. Validation of Setpoints Tracking in the SOF System

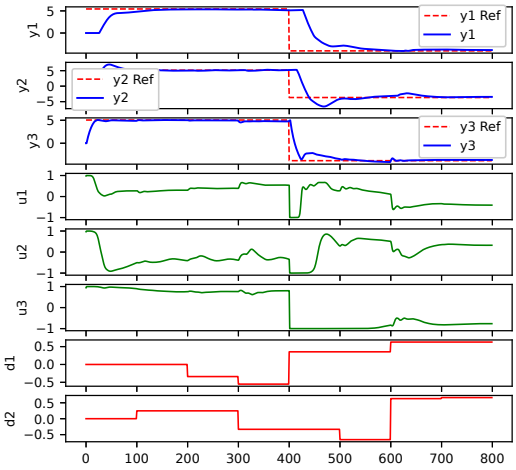



Fig. 7. Validation of Setpoint Tracking and Disturbance Rejection in the SOF System

disturbances while handling the setpoint tracking scenarios.

## V. Conclusive Remarks

This work proposed a smooth recurrent TD3 (SRTD3) algorithm, which embeds RNN architecture in the actor and critic networks of the state-of-the-art TD3 algorithm, along with regularization over action space, to address the challenges posed by process delays encountered in process industries. The simulation results reveal enhanced control performance compared to the existing architectures and the applicability is shown for a wider class of multivariate processes involving long and variable delays across input-output relationships. The proposed framework addresses the general inherent complexities prevalent in process industries and demonstrates significant practical relevance. Future directions involve extensions of present ideas to efficiently deal with time delays that have nonlinear dependencies with respect to the operating point.

## References


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

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

[3] G. Tesauro *et al.*, "Temporal difference learning and td-gammon," *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, 1995.

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[5] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[6] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, 2022.

[7] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE symposium series on computational intelligence (SSCI)*. IEEE, 2020, pp. 737–744.

[8] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2021.

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

[10] R. Nian, J. Liu, and B. Huang, "A review on reinforcement learning: Introduction and applications in industrial process control," *Computers & Chemical Engineering*, vol. 139, p. 106886, 2020.

[11] S. Spielberg, A. Tulsyan, N. P. Lawrence, P. D. Loewen, and R. B. Gopaluni, "Deep reinforcement learning for process control: A primer for beginners," *arXiv preprint arXiv:2004.05490*, 2020.

[12] S. Spielberg, R. Gopaluni, and P. Loewen, "Deep reinforcement learning approaches for process control," in *2017 6th international symposium on advanced control of industrial processes (AdCONIP)*. IEEE, 2017, pp. 201–206.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[14] N. P. Lawrence, P. D. Loewen, G. E. Stewart, and R. B. Gopaluni, "Data-driven process control via reinforcement learning and recurrent neural networks," *Foundations of Process Analytics and Machine Learning*, 2019.

[15] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," 2 2018.

[16] S. Mysore, B. Mabsout, R. Mancuso, and K. Saenko, "Regularizing action policies for smooth control with reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1810–1816.

[17] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour *et al.*, "Policy gradient methods for reinforcement learning with function approximation." in *NIPs*, vol. 99. Citeseer, 1999, pp. 1057–1063.

[18] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*. PMLR, 2014, pp. 387–395.

[19] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[20] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[21] R. Wood and M. Berry, "Terminal composition control of a binary distillation column," *Chemical Engineering Science*, vol. 28, no. 9, pp. 1707–1717, 1973.