Trajectory Tracking Control of a Batch Process using Deep Reinforcement Learning

M U Abuthahir * **Nabil Magbool Jan** *

* *Indian Institute of Technology Tirupati, Andhra Pradesh, 517619, India (e-mails: ch25s001@iittp.ac.in, nabil@iittp.ac.in).*

Abstract: Batch processes are indispensable for the production of low-volume and high-value products. However, control of a batch process is challenging due to the inherent nonlinearity, and time-varying characteristics. In this work, we consider the problem of nonlinear trajectory tracking control in batch processes. We assume that the reference trajectory is known in advance, and does not change between batches as it is common in repetitive processes. The main objective of this work is to develop a reinforcement learning approach to perform tracking control in a repetitive batch process. To this end, the batch process control is formulated as a Markov decision process with a suitable characterization of state vector, and reward design. A model-free function approximation approach based on deep Q learning is developed as a Deep Reinforcement Learning (DRL) controller. A nonlinear batch reactor system is used to demonstrate the efficacy of the proposed DRL controller. Further, we present the performance of the proposed DRL controller under feed and parametric uncertainties.

Keywords: tracking control, batch process, nonlinear model predictive control, deep Q learning, deep reinforcement learning

1. INTRODUCTION

Batch processes play an important role in low-volume, high-value products due to their flexibility in producing different products using the same processing equipment. However, control of batch process is challenging due to it's inherent nonlinear, time-varying and non-stationary behavior. Industrial applications of batch processes include the manufacture of pharmaceutical drugs and biochemicals like ethanol, specialty chemicals, and polymers (Bonvin et al., 2006). One of the most studied controllers for batch processes is nonlinear model predictive control due to its capability to handle constraints. However, it requires a reliable process model to achieve satisfactory performance (Rho et al., 1998; Corbett et al., 2013; Magbool Jan and Narasimhan, 2024). On the other hand, reinforcement learning provides a model-free approach in the control of process systems (Spielberg et al., 2019; Ankalugari et al., 2024). Reinforcement learning-based approaches have been studied in the optimization and control of batch processes (Yoo et al., 2021a). Some of the relevant works focus on maximizing economic objectives, production rates, product quality, or minimizing batch time. Peroni et al. (2005) developed an approximate dynamic programming approach to simultaneously minimize batch time and maximize production rates in the production of invertase. Very recently, Byun et al. (2020) developed a proximal policy optimization approach with the RL goal of maximizing the production rates in a fed batch bioethanol process. The focus of this work is to develop a Deep Reinforcement Learning (DRL) approach to perform a nonlinear setpoint tracking control in batch processes.

Reinforcement learning has also been used in the tracking control of batch processes. Yoo et al. (2021b) developed an RL approach to perform batch control based on process economic performance and path and end constraint satisfaction using Monte Carlo Deep Deterministic Policy Gradient (MC-DDPG) method. The above work studies the nonlinear setpoint tracking with the use of phase segmentation to account for rapidly changing slopes. MC-DDPG algorithm is also shown to handle non-stationary, and irreversible characteristics of a batch process. Very recently, Abuthahir and Jan (2024) presented the performance of tabular reinforcement learning approaches for tracking control in a batch reactor. Although this approach can deal with nonlinear setpoint tracking for within-batch control, it is not suitable for high-dimensional state and action space.

In this work, we develop the deep Q learning-based DRL controller for tracking a nonlinear setpoint trajectory without phase segmentation in a batch process. To this end, we propose time-augmented output variables as state variables to account for the time-varying nature of a batch process, and the reward design was based on the stage-wise cost of a model predictive controller. Finally, we demonstrate the efficacy of the aforementioned DRL controller in a simulated batch reactor system with feed uncertainty and parametric uncertainty.

The rest of this paper is organized as follows. Section 2 presents the problem statement for tracking control of a batch process. In section 3, we present the deep reinforcement learning framework for batch process control with continuous states and discrete actions. In section 3.1,

we formulate the optimal control problem as a Markov decision process, and propose the DRL controller design for nonlinear trajectory tracking. Finally, in section 4, we demonstrate the efficacy of tracking control in a batch reactor system.

2. PRELIMINARIES

2.1 System Description

Consider a discrete-time dynamic model of a batch process. It can be mathematically expressed in state space representation as follows:

$$x_{k+1} = f(x_k, u_k, d_k, \bar{\theta}) \tag{1}$$

$$y_k = h(x_k, u_k) \tag{2}$$

where $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_d} \to \mathbb{R}^{n_x}$ is a vector mapping of manipulated variables $u_k \in \mathbb{R}^{n_u}$, disturbance variables $d_k \in \mathbb{R}^{n_d}$, and state variables $x_k \in \mathbb{R}^{n_x}$ to the next state x_{k+1}, $h : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$ is a vector mapping of input variables u_k and state variables x_k to the output variables y_k, and $\bar{\theta}$ denotes the nominal values of parameters. These model equations are often derived from material and energy balances.

2.2 Optimal Control Problem

In general, the goal of within-batch control is to perform a setpoint tracking of one or more process variables in a batch process. The setpoint trajectory signifies the recipe obtained from recipe optimization. In this work, we assume the recipe is already available, and the objective is to find the optimal control policy to achieve the desired recipe as defined by the time-varying setpoint trajectory (denoted by $y_{sp,k}$). Thus, the discrete-time optimal control problem can be formulated as follows:

$$\min_{u_{0:N}} \quad \mathcal{J} = \sum_{k=0}^{N-1} \|y_k - y_{sp,k}\|_{\Phi}^2 + \|u_{k+1} - u_k\|_{\Lambda}^2 \tag{3}$$

$$s.t. \quad x_{k+1} = f(x_k, u_k, \bar{\theta}) \quad \text{given} \quad x_0 \tag{4}$$

$$y_k = h(x_k, u_k) \quad \forall k = 0 \text{ to } N \tag{5}$$

$$c(x_k, u_k) \leq 0 \tag{6}$$

where $\mathcal{J}$ is the cost function that we need to minimize to achieve the control goal, $\|a\|_{\Phi} = \sqrt{a^T Q a}$ denotes the quadratic norm of vector a with respect to matrix Φ, c denotes the set of inequalities for safety and environmental reasons, $y_{sp,k}$ denotes the time-varying setpoint trajectory as prescribed by recipe optimization, N denotes the number of time steps in a fixed batch time t_f, and x_0 denotes the initial values of state variables. Notice that the first term in the objective function minimizes the tracking error, whereas, the second term minimizes the control effort. Φ and Λ are weighting matrices that are chosen to achieve the trade-off between tracking error and control effort. The optimal solution $u^*_{0:N}$ is the control sequence one needs to implement to achieve the time-varying setpoint tracking control for the given initial state x_0 and nominal values of parameter $\bar{\theta}$. Owing to uncertainties and disturbances, the determined open-loop policy is often not suitable. Therefore, a Batch Nonlinear Model Predictive Control (BNMPC) is often used in a shrinking horizon fashion or a combination of shrinking and receding horizon fashion (Nagy and Braatz, 2010).

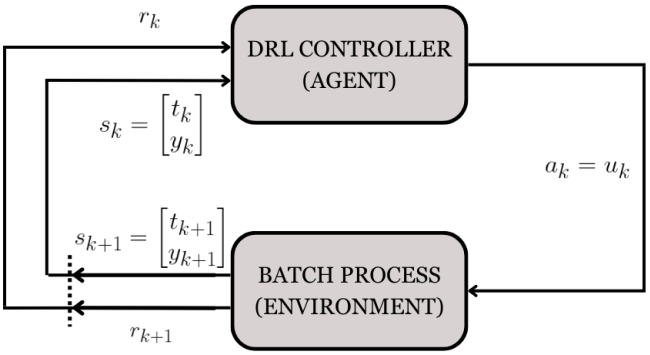

Fig. 1. Reinforcement learning based control of a batch process

3. DEEP REINFORCEMENT LEARNING BASED BATCH PROCESS CONTROL

In this section, we design a model-free deep Q learning based DRL controller for the time-varying setpoint trajectory tracking problem in a batch process. We assume that the reference trajectory is known in advance and does not change between batches. Reinforcement learning belongs to a class of machine learning algorithms in which the learning is achieved for the predefined objective by directly interacting with the environment or system (in our case, it is a batch process). The main philosophy of reinforcement learning in the context of batch process control is depicted in Figure 1. Reinforcement learning can be applied to sequential decision making problems some of which can be formulated as a Markov Decision Process. Markov decision process, defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, is a stochastic process that takes a state from the set $\mathcal{S}$ to the next state in the same set through probability transition function $\mathcal{P}$ by taking action from the set $\mathcal{A}$, and quantify the merit of this action in terms of the reward function $\mathcal{R}$.

State ($s_k \in \mathbb{R}^m = \mathcal{S}$) is the set of variables that describe the environment at k^{th} time step. State may contain observations, sensory inputs or any other relevant data that describes the environment that the agent can observe. Owing to the time-varying nature of a batch process, the state is given by $s_k = [t_k \; y_k]^T$. Thus, $m = n_y + 1$.

Action, denoted by $a_k \in \mathcal{A} = \{a^1, a^2, \cdots, a^n\}$, is the decision made by the agent. This action is executed on the environment and the next state s_{k+1} is observed. The agent executes the next action a_{k+1} at $k + 1^{th}$ time step, and so on. In this way, the agent interacts with the environment to find the best action for any state to achieve the desired goal. Actions can be either discrete or continuous, depending on the nature of the problem. For the batch process control problem, the action is the manipulated variable, that is, $a_k = [u_k]$. In this work, we consider discrete actions. The agent acquires a **reward** at each timestep after executing an action which quantifies the effectiveness of the action with respect to the desired goal. It is denoted by $r_k \in \mathbb{R} = \mathcal{R}$. Reward in our case is given by,

$$r_k = -\|y_k - y_{sp,k}\|_{\Phi}^2 - \|u_k - u_{k-1}\|_{\Lambda}^2$$

This is the same as the stage-wise cost in optimal control problem. **Policy** is a function mapping from state s_k to

action a_k that the agent follows to choose an action on a given state. It is denoted by π. The relation between policy π, state s_k and action a_k is given below:

$$\pi(s_k) = a_k \tag{7}$$

There may exist several policy or function mapping. Our aim is to determine the optimal policy π^* that enables us to accomplish our control goal.

Return is the exponentially discounted sum of rewards from k^{th} timestep till the termination of the state action sequence. It is denoted by $G_k \in \mathbb{R}$. For a sequence of rewards $r_{k+1}, r_{k+2}, r_{k+3}, ...$, we can define returns as,

$$G_k = \sum_{n=0}^{N} \gamma^k r_{k+n+1} \tag{8}$$

Where $\gamma \in [0, 1]$ is the discount factor and N is the timestep where the state action sequence will terminate. γ can be 1 when N is finite and cannot be 1 when N is infinite. When γ is 0, the agent gives importance to the immediate reward, whereas, when γ is 1, the agent tries to maximize long-term rewards.

Action-value function is the effectiveness of taking action a_k starting from state s_k following policy π. It is denoted by $q^\pi(s_k, a_k)$. Therefore, it is the expected return given the state s_k and action a_k, and is given by,

$$\begin{aligned} q^\pi(s_k, a_k) &= E_\pi[G_k | S_k = s_k, A_k = a_k] \\ &= E_\pi\Big[\sum_{n=0}^{N} \gamma^n r_{k+n+1} | S_k = s_k, A_k = a_k\Big] \end{aligned} \tag{9}$$

The process of obtaining the action-value function $q^\pi(s_k, a_k)$ for a policy π is called policy evaluation, and the process of finding a new better policy from the action-value function is called policy improvement. **Optimal policy** belongs to set of policies that has the highest value function out of all possible policies in the policy space Π. It is denoted by π^*. Thus, the optimal control problem is formulated as the one that determines the optimal control policy such that it maximizes the expected return. Therefore, the optimal control policy π^* can be expressed in terms of optimal action-value function q^* as:

$$\pi^*(s^i) = \underset{a^j \in \mathcal{A}}{\operatorname{argmax}}\, q^*(s^i, a^j) \quad \forall s^i \in \mathbb{R}^m \tag{10}$$

It is important to notice that value functions have a fundamental property that they satisfy a particular recursive relationship expressed by the Bellman equation, and it is given below.

$$q^*(s_k, a_k) = \mathbb{E}_{s_{k+1}}[r_{k+1} + \gamma \max_{a_{k+1}} q^*(s_{k+1}, a_{k+1}) | s_k, a_k] \tag{11}$$

In general, the set of algorithms that find an optimal policy based on repeated policy evaluation and policy improvement are known as generalized policy iteration methods.

3.1 Deep Q Learning based Batch Process Control

In this work, we use Deep Q learning to design a DRL controller that can achieve nonlinear setpoint tracking of a batch process. To improve the generalization performance and to make the training of RL agent more scalable for higher dimensions of states, we can use a parameterized function approximator $q^*(s_k, a_k; \theta) \approx q^*(s_k, a_k)$ where θ is the parameter. In this case, a neural network is used as a nonlinear function approximator. The DRL controller consists of a deep neural network that takes continuous states s_k at k^{th} time step as an input and outputs $q^*(s_k, a^j; \theta) \forall a^j \in \mathcal{A}$. The action a_k at k^{th} time step will be obtained based on the chosen exploration-exploitation trade-off strategy. In particular, we use the ϵ-greedy strategy.

In this work, we use a feedforward neural network $g : \mathbb{R}^m \longrightarrow \mathbb{R}^n$ as a function approximator for the action-value function. It consists of m neurons for the input layer, L hidden layers with n_i neurons per layer, and the output layer contains n neurons which correspond to n possible control actions. Given this definition, a neural network can be described as a composition of functions as given below:

$$g(s_k, \theta) = \lambda_{L+1} \circ z \circ \lambda_L \circ z \circ ... \circ z \circ \lambda_1(s_k) \tag{12}$$

where λ_l is an affine transformation of the output from the $(l-1)^{th}$ layer, and it is given by,

$$\lambda_l(\zeta_{l-1}) = W_l \zeta_{l-1} + \alpha_l \tag{13}$$

where $\zeta_{l-1} \in \mathbb{R}^{n_{l-1}}, \zeta_{l-1} = z(\lambda_{l-1}(\zeta_{l-2}))$ is the input to the current layer, and it is the output from the previous layer. It is important to notice that $\zeta_0 = s_k$ is the input of the neural network. Further, $z : \mathbb{R}^{n_l} \longrightarrow \mathbb{R}^{n_l}$ denotes a non linear activation function. In this work, we use a rectified linear activation function and it is given by,

$$z(\lambda_l(\zeta_{l-1})) = max(0, \lambda_l(\zeta_{l-1})) \tag{14}$$

The parameters of the neural network $\theta = \{W_1, \alpha_1, W_2, \alpha_2, ...W_{L+1}, \alpha_{L+1}\}$ contains the weights W_l and biases α_l and they are represented as follows,

$$W_l \in \mathbb{R}^{n_l \times n_{l-1}} \quad \forall l \in 1, 2, ...L+1 \tag{15}$$

$$\alpha_l \in \mathbb{R}^{n_l} \quad \forall l \in 1, 2, ...L+1 \tag{16}$$

Notice that $n_0 = m$ and $n_{L+1} = n$. The goal in Deep Q Learning is to learn the neural network parameters θ by interacting with the batch process environment.

To this end, the deep Q network is trained to reduce the mean squared error in the bellman equation where the optimal target values $r_{k+1} + \gamma \max_{a^j} q^*(s_{k+1}, a^j)$ are substituted with approximate target values, $y = r_{k+1} + \gamma \max_{a^j} q_t^\pi(s_{k+1}, a^j; \theta^-)$, where θ^- is the parameters of the target Q network, and these parameters are updated every C steps by resetting it to the same parameters of current Q network. This leads to a sequence of loss functions that changes at every interval (Mnih et al., 2015).

$$L_p(\theta_p) = \mathbb{E}_{s_k, a_k, r_{k+1}}[(\mathbb{E}_{s_{k+1}}[y | s_k, a_k] - q^\pi(s_k, a_k; \theta_p))^2] \tag{17}$$

$$\begin{aligned} &= \mathbb{E}_{s_k, a_k, r_{k+1}, s_{k+1}}[(y - q^\pi(s_k, a_k; \theta_p))^2] \\ &\quad + \mathbb{E}_{s_k, a_k, r_{k+1}}[\mathbb{V}_{s_{k+1}}[y]] \end{aligned} \tag{18}$$

where $\mathbb{E}_{s_k, a_k, r_{k+1}}[\mathbb{V}_{s_{k+1}}[y]]$ is the variance of targets y. This term is independent of θ_p, and thus the gradient of this term is zero, and removing this term does not affect the iterative updates to the parameters θ_p in the Q network. The agent is trained with experiences $e_p = (s_k, a_k, r_{k+1}, s_{k+1})$ that is stored in a dataset $\mathcal{D} = \{e_1, \cdots, e_p\}$ called replay memory. However, owing to memory limitations, this dataset $\mathcal{D}$ contains only the last D experiences (that is, $\mathcal{D} = \{e_{p-D}, \cdots, e_p\}$). D denotes the size of the replay memory. During learning, we apply Q learning updates on

minibatch of experiences $e \sim U(\mathcal{D})$ drawn uniformly at random from the replay memory. The loss function used is

$$L_p(\theta_p) = \mathbb{E}_{s_k,a_k,r_{k+1},s_{k+1}\sim U(\mathcal{D})}[(y - q^\pi(s_k, a_k; \theta_p))^2] \quad (19)$$

The loss function $L_p(\theta_p)$ is optimized using stochastic gradient descent to update the weights of the Q network for each iteration p. Taking the gradient of this loss function with respect to θ_p and simplifying, we get

$$\begin{aligned}\nabla_{\theta_p} L_p(\theta_p) = \mathbb{E}_{s_k,a_k,r_{k+1},s_{k+1}\sim U(\mathcal{D})}[(r_{k+1} \\ + \gamma \max_{a^j} q_t^\pi(s_{k+1}, a^j; \theta^-) \\ - q^\pi(s_k, a_k; \theta_i))\nabla_{\theta_p} q^\pi(s_k, a_k; \theta_p)] \end{aligned} \quad (20)$$

This version of Deep Q learning in which we use a separate target network to generate labels is called Double Deep Q learning (van Hasselt et al., 2016). The detailed algorithm for tracking control of a batch process is presented in Algorithm 1.

Algorithm 1: Deep reinforcement learning for batch process control

Initialize replay memory $\mathcal{D}$ of size D;
Initialize action-value function $q^\pi(\theta)$ with random weights θ;
Initialize target action-value function $q_t^\pi(\theta^-)$ with weights $\theta^- = \theta$;
repeat until convergence
  Choose initial state $s_0 \in \mathcal{S}$ s.t. all states have probability > 0;
  for *each step in the episode* **do**
    Select random control action a_k with probability ϵ ;
    Otherwise, $a_k = \underset{a^j \in A}{\operatorname{argmax}}\, q^\pi(s_k, a^j; \theta)$;
    Take control action a_k;
    Measure states s_{k+1} and compute reward r_{k+1};
    Store state transitions $(s_k, a_k, r_{k+1}, s_{k+1})$ in $\mathcal{D}$;
    Sample random minibatch of state transitions $(s_b, a_b, r_{b+1}, s_{b+1})$ from $\mathcal{D}$;
    for *each transition b in batch* **do**
      if *episode terminates at* $b+1$ **then**
        $y_b = r_{b+1}$;
      else
        $y_b = r_{b+1} + \gamma \max_{a^j \in A} q_t^\pi(s_{b+1}, a^j; \theta^-)$;
    Perform a stochastic gradient descent on (19) with respect to the network parameters θ;
    Reset $q_t^\pi(\theta^-) = q^\pi(\theta)$ every C steps ;
    $k \leftarrow k + 1$;
  until $k = N$;

4. CASE STUDY

In this section, we demonstrate the application of Deep Q learning in a nonlinear batch reactor system undergoing the $A \longrightarrow B$, at nonisothermal conditions in which the goal is to maximize the product quality (Lee et al., 2000; Abuthahir and Jan, 2024). This can be achieved by controlling the temperature trajectory of the reactor by manipulating the jacket temperature. Let $C_{A,k}$ denotes the concentration of reactant A inside the reactor at time step k, T_k denotes the temperature inside the reactor at time step k, $T_{j,k}$ denotes the jacket temperature at time step k. In this work, we consider the discrete-time, nonlinear state space model of the batch reactor. It is given by,

Table 1. Hyperparameter settings

Symbol	Description	Value
α	Learning rate	10^{-3}
μ	Decay rate	10^{-5}
γ	Discount factor	1
C	Target reset steps	9
ϵ	ϵ-greedy policy	0.05
-	Batch size	1.2×10^4
-	Hidden layer	$[400, 300, 200]$

$$C_{A,k+1} = C_{A,k} + \Delta t(-k_0 e^{\frac{-Ea}{RT_k}} C_{A,k}^2) \quad (21)$$

$$T_{k+1} = T_k + \Delta t(\beta_1(T_k - T_{j,k}) + \beta_2 k_0 e^{\frac{-E_A}{RT_k}} C_{A,k}^2) \quad (22)$$

where Δt is the sampling interval, $\beta_1 = \frac{-UA}{MC_p}$ is the constant relating overall heat transfer coefficient, $\beta_2 = \frac{(-\Delta H)V}{MC_p}$ is the constant relating heat of reaction, k_0 is the reaction rate constant, $\frac{E_A}{R}$ is the constant related to activation energy, T_0 is the initial reactor temperature, C_{A0} is the initial reactor concentration and t_f is the batch time. For a description of the model parameters and the corresponding values used in the simulation, an interested reader is referred to the work of (Abuthahir and Jan, 2024). To formulate the batch process control problem as a Markov decision process, we define the following state measurements as $x_k = [t_k \; T_k \; C_{A,k}]^T$, and manipulated variable as $u_k = [T_{j,k}]$. Assuming the states are directly measured (that is, $y_k = x_k$), the reward function is given by,

$$r_k = -\|y_k - y_{k,ref}\|_\Phi^2 - \|u_{k+1} - u_k\|_\Lambda^2 \quad (23)$$

where Φ and Λ are chosen to be I and 0.1, respectively.

For training, the episodes start with a state $s_0 = [0 \; T_0 \; C_{A0}]^T$ where T_0 is randomly chosen in the interval $[293, 308]$ and C_{A0} is randomly chosen in the interval $[0, 1]$ with uniform probabilities respectively. The hyperparameter settings used are tabulated in Table 1. The designed DRL controller is trained with 70000 episodes for $N = 30$ timesteps, and 100000 episodes for $N = 40$ timesteps. Further, we use $\epsilon-$greedy exploration strategy during training. Figure 2 shows the episode versus return during the training of DRL controller. It can be noticed that as the number of episodes increases, the return increases and then flattens which signifies the convergence of DRL algorithm. In the testing phase, we initialize the episode with a state $s_0 = [0 \; 298 \; 0.6]^T$ with a greedy policy to assess the tracking performance. Figure 3 presents the tracking performance of the proposed DRL controller obtained with 40 uniform discretization points in time space.

In Figure 3, the top row presents the tracking performance under nominal feed condition of $T_0 = 298$ K and $C_{A0} = 0.6$ mol/l. It can be observed that the temperature inside the reactor closely follows the desired reference trajectory. The corresponding concentration profiles inside the batch reactor, and the required changes in jacket temperature to achieve the tracking control is also presented in the top row of Figure 3.

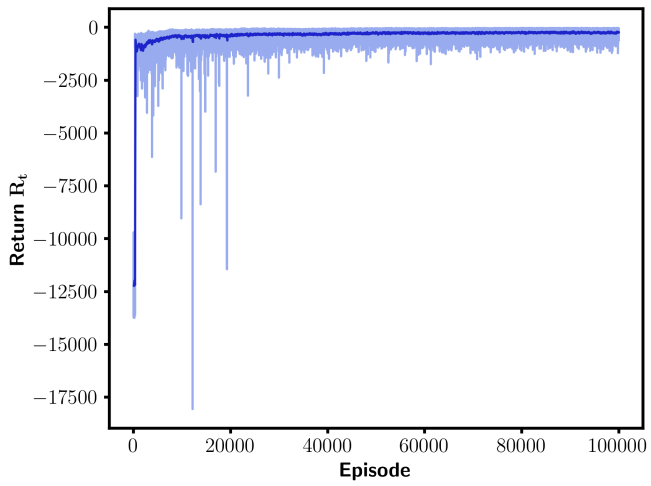

Fig. 2. Episode vs return obtained using Deep Q Learning

4.1 Effect of Feed Variations

Feed variations (such as initial feed composition or feed temperature) is a common occurrence in batch processes, and it often affects the product quality. In this subsection, we aim to show the tracking performance of the designed DRL controller despite feed variations. The middle row of Figure 3 shows the tracking performance for two different feed conditions (that is, Feed Condition 1: $T_0 = 295\ K$ and $C_{A0} = 0.8\ mol/l$ and Feed Condition 2: $T_0 = 302\ K$ and $C_{A0} = 0.5\ mol/l$) are presented. Despite uncertainties in the feed, the designed DRL controller was able to track the desired reactor temperature profile. However, due to the feed variations, it can be inferred that it takes a few time steps to achieve good tracking performance. Further, the corresponding concentration profiles inside the batch reactor for two different feed conditions, and the required changes in jacket temperature for these feed conditions are also presented in the bottom row of Figure 3. It is important to notice that due to the feed variations, the initial control moves under feed uncertainties are different from the nominal control moves obtained under nominal feed conditions.

Table 2 presents the performance metrics of the designed DRL controller for different feed conditions. As the feed condition in the nominal case is the same as the initial point in the reference trajectory, the mean absolute error and root mean square error are the least. However, for the non-nominal feed conditions, due to a different initial point than the reference point, the controller takes a few initial time steps to follow the reference trajectory. This can be inferred from the higher mean absolute error and root mean square error for the non-nominal feed conditions. On the other hand, the control effort required is large under nominal conditions. This can be attributed to a large step change in jacket temperature in the initial time step as can be observed in Figure 3.

For the purpose of comparison, we consider the state-of-the-art BNMPC technique with a prediction and control horizon of 10. All other settings for BNMPC are similar to RL settings. Following the work of Nagy and Braatz (2010), we employ a combination of receding horizon and shrinking horizon. It can be observed that BNMPC emphasizes on the control effort at the cost of tracking error. This can also be observed from Table 2.

Table 2. Performance metrics

Controller	**MAE**	**RMSE**	**Control Effort**
BNMPC	0.57	0.92	4.30
DRL Controller (Nominal)	0.32	0.45	37.67
Feed Variation			
Feed Condition 1	0.46	0.68	17.84
Feed Condition 2	0.53	0.92	20.76
Parameter Variation			
Parameter Condition 1	0.396	0.51	49.52
Parameter Condition 2	0.37	0.475	12.51

4.2 Effect of Parametric Uncertainty

In this subsection, we study the effect of parametric uncertainty on the designed DRL controller. To this end, we consider that the reaction frequency factor k_0 is known with uncertainty. Figure 3 shows the tracking performance of the trained DRL controller for different values of k_0 (that is, Parameter Condition 1: $k_0 = 1.9 \times 10^{19}\ l\ mol^{-1}min^{-1}$ and Parameter Condition 2: $k_0 = 4.4 \times 10^{19}\ l\ mol^{-1}min^{-1}$), and it can be observed that the proposed DRL controller is robust to parametric uncertainty. The corresponding tracking performance of DRL controller is presented in Table 2.

5. CONCLUSIONS

A model-free, function approximation-based DRL controller was designed to compute the optimal policy for the manipulated input. The proposed approach shows excellent tracking performance despite feed and parametric uncertainty. It is important to notice that this approach scales well in high-dimensional state space, but does not scale well for action space. It is worth mentioning that the number of interactions between the DRL controller and the batch process to attain satisfactory training is very high. Therefore, we are currently investigating on improving the sample efficiency of the proposed methodology. Furthermore, we are working on extending the approach to account for batch-to-batch variations.

REFERENCES

Abuthahir, M.U. and Jan, N.M. (2024). Time-varying set-point tracking for batch process control using reinforcement learning. In *2024 18th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 1148–1153.

Ankalugari, R.Y., Abuthahir, M.U., Jan, N.M., and Joseph, A.G. (2024). Control of van de vusse reactor using deep reinforcement learning. In *2024 18th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 1154–1159.

Bonvin, D., Srinivasan, B., and Hunkeler, D. (2006). Control and optimization of batch processes: Improvement of process operation in the production of specialty chemicals. *IEEE Control Systems Magazine*.

Byun, H.E., Kim, B., and Lee, J.H. (2020). Robust dual control of batch processes with parametric uncertainty

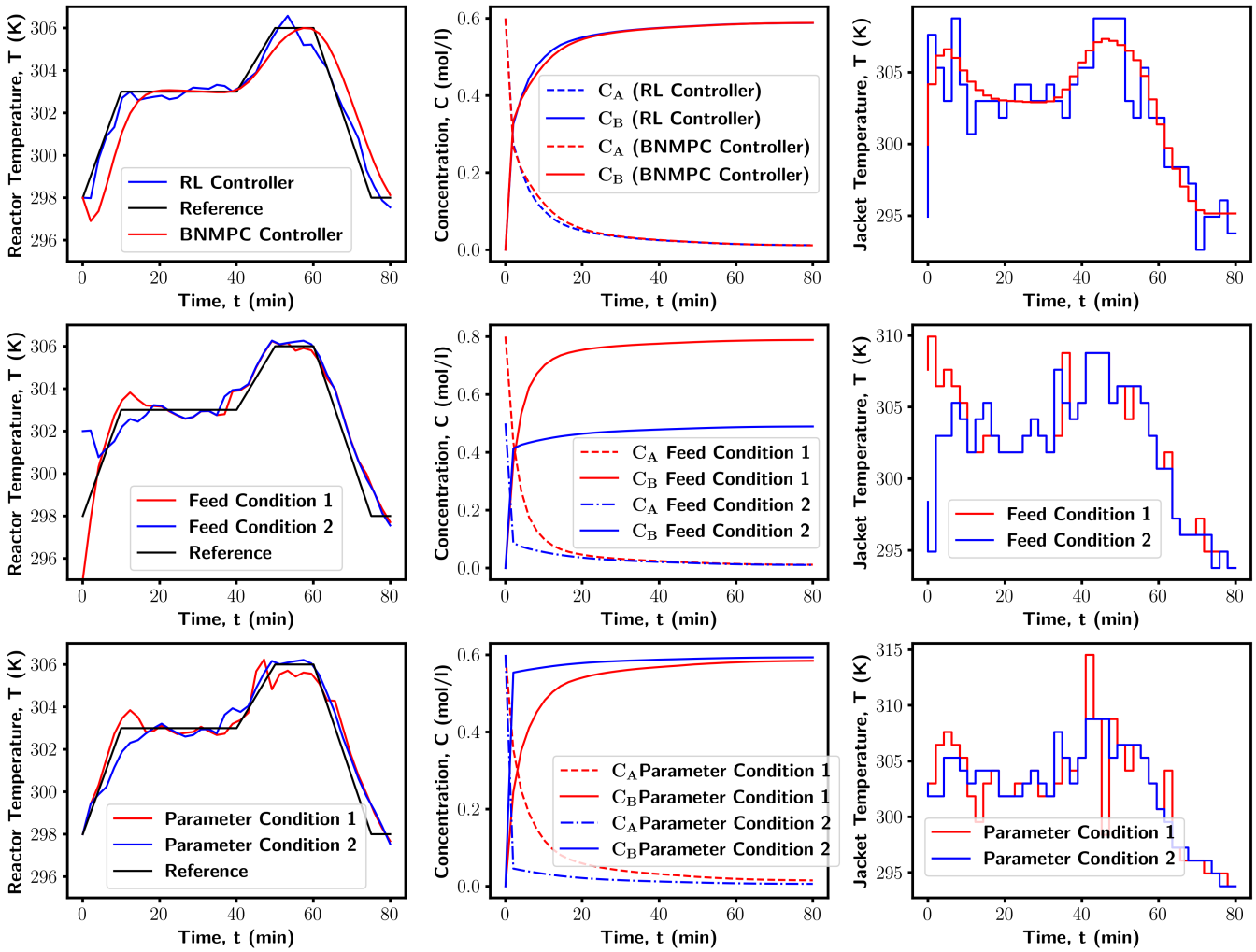

Fig. 3. Performance of the DRL controller: (a) Top row - under nominal condition, (b) Middle row - under two different feed conditions, (c) Bottom row - under two different parameter conditions

using proximal policy optimization. In *2020 59th IEEE Conference on Decision and Control (CDC)*, 3016–3021.

Corbett, B., Macdonald, B., and Mhaskar, P. (2013). Model predictive quality control of polymethyl methacrylate. In *2013 American Control Conference*, 3942–3947.

Lee, J.H., Lee, K.S., and Kim, W.C. (2000). Model-based iterative learning control with a quadratic criterion for time-varying linear systems. *Automatica*, 36(5), 641–657.

Magbool Jan, N. and Narasimhan, S. (2024). Economic performance of model predictive control at back-off operating point. *Journal of Process Control*, 139, 103231.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540).

Nagy, Z.K. and Braatz, R.D. (2010). *Control system applications*. CRC press.

Peroni, C., Kaisare, N., and Lee, J. (2005). Optimal control of a fed-batch bioreactor using simulation-based approximate dynamic programming. *IEEE Transactions on Control Systems Technology*, 13(5), 786–790.

Rho, H.J., Huh, Y.J., and Rhee, H.K. (1998). Application of adaptive model-predictive control to a batch mma polymerization reactor. *Chemical Engineering Science*, 53(21), 3729–3739.

Spielberg, S., Tulsyan, A., Lawrence, N.P., Loewen, P.D., and Bhushan Gopaluni, R. (2019). Toward self-driving processes: A deep reinforcement learning approach to control. *AIChE Journal*, 65(10), 1–20.

van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1).

Yoo, H., Byun, H.E., Han, D., and Lee, J.H. (2021a). Reinforcement learning for batch process control: Review and perspectives. *Annual Reviews in Control*, 52, 108–119.

Yoo, H., Kim, B., Kim, J.W., and Lee, J.H. (2021b). Reinforcement learning based optimal control of batch processes using monte-carlo deep deterministic policy gradient with phase segmentation. *Computers & Chemical Engineering*, 144, 107133.