Title

Constrained Control of a Process Network using Multi-Agent Reinforcement Learning

Author

Korimerla, Krishna Teja

Publication Date

2024

Peer reviewed|Thesis/dissertation

Constrained Control of a Process Network using Multi-Agent Reinforcement Learning

By

KRISHNA TEJA KORIMERLA

THESIS

Submitted in partial satisfaction of the requirements for the degree of

Master of Science

in

Chemical Engineering

in the

Office of Graduate Studies

of the

University of California

Davis

Approved:

Nael H. El-Farra, Chair

Ahmet N. Palazoglu

Matthew J. Ellis

Committee in Charge

2024

Acknowledgements

I would like to express my deepest gratitude to the following individuals and organizations whose unwavering support and contributions have been instrumental in the completion of my Master's thesis:

My Thesis Advisor, Dr. Nael El-Farra, Committee members, Dr. Ahmet N. Palazoglu and Dr. Matthew J. Ellis: Your guidance, expertise, and patience have been invaluable throughout this research journey. Your mentorship has not only shaped this thesis but also my growth as a scholar.

My Family: I am profoundly thankful to my parents, for their unconditional love and unwavering belief in my abilities. Your sacrifices and encouragement have been my driving force.

Friends and Peers: To my friends and fellow students who provided a shoulder to lean on, listened to my ideas, and offered their insights, thank you for your camaraderie and moral support.

I am grateful to my Department and University of California, Davis for providing financial support that enabled me to conduct my research and complete my degree.

Finally, I would like to express my gratitude to everyone who has played a part in this academic endeavor, no matter how big or small. Your contributions have made a lasting impact on my academic and personal growth.

Thank you all for your unwavering support and belief in my abilities. This thesis would not have been possible without you.

Krishna Teja Korimerla

Constrained Control of a Process Network Using Multi-Agent Reinforcement Learning

Abstract

This thesis examines the application of model-free multi-agent reinforcement learning for autonomous model-free control and constrained optimization of process networks comprised of multiple interconnected processes. Initially, different types of multi-agent reinforcement learning algorithms are outlined, and the challenges faced in their implementation for process control are identified. Reinforcement learning is then combined with optimal control, where the reward function is designed based on quadratic penalty functions to improve computational time and achieve improved performance compared to sparse rewards proposed in previous research studies. Several multi-agent reinforcement learning strategies, including centralized, decentralized, and mixed-type (centralized action and decentralized execution) are considered and compared. An assessment of the benefits and drawbacks of each control strategy is presented. Finally, an evaluation of the robustness of the control system against parametric uncertainties and sensor noise which are of practical significance is given. Throughout the thesis, a model system comprising of two interconnected non-isothermal chemical reactors with a recycle stream and multiple reactions is used to illustrate the design and implementation of the reinforcement learning agents. The control objective is to regulate the reactors' temperatures and concentrations at desired set-points. The proposed framework can be applied to more general process networks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

A multi-agent system can be defined as a group of autonomous,interacting entities sharing a common environment, which they perceive with sensors and upon which they act with actuators [38, 35].Multi-agent systems are finding applications in a wide variety of domains including robotic teams, distributed control, resource management, collaborative decision support systems, and data mining, to name a few [21]. The complexity of many tasks arising in these domains makes them difficult to solve with pre-programmed agent behaviors. The agents must, instead, discover a solution on their own, using learning. A significant part of the research on multi-agent learning concerns Reinforcement Learning (RL) techniques. Well-understood algorithms with good convergence and consistency properties are available for solving the single-agent RL task, both when the agent knows the dynamics of the environment and the reward function (the task model), and when it does not.

In the context of chemical process control, one of the early works involving reinforcement learning focused on temperature control of a nonlinear chemical reactor using only

information about the states without any model, and was conducted by Hoskins and Himmelblau [12]. The algorithm consists of two Artificial Neural Networks (ANNs). One ANN (called the evaluation ANN) evaluates the actions of the other ANN (called the action ANN). The action ANN gives the controlled input to the system. The weights of both ANNs are updated by a Reinforcement Learning algorithm based on the reward function.

While this work did not receive significant attention, due in part to the sub-optimal performance of the model-free controller compared to the PID controller, and also due to the computationally-intensive requirements of the algorithm (which took several hours and multiple computers to achieve the control task using data-driven and black-box model of the ANN), it paved the way for subsequent important developments in the area. For example, Syafiie, Fernando and Martinez [34] developed a Model-Free Learning Control (MFLC) strategy for pH process control based on reinforcement learning [32] and hierarchical reinforcement learning [33]. A single RL agent was designed to control the pH of a chemical process. Instead of using neural networks for policy and value functions, their work used an algorithm based on Q-Learning and a multi-step action framework [26]. Starting in 2012 with the advancements in Deep Learning, deep neural networks(DNNs) began to be used as feature extractors. This allowed the application of reinforcement learning in problems with high-dimensional state space (i.e., deep reinforcement learning (DRL)) such as, for example, cyber-physical systems and complex large-scale environments.

For these reasons, the number of publications on reinforcement learning applied to process control began to grow again. Martínez and Rodríguez [20] developed a controller

based on a deep reinforcement learning methodology in order to keep the level and composition of an exothermic continuous stirred-tank reactor under control. They implemented a single Twin Delayed Deep Deterministic Policy Gradient (DDPG:TD3) agent to control both the level and concentration of the reactor. The observations to the agent were the error, the integral error, and its derivative, which is quite similar to the concept of the PID controller. Their work, however, did not assess the robustness of the controller to external disturbances, sensor noise and controller performance at multiple set-points. Alhazm and Sarathy [1] developed a single DDPG RL agent to control the temperature of a complex reaction network in a continuous stirred-tank reactor. The process considered was a fourth order nonlinear dynamical system, with unstable zero dynamics, and could not be effectively controlled using linear controllers, resembling many real world control problems. The observation space for the RL agent consisted of the proportional error (ϵ), the integral error ($\int \epsilon dt$), the differential error ($\frac{d}{dt}\epsilon$), and the reactor temperature T_r. The reward function was chosen as $r_t = 10(|\epsilon| < 0.005) - 1(|\epsilon_t| \geq |\epsilon_{t-1}|)$, Where ϵ is the error between the process variable and the set-point. Ten points were rewarded if the error was less than 5% and one point was subtracted if the error increased. The performance of the RL controller was tested for set-point variations, disturbance rejection and sensor noise. While the results of this study are promising, the controller was designed to control only a single variable (reactor temperature). The methodology to design an RL control system for multiple process outputs was not studied in this work.

Other efforts involving the use of single-agent RL in chemical process control include the works in [28, 14, 27, 40]. In all these works, a single RL agent was used to control the process variables in a single (stand-alone) process unit. Work on simultaneous use

of multiple DRL agents towards optimal control includes the studies in [25, 6]. The work in [25] used multi-agent reinforcement learning to develop operational strategies for semi-batch reactors. The RL-based operation strategy uses an RL agent both for the feeding and mixing phase. The RL controller in the feeding phase is a simple single-loop controller, while it acts as a primary controller in a cascade-loop during the mixing phase. During the feeding phase, the temperature was controlled by varying the feeding rate, and in the mixing phase, the temperature was controlled by manipulating the coolant rate. The operation phases from the control point are independent, so the two RL agents are decoupled and independent.

The work in [6] focused on using multi-agent reinforcement learning for optimal control of wastewater treatment plants. The study used a novel technique, multi-agent deep reinforcement learning (MADRL), to simultaneously optimize dissolved oxygen and chemical dosage in a wastewater treatment plant. The reward function was specially designed from a life cycle perspective to achieve sustainable optimization. In this work, two agents were deployed in terms of a multi-agent paradigm, i.e., MADDPG. Compared to single-agent reinforcement learning, agents of MARL are merely able to receive local information, which is consistent with practical control scenarios. The main focus in this work was to control multiple variables in a single process. Their work, however, did not consider multiple processing units or coupling between the controlled variables.

Other works involving the application of multi-agent RL control strategies to a single continuous-stirred tank reactor (CSTR) include the study in [39].In this work, a multi-agent reinforcement learning (MARL) system was designed to control a multiple-input multiple-output (MIMO) process. The MARL system resembles a multi-loop control

system that is comprised of two control loops. In each loop, two RL agents are trained in a feed forward-feedback control configuration to control the process. Two controlled variables were identified in the CSTR model. These were the reactor temperature and reactor volume. For each control loop, a pair of TD3 agents were created and trained to control the controlled variable in a feed forward-feedback configuration. The TD3 agents interact with the CSTR model by making observations of the controlled variables and disturbances to maintain steady-state operation. Reward functions were applied to reward stable control and penalize deviations in the controlled variables. The objective of the MARL control system was to regulate the reactor temperature and volume so as to maintain a specified product concentration. A total of four TD3 agents were created; one feedback controller and one feed-forward controller each of the two control loops. Each feedback controller resembles a PID controller, and the feed-forward controller is a simple Proportional (P) controller. The reward function used was of the form

$$r_V(t) = \begin{cases} -|V_{\text{err}}(t)|, & \text{if} |V_{\text{err}}(t)| \geq 0.5 \\ 0, & \text{if} |V_{\text{err}}(t)| < 0.5 \end{cases}$$

The reward function resembles the criteria of the integral of the absolute value of the error (IAE) used for controller tuning. The main objective of the TD3 agent is to minimize the long-term cumulative IAE. Both agents controlling the temperature share the same reward function:

$$r_T = \begin{cases} -|T_{\text{err}}|, & \text{if} |T_{\text{err}}| \geq 0.1 \\ 0, & \text{if} |T_{\text{err}}| < 0.1 \end{cases}$$

The main limitation of this work is the fact that each RL agent is pre-trained individually and once all four agents are trained they are then loaded into the common

environment and then trained simultaneously. This might not be feasible in practical scenarios for such multiple training. The work also did not take into account the dynamic nature of the environment due to use of multiple agents. The environment is constantly changing due to the actions of multiple agents and it is imperative to develop RL agents such that they can communicate and work with each other to reach a common goal.

An examination of the available literature shows that, to date and within the chemical process control research area, no work has been conducted on applying deep reinforcement learning to the control of large-scale process networks comprised of interconnected processing units. The current work aims in part to bridge this gap. It should be noted that while significant work has been reported on the application of traditional model-based control methods to interconnected systems, these method are mostly focused on decentralized control due to the practical constraints on scaling a centralized controller to a vast number of interconnected units. A number of different decentralized feedback control methods can be found in [23, 29, 3, 19].

The goal of our work is to study the MIMO control of interconnected processes using multi-agent deep reinforcement learning. Within this context, the research is mainly concerned with the application of different multi-agent reinforcement learning frameworks, i.e.,centralized training for centralized control (CTCC), decentralized training for decentralized control (DTDC) and centralized training and decentralized Control (CTDC) to develop a model-free control strategy to control a process network comprised of interconnected units. An essential aspect for the reinforcement learning agent to reach its goal is the design of appropriate reward functions. In this research, the control goals are defined and this poses an additional challenge for the RL agent as it needs to achieve its goal

under constraints. To address this issue, the reward function is designed using exterior penalty functions. Specifically, quadratic penalty functions are used, which transforms the constrained problem into a series of unconstrained problems, and a large negative penalty is applied whenever the agent actions lead to outcomes outside the control goals. An additional reason for using penalty functions is their effectiveness in nonlinear programming as the processes considered in this work are highly nonlinear. Throughout this work, we outline the procedure for designing the actor and critic functions to approximate the policy and value functions in the reinforcement learning agents. We also outline the design of observation space for the reinforcement learning agents based on control theory. The results are applied to a process network comprised of two interconnected continuous-stirred tank reactors.

1.2 Objectives

Motivated by the considerations mentioned above, the broad objectives of this thesis are to:

- Develop a model-free multi-agent reinforcement learning algorithm to control the temperature and concentration of a process network made up of a cascade of two interconnected non-isothermal continuous-stirred tank reactors with recycle.
- Design a reward function for the reinforcement learning agent to achieve the control goals and also satisfy process constraints.
- Implement decentralized MARL, centralized MARL and centralized action and decentralized execution methods, and compare their effectiveness using a simulated process network example.

- Evaluate the robustness of the model-free controllers under parameter uncertainty and sensor noise.
- Highlight the benefits, challenges and possible future directions of research in this area.

Chapter 2

Review of Reinforcement Learning

2.1 Fundamentals of Reinforcement Learning

Fig 2.1 [32] illustrates how a single RL agent interacts with its environment. At time step t, the agent takes an action A_t. This results in a transition in the state of the environment from S_t to S_{t+1} and the reward function generates an immediate reward R_{t+1}. The agent receives observation of the state of the environment, S_{t+1}, and the immediate reward, R_{t+1}, in the next time step. Subsequently, the agent uses a policy function π to map current state s to action a, such that $a = \pi(s)$. The agent takes the next action A_{t+1} and the process is repeated [32].

The value function $V_\pi(s)$ is the expected reward of an agent starting from state s, and subsequently following policy π.

$$V_\pi(s) = \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid S_t = s \right\} \tag{1}$$

where $\gamma \in [0, 1]$ refers to the discount factor that determines the weight of future rewards, E_π denotes the expectation of the value of following the policy π. Correspondingly, the action-value function $Q_\pi(s, a)$ is the expected reward of starting from state s, taking

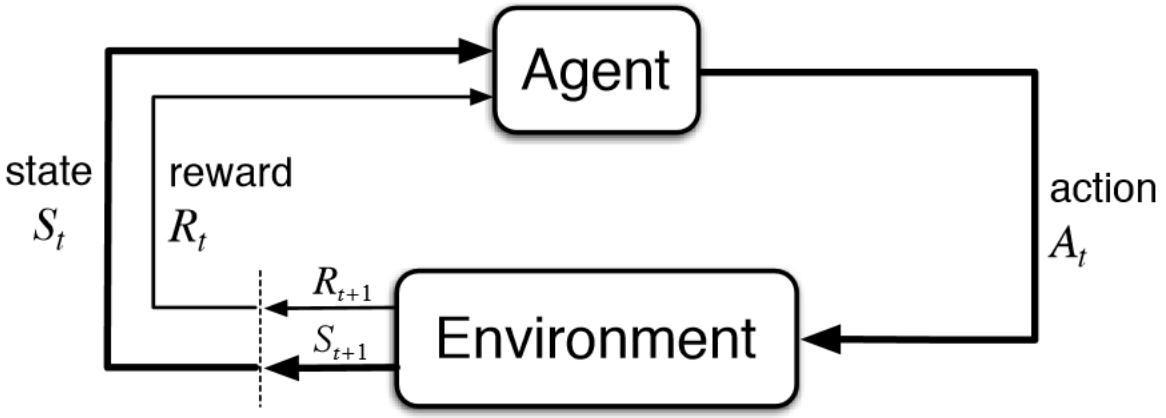

Figure 2.1: Reinforcement Learning framework.

action a and following policy π [32].

$$Q_\pi(s,a) = \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid S_t = s, A_t = a \right\} \tag{2}$$

The objective of the RL agent is to find an optimal policy that maximises the long-term reward. The optimal policy π^* is defined as the policy that maximizes future rewards. π^* can be derived from the optimal value function $V^*(s)$ [32]. This is also known as the optimal Bellman equation:

$$\begin{aligned} V^*(s) &= \max_{a(s)} Q^*(s,a) \\ &= \max_{a(s)} \sum_{s',r} P^a_{ss^r}[r(s,a) + \gamma^*(s')] \end{aligned} \tag{3}$$

where $Q^*(s,a)$ is the optimal action-value function, and the term $P^a_{ss^r}$ is the transitional probability of the environment, which is the probability that the current environment at state s and action a will transition to the next state s' and receive the reward r. The optimal policy $\pi^*(s)$ can be derived from $V^*(s)$ once Eq. 3 is solved.

$$\pi^* = \arg\max_a Q^*(s, a) \tag{4}$$

In practice, the optimal Bellman equation is very difficult to solve. Hence, Neural Networks (NNs) are utilized as function approximators that are trained to estimate the true value function V^* and find the optimal policy function π^*.

2.2 Deep Reinforcement Learning

NNs are a network of neurons arranged in multiple layers and are commonly utilized as function approximators in RL. Deep Neural Networks (DNNs) [24] is a class of NNs that have multiple hidden layers in between the input and output layers. Deep Reinforcement Learning (DRL) [36] is a field in RL that utilizes powerful DNN as function approximators in RL. In the section below, we briefly review the Twin-Delayed Deep Deterministic Policy Gradients (TD3) algorithm which is used in this research.

2.3 Twin-Delayed Deep Deterministic Policy Gradients(TD3) Algorithm

For continuous action spaces, a popular DRL algorithm is the Deep Deterministic Policy Gradient (DDPG) algorithm. DDPG is a model-free, off-policy actor-critic algorithm that can learn deterministic policies in continuous action spaces [16]. An extension of the DDPG, the Twin-Delayed Deep Deterministic Policy Gradient (TD3) algorithm is developed to further address problems with value-function overestimation in DDPG [10]. In Q-learning, the value function is updated based on the Bellman equation. For deterministic policies, the action-value function $Q_\mu(s, a)$ is parameterized by μ can be expressed

using the Bellman equation [16].

$$Q_{\mu}(s_t, a_t) = \mathbb{E}_{s_{t+1}}[r(s_t, a_t) + \gamma Q_{\mu}(s_{t+1}, \mu(s_{t+1}))] \tag{5}$$

The function approximators in the actor and critic networks parameterized by θ^Q can be optimized by minimizing the Loss function $L(\theta^Q)$ [37]:

$$L(\theta^Q) = \mathbb{E}[(Q(s_t, a_t \mid \theta^Q) - y_t)^2] \tag{6}$$

where y_t is known as the target value:

$$y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1}) \mid \theta^Q) \tag{7}$$

The DDPG algorithm introduces the concepts of experience buffer and target networks to achieve stability in learning [16]. The TD3 algorithm is an extension of DDPG. TD3 agents utilizes two target critic networks, denoted by $Q_{\theta_i{'}}$, where $i \in \{1,2\}$, and one actor network π_{ϕ}, parameterized by ϕ'. The target value y_t at a particular time step t is updated by the minimum estimation of the target Q-values to resolve the issue of value function overestimation [10]:

$$y_t = r(s_t, a_t) + \gamma \min_{i=1,2} Q_{\theta_i^I}(s_{t+1}, \pi_{\phi}, (s_{t+1})) \tag{8}$$

At the start of each iteration, the agent takes action a_t based on current policy π from the actor with exploration noise $\mathcal{N}$.

$$a_t = \pi_{\phi}(s_t) + \mathcal{N} \tag{9}$$

The agent will execute action α_t, observe the transition in state from s_t to s_{t+1}, and receive reward r_t. This data is stored in the replay buffer R as a tuple (s_t, a_t, r_t, s_{t+1}). Thereafter, a mini batch of N transition is sampled normally from R. The target value y_t will be computed using Eq. 8. The two critic networks are updated by minimizing the loss function:

$$L(\theta_i) = \frac{1}{N} \sum_j [y_j - Q_{\theta_i}(s_j, \alpha_j)]^2 \tag{10}$$

And the actor network is updated using the policy gradient from the sample of transitions [30]:

$$\nabla_\phi J(\phi) = \frac{1}{N} \sum_j \nabla_a Q_{\theta_1}(s_j, a_j) \nabla_\phi \pi_\phi(s_j) \tag{11}$$

Finally, the parameters of target networks are updated to slowly track the parameters from the learned actor parameter β and critic parameter θ using a slow tracking parameter τ:

$$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i' \tag{12}$$

$$\phi' \leftarrow \tau\phi + (1-\tau)\phi' \tag{13}$$

2.4 Policy Gradient for Multi-Agent Systems

In this section, we review extensions of single-agent policy gradient methods to cooperative multi-agent settings. The research work is built upon the concepts introduced in [5, 11]. We shall distinguish between three paradigms: Centralized Training for Centralized Control (CTCC) vs. Distributed Training for Decentralized Control (DTDC) vs. Centralized Training for Decentralized Control (CTDC), illustrated in Fig. 2.2 [5].

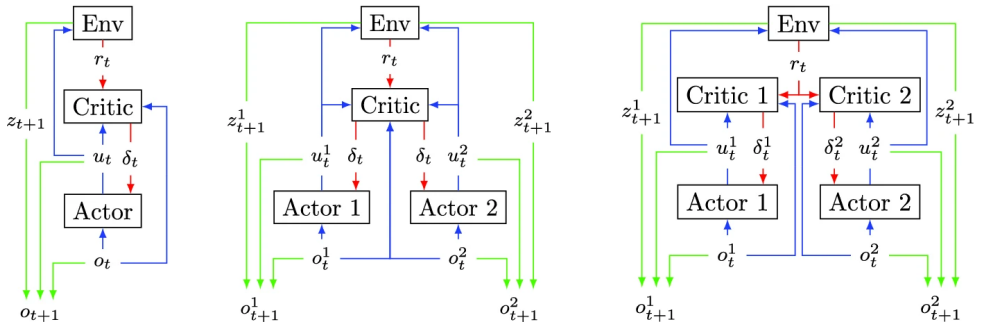

Figure 2.2: For each paradigms—(left) CTCC; (center) CTDC; and (right) DTDC—we describe actor-critic algorithmic schemes. The blue, green and red arrows represent, respectively, the forward control flow; the aggregation of information for the next time step; and the feedback signals back-propagated to update all parameters.

2.5 Centralized Training for Centralized Control (CTCC)

Some cooperative multi-agent applications have cost-free instantaneous communications. Such applications can be modeled as Partially Observable Markov Decision Processes (POMDPs) [15], making it possible to use single-agent policy gradient methods. In such a CTCC paradigm (see Fig. 2.2 (left)), centralized single-agent policy gradient methods use a single critic and a single actor. The major limitation of this paradigm is also its strength: the requirement for instantaneous, free and noiseless communications among all agents till the end of the process both at the training and execution phases.

2.6 Distributed Training for Decentralized Control (DTDC)

Perhaps surprisingly, the earliest multi-agent policy gradient method aims at learning in a distributed manner policies that are to be executed in a decentralized way, e.g., distributed Reinforce [22]. In this DTDC paradigm (see Fig. 2.2 (right)), agents simultaneously, but independently, learn via Reinforce their individual policies using multiple critics and multiple actors. The independence of parameter vectors $\theta^1_{0:T}, \ldots, \theta^n_{0:T}$ leads to the following distributed update rule:

$$\Delta\theta_t^i = \alpha \mathbb{E}_{\mathcal{D}} \left[R(\omega_{0:T}) \frac{\partial \log a_t^i(u_t^i|o_t^i)}{\partial \theta_t^i} \right], \quad \forall t = 0, 1, \ldots, T, \forall i \in I_n \tag{14}$$

Interestingly, the sum of individual policy gradient estimates is an unbiased estimate of the joint policy gradient. However, how to exploit insights from actor-critic methods to combat high-variance in the joint policy gradient estimate remains an open question. Distributed Reinforce restricts to on-policy setting; off-policy methods instead can significantly improve the exploration, i.e., learns target joint policy π while following and obtaining trajectories from behavioral joint policy $\overline{\pi}$ [7].

2.7 Centralized Training for Decentralized Control (CTDC)

The CTDC paradigm has been successfully applied in planning [2, 4, 8] and learning [13, 17, 18]. In such a paradigm, a centralized coordinator agent learns on behalf of all agents at the training phase and then assigns policies to corresponding agents before the execution phase takes place. Actor-critic algorithms in this paradigm (refer to Fig. 2.2 (center)) maintain a centralized critic but learn multiple actors, one for each agent. The critic framework used in this work is individual critics with shared parameters [11].

Chapter 3

Application of Multi-Agent RL to a Process Network

In this chapter, a multi-agent RL control system is applied to the control of two interconnected chemical reactors. Each reactor has two controlled variables, reactor temperature and reactant concentration, which are dependent on the reactor temperature and reactant concentration of the other reactor. The control objective is to maintain each the concentration and temperature of each reactor at a desired set-point using the heat input rate and the inlet reactant concentration as manipulated inputs for the first reactor and using the heat input rate and the inlet reactant concentration as manipulated inputs for the second reactor. A detailed description of the design of RL control agents using the CTCC, DTDC and CTDC frameworks, as well as the reward function design, is provided. The objective of this study is to demonstrate how model-free MARL systems can be trained and implemented to control complex interconnected systems.

3.1 Simulation setup

Consider a plant composed of two well-mixed,non-isothermal continuous stirred-tank reactors (CSTRs) with interconnections [31], where three parallel irreversible elementary exothermic reactions of the form $A \xrightarrow{k_1} B$, $A \xrightarrow{k_2} U$ and $A \xrightarrow{k_3} R$ take place, where A is the reactant species, B is the desired product, and U and R are undesired byproducts. Feed to CSTR 1 consists of two streams, one containing fresh A at flow rate F_0, molar concentration C_{A0}, and temperature T_0; and another containing recycled A from the second reactor at flow rate F_r, molar concentration C_{A2}, and temperature T_2. The feed to CSTR 2 consists of the output of CSTR 1, and an additional fresh stream feeding pure A at flow rate F_3, molar concentration C_{A03}, and temperature T_{03}. The output of CSTR 2 is passed through a separator that removes the products and recycles unreacted A to CSTR 1. Due to the non-isothermal nature of the reactions, a jacket is used to remove or provide heat to both reactors. Under standard modeling assumptions, a plant model of the following form can be derived:

$$\begin{aligned}
\dot{T_1} =& \frac{F_0}{V_1}(T_0 - T_1) + \frac{F_r}{V_1}(T_2 - T_1) + \sum_{i=1}^{3} G_i(T_1)C_{A1} + \frac{Q1}{\rho * C_p * V_1} \\
\dot{C_{A1}} =& \frac{F_0}{V_1}(C_{A0} - C_{A1}) + \frac{F_r}{V_1}(C_{A2} - C_{A1}) - \sum_{i=1}^{3} R_i(T_1)C_{A1} \\
\dot{T_2} =& \frac{F_1}{V_2}(T_1 - T_2) + \frac{F_3}{V_2}(T_03 - T_2) + \sum_{i=1}^{3} G_i(T_2)C_{A2} + \frac{Q2}{\rho * C_p * V_2} \\
\dot{C_{A2}} =& \frac{F_1}{V_2}(C_{A1} - C_{A2}) + \frac{F_3}{V_2}(C_{A03} - C_{A2}) - \sum_{i=1}^{3} R_i(T_2)C_{A2}
\end{aligned} \tag{15}$$

where $R_i(T_j) = k_{io} \exp(-E_i/RT_j)$, $G_i(T_j) = ((-\Delta H_i)/\rho C_p)R_i(T_j)$ for $j = 1, 2$. T_j, C_{Aj}, Q_j, and V_j denote the temperature of the reactor, the concentration of A, the rate of heat input to the reactor, and the reactor volume, respectively, with subscript i denoting the

i-th CSTR. ΔH_i, k_i, E_i, for $i = 1, 2, 3$, denote the enthalpies, pre-exponential constants, and activation energies of the three reactions, respectively; c_p and ρ denote the heat capacity and density of fluid in the reactor, respectively. For typical values of the process parameters as given in Table 3.1, the plant (with $Q1 = 0$, $Q2 = 0$, $C_{A0} = C^s_{A0}$, $C_{A03} = C^s_{A03}$ and a recycle ratio of $r = 0.5$) has three steady states: two locally asymptotically stable, and one unstable. The non-linear system is linearized at one of the steady state operating points, and the MARL algorithm is used to control the plant outputs. Also, while designing the MARL algorithms it is assumed that there is no communication delay in receiving the state information from different units, i.e., the MARL agent receives the state signals continuously without any delay. Below are the detailed guidelines to design and implement the MARL control system for the interconnected CSTR network in series. The CSTR model is created in the MATLAB Simulink environment, and the steady state process variables are validated.

Table 3.1: Process parameter values for the chemical reactors of Eq.15.

$F_0 = 4.998 m^3/hr$
$F_1 = 39.996 m^3/hr$
$F_3 = 30.0 m^3/hr$
$F_r = 34.998 m^3/hr$
$V_1 = 1.0 m^3$
$V_2 = 3.0 m^3$
R = $8.314 kJ/kmolK$
$T_0 = 300.0 K$
$T_{03} = 300.0 K$
$C^s_{A0} = 4.0 kmol/m^3$
$C^s_{A03} = 2.0 kmol/m^3$
$\Delta H_1 = -5.0 * 10^4 KJ/kmol$
$\Delta H_2 = -5.2 * 10^4 KJ/Kmol$
$\Delta H_3 = -5.4 * 10^4 KJ/kmol$
$k_{10} = 3.0 * 10^6 h^-1$
$k_{20} = 3.0 * 10^5 h^-1$
$k_{30} = 3.0 * 10^5 h^-1$
$E_1 = 5.0 * 10^4 KJ/kmol$
$E_2 = 7.53 * 10^4 KJ/kmol$
$E_3 = 7.53 * 10^4 KJ/kmol$
$\rho = 1000.0 kg/m^3$
$C_p = 0.231 KJ/kgK$

3.2 Centralized Training for Centralized Control (CTCC)

The reinforcement learning algorithm that is used for CTCC is called the Twin-Delayed Deep Deterministic Policy Gradient (TD3) algorithm. It is a model-free, online, off-policy reinforcement learning method. A TD3 agent is an actor-critic reinforcement learning agent that searches for an optimal policy that maximizes the expected cumulative long-term reward.

3.2.1 TD3 Agent Design Procedure

To estimate the policy and value function, the TD3 Agent has actor and critic function approximators which are designed using Deep Neural Networks(DNNs).

3.2.2 Design Procedure for Actor and Critic Functions

The Twin-Delayed Deep Deterministic Policy Gradient (TD3) agent uses two parameterized Q-value function approximators to estimate the value (that is the expected cumulative long-term reward) of the policy. To model the parameterized Q-value function within both critics, a neural network with two inputs (the observation and action) and one output (the value of the policy when taking a given action from the state corresponding to a given observation) is designed. Each network path is defined as an array of layer objects. In this work, three layers are defined. A state input layer, an action input layer, and a common path. The three layers are then combined to create the **Critic Net** in each Critic Function. Below is the image of the **Critic Net** which has relu activation function and fully connected layers.

In Fig. 3.1, the left hand side (fc1) has 256 neurons and the layers take the observations

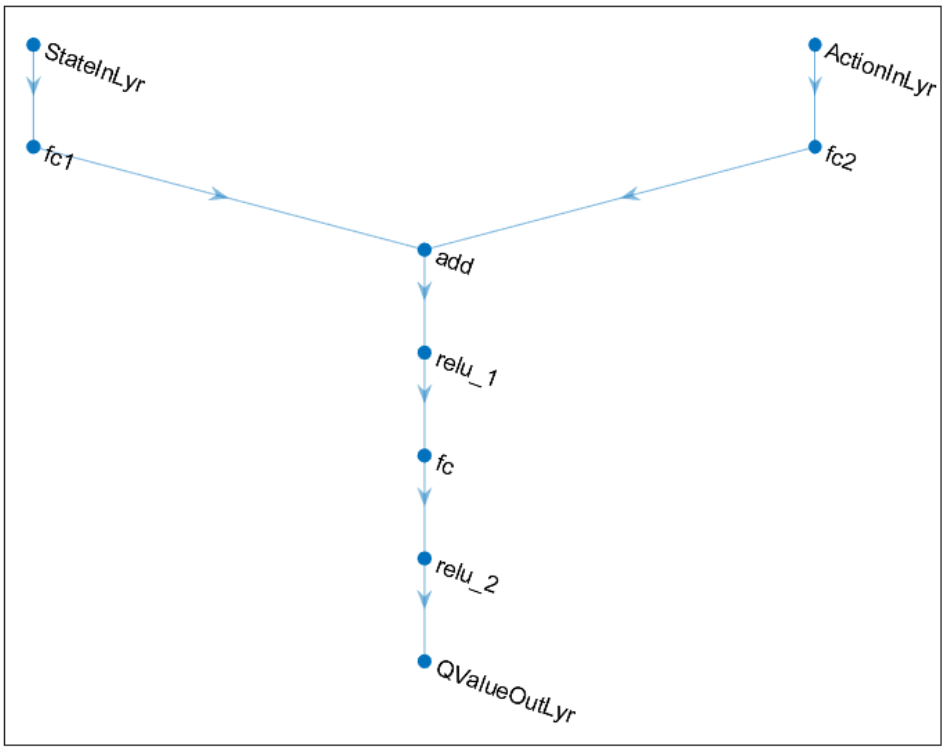

Figure 3.1: Critic Network Architecture.

of the RL agent as input. For CTCC, we have a total of 48 observations. The right hand side (fc2) has 256 neurons and takes four action variables in CTCC as input. These two layers are merged with another layer which has two relu activation functions and two fully connected layers. Table 3.2 provides a detailed description of the number of neurons present in each layer.

Table 3.2: Critic Net Design Information.

Critic Net Layers	Number of Neurons
State input fc-1	256
Action input fc-2	256
Common path fc	256
QValueOut layer	1

Once the **Critic Net** is built, it is randomly initialized using a random seed generator

in MATLAB. **Critic Net** is then used to build **Critic function**. It is built in MATLAB using the following Command:

$$critic1 = rlQValueFunction(initialize(criticDLNet), Observations, Actions)$$

where **rlQvalueFunction** is an object in MATLAB which implements a Q-value function approximator. A Q-value function (also known as action-value function) is a mapping from an environment observation-action pair to the value of a policy. Specifically, its output is a scalar that represents the expected discounted cumulative long-term reward when an agent starts from the state corresponding to the given observation, executes the given action, and keeps on taking actions according to the given policy afterwards. A Q-value function critic, therefore, needs both the environment state (observations) and actions as inputs.

Critic function 2 also shares the same neural network architecture as **Critic function 1**. Although **Critic function 2** can have different design, we choose to keep it the same for simplicity. But it is important to explicitly initialize the network each time (to make sure weights are initialized differently) before passing it to **rlQValueFunction**.

Actor Function:TD3 agents use a parameterized deterministic policy over continuous action spaces, which is learned by a continuous deterministic actor. This actor takes the current observation as input and returns as output an action that is a deterministic function of the observation. To create a parameterized policy within the actor, a neural network with one input layer (which receives the content of the environment observation channel) and one output layer (which returns the action to the environment action channel) with intermediate layers is designed in MATLAB. Below is the figure

of the architecture of the **Actor Network**. Actor Function is created using the Ob-

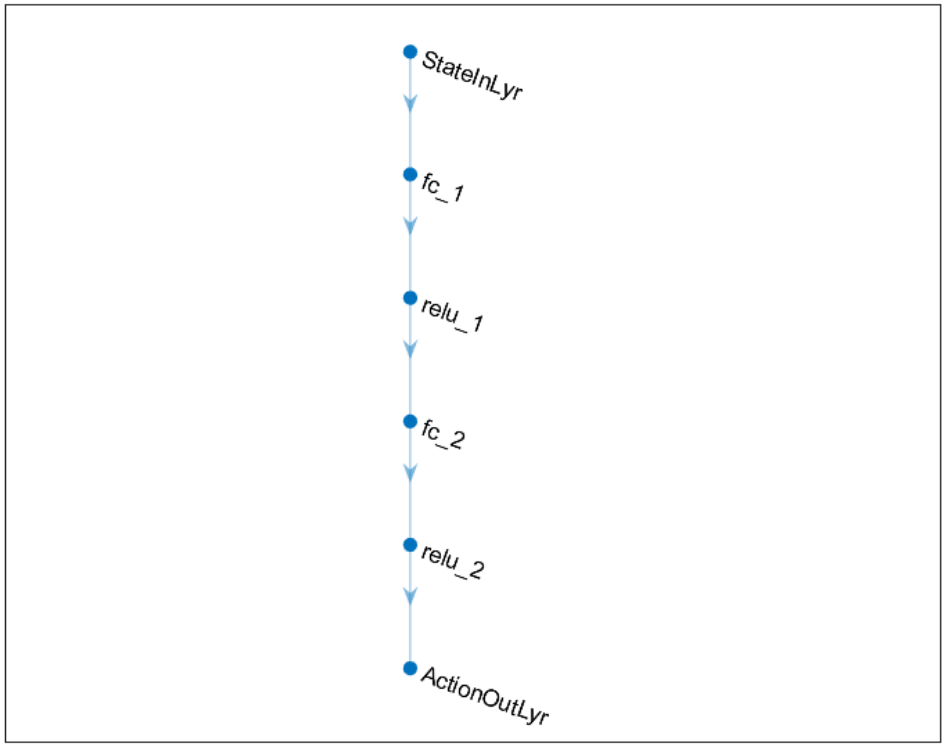

Figure 3.2: Actor Network Architecture.

ject **rlContinuousDeterministicActor** in MATLAB passing the actor network and the environment specifications as input arguments.

$$ActorFunction = rlContinuousDeterministicActor(ActorNetwork, Observations, Actions)$$

The **rlContinuousDeterministicActor** object implements a function approximator to be used as a deterministic actor within a reinforcement learning agent with a continuous action space. A continuous deterministic actor takes an environment observation as input and returns as output an action that is a parameterized deterministic function of the observation, thereby implementing a parameterized deterministic policy. Below is a table

with the number of neurons in each layer. The activation function used for each layer is relu.

Table 3.3: Actor Net Layers Design Information.

Actor Net Layers	Number of Neurons
fc-1	256
fc-2	256
Action Layer	No of Action Outputs(4 in case of CTCC)

3.2.3 Initializing and Optimizing Parameters for TD3 Agent

Table 3.4: TD3 Agent Hyper-parameters.

TD3 Agent Hyper-parameters	Values
Sample Time	0.1
Discount Factor	0.999
Experience BufferLength	10000
Mini BatchSize	64
Number of Steps To Look Ahead	1
Target SmoothFactor	0.005
Target Update Frequency	2

Table 3.5: Critic Hyper-parameters.

Critic Hyper-parameters	Values
Learn rate	10^{-4}
Gradient threshold	1
Optimizer	adam
Denominator offset	10^{-8}
Gradient decay	0.9
Squared gradient decay	0.999
Gradient threshold method	i2norm
L2 regularization factor	0.0001

Both **critics** share the same learn rate and gradient threshold mentioned in Table 3.5. The learn rate is chosen as 10^{-4} to improve accuracy. Smaller values to improve the accuracy can be chosen, but it takes too much computational resources. There is a trade off between accuracy and computational time. The RL agent is trained on NVIDIA MX150 GPU for 2 hours.

In CTCC, a single RL agent is used to control both reactors. The RL agent has the

Table 3.6: Actor Hyper-parameters.

Actor Parameters	Values
Learn Rate	10^{-4}
Gradient Threshold	1
Optimizer	adam
Denominator offset	10^{-8}
Gradient decay	0.9
Squared gradient decay	0.999
Gradient threshold method	i2norm
L2 Regularization Factor	0.001
Exploration Model Variance	0.05
Exploration Model Variance Decay Rate	$1 * 10^{-5}$
Exploration Model Variance Min	0.001
Target Policy Smooth Model Variance	0.1
Target Policy Smooth Model Variance Decay Rate	10^{-4}

following observations and reward structure.

3.2.4 Observation Space

The observations to the RL Agent form the simulation Environment consists of:

Table 3.7: CTCC TD3 RL Agent Observation States.

	Quantity	Purpose
e	Proportional error	Accounts for present error value
$\int e\,dt$	Integral error	Accounts for cumulative error
T_1	Reactor 1 Temperature	To perceive the state
C_{A1}	Reactor 1 Concentration	To perceive the state
T_2	Reactor 2 Temperature	To perceive the state
C_{A2}	Reactor 2 Concentration	To perceive the state
T_1SP	Reactor 1 Temperature Set point (SP)	To perceive the current SP
C_{A1} SP	Reactor 1 Concentration Set point (SP)	To perceive the current SP
T_2 SP	Reactor 2 Temperature Set point (SP)	To perceive the current SP
C_{A2} SP	Reactor 2 Concentration Set point(SP)	To perceive the current SP

For both reactors, the past five concentration and temperature measurements and set-points are provided so that the agent has knowledge of the evolution of the corresponding Variables. The total number of states in the observation space in the CTCC framework are 48. As the number of units increases, the number of observation states increase exponentially leading to the curse of dimensionality and increasing computational resources. This factor is a major limitation in applying a centralized control strategy for a large

number of units.

3.2.5 Reward Functions

The main objective of the reward function is to guide the reinforcement learning algorithm to achieve the desired output. Safety is a critical factor in the operation of chemical reactors and other unit operations in the Chemical Industry. Sparse rewards like those mentioned in [39] are not feasible from a practical viewpoint. As unconstrained exploration by RL algorithms might lead to huge computational resources, equipment loss and loss of life. Taking this into account, we define the reward function as a constrained optimization problem and impose a high negative reward on the agent when its actions go beyond the design criteria in Table 3.8. In this work, a quadratic loss function, which falls under Exterior Penalty methods, is defined based on initial value, desired final value, rise time, settling time, percent overshoot, percent rise, percent settling and percent undershoot. These are the desired parameters chosen for the temperature of CSTR 1. For all other controlled variables, only the final value (set-point) is changed. All the other parameter values are fixed in this research work. These can be modified as desired for each controlled variable.

Table 3.8: Control Objectives for the Temperature of CSTR 1.

step time (seconds) = 0	
Intial value = 0	Final value = 1.2
Rise time(seconds) = 2	% Rise = 80
Settling time(seconds): = 5	% settling = 2
% Overshoot = 10	%Undershoot = 5

3.2.6 Quadratic Penalty Function

The key significance of using penalty functions in reward functions is that they "absorb" the equality and inequality constraints into a penalizing term that augments the original objective function, along with weighting parameters, to enforce increasing satisfaction of the constraints involved. This ensures that the RL agent receives huge negative penalty as long as the desired outcome is outside the parameters described in the above table and their effectiveness in solving nonlinear programming problems. The algorithm is developed based on the work in [9] which contains all the theorems and proofs for penalty and barrier functions. The reward function is defined as:

$$R(x,t) = -Weight * Penalty$$

where x is the desired value of the Controlled variable and t is the time where the controller achieves the desired set-point. Weight is a tunable parameter, which is chosen to be 10 for the purpose of our study. The same framework is used to design the reward functions for the other three Controlled Variables. For CTCC, the reward is the sum of the individual reward functions, i.e.,

$$R = \sum_{i=1}^{4} R_i(x_i, t)$$

The source code for the algorithm below is adapted from the MATLAB Quadratic Penalty documentation and is modified according to the current research problem.

Algorithm 1 Quadratic Penalty Based Reward Function Algorithm.

$IntialValue \leftarrow UserDefined$
$FinalValue \leftarrow UserDefined$
$StepTime \leftarrow 0$
$StepRange = FinalValue - InitialValue$
$MinRise = InitialValue + (StepRange * \%Rise)$
$MaxSettling = InitialValue + StepRange * (IntialValue + FinalValue/100)$
$MinSettling = InitialValue + StepRange * (IntialValue - FinalValue/100)$
$MaxOvershoot = InitialValue + StepRange * (1 + \%Overshoot)$
$MinUndershoot = InitialValue - StepRange * \%Undershoot$
if $t \geq$ steptime **then**
  if Intialvalue $\leq$ Finalvalue **then**
    $UpperBoundTimes \leftarrow [0, 5; 5, max(5 + 1, t + 1)]$;
    $UpperBoundAmplitudes \leftarrow [MaxOvershoot, MaxOvershoot; MaxSettling, MaxSettling]$
    $LowerBoundTimes = [0, 2; 2, 5; 5, max(5 + 1, t + 1)]$

    $LowerBoundAmplitudes = [MinUndershoot, MinUndershoot; MinRise, Block1_MinRise; MinSettling, MinSettling]$
  else
    $UpperBoundTimes \leftarrow [0, 2; 2, 5; 5, max(5 + 1, t + 1)]$;

    $UpperBoundAmplitudes \leftarrow [MinUndershoot, MinUndershoot; MinRise, MinRise; MinSettling, MinSettling]$
    $LowerBoundTimes \leftarrow [0, 5; 5, max(5 + 1, t + 1)]$;
    $LowerBoundAmplitudes \leftarrow [MaxOvershoot, MaxOvershoot; MaxSettling, MaxSettling]$
  end if$xmax = zeros(1, size(UpperBoundTimes, 1))$
  for $idx = 1 : size(xmax)$ **do**
    $tseg \leftarrow UpperBoundTimes(idx, :)$
    $xseg \leftarrow UpperBoundAmplitudes(idx, :)$
    $xmax(idx) \leftarrow interp1(tseg, xseg, t, 'linear', NaN)$
  end for
  if $all(isnan(xmax))xmax \leftarrow Inf$ **then**
    $xmax = Inf$;
  else
    $xmax \leftarrow max(xmax, [], 'omitnan')$
  end if$xmin = zeros(1, size(LowerBoundTimes, 1))$
  for $xmin = zeros(1, size(LowerBoundTimes, 1))$ **do**
    $tseg \leftarrow LowerBoundTimes(idx, :)$
    $xseg \leftarrow LowerBoundAmplitudes(idx, :)$
    $Block1_xmin(idx) \leftarrow interp1(tseg, xseg, t, 'linear', NaN)$
  end for
  if $all(isnan(Block1_xmin))$ **then**
    $xmin = -Inf$
  else
    $xmin = max(xmin, [], 'omitnan')$
  end if
else
  $xmin = -Inf$
  $xmax = Inf$;
  $Weight \leftarrow 10$
  $Penalty \leftarrow sum(exteriorPenalty(x, xmin, xmax, 'Quadratic'))$
  $reward = -Weight * Penalty$
end if

3.2.7 Results

Below are the set-point tracking plots for the temperature and concentration of CSTR 1 and CSTR 2 using CTCC.

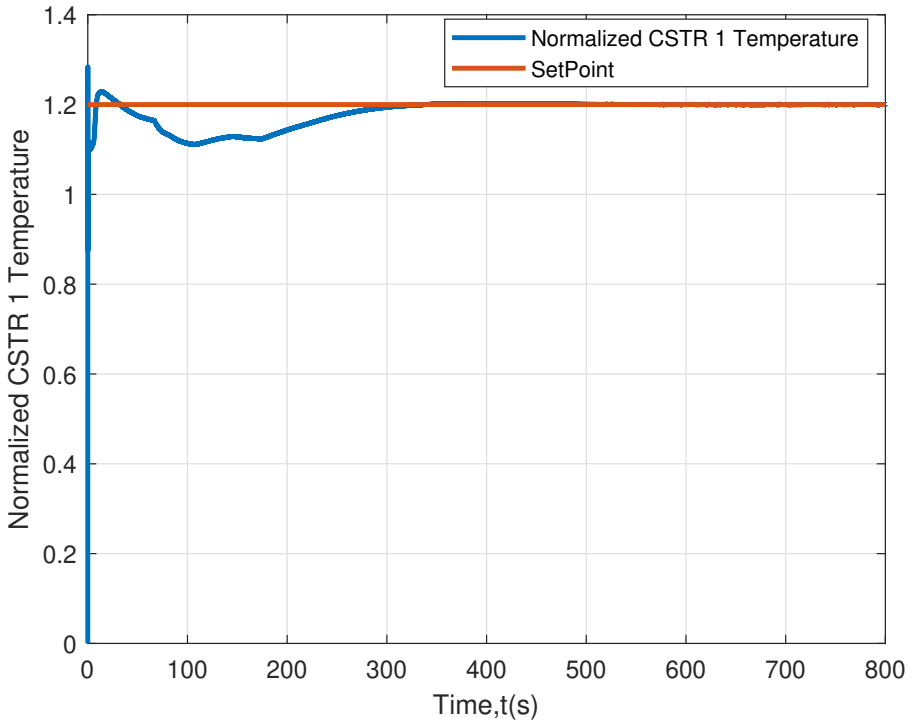

Figure 3.3: Temperature set-point tracking for CSTR 1 using CTCC.

The controller is able to reach the desired temperature set-point for CSTR 1. The offset is minimal with error less than 1%. Although the overshoot is within the design limit of 10%, the controller was unable to maintain the undershoot under 5% as it can be observed from Fig. 3.3. The design goal is to reach 80% of the set-point under 25 seconds. Although this has been achieved, the settling time took much longer to achieve.

While the controller is able to achieve the desired concentration set-point for CSTR 1 as can be seen from Fig. 3.4, it took a longer time to settle and the undershoot was also significantly higher than the design goal of 5%.

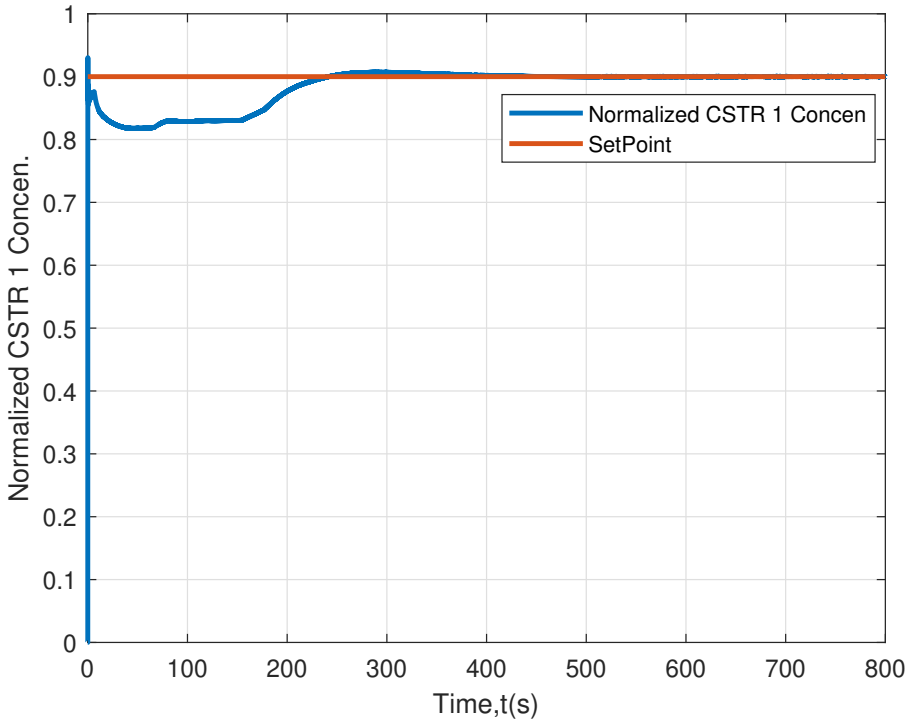

Figure 3.4: Concentration set-point tracking for CSTR 1 using CTCC.

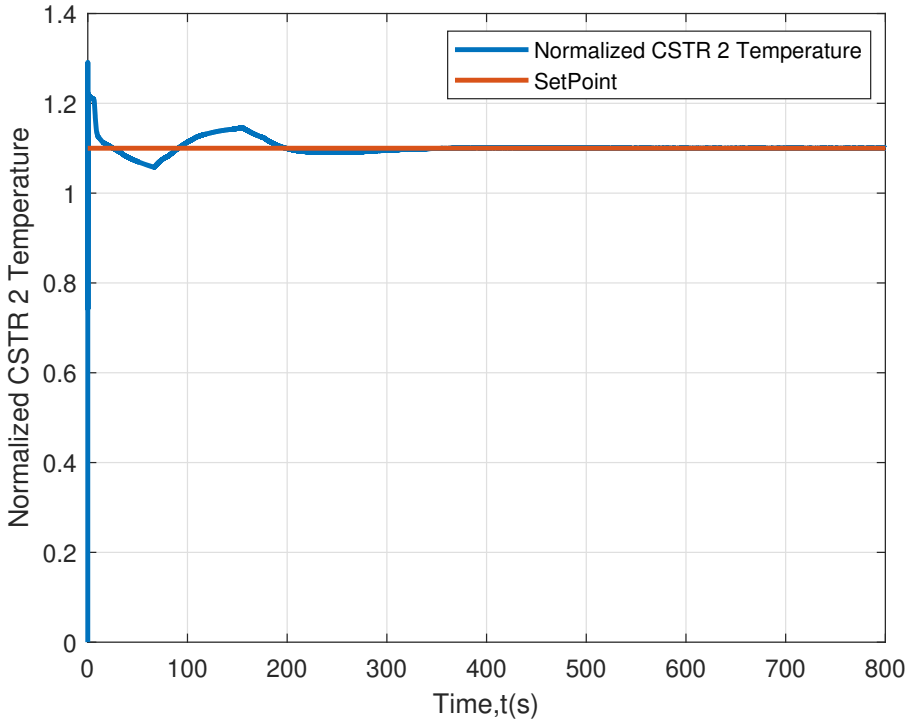

Figure 3.5: Temperature set-point tracking for CSTR 2 using CTCC.

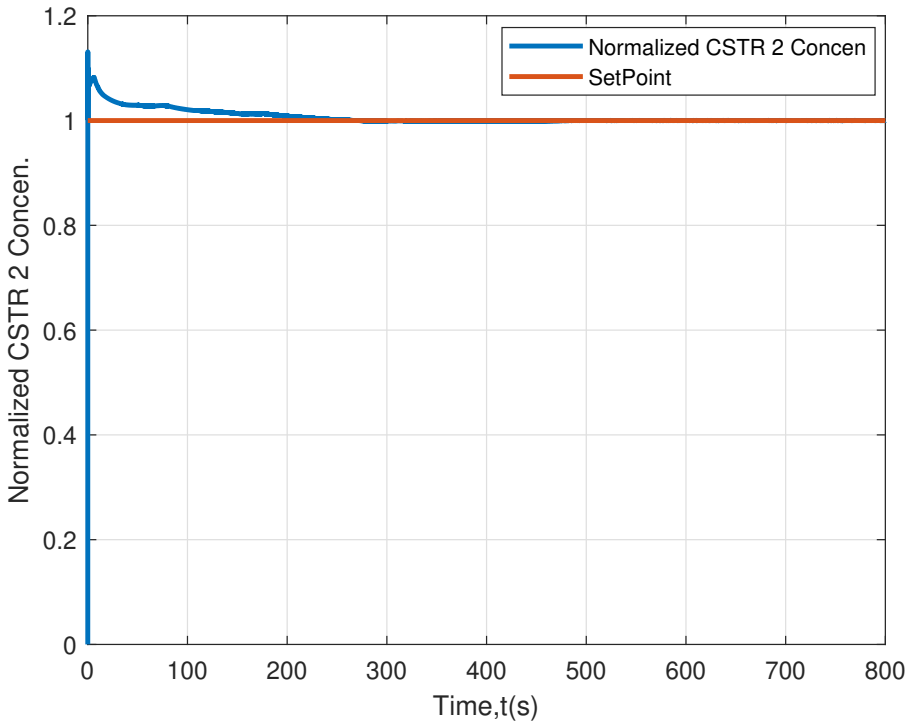

Figure 3.6: Concentration set-point tracking for CSTR 2 using CTCC.

For the temperature control of CSTR 2, the results are similar to the those for CSTR 1 as seen in Fig. 3.5 in that the settling time is longer than the design objective. Also, the overshoot is above 10% for the temperature response of CSTR 2 and the undershoot remains within the target range of 5%.

Figure 3.6 shows the response of the concentration for CSTR 2. Although the set-point tracking objective has been met and the error is less than 1%, the settling time is longer than the design goal and the controller overshoot is also higher than the 10% design objective. Figures 3.7 and 3.8 show the corresponding evolution of the manipulated variables for the two reactors.

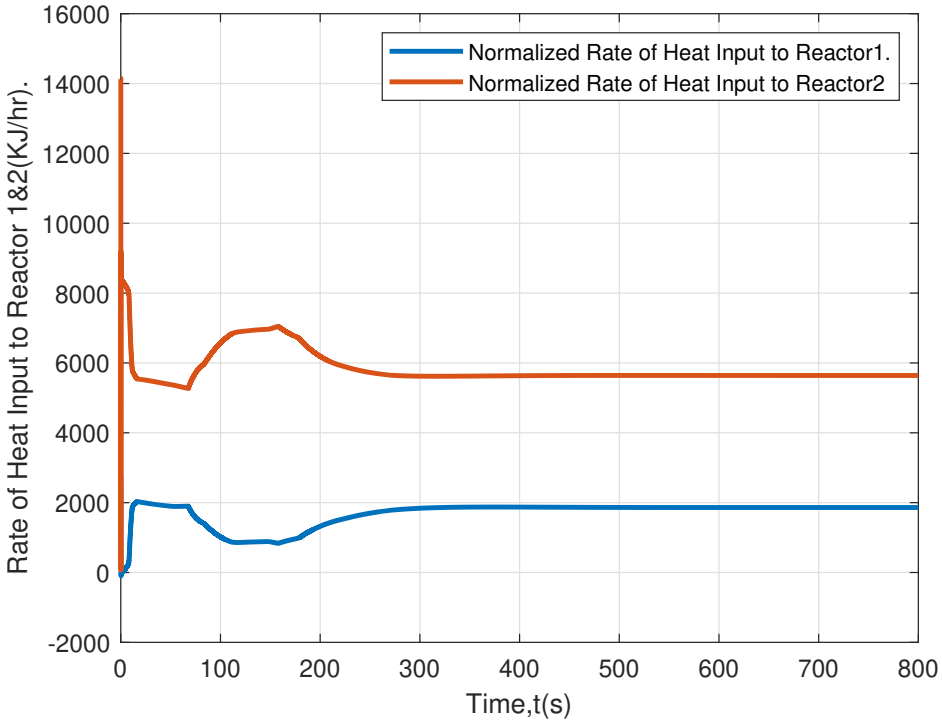

Figure 3.7: Rates of heat input for CSTR 1 and CSTR 2 under CTCC.

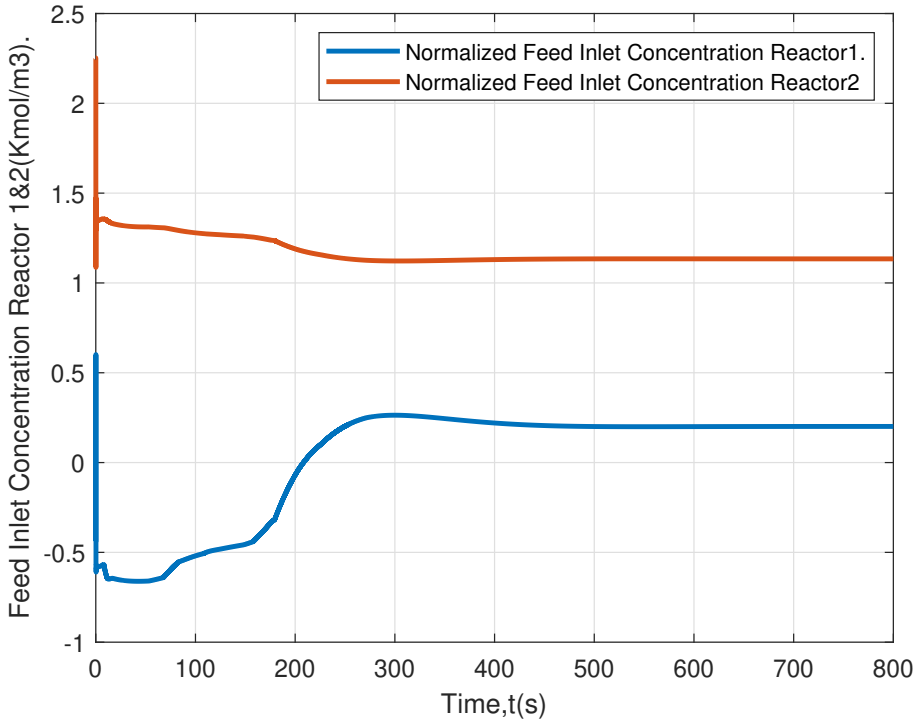

Figure 3.8: Inlet reactant concentrations for CSTR 1 and CSTR 2 under CTCC.

3.2.8 Parameter Variations

The TD3 Agent is trained at the operating conditions as stated in Table 3.1. In practical applications, disturbances are common and the efficiency of the equipment also gradually deteriorates due to wear and tear, fouling and other factors. To test the robustness of the model-free controller, the following parameters are chosen upon careful consideration based on practical considerations, such as fluctuations in flow rates, input feed concentrations, and deactivation of catalyst, and are varied as stated in Table 3.9. The new operating conditions are updated in the simulation model and the pre-trained RL CTCC control agent is used in the updated model to test set-point tracking. Note that, instead of introducing uncertainties in the middle of the simulation at a certain time, here the model parameters are changed.

Table 3.9: Details of Parameter Variations.

Parameter	Description	Variation
T_0	Temperature of Feed A to CSTR 1	10%
T_03	Temperature of Feed A to CSTR 2	10%
Ca_0	Concentration of Feed A to CSTR 1	10%
Ca_{03}	Concentration of Feed A to CSTR 2	10%
E_1	Activation energy of Reaction 1	40%
E_2	Activation energy of Reaction 2	40%
E_3	Activation energy of Reaction 3	40%
F_0	Fresh A Feed flow to CSTR 1	10%
F_r	Recycled A flow from CSTR 2 to CSTR 1	14%
F_3	Fresh A Feed flow to CSTR 2	10%

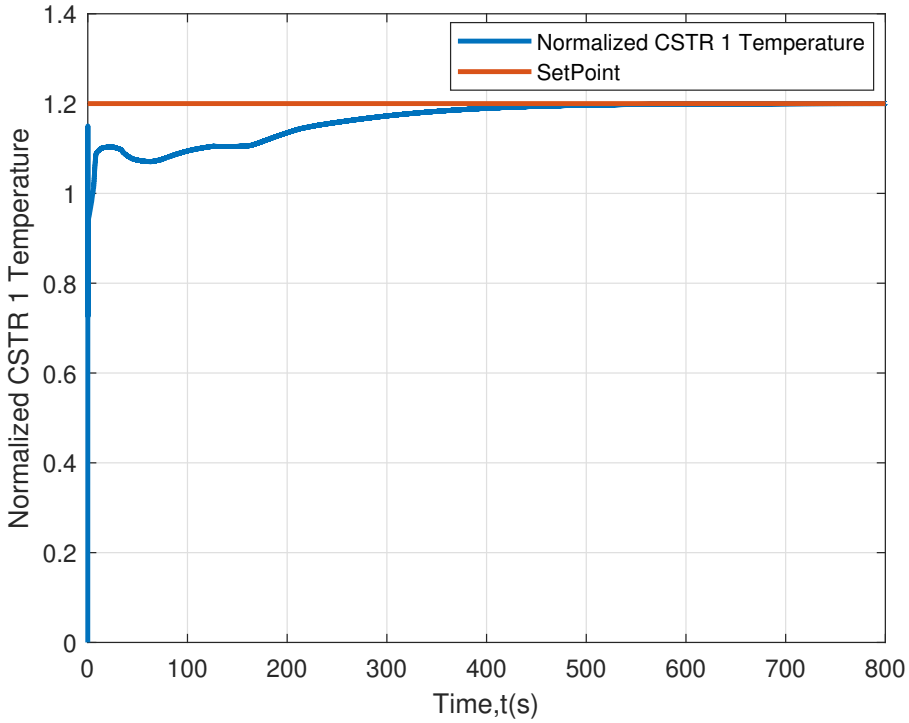

Figure 3.9: Temperature set-point tracking for CSTR 1 under CTCC and parametric variations.

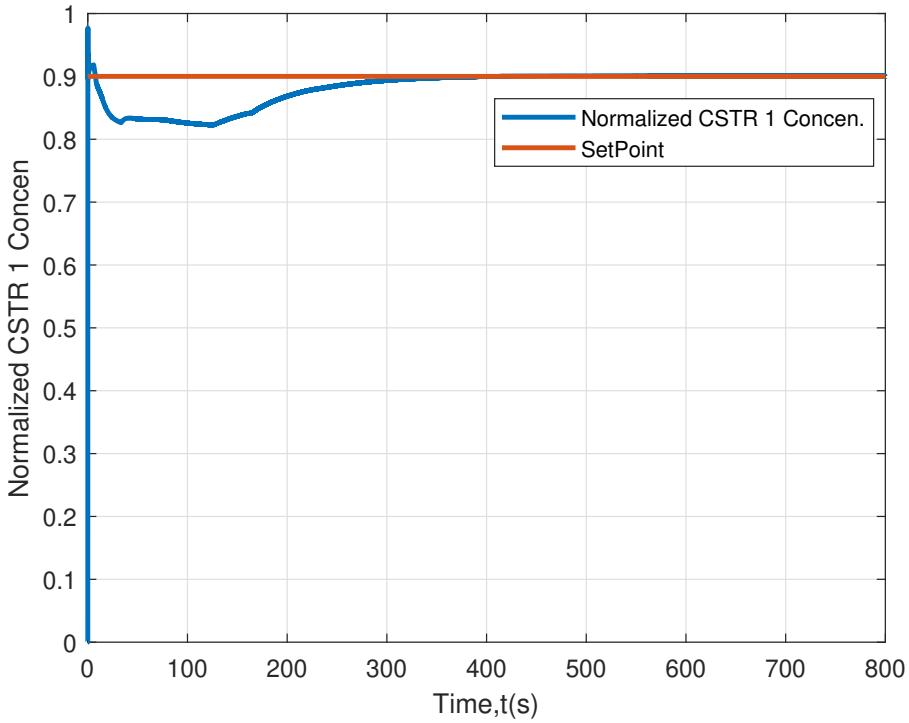

Figure 3.10: Concentration set-point tracking for CSTR 1 under CTCC and parametric variations.

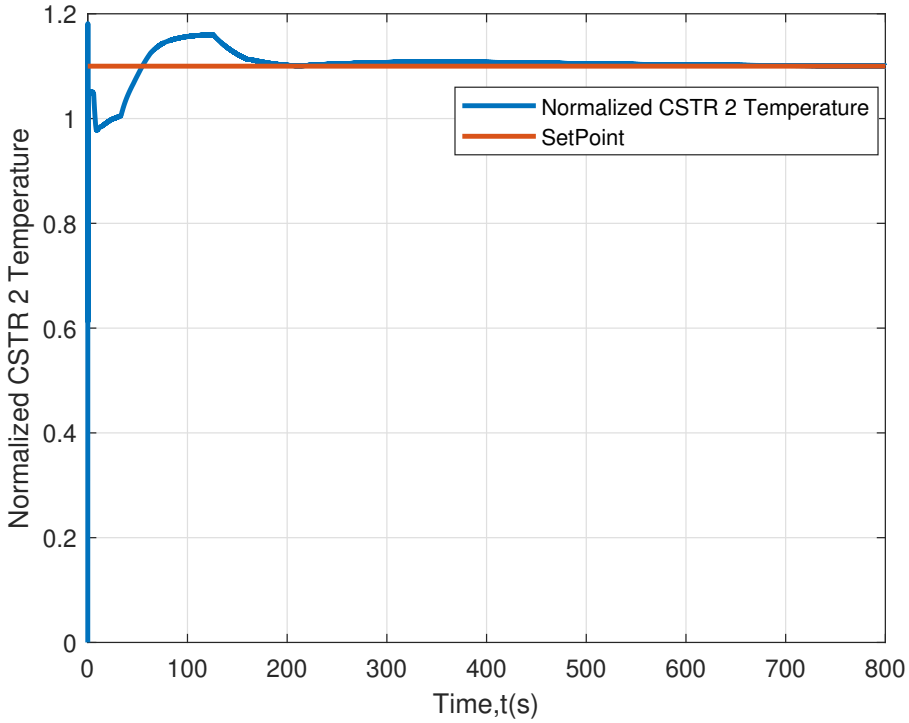

Figure 3.11: Temperature set-point tracking for CSTR 2 under CTCC and parametric variations.

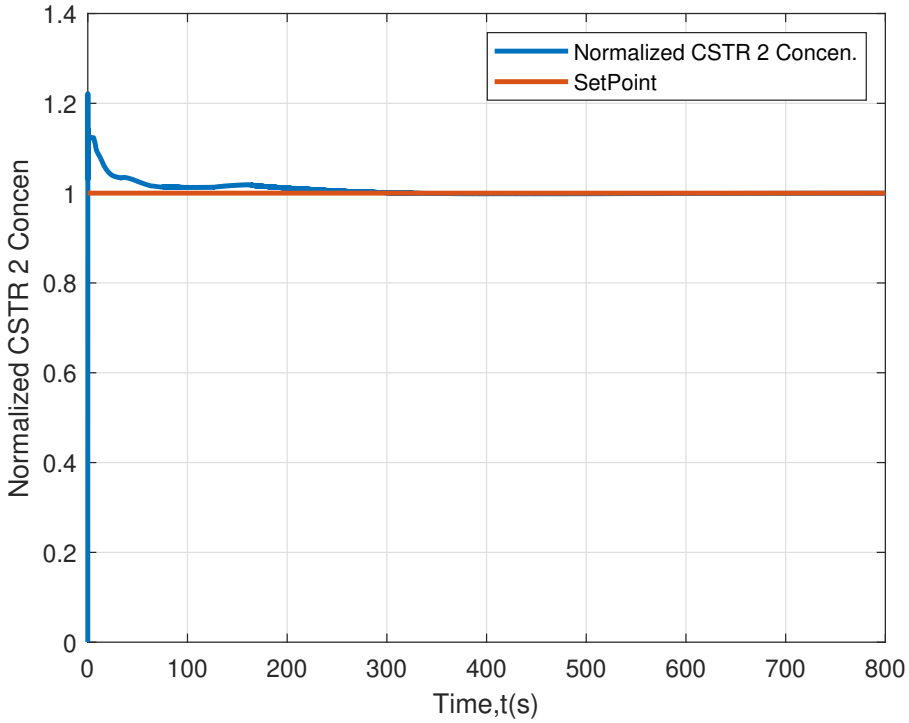

Figure 3.12: Concentration set-point tracking for CSTR 2 under CTCC and parametric variations.

Parameter uncertainty has the most effect on temperature control of CSTR 1. Examining Figs. 3.9-3.12, it can be seen that introducing uncertainties slowed down the temperature response time but there is no offset in reaching the desired set-point. For the other controlled variables, parameter variations have no significant impact on the controller response.

3.2.9 Effect of Sensor Noise

For each sensor, Gaussian random noise with a mean of 0.01 and variance of 10^{-5} was introduced to evaluate the controller performance with respect to random noise. The plots below show that the controller performance is robust in the presence of the sensor noise considered.

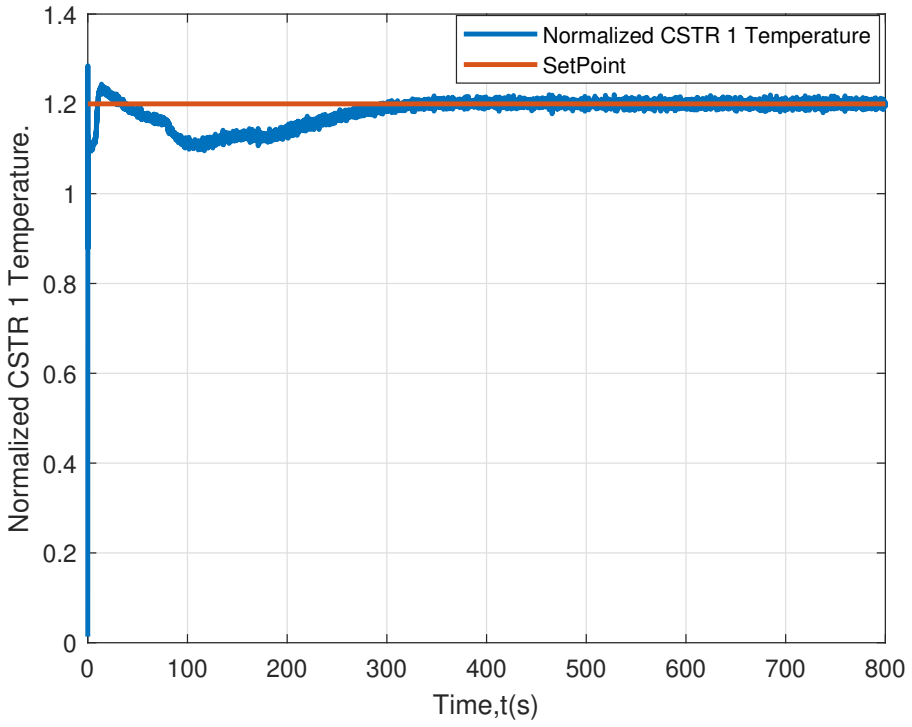

Figure 3.13: Temperature set-point tracking for CSTR 1 under CTCC and sensor noise.

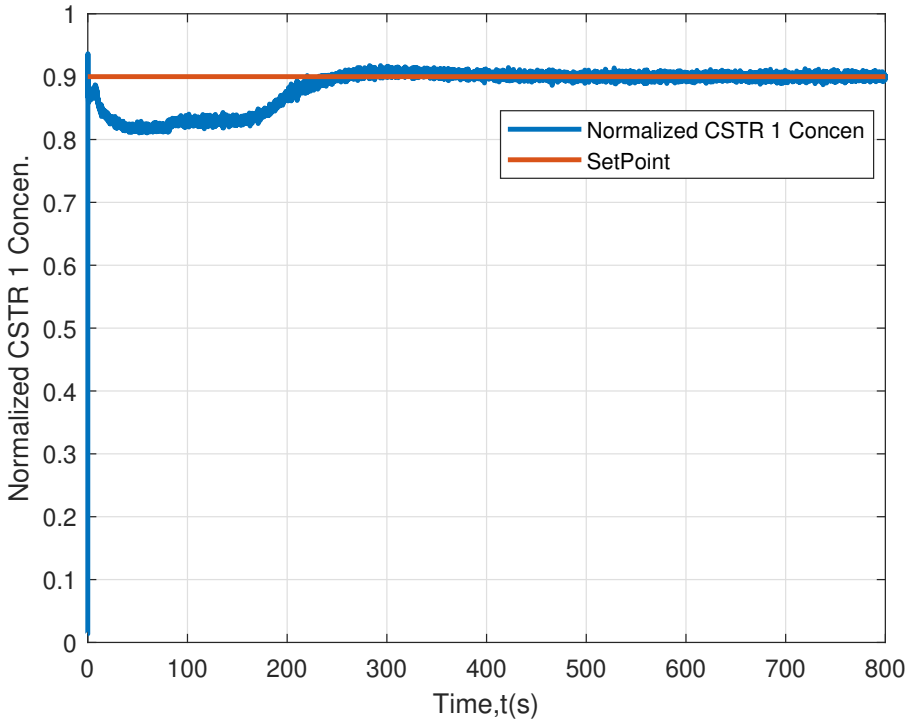

Figure 3.14: Concentration set-point tracking for CSTR 1 under CTCC and sensor noise.

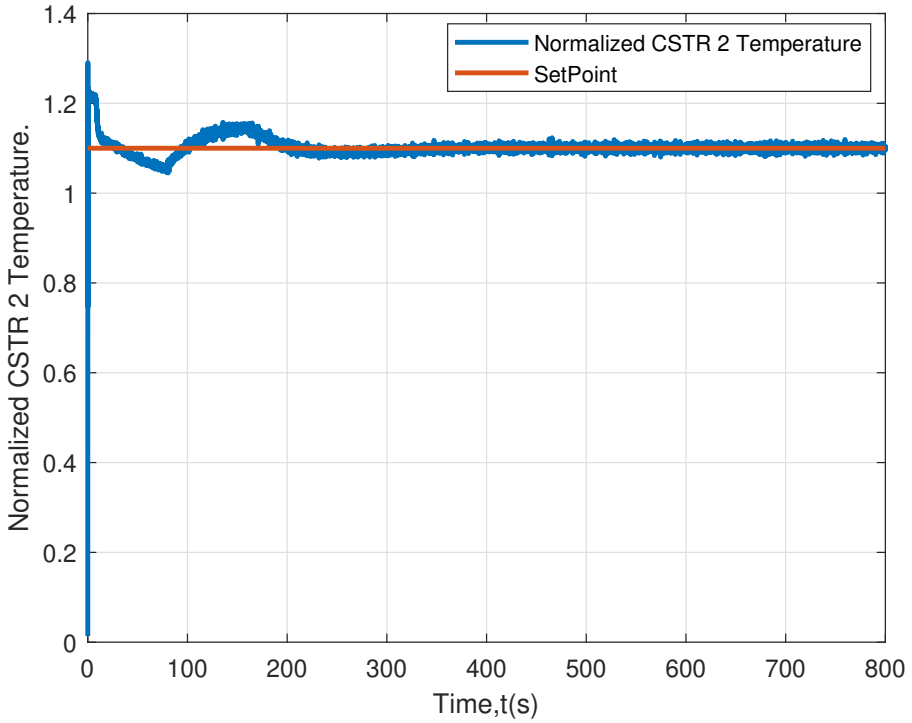

Figure 3.15: Temperature set-point tracking for CSTR 2 under CTCC and sensor noise.

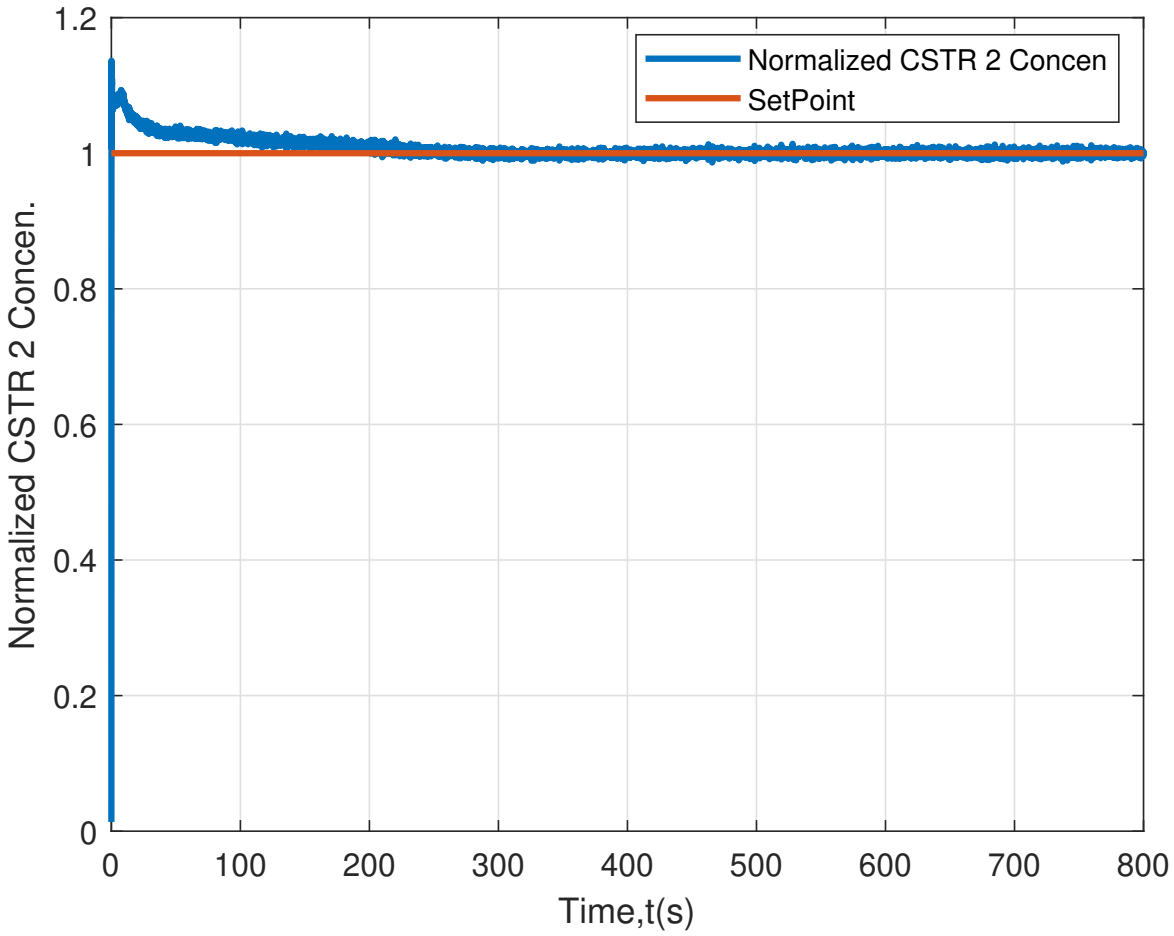

Figure 3.16: Concentration set-point tracking for CSTR 2 under CTCC and sensor noise.

From the results in Figs. 3.13-3.16, the controller performance appears to be robust to the sensor noise considered. Although there is some noise in the controller output, both the concentration and temperature set-points for CSTR 1 and CSTR 2 are still achieved without any deterioration in controller performance with respect to the design objectives. It should be noted here that sensor noise was not considered while training the RL agent.

3.3 Distributed Training for Decentralized Control (DTDC)

In the DTDC framework, each controlled variable is controlled by an independent RL agent with observation space having variables from that subsystem. Each agent has partial

observability of the plant environment, takes independent actions with the objective of maximizing their own reward without considering the impact of their actions on other agents. This can lead to convergence issues, i.e., the controller not able to reach the desired goal. Unlike the CTCC framework, where a single RL agent controls all four controlled variables, four decentralized individual RL agents are designed to control the four controlled variables with each corresponding to a single control variable.

3.3.1 TD3 Agent Design Architecture

The TD3 agents are designed based on the same framework used in the CTCC control framework with some modifications in the **Actor Network** and **Critic Network** architectures, as well as tuning of hyper-parameters. Below are the images of the **Critic** and **Actor Network** architectures used in each TD3 Agent.

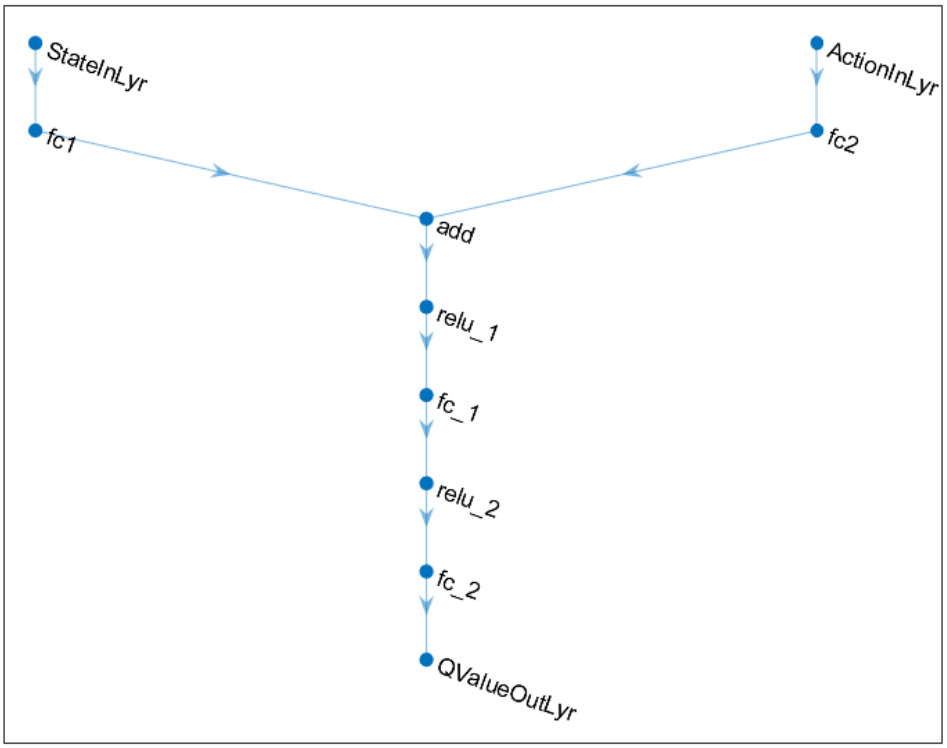

Figure 3.17: Critic Network Architecture.

Table 3.10: Critic Net Design Information.

Critic Net Layers	Number of Neurons
State input fc-1	256
Action input fc-2	256
common path fc-1	256
common path fc-1	256
QValueOut layer	1

Table 3.11: Actor Net Layers Design Information.

Actor Net Layers	Number of Neurons
fc-1	256
fc-2	256
fc-3	256
Action Layer(tanhLayer)	No. of Action Outputs (1 in case of DTDC)

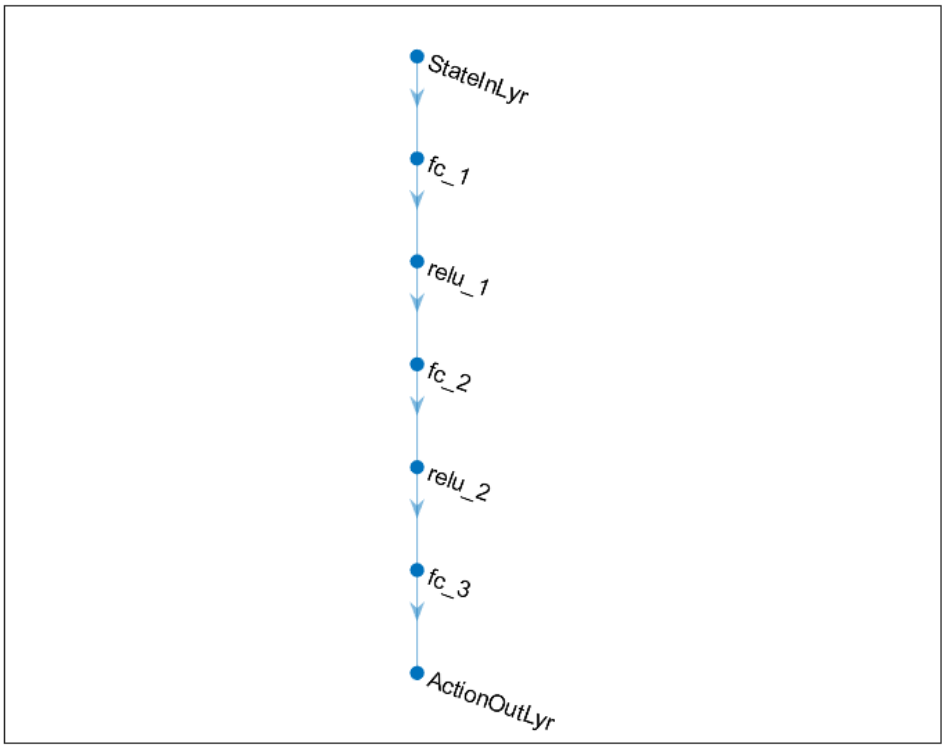

Figure 3.18: Actor Network Architecture.

3.3.2 TD3 Agent Hyper-parameters

Table 3.12: TD3 Agent Hyper-parameters.

TD3 Agent Hyper-parameters	Values
Sample Time	0.1
Discount Factor	0.999
Experience BufferLength	$2 * 10^6$
Mini BatchSize	512
Number of Steps To Look Ahead	1
Target SmoothFactor	0.005
Target Update Frequency	10

Table 3.13: Critic Hyper-parameters.

Critic Hyper-parameters	Values
Learn rate	10^{-4}
Gradient threshold	1
Optimizer	adam
Denominator offset	10^{-8}
Gradient decay	0.9
Squared gradient decay	0.999
Gradient threshold method	i2norm
L2 regularization factor	0.0001

Table 3.14: Actor Hyper-parameters.

Actor Parameters	Values
Learn rate	10^{-4}
Gradient threshold	1
Optimizer	adam
Denominator offset	10^{-8}
Gradient decay	0.9
Squared gradient decay	0.999
Gradient threshold method	i2norm
L2 regularization factor	0.001
Exploration model variance	0.05
Exploration model variance decay rate	$2 * 10^{-4}$
Exploration vodel variance min.	0.001
Target policy smooth model variance	0.1
Target policy smooth model variance decay rate	10^{-4}

3.3.3 Observation Space

There are total four TD3 agents with each agent having their own local observations. The observations to the RL agent from the simulation environment consist of the error, the integral error of the controlled process variable, the current state of the control variable and its past four states, and the current set-point and past four values (the set-point is varied randomly to prevent over fitting during training).

Table 3.15: TD3 Agent 1 Observation Space for CSTR 1 Temperature Control.

Process parameter	Quantity	Purpose
e	Proportional error	Accounts for present error value
$\int e\,dt$	Integral error	Accounts for cumulative error
T_1	Reactor 1 Temperature	To perceive the state
T_1 SP	Reactor 1 Temperature Set point(SP)	To perceive the SP

Table 3.16: TD3 Agent 2 Observation Space for CSTR 1 Concentration Control.

	Quantity	Purpose
e	Proportional error	Accounts for present error value
$\int e\,dt$	Integral error	Accounts for cumulative error
C_{A1}	Reactor 1 Concentration	To perceive the state
C_{A1} SP	Reactor 1 Concentration Set-point (SP)	To perceive the SP

Table 3.17: TD3 Agent 3 Observation Space for CSTR 2 Temperature Control.

	Quantity	Purpose
e	Proportional error	Accounts for present error value
$\int e\,dt$	Integral error	Accounts for cumulative error
T_2	Reactor 2 Temperature	To perceive the state
T_2 SP	Reactor 2 Temperature Set-point (SP)	To perceive the SP

Table 3.18: TD3 Agent 4 Observation Space for CSTR 2 Concentration Control.

	Quantity	Purpose
e	Proportional error	Accounts for present error value
$\int e\,dt$	Integral error	Accounts for cumulative error
C_{A2}	Reactor 1 Concentration	To perceive the state
C_{A2} SP	Reactor 1 Concentration Set point(SP)	To perceive the SP

3.3.4 Reward Functions

The reward function framework and objective is similar as described earlier in the CTCC framework. The main difference here is that each RL agent has its own reward function with the same design goals as in CTCC.

3.3.5 Results

Compared to the CTCC framework, the DTDC control strategy yields superior controller performance with respect to the speed of the response time, the %undershoot and %overshoot. Below are the plots for the temperature and concentration set-point tracking for CSTR 1 and CSTR 2, respectively. It should be noted that there is an offset of about 10% divergence from the set-point for temperature control of CSTR 2. This is due to the convergence issue with the DTDC algorithm, as the RL agents are completely decentralized with no information exchange between them. The individual agents are unaware

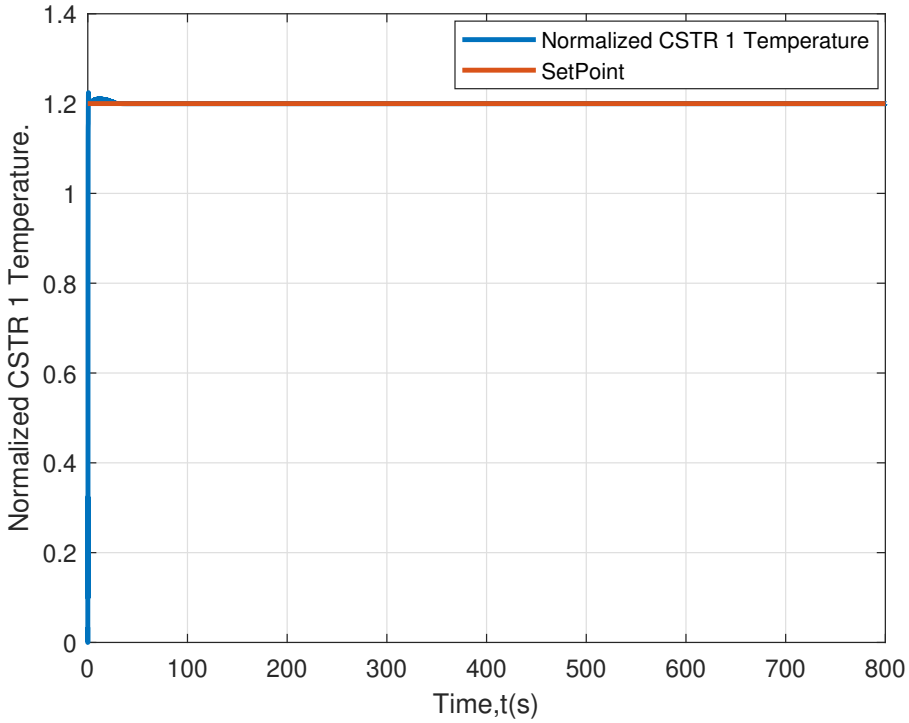

Figure 3.19: Temperature set-point tracking for CSTR 1 under DTDC.

of the other agents' actions and, since in an interconnected system variation in one controlled variable affects the other variables, there is a convergence issue due to the partial observability of states and non-stationary environment.

As observed from Fig. 3.19, the DTDC controller was able to achieve set-point tracking for the temperature of CSTR 1 with less than 2% error. The settling time, the rise time, the % overshoot and % undershoot also met the design goals.

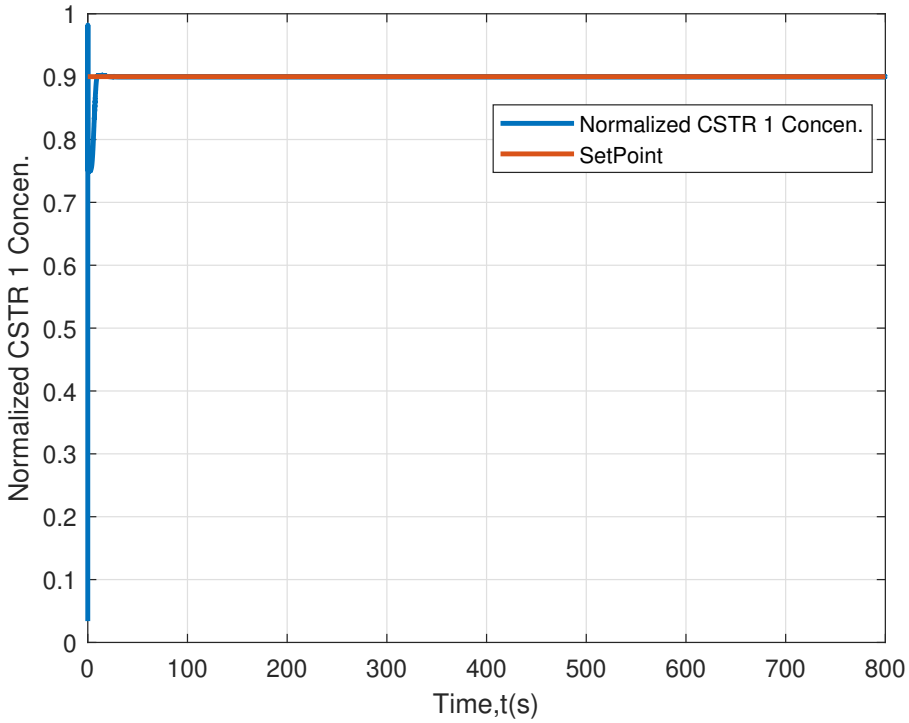

Figure 3.20: Concentration set-point tracking for CSTR 1 under DTDC.

From Fig. 3.20, it can be seen that the controller was able to successfully achieve the desired set-point for the reactant concentration for CSTR 1. From the plot, the set-point tracking error is less than 2%. The rise time is slower compared to the other controlled variables. Except for the rise time, all the other design goals have been met.

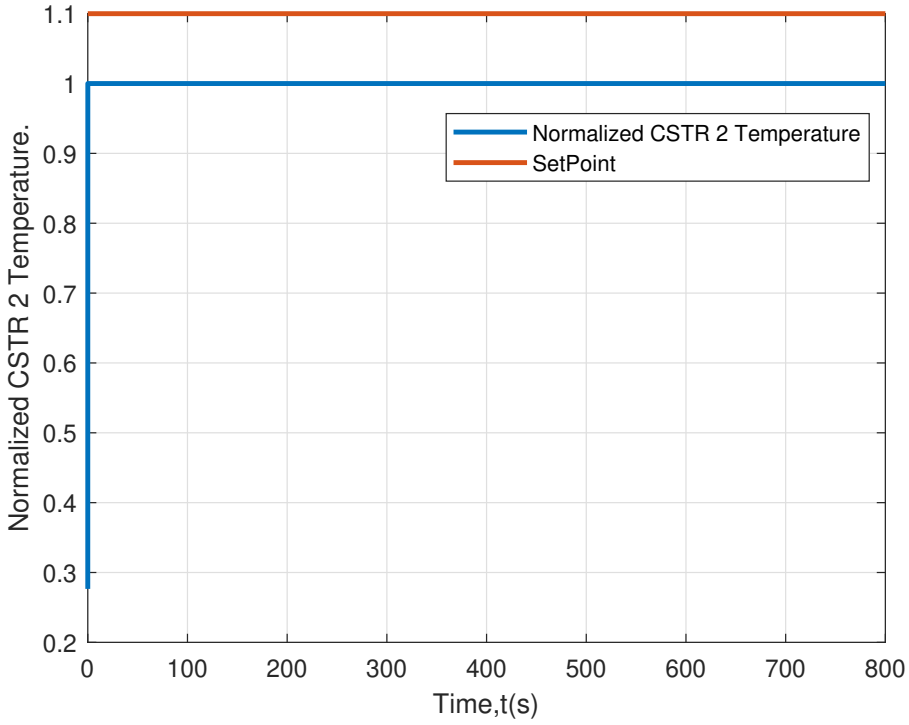

Figure 3.21: Temperature set-point tracking for CSTR 2 under DTDC.

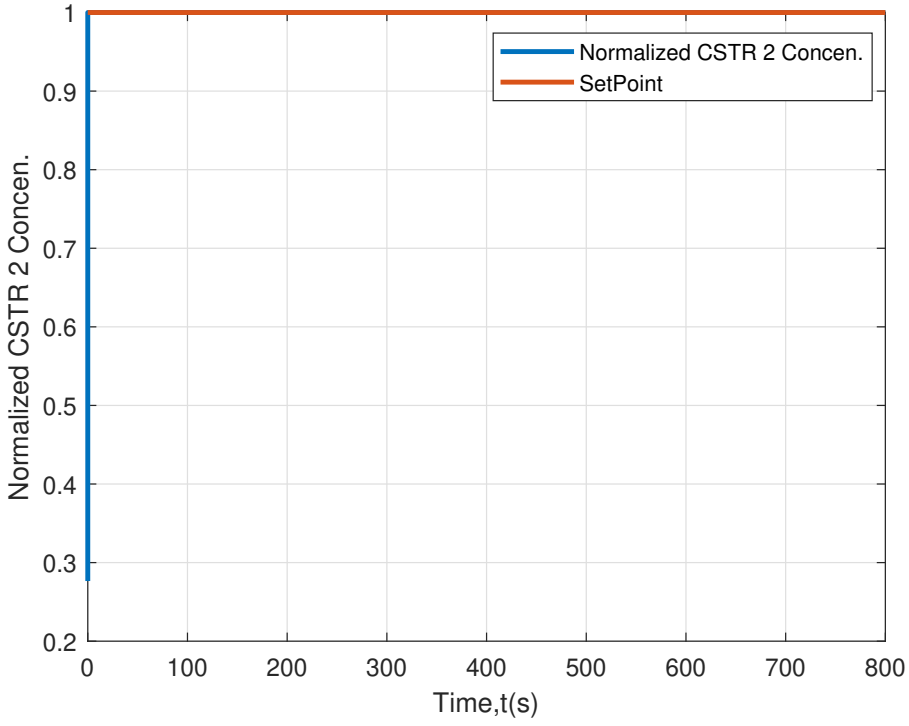

Figure 3.22: Concentration set-point tracking for CSTR 2 under DTDC.

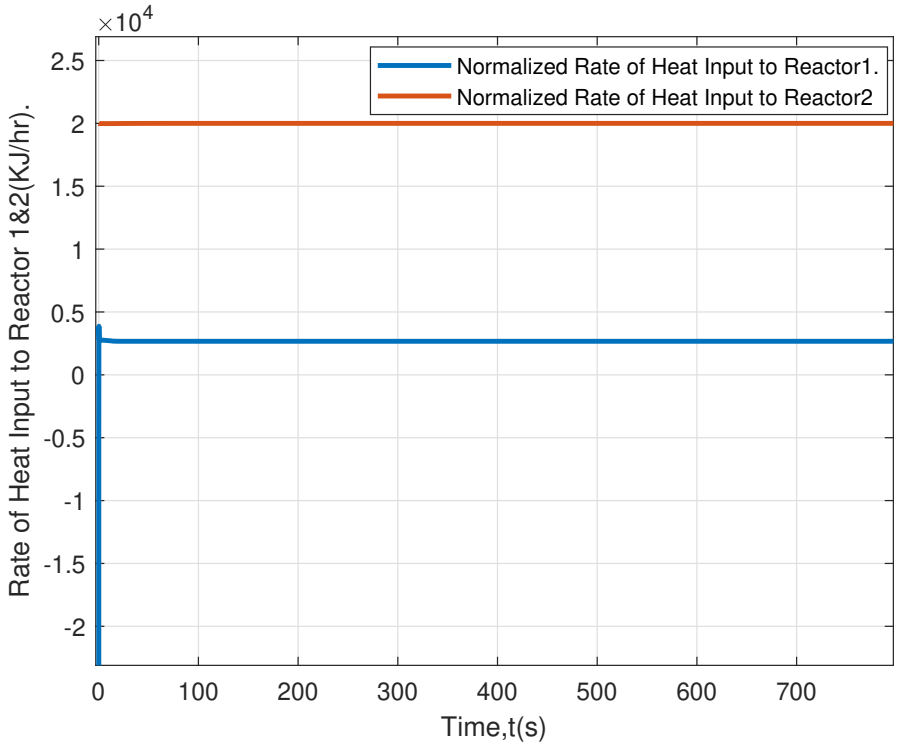

Figure 3.23: Rates of heat input for CSTR 1 and CSTR 2 under DTDC.

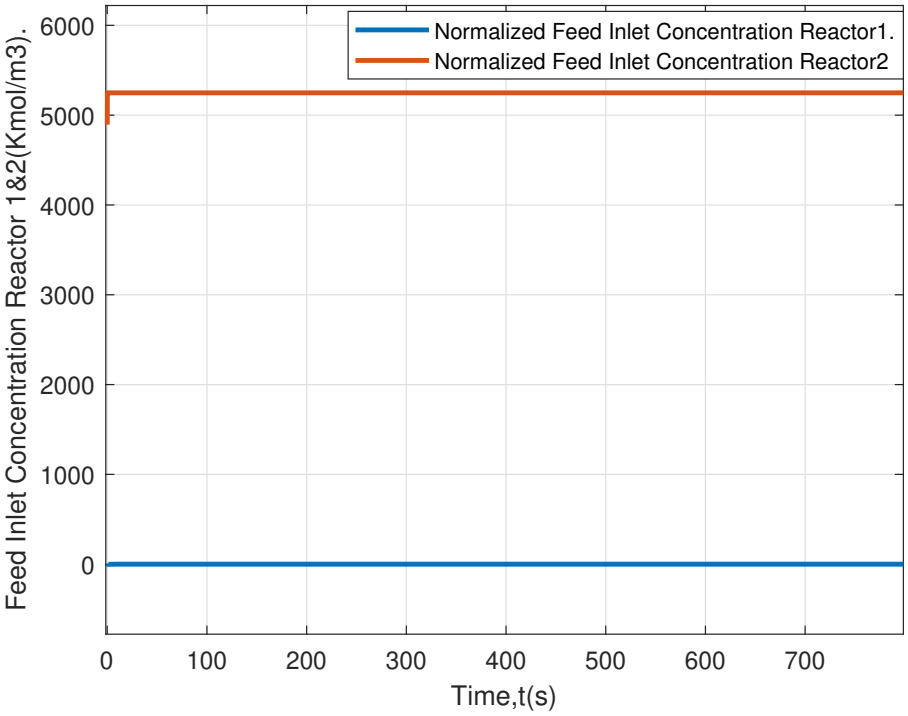

Figure 3.24: Inlet reactant concentrations for CSTR 1 and CSTR 2 under DTDC.

Figure 3.21 shows that the controller failed to achieve the temperature set-point for CSTR 2. The set-point error is about 10%. Except for the offset in the set-point, the other design goals for rise time, the % undershoot, %overshoot, and settling time have been successfully achieved. More control effort is needed to reach the set-point compared to the CTCC controller, and the control effort to maintain the concentration of CSTR 2 (see Figs. 3.23-3.24 is too high and might not be feasible from a practical standpoint. In decentralized control, the individual controllers act independently, with no knowledge of the impact of their actions on other controllers. Each controller has only information about their local states. Since each agent has partial observability of the plant environment, this leads to non-convergence issue for an optimization problem.

In the case of the reactant concentration in CSTR 2, the Controller was able to reach the desired set-point with error under 2% as can be seen from Fig. 3.22. The rise time is under 2 seconds, while the settling time is under 5 seconds. There is almost no overshoot or undershoot. All the design goals have been met. Except for the temperature offset in CSTR 2, the decentralized RL agents performance is in line with the design objectives.

3.3.6 Parameter Variations

The same parameter variations considered under CTCC control (see Table 3.9) are implemented in the DTDC framework to assess the controller performance. As observed from the plots in Figs. 3.25-3.28, the parameter uncertainty had the most impact on the temperature of CSTR 1, where it took longer to reach the desired set-point compared to the other controlled Variables. The DTDC algorithm based controller is robust to the parameter variations considered as can be observed from the plots, and the controllers are able to achieve their desired set-points except for temperature control of CSTR 2 which

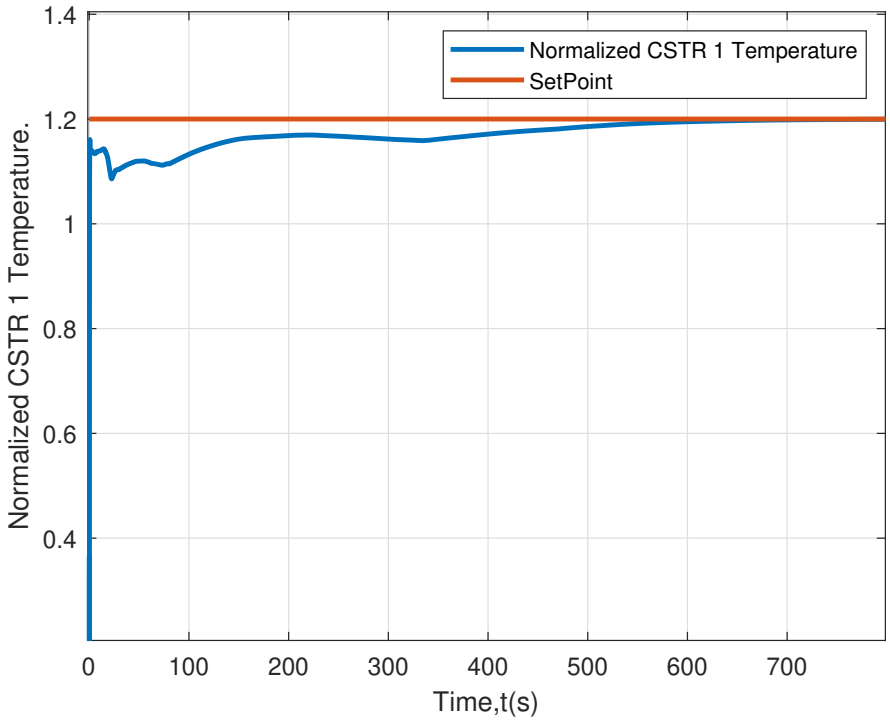

Figure 3.25: Temperature set-point tracking for CSTR 1 under DTDC and parameter variations.

has an offset of 10%.

3.4 Effect of Sensor Noise

For each sensor, Gaussian random noise with a mean of 0 and a variance of 10^{-5} was introduced to evaluate the controller performance with respect to random sensor noise.

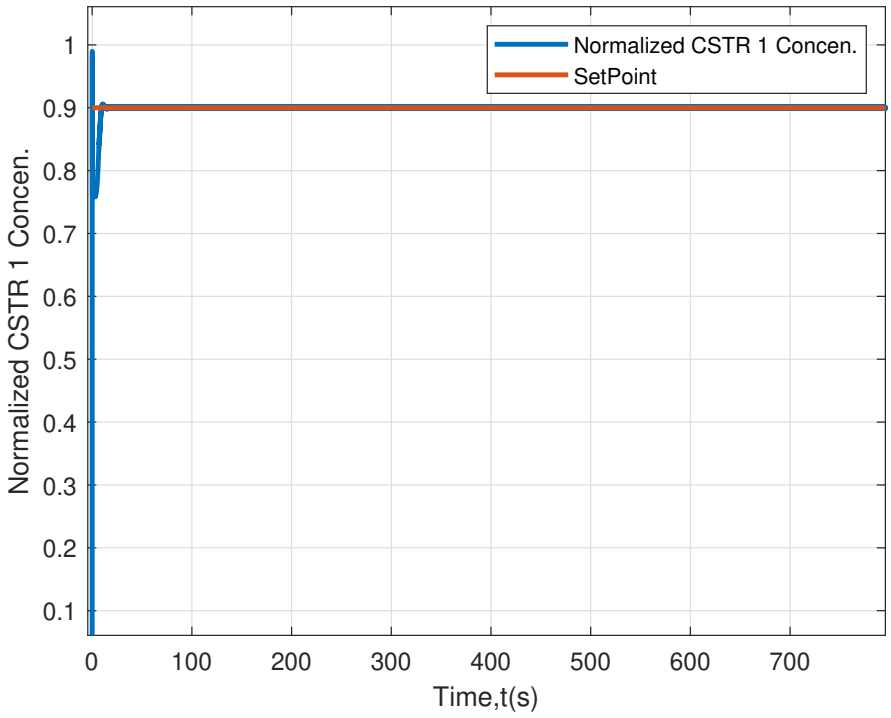

Figure 3.26: Concentration set-point tracking for CSTR 1 under DTDC and parameter variations.

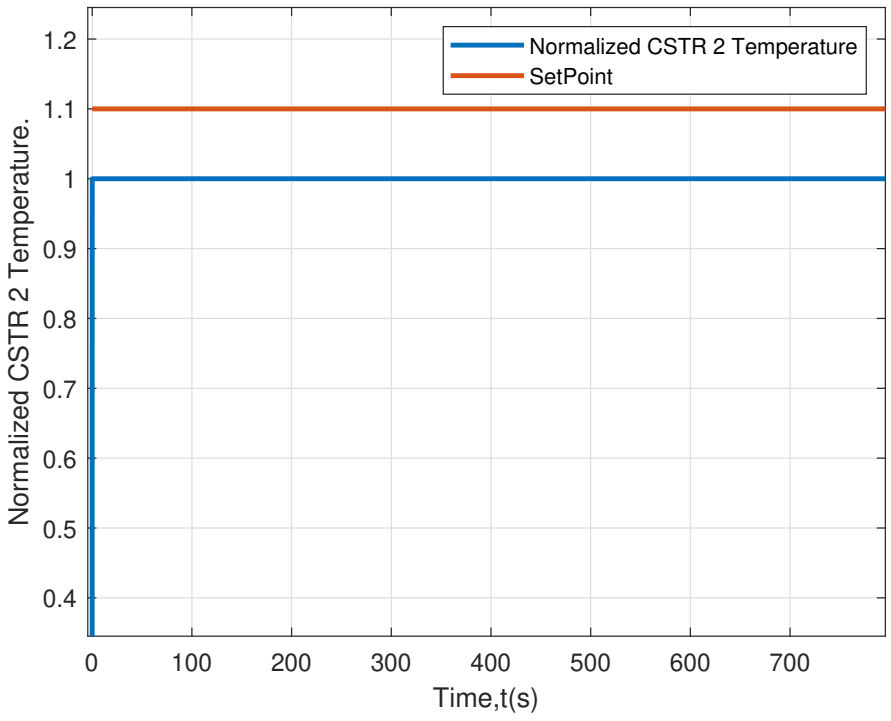

Figure 3.27: Temperature set-point tracking for CSTR 2 under DTDC and parameter variations.

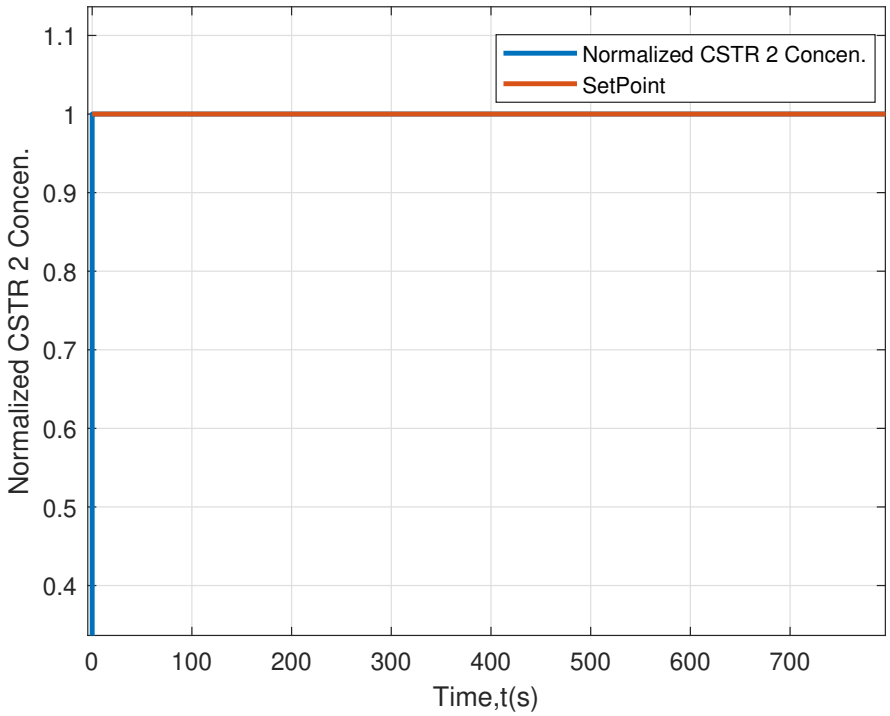

Figure 3.28: Concentration set-point tracking for CSTR 2 under DTDC and parameter variations.

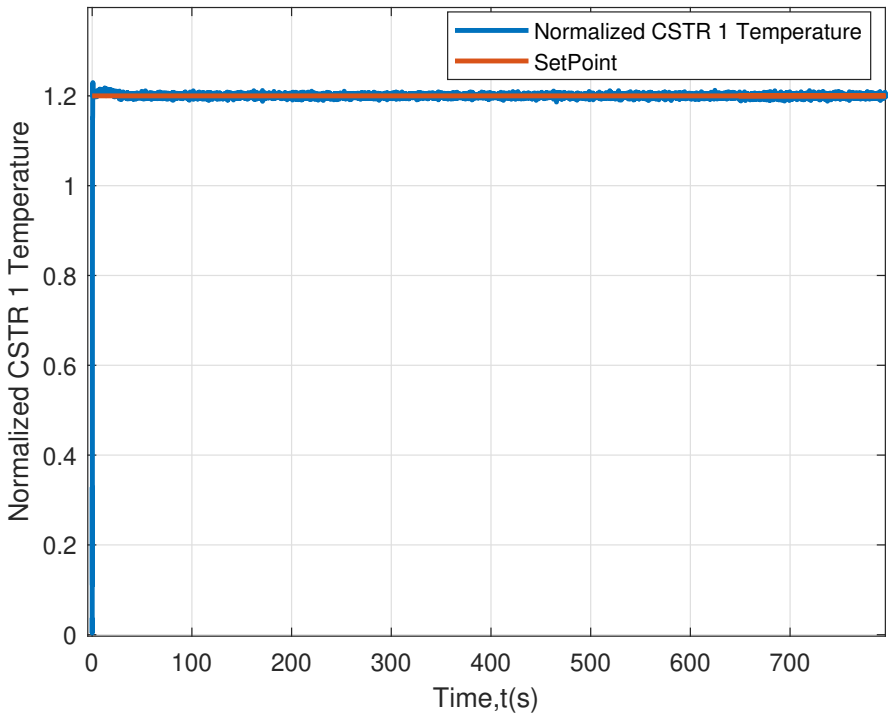

Figure 3.29: Temperature set-point tracking for CSTR 1 under DTDC and sensor noise.

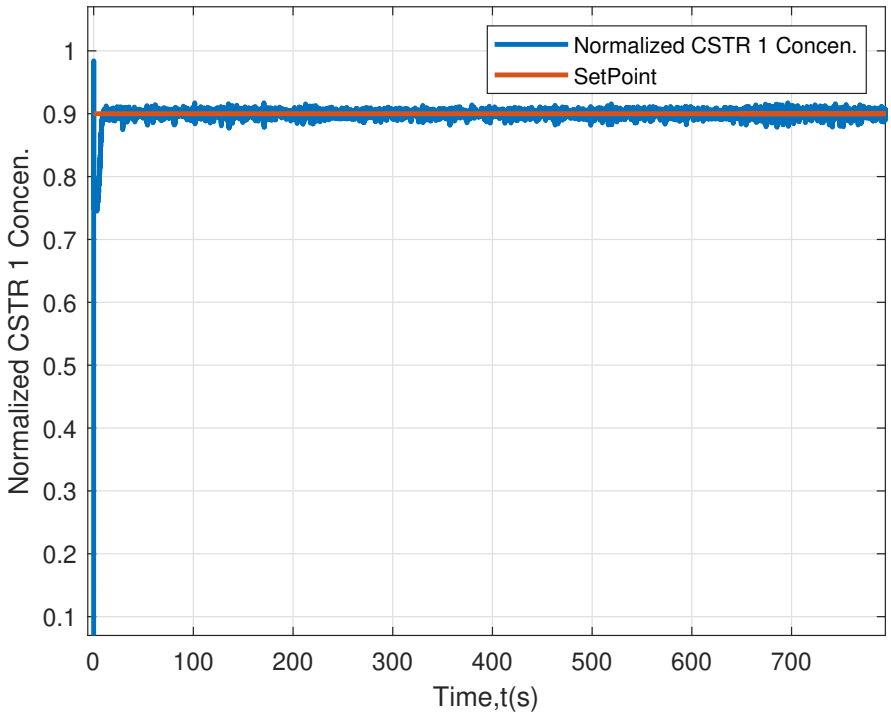

Figure 3.30: Concentration set-point tracking for CSTR 1 under DTDC and sensor noise.

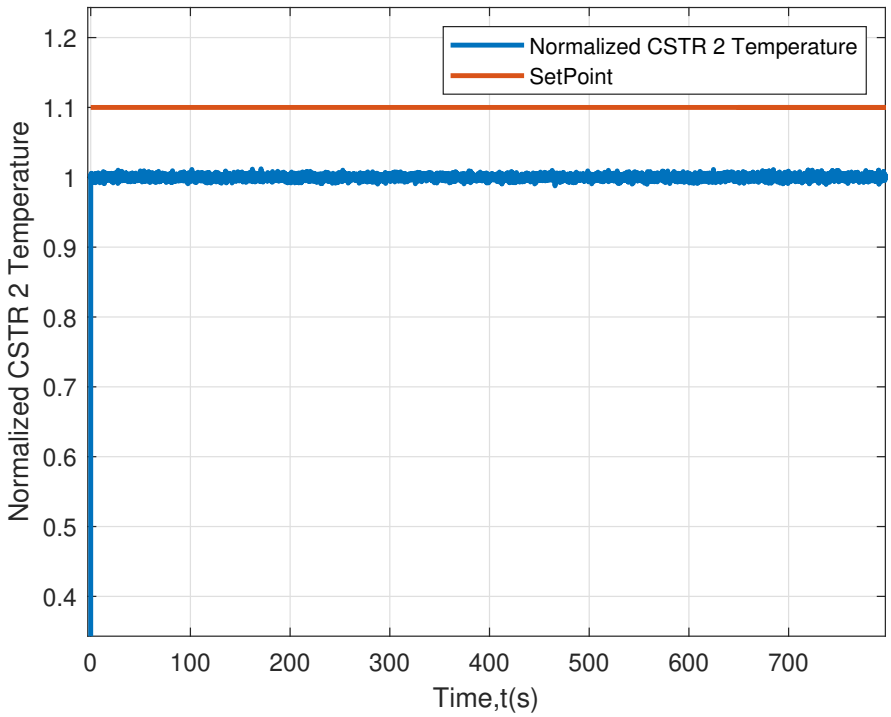

Figure 3.31: Temperature set-point tracking for CSTR 2 under DTDC and sensor noise.

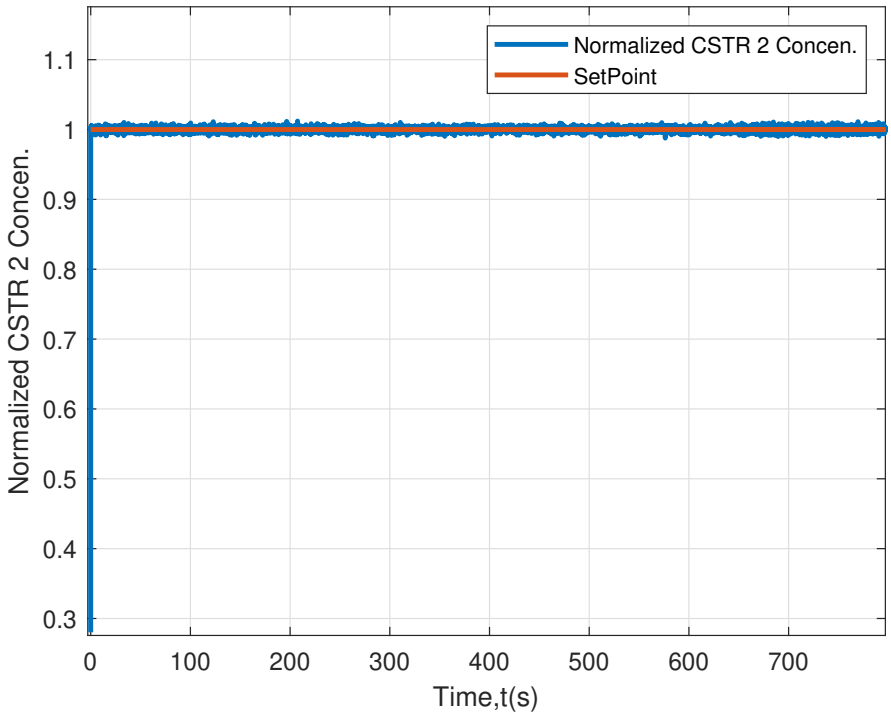

Figure 3.32: Concentration set-point tracking for CSTR 2 under DTDC and sensor noise.

From the results in Figs. 3.29-3.32, it is evident that the controller was still able to reach the desired set-point, albeit with some noise. We ran the simulations with the strength of the noise much lower than that of the response signal. Still the Controller was able to perform well considering we have not considered sensor noise while training the RL controller.

3.5 Centralized Training for Decentralized Control (CTDC)

In the CTDC framework, each agent has its own observation space but share a common critic network. Once the individual actors receive their localized observations, the common critic has the collective knowledge of all the other actors and guides the actors to achieve the collective goal. The actor and critic structure for each agent is the same as in DTDC control with the main difference being that all four TD3 agents critics have same hyper-

parameters and initialization so that all four TD3 agents share the same critic function. The actor and agent hyper-parameters are also kept same as in the DTDC algorithm for simplicity (these can be changed for each agent but the critic has to be common for all four agents). The observation space is also the same as in the DTDC framework.

3.5.1 Reward Functions

The reward function framework in CTDC slightly differs from the other two methods. Specifically,

$$R = R_i(x_i, t) + R_C$$

where R_C is a common reward received by each agent if all the errors (i.e., error 1, error 2, error 3 and error 4 corresponding to the temperatures and concentrations of CSTR 1 and CSTR 2, respectively) are less then 0.05. The reward function is designed such that the agents collaborate with each other to reach their target goal.

3.5.2 Results

In the CTDC framework, each RL agent receives local observations of the states but they share a common critic network structure as represented in Fig. 2.2(center). Once the local RL agents receive their observations, the critic neural network has access to all the agents' local observations, and during training guides the individual agents to arrive towards the common goal. This solves the partial observability problem in the DTDC algorithm and also the computational burden and complexity of a central controller in the CTCC framework. It can be observed from the results (see Figs. 3.33-3.36) that the CTDC controller enforces the desired set-points without any offset. The controller also shows robust performance to parameter uncertainty that is better than the other two

control algorithms. However, compared to CTCC and DTDC, a slight overshoot above the set-points is observed which is not present in the other two control schemes.

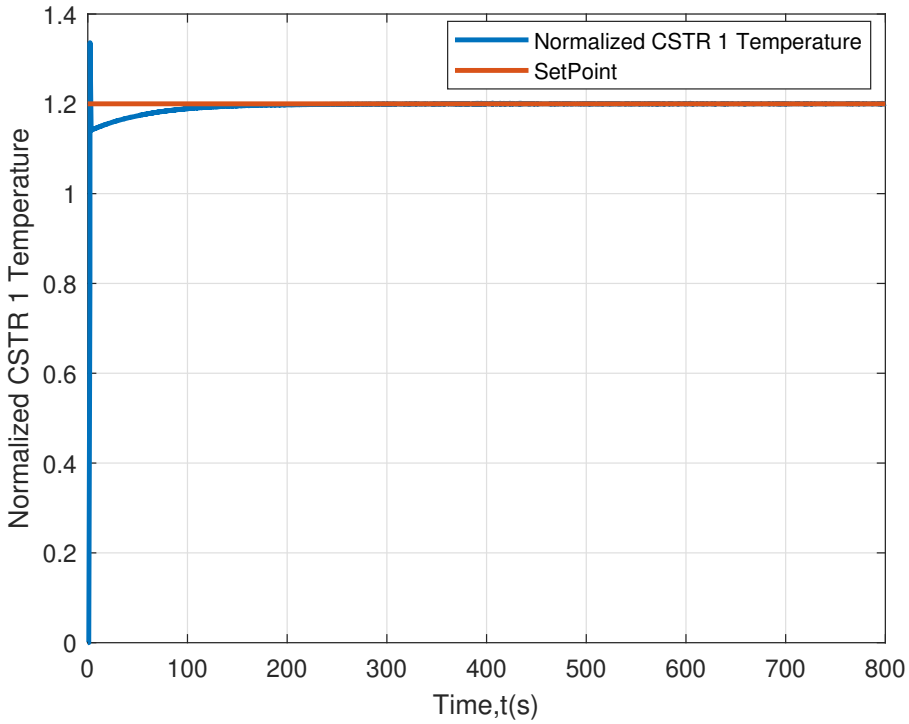

Figure 3.33: Temperature set-point tracking for CSTR 1 under CTDC.

Figure 3.33 shows that the controller was able to reach the temperature set-point with offset less than 1%. The rise time and the settling time are also according to the design specifications. Compared to the decentralized control scheme, where there is minimal overshoot, here we can notice that there is a slight overshoot of around 10% which is within the range of the design objective. Also, it took a longer time for the controller to reach the set-point with error less than 1% than in the DTDC control setting.

For the reactant concentration set-point tracking in CSTR 1, Fig. 3.34 shows that although the rise time and settling time design objectives have been met, the controller took longer time to stabilize the control action than in DTDC control. Furthermore, there is a slight undershoot in this scenario, though it is around 2% and much less than the

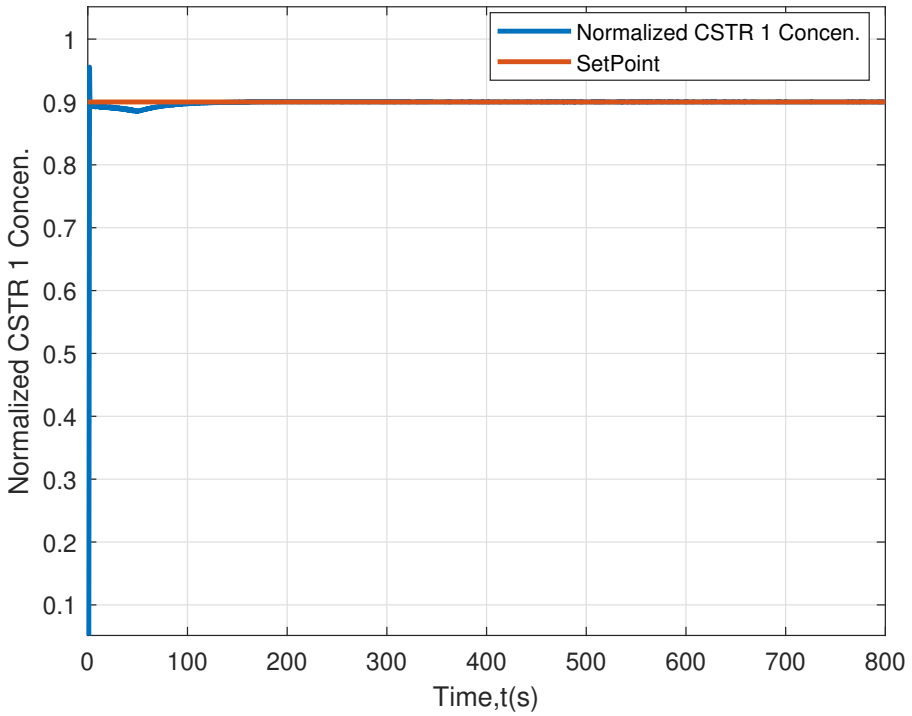

Figure 3.34: Concentration set-point tracking for CSTR 1 under CTDC.

design tolerance limit. Also, at the beginning the overshoot is around 8% which is well within the design objective.

For temperature control of CSTR 2, the overshoot is around 5% and there is almost no overshoot (see Fig. 3.35). The rise time and settling time objectives have been met. As with the previous controlled variables, it took longer time for the controller to achieve the set-point with error under 1% compared to DTDC control. Also, no undershoot is observed in this scenario. The control response profile is very similar to that of the temperature control of CSTR 1.

The controller response for the reactant concentration control of CSTR 2 is similar to that of the concentration profile of CSTR 1 (see Fig. 3.36. The overshoot though is much less than that in the case of the concentration control of CSTR 1. The rise time, the settling time, the undershoot and overshoot are well within the design Objectives, albeit

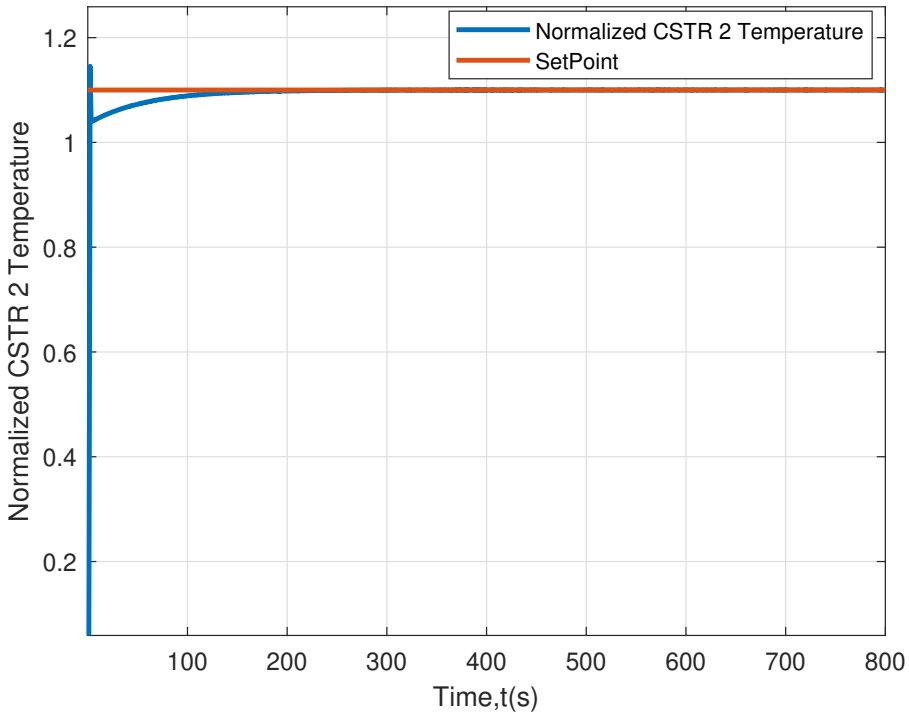

Figure 3.35: Temperature set-point tracking for CSTR2 under CTDC.

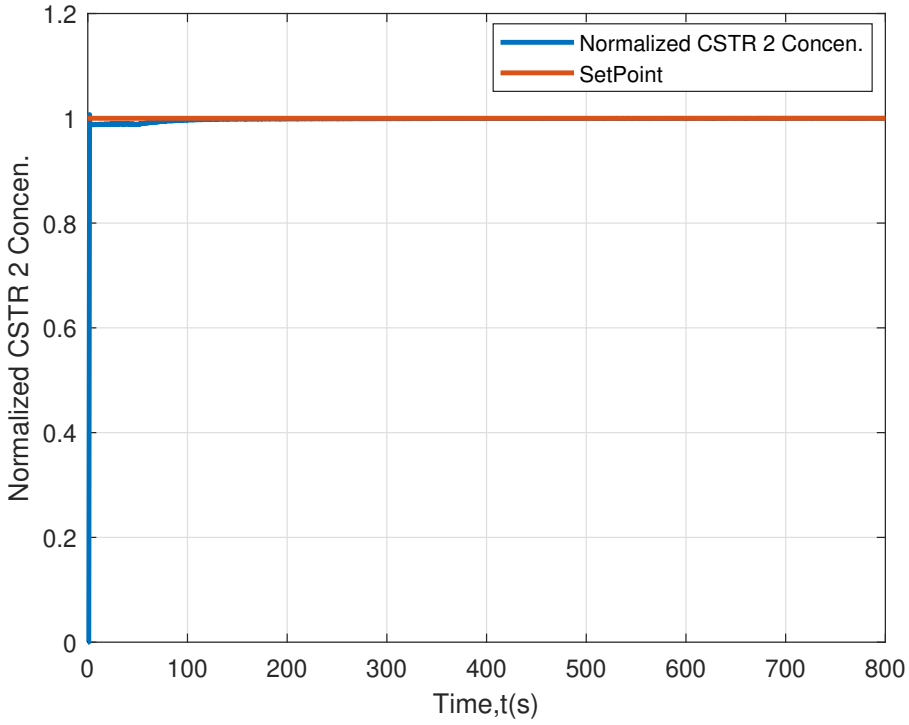

Figure 3.36: Concentration set-point tracking of CSTR 2 under CTDC.

slower compared to the DTDC control strategy. As shown from the manipulated input profiles in Figs. 3.37-3.38, the control effort in this case is higher than that in the CTCC control scheme but lower than that of the DTDC scheme.

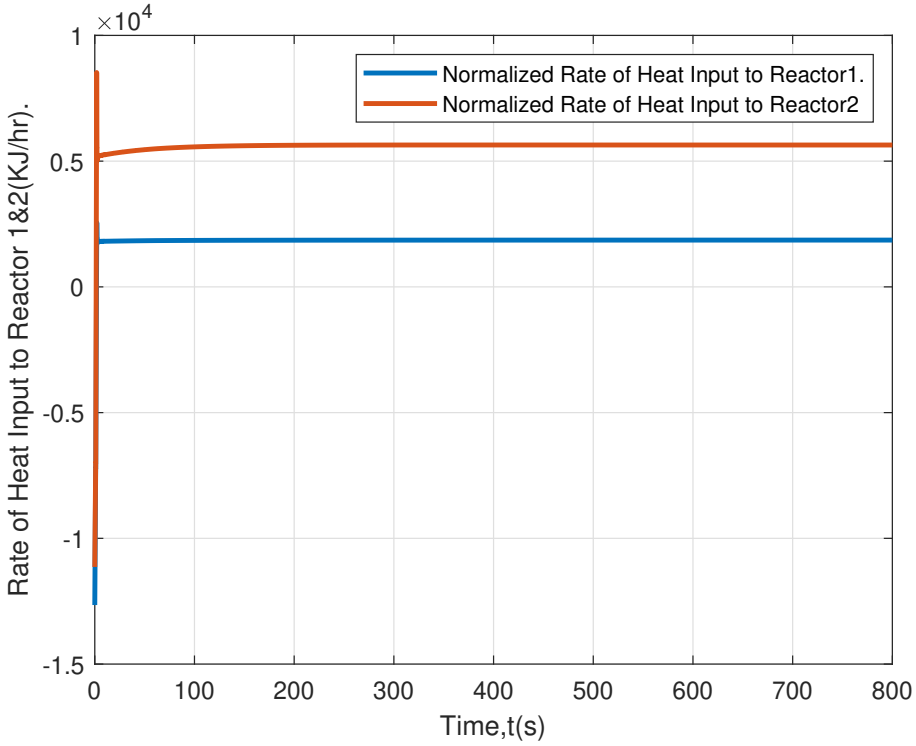

Figure 3.37: Rates of heat input under CTDC.

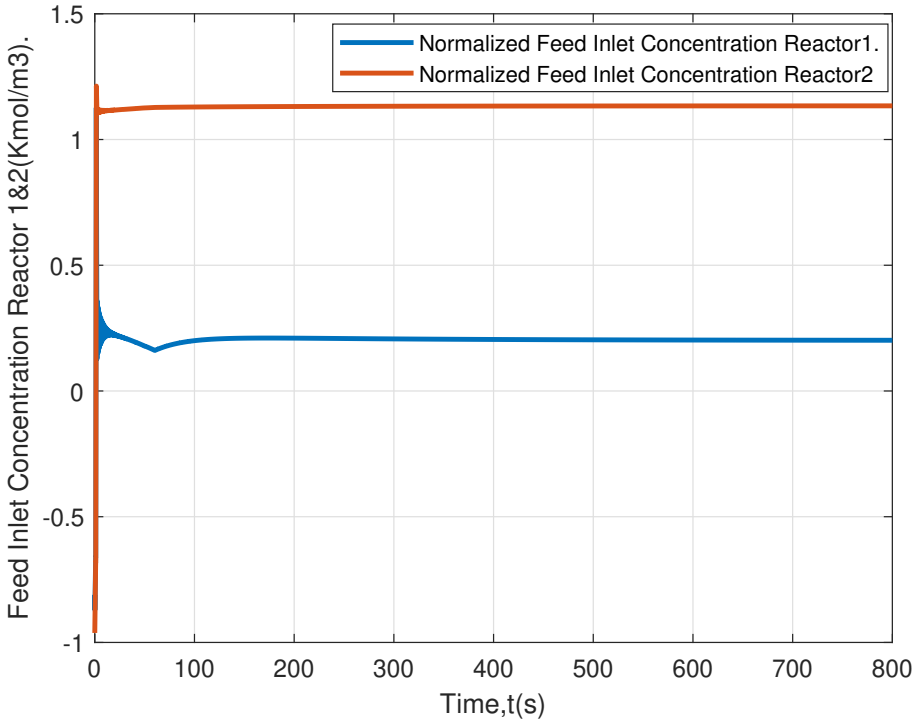

Figure 3.38: inlet reactant concentrations under CTDC control.

3.5.3 Parameter Variations

The trained RL agent was evaluated for robustness by changing the model parameters and observing the controller set-point tracking performance on the new simulated model with the updated parameters. The same parameter variations which are highlighted in Table 3.9 are used. As it can be observed from the plots in Figs. 3.39-3.42, the controller was able to achieve the desired set-points even when the parameters were modified, and no offsets were detected.

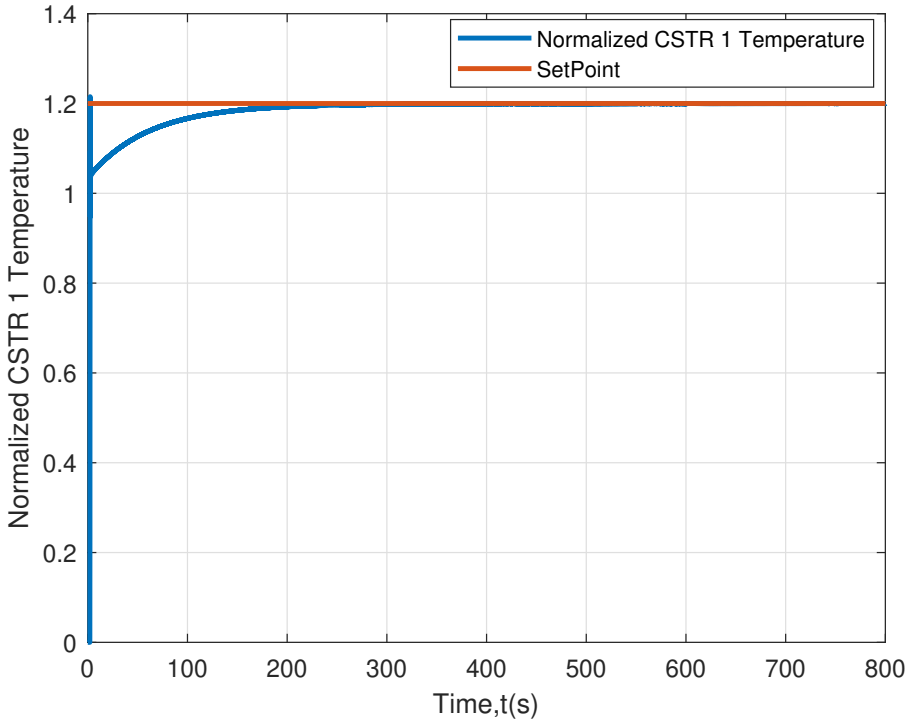

Figure 3.39: Temperature set-point tracking for CSTR 1 under CTDC and parameter variations.

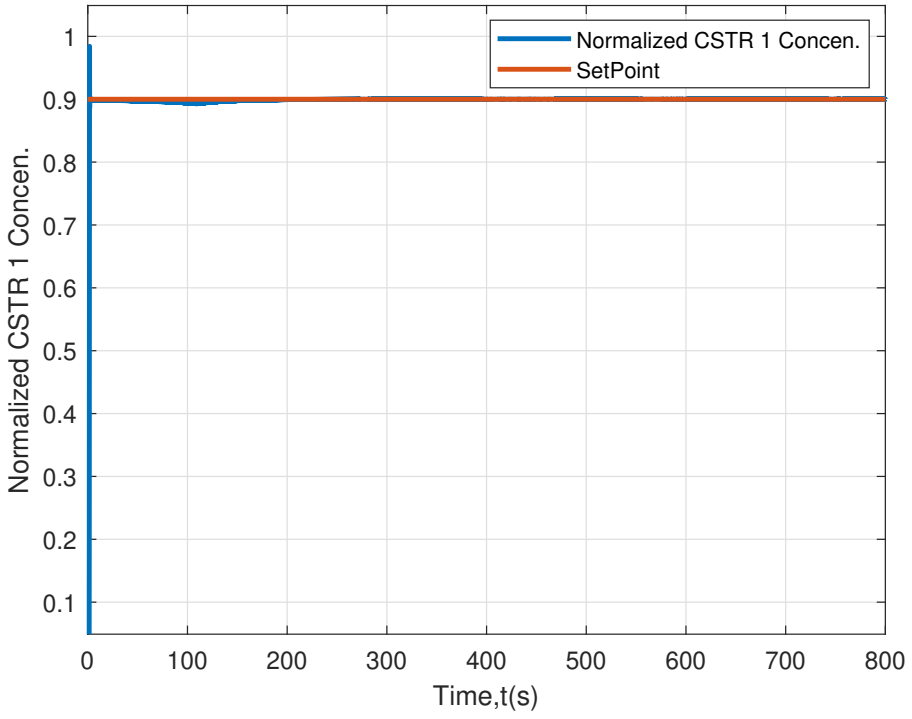

Figure 3.40: Concentration set-point tracking for CSTR 1 under CTDC and parameter variations.

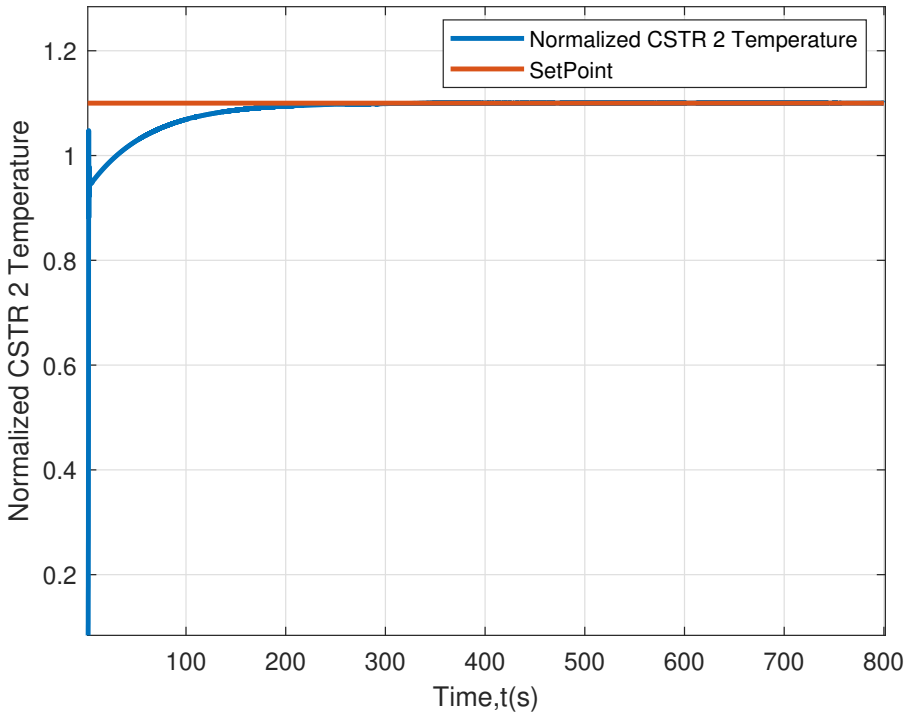

Figure 3.41: Temperature set-point tracking for CSTR 2 under CTDC and parameter variations.

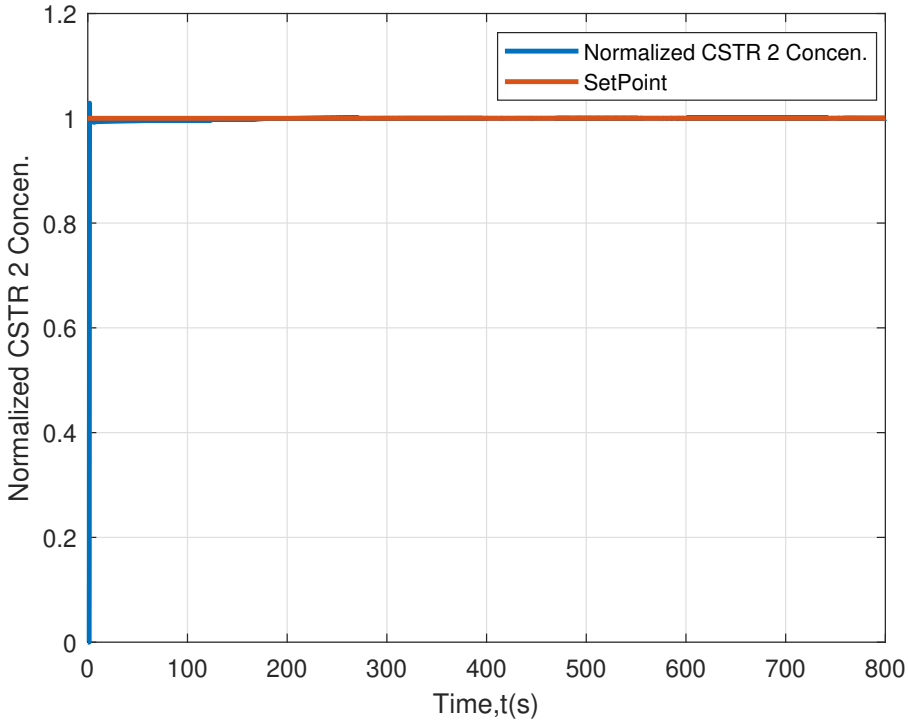

Figure 3.42: Concentration set-point tracking for CSTR 2 under CTDC and parameter variations.

All the parameters that can be varied in a practical scenario are considered, and the limits are chosen such that the percentage variations can be expected in real-life scenarios due, for example, to catalyst deactivation, changes in flow rates due to pump problems, changes in concentration of the fresh feeds to the CSTRs due to any unknown factors. From the plots, it can be seen that the parameter uncertainties have the most effect on the temperature of CSTR 1 and it took a long time to reach the desired set-point (around 200 seconds) compared to the other controlled parameters. All the other controlled variables were able to reach their specified set-points rapidly. Compared to the DTDC and CTCC control frameworks, the CTDC framework based controller shows improved performance under parameter uncertainty. From the plots, the CTDC controller took less time to stabilize and reach the set-points compared to the other controllers. It is important to note that the individual RL agents were not trained under parameter uncertainty. The ability of RL agents to robustly maintain controlled variables at the desired set-point under parameter variations shows the promising applicability of RL agents for control system applications when a reliable model is not available or when the system is too complex to generate a model.

3.5.4 Effect of Sensor Noise

For each sensor, Gaussian random noise with a mean of 0.01 and variance of 10^{-5} was introduced to evaluate the controller performance with respect to random noise. From the plots given in Figs. 3.43-3.46, we conclude that the controller performance is robust in the presence of the sensor noise considered, although some slight oscillatory behavior is observed at the very beginning and then response becomes smoother quickly.

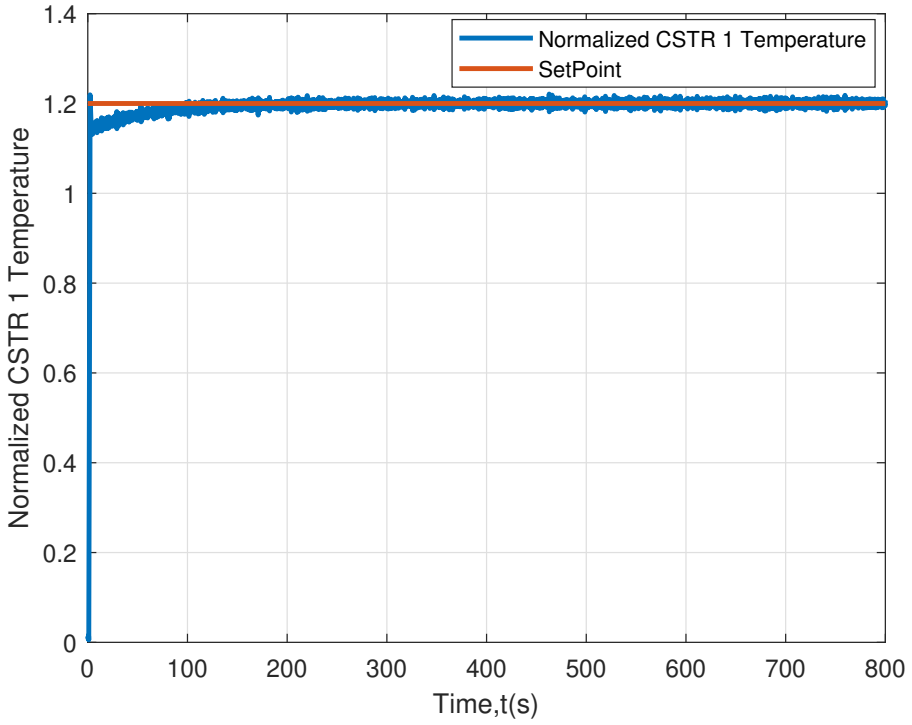

Figure 3.43: Temperature set-point tracking for CSTR 1 under CTDC and sensor noise.

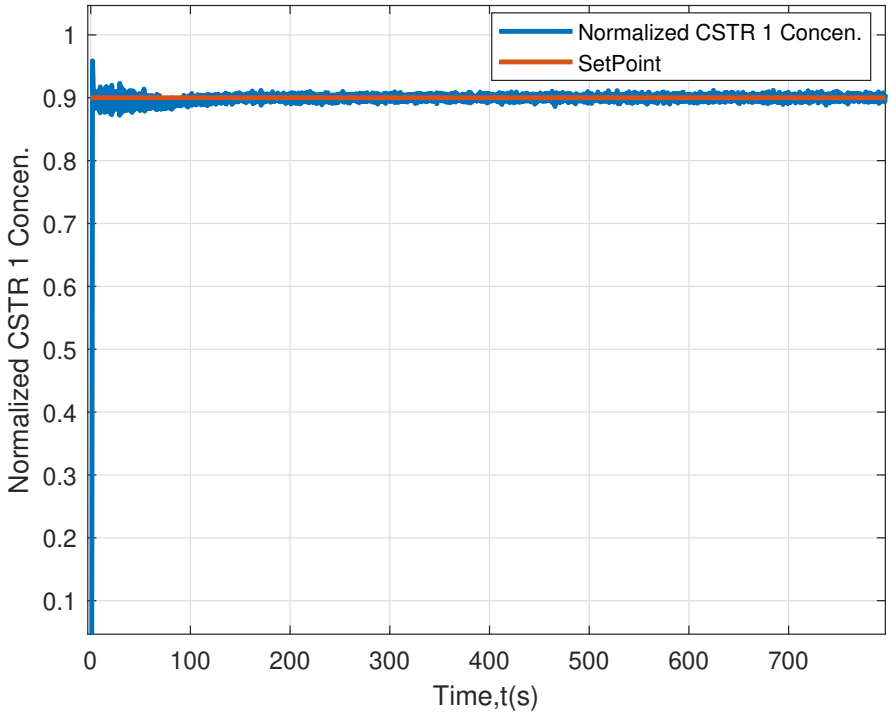

Figure 3.44: Concentration set-point tracking for CSTR 1 under CTDC and sensor noise.

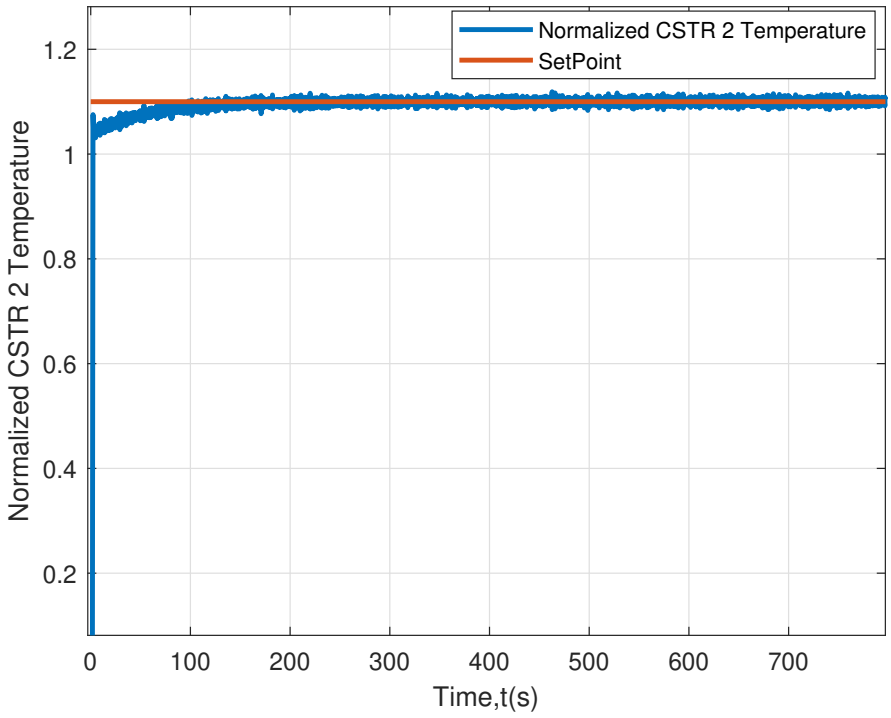

Figure 3.45: Temperature set-point tracking for CSTR 2 under CTDC and sensor noise.

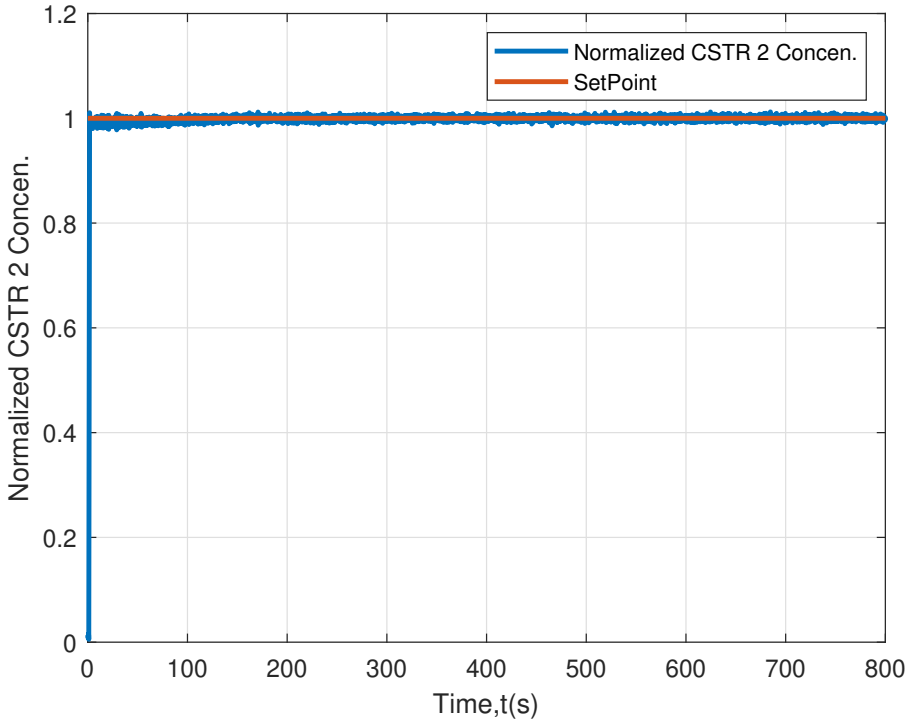

Figure 3.46: Concentration set-point tracking for CSTR 2 under CTDC and sensor noise.

The concentration profile of CSTR 1 shows oscillations under sensor noise at the beginning and eventually became smoother. All the other controlled variables are able to achieve the desired set-points with some background noise in the control signal. These results illustrate the robustness of the controller to sensor noise, although the controller is not trained under sensor noise.

Chapter 4

Conclusions and Possible Future Extensions

This work illustrated the design and implementation of three different multi-agent RL control strategies to a process network comprised of two interconnected chemical reactors with a recycle stream. A a centralized strategy (CTCC) was considered first, where a central RL agent has the information of all the states of the system. A fully-decentralized control strategy (DTDC) was presented next, where each RL agent acts on its own without interacting with other agents or considering the impact of their actions on other agents. The third strategy involved decentralized execution with centralized control (CTDC), where all the RL agents share a common critic network parameters.

The CTCC control strategy requires instantaneous knowledge of all the states which may not be feasible practically if the interconnected process units are located far from one another and the RL agent may receive state signals at different time intervals from the different units. Also, the observation space increases exponentially with an increase in the number of units, leading to computational burden on the central controller, as evident from the results in which the CTCC based controller took longer to reach the control

goals compared to the DTDC and CTDC based controllers, which makes this approach practically challenging.

In the DTDC control strategy, four individual RL agents were used to control the process variables. This approach resolved the issue of availability of all the states at the same instance faced in the CTCC control strategy, but was not able to achieve the desired set-point for the temperature of the second reactor. The control strategy showed excellent disturbance rejection and robust performance to sensor noise even though these considerations were not accounted for during the training phase. The main limitation of this strategy is the partial observability of the states and non-stationary environment. Each agent has no knowledge of the complete states of the system and there is also no communication between the agents. This can lead to non-convergence and competition between the agents. The offset is the result of the failure of convergence due to this issue. Further research is needed to design the decentralized RL agents to cooperate with one another to achieve their individual goals.

Compared to the CTCC and DTDC strategies, the CTDC control strategy showed promising results, as it was able to achieve the desired set-point tracking using moderate control effort, and exhibited robustness to parameter variations and sensor noise. This approach overcomes the limitations of both the CTCC and DTDC based control frameworks; however, the response time is slower than that of the DTDC strategy and faster than that of the CTCC strategy. In all three control schemes, similar observation space and reward function structure were used for comparing the results. This work shows that multi-agent RL can effectively control highly nonlinear interconnected systems when a model of the system is not known. The reward function and observation space were

designed using optimization techniques and concepts from PI control.

While the results of this comparative study of the different multi-agent RL control systems show promise, several issues of practical concern remain to be addressed and can be the subject of future extensions. These include developing multi-agent RL control systems that explicitly account for communication delays, and investigating the effects of sensor faults which result in loss of state information to the RL agents, degradation of the control system performance and raises safety concerns to assets. Further research is needed to develop robust fault-tolerant RL control systems for interconnected process networks.

Bibliography

[1] Khalid Alhazmi and S Mani Sarathy. Continuous control of complex chemical reaction network with reinforcement learning. In *2020 European Control Conference (ECC)*, pages 1066–1068. IEEE, 2020.

[2] Christopher Amato, Jilles Dibangoye, and Shlomo Zilberstein. Incremental policy generation for finite-horizon dec-pomdps. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 19, pages 2–9, 2009.

[3] Lubomir Bakule. Decentralized control: An overview. *Annual reviews in control*, 32(1):87–98, 2008.

[4] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

[5] Guillaume Bono, Jilles Steeve Dibangoye, Laëtitia Matignon, Florian Pereyron, and Olivier Simonin. Cooperative multi-agent policy gradient. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part I 18*, pages 459–476. Springer, 2019.

[6] Kehua Chen, Hongcheng Wang, Borja Valverde-Pérez, Siyuan Zhai, Luca Vezzaro, and Aijie Wang. Optimal control towards sustainable wastewater treatment plants based on multi-agent reinforcement learning. *Chemosphere*, 279:130498, 2021.

[7] Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*, 2012.

[8] Jilles Steeve Dibangoye, Christopher Amato, Olivier Buffet, and François Charpillet. Optimally solving dec-pomdps as continuous-state mdps. *Journal of Artificial Intelligence Research*, 55:443–497, 2016.

[9] Robert M Freund. Penalty and barrier methods for constrained optimization. *Lecture Notes, Massachusetts Institute of Technology*, 2004.

[10] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[11] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16*, pages 66–83. Springer, 2017.

[12] JC Hoskins and DM Himmelblau. Process control via artificial neural networks and reinforcement learning. *Computers & chemical engineering*, 16(4):241–251, 1992.

[13] Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016.

[14] Shumpei Kubosawa, Takashi Onishi, and Yoshimasa Tsuruoka. Synthesizing chemical plant operation procedures using knowledge, dynamic simulation and deep reinforcement learning. *arXiv preprint arXiv:1903.02183*, 2019.

[15] Mikko Lauri, David Hsu, and Joni Pajarinen. Partially observable markov decision processes in robotics: A survey. *IEEE Transactions on Robotics*, 39(1):21–40, 2022.

[16] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[17] Miao Liu, Christopher Amato, Emily Anesta, John Griffith, and Jonathan How. Learning for decentralized control of multiagent systems in large, partially-observable stochastic environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[18] Miao Liu, Christopher Amato, Xuejun Liao, Lawrence Carin, and Jonathan P How. Stick-breaking policy learning in dec-pomdps. *arXiv preprint arXiv:1505.00274*, 2015.

[19] Magdi S Mahmoud. Reliable decentralized control of interconnected discrete delay systems. *Automatica*, 48(5):986–990, 2012.

[20] Borja Martínez, Manuel Rodríguez, and Ismael Díaz. Cstr control with deep reinforcement learning. In *Computer Aided Chemical Engineering*, volume 49, pages 1693–1698. Elsevier, 2022.

[21] H Van Dyke Parunak. Industrial and practical applications of dai. *Multiagent systems: a modern approach to distributed artificial intelligence*, pages 337–421, 1999.

[22] Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau, and Leslie Pack Kaelbling. Learning to cooperate via policy search. *arXiv preprint cs/0105032*, 2001.

[23] G Pujol, J Rodellar, JM Rossell, and F Pozo. Decentralised reliable guaranteed cost control of uncertain systems: an lmi design. *IET Control Theory & Applications*, 1(3):779–785, 2007.

[24] Rabia Saleem, Bo Yuan, Fatih Kurugollu, Ashiq Anjum, and Lu Liu. Explaining deep neural networks: A survey on the global interpretation methods. *Neurocomputing*, 2022.

[25] Ádám Sass, Alex Kummer, and János Abonyi. Multi-agent reinforcement learning-based exploration of optimal operation strategies of semi-batch reactors. *Computers & Chemical Engineering*, 162:107819, 2022.

[26] Ralf Schoknecht and Martin Riedmiller. Learning to control at multiple time scales. In *International Conference on Artificial Neural Networks*, pages 479–487. Springer, 2003.

[27] Dale E Seborg, Thomas F Edgar, Duncan A Mellichamp, and Francis J Doyle III. *Process dynamics and control*. John Wiley & Sons, 2016.

[28] Hareem Shafi, Kirubakaran Velswamy, Fadi Ibrahim, and Biao Huang. A hierarchical constrained reinforcement learning for optimization of bitumen recovery rate in a primary separation vessel. *Computers & Chemical Engineering*, 140:106939, 2020.

[29] Dragoslav D Siljak. *Decentralized control of complex systems*. Courier Corporation, 2011.

[30] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. Pmlr, 2014.

[31] Yulei Sun and Nael H El-Farra. Quasi-decentralized model-based networked control of process systems. *Computers & Chemical Engineering*, 32(9):2016–2029, 2008.

[32] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[33] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[34] S Syafiie, Fernando Tadeo, and E Martinez. Model-free learning control of neutralization processes using reinforcement learning. *Engineering Applications of Artificial Intelligence*, 20(6):767–782, 2007.

[35] Nikos A. Vlassis. A concise introduction to multiagent systems and distributed artificial intelligence. In *A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence*, 2007.

[36] Xu Wang, Sen Wang, Xingxing Liang, Dawei Zhao, Jincai Huang, Xin Xu, Bin Dai, and Qiguang Miao. Deep reinforcement learning: a survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[37] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[38] Gerhard Weiss. *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press, 1999.

[39] Yue Yifei and S Lakshminarayanan. Multi-agent reinforcement learning system for multiloop control of chemical processes. In *2022 IEEE International Symposium on Advanced Control of Industrial Processes (AdCONIP)*, pages 48–53. IEEE, 2022.

[40] Lingwei Zhu, Yunduan Cui, Go Takami, Hiroaki Kanokogi, and Takamitsu Matsubara. Scalable reinforcement learning for plant-wide control of vinyl acetate monomer process. *Control Engineering Practice*, 97:104331, 2020.