Computers and Chemical Engineering 212 (2026) 109704

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/cace

Reactor runaway aware RL Agents for safe reactor operation

Balázs Fricz [a], Kinga Szatmári [a,*], Sándor Németh [a], Lajos Nagy [a], Wenshuai Bai [b], Alex Kummer [a,c]

[a] Department of Process Engineering, University of Pannonia, Egyetem st. 10, H-8200 Veszprém, Hungary
[b] Tianjin Key Laboratory of Chemical Process Safety, School of Chemical Engineering and Technology, Hebei University of Technology, Tianjin 300130, China
[c] HUN-REN-PE Complex Systems Monitoring Research Group, University of Pannonia, Egyetem st. 10, H-8200 Veszprém, Hungary

ARTICLE INFO

Keywords:
Reinforcement learning
Chemical process control
Thermal runaway
Process safety

ABSTRACT

The control of semi-batch reactors is a challenging task due to the fact that these systems are non-stationary and often non-linear. Thermal runaway may also occur without strict operator supervision and rigorous control systems, which can lead to critical safety incidents. Reinforcement learning (RL) can offer a solution to the mentioned issues for autonomous process control by handling non-linearities and learning the uncertainties of the process. We applied Twin Delayed Deep Deterministic Policy Gradient (TD3) agents with different penalty weight factors in the reward function and compared their control performance on a reactor case study. Assessment of a thermal runaway criterion was included in the training of the agents to make them aware of potential future operating conditions and to ensure safe and stable process behavior. The proposed approach shows that applying RL for non-continuous systems is promising due to the observed flexibility and ease of setting up the system.

1. Introduction

In the case of highly exothermic reactions, the increase of the reaction rate can become large enough that the control structure cannot compensate the temperature change. It can lead to thermal runaway of the reactor that can result the loss of raw materials, process equipment, or in extreme cases, personal life (Dakkoune et al., 2019). In semi-batch reactors (SBRs), thermal runaway can be prevented by controlling the reagent feeding rate (Westerterp and Molga, 2006) or by changing the flow rate of the reactor jacket during the entirety of the reactor operation (Kummer and Varga, 2021). In our work, we designed a reinforcement learning based control agent that is tasked with the temperature control of a semi-batch reactor. During training, the agent was made aware of a runaway criterion metric, and we have investigated the effect of the dominance of this metric in the reward function.

When designing a controller for a semi-batch reactor, the task is to create a control system that maintains a high production efficiency while also operating the reactor system safely (Yang et al., 2024). The approaches to solving this can range from modified PID algorithms to model predictive controllers and data-based solutions based on machine learning algorithms.

Reactor runaway is one of the major risks associated with highly exothermic reactions, therefore, the design of protection systems is critical, particularly for handling emergency situations caused by sudden technical failures. In reducing the risks related to runaway reactions, a layered protection strategy is required. The first priority is to implement measures that prevent the runaway. If a runaway occurs despite these measures, preventive actions should be available to stop or limit its development. As a final layer of protection, emergency measures should mitigate the consequences of a runaway that can no longer be avoided. In the process industries, safety has traditionally been ensured through dedicated protection layers, most notably safety instrumented systems (SIS), which are designed according to standards such as IEC 61511. These systems are intended to achieve or maintain a safe state of the process, but they typically operate as independent protection layers rather than as performance-oriented control systems (Stoessel, 2021).

The first layer of protection is the process itself, meaning that the reactor and operating conditions should be designed in such a way that runaway is inherently unlikely to occur. In practice, this includes measures such as limiting reactant accumulation, reducing the energy release potential of the reaction, and applying inherently safer design principles such as substitution, attenuation, and process intensification. Thus, before introducing additional protective or emergency layers, the process design and control strategy should already contribute to reducing runaway risk at its source (Stoessel, 2021).

To bridge the gap between safety and control, increasing attention has been devoted to integrating safety-related information directly into

* Corresponding author.
E-mail address: szatmari.kinga@mk.uni-pannon.hu (K. Szatmári).

https://doi.org/10.1016/j.compchemeng.2026.109704
Received 5 February 2026; Received in revised form 4 May 2026; Accepted 11 May 2026
Available online 19 May 2026

the control layer. In particular, process operational safety has been addressed through model predictive control (MPC) formulations that incorporate safety metrics, or safety indices into the control design, for example, a Safeness-Index-based MPC defines a safe operating region explicitly within the controller (Zhang et al., 2019). These approaches are closely related to safety-constrained control strategies, where unsafe operating regions are avoided by design through explicit constraints or safety-oriented optimization criteria. In the context of chemical reactors, this has included MPC formulations that integrate runaway criteria to prevent thermal runaway while maintaining acceptable performance. Semi-batch reactor case studies demonstrated that thermal runaway criteria can be incorporated directly into NMPC formulations for safe operation (Kummer et al., 2020).

Intelligent control represents the use of artificial intelligence and machine learning methods (Kavitha, 2019), where systems are capable of learning from signals received from their environments, so they are able to upgrade themselves to perform better in control tasks. The control agent then uses the signals to make more informed control decisions compared to standard control methods (Hangos et al., 2001).

One method of creating a machine learning based controller is to use reinforcement learning (RL), which is a modern research direction. The basis of the RL algorithm is a rewarding system. It requires an agent that learns to map states to actions and an environment with which it interacts through actions. The learning process is determined by the states of the environment, the actions taken by the agent, and a reward that is based on the action of the agent and its effect on the environment. This control method needs a plan to follow when assigning an action to a given state, which is generally called a *policy*, and the objective of model training is to generate the best possible policy for the task (Barto, 2021). In the case of process control, both the state and action spaces are often continuous, which requires additional considerations compared to discrete RL methods.

The basis of most modern RL applications is Deep Q-Networks (DQNs). These by themselves are neural nets with discrete outputs, however, they have found use in process control. A DQN with Long Short-Term Memory (LSTM) net was applied to the control of a divided wall column. The net was trained offline and was compared to a continuous method, where DQN showed better control capabilities (Park et al., 2025). However, the performance of a continuous method was found to be better compared to DQN when the model of a bioreactor was available in Rajasekhar et al. (2024). These show that it is not always clear whether a process control task requires discrete or continuous action space, yet since we had a model of the process available, we choose to use a continuous method. As such, our research reviewed and applied a modern training method, namely the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm.

TD3 is an improved algorithm based on the Deep Deterministic Policy Gradient (DDPG) algorithm, containing small albeit significant changes. Despite the existence of TD3, both algorithms have been applied independently in many different case studies. DDPG was applied to control a continuously stirred tank reactor (Panjapornpon et al., 2022) and a polymerization reactor (Ma et al., 2019; Yoo et al., 2021), and compared to a tuned PI controller, where the setpoints are reached faster with the RL algorithm. A model predictive controller was also compared to DDPG for an anaerobic digestion reactor (Mendiola-Rodriguez and Ricardez-Sandoval, 2022), in an electrowinning process (Shi et al., 2020), and in a reactor and a simulation of an industrial heat recovery system (Lin et al., 2023), and in each case the DDPG provided better results. A cascade control structure was designed using PID and DDPG based RL controllers for temperature control in a polyethylene producing fluidized bed reactor, where the use of RL improved the speed of setpoint change (Hong et al., 2024).

Multiple PID configurations along with DDPG and unmodified TD3 were compared to an improved TD3 for the control of an electrolysis reactor, where the proposed algorithm outperformed the other tested methods to maintain temperatures at setpoints (Ma et al., 2024). TD3 algorithm with two actors was tested in two batch processes, where the proposed algorithm was compared with other RL algorithms (DQN, DDPG, TD3) and was shown to be better at precise control of reactor temperature (Joshi et al., 2021). A TD3 based controller was utilized for the control of a polymerization reactor (Ballard et al., 2024), and a modified TD3 algorithm was used as a voltage controller in a solid oxide fuel cell and compared with multiple RL algorithms (Li et al., 2021). Two TD3 based control agents were used for the control of a distillation column model with strong loop interaction. Two different control strategies were tested and compared, and the two agents that had a common reward function provided better control than when each had its own reward function (Yifei and Lakshminarayanan, 2023).

The motivation for our work is to investigate the capabilities of RL-based control to operate SBRs when combined with safety considerations. The proposed control method uses an RL agent on an SBR that carries out exothermic reactions, for the feeding phase of the operation. The reactor model includes the selected runaway criterion, and the trained agent is only aware of this criterion through its reward structure. We trained multiple RL agents with different penalty weight factors in the reward function and tested and compared them in their ability to reach and maintain the temperature setpoint while avoiding a potential runaway. The weight of the quantified runaway criterion was varied over a wide magnitude range to analyze the sensitivity of the agent to its dominance in the reward function. We carried out additional controller tests to check what would happen in case of a coolant failure when the agent was trained using different criterion weighting parameters. The novelty of our work is the following.

- Developing a reinforcement learning-based controller for safe semi-batch reactor operation, where the RL agent can prevent the reactor runaway.
- Integrating a modern thermal runaway criterion, namely Modified Dynamic Condition with RL-based controller to enhance the safe operation of SBRs.
- Comparing the impact of different penalty factors in the reward function of a reinforcement learning algorithm for process control.
- Handling and quantifying the impact of uncertain and non-observable states for RL-based semi-batch reactor operation.

2. Methodology

Reinforcement learning is a suitable approach for decision making in chemical process control, where control actions must be continuously selected under dynamic conditions. Classic controllers, such as PID or MPC may have difficulties with non-linearities, uncertainties, or time-varying dynamics, making reinforcement learning a promising alternative control method (Yoo et al., 2021). In particular, although MPC is a well-established method for reactor temperature regulation, its performance depends on the availability of an accurate process model and on the proper selection of tuning parameters, such as the prediction horizon. In highly nonlinear and time-varying semi-batch reactors, model-plant mismatch and horizon selection may influence the controller's ability to anticipate unsafe operating conditions (Kummer et al., 2020).

Reinforcement learning provides an alternative framework in which the control policy is learned directly from system interactions, thereby reducing the dependence on an accurate process model. Moreover, unlike MPC, which requires repeated online optimization, RL performs most of the computational effort offline during training. As a result, the learned policy can be evaluated rapidly online, which is advantageous when fast control actions are required (Nian et al., 2020). In addition, thermal runaway criteria can be incorporated directly into the reward function. Therefore, the use of RL in this work is motivated not only by its reduced reliance on accurate process models and its ability to handle nonlinear, time-varying, and partially observable reactor dynamics, but

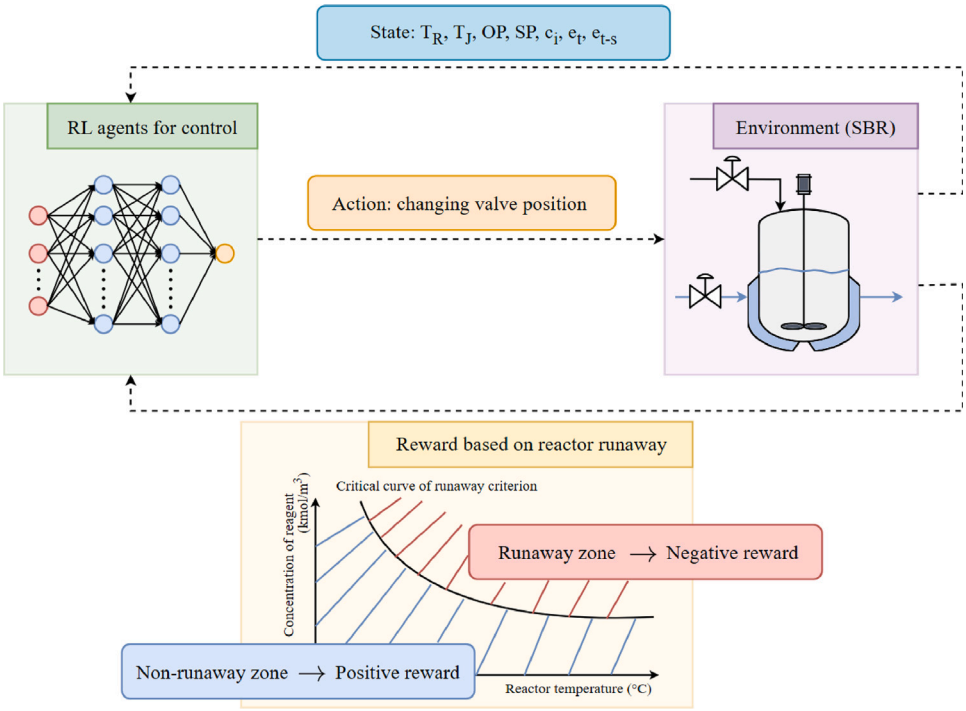


Fig. 1. Reinforcement learning for chemical process control to prevent reactor runaway.

also by its capability to embed the runaway criterion directly into the learning objective. This enables the controller to learn an explicit performance-safety trade-off from interaction, rather than addressing safety solely through model-based online optimization.

Fig. 1 shows the methodology of our work in a reinforcement learning framework, where the agent observes the state, selects an action, receives a reward, and transitions to a new state. The environment is the chemical process, in our case a semi-batch reactor. The state is defined by key process and control system variables, including reactor and jacket temperatures (T_R, T_J), setpoint (SP), concentrations (c_i), manipulated variable (OP), and control errors (e_t, e_{t-s}). Based on these states, the agent selects an action by adjusting the valve position. The reward provides a positive term when the reactor temperature is close to the setpoint, and includes a continuous penalty based on the Modified Dynamic Condition (MDC) runaway criterion. When the criterion becomes positive, the agent operates above the critical curve, and the thermal runaway is possible. In this case, the reward contains a penalty weight factor that motivates the agent to learn to avoid the runaway zone and maintain safe operation.

In this work, we tested the TD3 algorithm as the RL agent that controls the reagent feed valve, and compared its performance with DDPG, and NMPC. TD3 is considered an appropriate algorithm between robustness, continuous control capability, and implementation simplicity for investigating the effect of the MDC-based reward formulation on runaway-aware reactor control. The manipulated variable is continuous, which makes actor–critic algorithms designed for continuous control particularly suitable. Compared to DDPG, TD3 improves training stability and reduces overestimation bias through clipped double critics (Fujimoto et al., 2018). PPO (Proximal Policy Optimization) is typically less sample-efficient than TD3 because it is an on-policy method and requires more training data (Yu et al., 2022), which is disadvantageous given the simulation time available in this work. SAC is effective in stochastic environments and long-term planning tasks, but it encourages stronger exploration through entropy maximization, which introduces additional variability in the control actions (Haarnoja et al., 2018). In a safety reactor control problem, where stable and predictable actuator behavior is preferred, this stronger exploration may be less desirable. Model-based reinforcement learning is also a relevant alternative, since it can improve sample efficiency, but its application requires learning or specifying an environment model, which increases modeling and implementation complexity (Moerland et al., 2023).

2.1. Description of the training algorithm

In our work, we applied the DDPG algorithm, and its improvement, the TD3 algorithm. The following section aims to provide a brief mathematical description of DDPG first in Section 2.1.1 and by building on these basics show the improvements of TD3 in Section 2.1.2.

2.1.1. DDPG algorithm

DDPG is a deterministic method that uses an actor–critic structure, the actor learning the policy while the critic learns the reward function. Both actor and the critic are updated in each episode, but are also assigned a copy of themselves called target networks. The actor target network provides the action approximations (μ') for the calculation of the critic target Q-values (Q'). The critic target network provides the approximated baseline to which the critic Q-network (Q) is optimized, as shown in Eq. (1):

$$L = \frac{1}{N}\sum_i \left(r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}) - Q\left(s_i, a_i|\theta^{Q}\right)\right)^2 \quad (1)$$

where N is the number of samples used for optimization and r_i is the reward received for taking action a_i at state s_i at time i. $\theta^{\mu'}$, $\theta^{Q'}$ and θ^{Q} are the collections of weight parameters of the action target, the critic target and critic networks, and γ is the discount factor. Following the optimization of the Q-network, the critic provides the gradients ($\nabla J(\theta^{\mu})$) for the optimization of the actor network in Eq. (2):

$$\nabla J(\theta^{\mu}) \approx \frac{1}{N}\sum_i \nabla_{\theta^{\mu}} Q(s_i, \mu(s_i|\theta^{\mu})|\theta^{Q}) \quad (2)$$

After the optimizations of the actor and critic networks, the target networks are "soft updated", meaning only fractions of the optimized network values are used to change the weights of the neural network. In this way a stable target is provided for the training while also

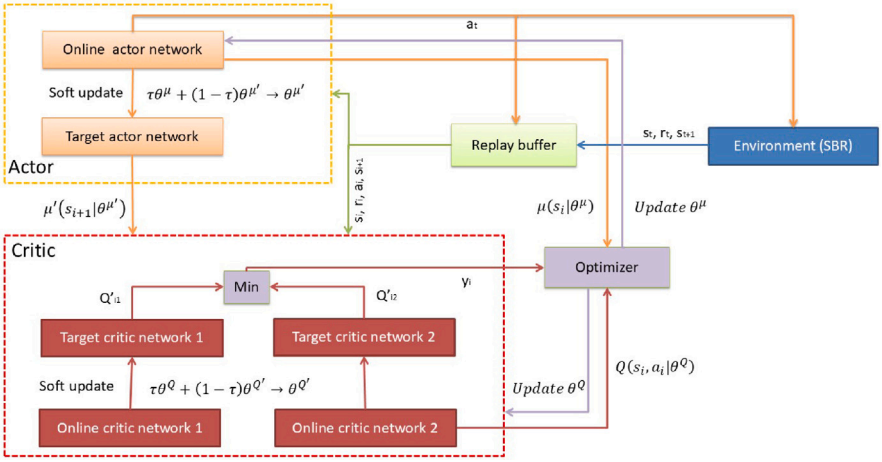


Fig. 2. Structure of the TD3 training algorithm. The actor network selects actions based on the current state. The resulting transitions (state, action, reward, new state) are stored in the replay buffer, from which minibatches are randomly sampled to train the networks. The critic networks estimate the Q-values of the state–action pairs and are updated by minimizing the loss defined in Eq. (5). The actor network is updated using the deterministic policy gradient described in Eq. (2). The target actor and critic networks are softly updated according to Eq. (3), such that their weights are gradually shifted toward the updated actor and critic weights, thereby improving training stability.

converging to the optimum solution. The method in Eq. (3) describes how the "soft update" happens in practice.

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta' \tag{3}$$

where θ' are the weight parameters of the target networks and τ is the weight parameter. As a deterministic policy would always choose the greedy action in any given state, exploration happens by adding a noise ($\mathcal{N}$) to the action of the actor network, shown in Eq. (4):

$$\mu'(s_t) = \mu(s_t|\theta_t^{\mu}) + \mathcal{N} \tag{4}$$

where $\mathcal{N}$ can be an Ornstein–Uhlenbeck process, for example.

Many modern RL methods also make use of a memory buffer, which is a fixed size storage for previously done actions. It contains the (s_t, a_t, s_{t+1}, r_t) tuples, and when the agent is trained, these tuples are randomly sampled and used in the network improvement equations described above. This is done, so the agent learns from discrete samples that have no temporal correlations, making learning more stable. The memory buffer is sampled in batches, and both the memory size and batch size being potential hyperparameters during training.

2.1.2. TD3 algorithm

It has been shown that DDPG can overestimate the Q values (Fujimoto et al., 2018), which can lead to the training of the actor to use overestimated actions, thus propagating the estimation bias. Another issue that can arise is that for every estimation made by the value function, it uses the subsequent state estimate, which leads to an error accumulating. This accumulation leads to the value function estimating the expected return minus the accumulated and discounted error instead of the estimated return, and this leads to overestimation.

Several modifications have been made to solve the mentioned issues, the first being the use of a pair of critic networks and clipping of the value function. This leads to underestimation, but that is preferred to overestimation as it cannot propagate with policy updates, as the low value estimates can be avoided by the policy. The second addition is the delay of the actor updates until the value network is stabilized, preventing the overestimation of a non-optimal policy and then propagating the error. Another issue solved by TD3 is that deterministic policy methods are sensitive to errors of the function approximation because these increase the variance of the target. The used solution was to smooth out the value of the target policy by using an area around the target action for value estimation, which means similar state–action values are used, thus preventing the algorithm from fitting to narrow peaks in value (Fujimoto et al., 2018).

As this algorithm uses two critic networks in parallel (Q_1, Q_2 and Q'_1, Q'_2 respectively), each of them provides a separate target estimate. One of the two estimates will always be greater than the other and using the larger value would induce overestimation. To prevent this, Eq. (5) was modified to use the minimum of the two target estimates, so only creating underestimation (Fujimoto et al., 2018). Fig. 2 shows the makeup of the training algorithm and the connections between each element and the pseudocode of the training is provided in Alg. 1.

$$L = \frac{1}{N}\sum_i \left(r_i + \gamma \min_{j=1,2} Q'_j(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}) - Q\left(s_i, a_i|\theta^{Q}\right)\right)^2 \tag{5}$$

Algorithm 1 TD3 training algorithm

Randomly initialize critics $Q_1(s,a|\theta^{Q,1})$, $Q_2(s,a|\theta^{Q,2})$ and actor network $\mu(s|\theta^{\mu})$ with weights $\theta^{Q,1}$, $\theta^{Q,2}$ and θ^{μ}
Initialize target network Q'_1, Q'_2 and μ' with weights $\theta^{Q',1} \leftarrow \theta^{Q,1}$, $\theta^{Q',2} \leftarrow \theta^{Q,2}$ and $\theta^{\mu'} \leftarrow \theta^{\mu}$
Initialize replay buffer R
for $episode = 1$ *to* M **do**
  Initialize reactor model with fixed reactor temperature (T_R) and reagent moles ($n_{i,0}$)
  Receive initial observation state s_1
  while $t < T$ **do**
    Select action $a_t = \mu(s_t|\theta^{\mu}) + \mathcal{N}_t$
    Execute action a_t and observe reward r_t and observe new state $s_{t+t_{sampling}}$
    Store transition$(s_t, a_t, r_t, s_{t+t_{sampling}})$ in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    $\hat{a} = \mu(s_t|\theta^{\mu'}) + \epsilon, \quad \epsilon \sim clip(\mathcal{N}_t, -c, c)$
    Set $y_i = r_i + \gamma \min_{j=1,2} Q'_j(s_{i+1}, \hat{a})$
    Update critic by minimizing the loss:
      $L = \frac{1}{N}\sum_i \left(y_i - Q\left(s_i, a_i|\theta^{Q}\right)\right)^2$
    if t mod d **then**
      Update the actor policy using the sampled policy gradient:
        $\nabla_{\theta^{\mu}} J \approx \frac{1}{N}\sum_i \nabla_{\mu} Q\left(s, a|\theta^{Q}\right)|_{s=s_i, a=\mu(s_i|\theta^{\mu})}$
      Update target networks:
        $\theta^{Q',i} \leftarrow \tau\theta^{Q,i} + (1-\tau)\theta^{Q',i}$
        $\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$
    end if
  end while
end for

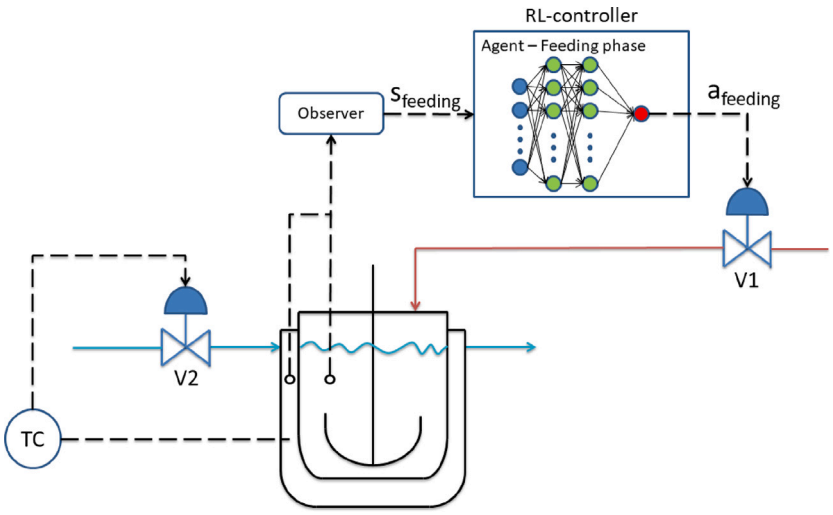


Fig. 3. Structure of the reactor. *Source:* The image was taken from Sass et al. (2022).

2.2. Definition of the control objective

The structure of the control system is illustrated in Fig. 3, where two separate operating phases are required for the operation of the reactor. In the first phase, labeled as "feeding" phase, the reagent is continuously fed into the reactor until the pre-defined batch volume or weight has been added to the reactor. In the second phase, labeled as "mixing" phase, the reactor is maintained at the setpoint reactor temperature and stirred to maintain homogeneity. Due to the sequential operation requiring different manipulated variables and being separated in time, two different control loops need to be designed. The objective in both defined operation phases is to maintain the temperature of the reactor at the provided setpoint, thus maximizing the conversion. In the feeding phase, it is achieved by keeping the jacket flow at a pre-defined setpoint while modifying the opening percentage of the reagent feed valve (V1). In the mixing phase, the reagent feed valve is shut off and the jacket coolant flow rate (V2) is manipulated. However, in the mixing phase the reaction is already slowly slowing down, hence thermal runaway is a far less likely to be an issue. Due to this, we have elected to only focus on the design of the feeding phase controller, which manipulates V1.

Because of the increasing liquid volume in the reactor and the varying concentration of the accumulated reagent, the system can present high levels of non-linearity. Based on this, the use of non-linear control methods is advised, as such we propose an RL-based controller to effectively handle the reactor non-linearities. The RL-controller is made up of a neural network with trainable edge weights. The agent receives information from the environment (the reactor) in the form of a state tuple: s, and takes an action (a) to change the manipulated variable. The optimization of such a controller is done by letting the agent interact with the environment and training it to learn from the received reward values.

In the feeding phase, the RL controller adjusts the feed rate to achieve the operating temperature setpoint starting from an initial temperature. The controller receives the state tuple shown in Eq. (6) as input (with min–max normalized states) to define the necessary actions. The agent takes action at 50 s intervals. The action in this case is the change in actuator position of V1 inlet valve, which was bounded in the $[-5\%; 5\%]$ range to prevent drastic changes in the valve that could damage the equipment.

$$s = \{T_R, T_J, OP, c_i, \dots, c_{NR}, e_t, e_{t-s}, (T_R - T_J)\} \tag{6}$$

In the reward function, a thermal runaway criterion is applied, namely the Modified Dynamic Condition (MDC) presented by Kummer and Varga (2019). The criterion in Eq. (7) was evaluated across multiple case studies, and demonstrated that it does not produce false negatives—cases where runaway occurs but no indication is given. The reactor temperature is determined by the balance between the generated heat flow (q_{gen}) and the removed heat flow (q_{rem}). So, the reactor temperature increases when the generated heat flow is higher than the removed heat flow, and decreases otherwise. The MDC does not only consider the temperature sensitivity of heat generation and heat removal, but also the effect of concentrations changes (Kummer and Varga, 2019).

$$\frac{\partial q_{gen}}{\partial T_R}\Big|_c + \frac{\partial m}{\partial c}\Big|_{T_R} \leq \frac{q_{gen}}{q_{rem}} \frac{dq_{rem}}{dT_R} \tag{7}$$

For the applied reaction system, which is described in Appendix A, the derivatives of the variables are given as follows. The generated heat flow can be described in Eqs. (8) and (9), where $\beta = -\frac{\Delta H_r}{\rho c_p}$ (Kummer and Varga, 2019).

$$q_{gen} = \beta r_R \tag{8}$$

$$q_{gen} = \beta k_0 \exp\left(\frac{-E_A}{R_g T_R}\right) c_A c_B \tag{9}$$

The partial derivative of the generated heat flow with respect to the reactor temperature is described in Eq. (10).

$$\frac{\partial q_{gen}}{\partial T_R}\Big|_c = \beta k_0 \exp\left(\frac{-E_A}{R_g T_R}\right) c_A c_B \left(\frac{E_A}{R_g T_R^2}\right) \tag{10}$$

The removed heat flow is described in Eq. (11), and its derivative with respect to the reactor temperature is shown in Eq. (12).

$$q_{rem} = \frac{UA(T_R - T_J)}{V \rho c_p}, \tag{11}$$

$$\frac{dq_{rem}}{dT_R} = \frac{UA}{V \rho c_p} \tag{12}$$

The derivatives of the mass balance function with respect to the concentrations are given in Eq. (13).

$$\frac{\partial m}{\partial c}\Big|_{T_R} = -\left(\frac{\partial r_R}{\partial c_A} + \frac{\partial r_R}{\partial c_B}\right)\Big|_{T_R} = -k_0 \exp\left(\frac{-E_A}{R_g T_R}\right)(c_A + c_B) \tag{13}$$

Substituting into Eq. (7), dividing by $k_0 \cdot \exp(-\frac{E_A}{R_g T_R})$ and simplifying and reorganizing gives:

$$\beta c_A c_B \left(\frac{E_A}{R_g T_R^2} \right) - (c_A + c_B) - \frac{\beta c_A c_B}{(T_R - T_J)} \leq 0 \tag{14}$$

The left side of Eq. (14) can be used as the definition of the MDC value:

$$MDC = c_A c_B \frac{E_A}{R_g T_R^2} - \frac{c_A + c_B}{\beta} - \frac{c_A c_B}{T_R - T_J} \tag{15}$$

This MDC value can be used as an indicator for thermal runaway. When MDC values are greater than 0, the system has a potential for thermal runaway. When MDC is below 0, the system is stable, as heat dissipation exceeds heat generation. This MDC value can augment the reward function that is used for the training for the reagent feeding agent. A general reward function presented in Eq. (16) does not contain the thermal runaway penalty in its original state, only a reward for maintaining the reactor temperature within an δ boundary of the setpoint. Additionally, there is also a penalty for the temperature not being within the previously described boundary, which is the negative absolute value of the error between the reactor temperature and the setpoint.

$$r_t = \begin{cases} +10 & \text{if} \quad |T_{R,t} - SP| \leq \delta \\ -|T_{R,t} - SP| & \text{otherwise} \end{cases} \tag{16}$$

In our work, this reward function is augmented by the MDC awareness component described in Eq. (17) as:

$$r_t^* = \begin{cases} r_t - \text{MDC} \cdot \lambda & \text{if} \quad \text{MDC} > 0 \\ r_t & \text{otherwise} \end{cases} \tag{17}$$

where λ is a penalty weight factor that is used to influence the strength of the MDC penalty. One of our tasks in our research was to investigate the influence of this factor on the learned agent behavior, and through that the temperature profiles and safe operation.

3. Results and discussion

To test the usefulness of the proposed reward function and the capabilities of the trained networks, we have used the semi-batch reactor structure shown in Section 2.2. This study is based on a dynamic reactor representation that has been used in the literature as a benchmark system for reactor control and safety studies. The applied formulation captures the main reaction kinetics and thermal dynamics of the process and has been validated previously in both experimental and simulation studies reported in the literature (Kummer et al., 2020; Sass et al., 2022). The system was modeled and parameterized according to Appendix A.

The reactor dynamics were simulated using a first-principles dynamic model discretized with an explicit Euler integration scheme, where we investigated the effect of the numerical step size on the solution accuracy. Simulations were repeated with time steps of 1.0, 0.5, 0.1, 0.05, and 0.01 s. The state trajectories showed only negligible differences between 0.1, 0.05, and 0.01 s, the selected time step of 0.1 s was considered sufficiently accurate, while also keeping the computational cost acceptable for the performed calculations. The RL agent selected a new action every 50 s, and the reward was evaluated on the same timescale. To provide enough time for the agent to take actions and learn the optimal policy, an episode is 4 h long and it is terminated early if the agent finishes feeding in the required amount of reactant ($n_{A,feed}$).

We have previously investigated the same semi-batch reactor using advanced model-based control methods, including nonlinear model predictive control (NMPC), where safe operation and thermal runaway avoidance were addressed within a model-based optimization framework (Kummer et al., 2020). Reinforcement learning-based operation of the same reactor with DDPG has also been explored previously, including a comparison of RL and NMPC (Sass et al., 2022).

Table 1
Parameters of the trained neural network.

Learning rate	10^{-4}	Hidden layers	2
τ	10^{-3}	Neurons per layer	256
γ	0.99	Batch size	512
Memory size	10^5	Number of episodes	10^4

Compared to these earlier studies, the present work investigates how the agent learns when the Modified Dynamic Condition (MDC) is applied directly into the reward function. By incorporating the MDC term into the reward function, the learned control policy results in a more gradual temperature increase, thereby reducing the risk of thermal runaway while maintaining setpoint tracking with minimal or no overshoot.

In this study, the reinforcement learning controller was implemented using the TD3 algorithm. To assess the effectiveness of the proposed reward formulation, in Section 3.1 the TD3-based controller was compared with a DDPG controller and with an NMPC-based benchmark. This comparison enables the evaluation of the proposed MDC-augmented reward function not only against another actor–critic reinforcement learning method, but also against a model-based predictive control strategy.

The neural network structure of the TD3 RL agent was first optimized by trial-and-error testing, followed by the training of the control task with varying MDC factors. The parameters of the used networks are presented in Table 1.

The agent was trained for 10 000 episodes, which started with a 750 episode long pretraining section. During this training period, an episode started with the opening of the feed valve at 50%. This was done because the agent takes random and incorrect actions at the beginning of the training, thus if the valve opening starts at 0%, the agent will not be able to see and store experiences of the effects of closing the valve, i.e. to see the effects of negative actuator position changes. This allowed us to partially fill the memory buffer with useful experiences before the training was switched over to the 0% valve opening case, which is the standard mode of operation the agent was designed to learn.

3.1. Penalty weight factor tests

We have conducted tests at $\lambda = 0, 1, 10, 100, 750, 1000, 1250, 2000, 3000, 10\,000$ factor values in the reward function as a sensitivity analysis to learn when the MDC values start to influence the rewards, and through that the operation of the agent.

We used the MDC value directly without normalization, so the parameter λ does not have a direct physical meaning. Instead, it acts as a weighting factor that determines how strongly runaway avoidance influences the reward compared to temperature setpoint tracking. Accordingly, λ is not interpreted as a physical process parameter, but as a tunable reward-scaling coefficient.

A continuous MDC-based penalty over the entire operating region was not used, because we also want the agent to remain primarily motivated to reach and maintain the temperature setpoint. The applied penalty is therefore continuous only in the unsafe region, where larger positive MDC values produce a larger penalty. Since positive MDC values indicate a potential runaway region, the penalty is applied only when the MDC becomes positive, while negative MDC values are treated as thermally stable operation.

3.1.1. Investigation of TD3 controller performance

The results can be separated into the [0, 1, 10, 100] part, where the effect of the MDC and its factor is negligible, which is shown in Fig. 4. Each of the trained agents follows closely the same trajectory, with the received rewards changing minimally. As the trained agents follow the same temperature trajectories, the MDC values are closely the same. Any visible slight deviations are from the small variations of the trained

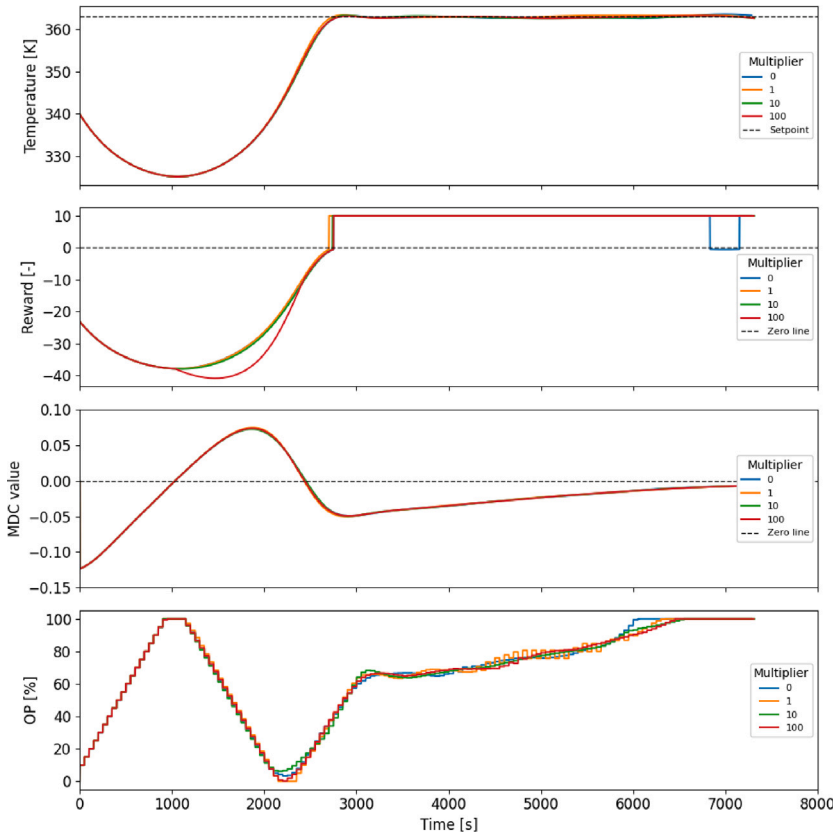


Fig. 4. Temperature, reward, MDC value, and control action plots for different factor values using TD3.

weights, as training samples were selected stochastically. Each version is a robust controller, with only minimal errors. The second factor subsection we have created is the [100−10 000] range. It was selected because from 750 onward there is a visible change in the temperature trajectories. This can be explained by the fact that, at $\lambda = 750$, the MDC-based negative reward becomes stronger than the positive reward for setpoint tracking. As a result, when the agent enters a region with positive MDC, the penalty has a larger effect on the total reward than the benefit of remaining close to the setpoint. Consequently, the MDC term starts to influence learning more strongly, and the agent changes its behavior to avoid positive MDC regions more clearly.

In Fig. 5 it can be seen that at 750 there is a visible change in the reward plot, showing a large dip in the reward received by the agent. Paired with the MDC plot, it shows that even lower positive MDC can influence agent behavior at this factor magnitude. At 1000 and 1250, the agent starts to deviate significantly from the baseline where the agent is unguided by the potential for thermal runaway. These cases show a slower temperature increase because the reactant A is fed into the reactor at a lower rate, slowing down the exothermic reaction and thus the change in temperature. This slower change in concentrations and temperature results in lower positive MDC values, however, it also extends the time period where the system is in a potential runaway situation.

The corresponding control actions further confirm the observed changes in temperature and MDC trajectories. For higher λ values, the agent applies smoother and less aggressive actuation, avoiding rapid increases in feed rate. This leads to a more gradual temperature rise and reduced peak MDC values, indicating a lower instantaneous runaway risk. At the same time, the reduced aggressiveness of the control input results in slower process dynamics.

In each case in Fig. 5 when the MDC value never crosses 0 the agent is incapable of heating up the reactor. This is a logical outcome from the fact that for this kind of semi-batch reactor the heatup comes from a tightly-controller and moderated reactor runaway. If the MDC value is overvalued, this runaway cannot take place. However, the unmoderated and over-moderated system are two edge cases, selecting proper λ values can help balance between moderation and cycle-time of the processing of the reactor batch.

This behavior reflects a trade-off between safety and productivity. Increasing λ improves safety by reducing peak MDC values and enforcing more stable and predictable control actions, but it also decreases productivity, as the slower heating rate delays reaching the desired operating temperature and extends the batch time. In addition, unsuitable λ values can lead to poor learning. If λ is too small, the safety penalty is too weak, so the agent does not pay enough attention to runaway risk. If λ is too large, the penalty dominates the reward, making the agent too conservative and, in extreme cases, unable to control the temperature effectively.

The plot in Fig. 6 shows the 10-element moving average of the episodic rewards for the $\lambda = 100, 1000, 2000, 10\,000$ scenarios. The averaging was performed to filter out the small variations in the reward values, which could hinder the readability of the plot. It is visible that as the multiplier increases, the training of the agent is slower to stabilize due to the unobserved MDC value. The relationship between the reward and the agent actions becomes less straightforward, inducing cases where the agent training diverges, but all of these cases are observed to be recoverable. For every tested λ value, learning stabilizes again and resumes its trajectory toward the optimal policy. The numerical values of the episodic reward are not relevant in this case study because the λ values heavily influence the rewards, and the goal of our investigation was to observe the underlying trends in the agent training.

Quantitative measures are provided in Table 2, including the maximum MDC value, the integral of absolute error (IAE), the time spent in the runaway-indicated region, and the time required to reach the setpoint for each tested λ value. The results show that the penalty

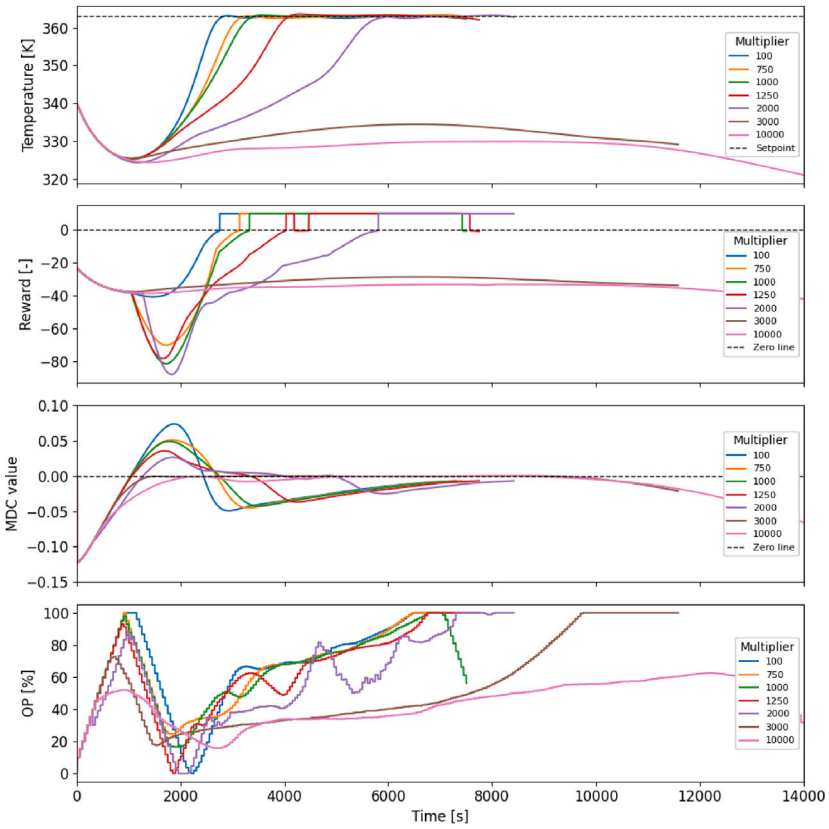


Fig. 5. Temperature, reward, MDC value, and control action plots for different factor values using TD3.

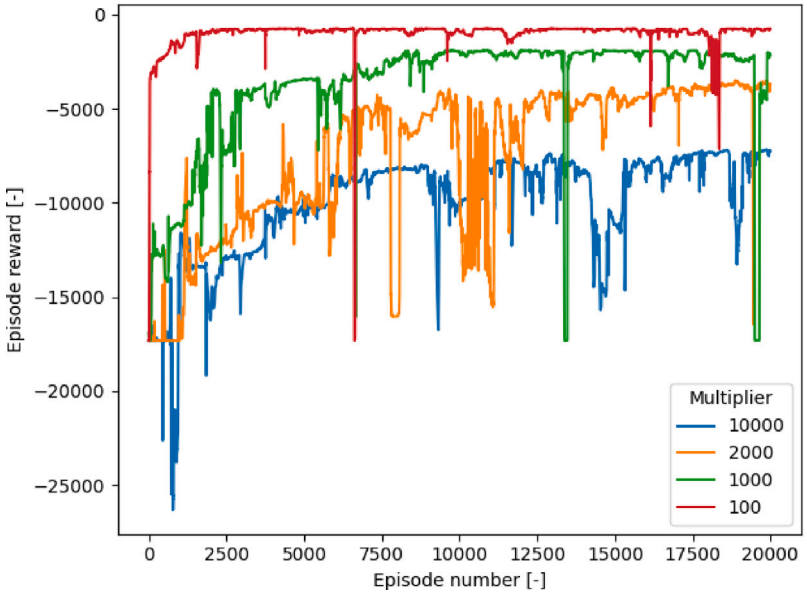


Fig. 6. Moving average of the episodic reward for selected penalty factor cases using TD3.

factor has a clear effect on the learned policy. For smaller λ values, the controller reaches the setpoint faster, and achieves lower IAE values, but the maximum MDC values are higher. As λ increases, the controller becomes increasingly conservative, the maximum MDC value decreases, while both the IAE and the time required to reach the setpoint increase significantly, while the time to reach the setpoint becomes significantly longer. For very large λ values, the agent is no longer able to reach the setpoint within the simulation horizon. This also shows that stronger penalization can reduce unsafe operation, but at the cost of reduced control performance.

3.1.2. *Investigation of DDPG controller performance*

The same tests were also performed using the DDPG algorithm to investigate the effect of the MDC-based reward. For consistency with the TD3 experiments, the neural network architecture, number of training episodes, replay buffer size, batch size, discount factor, and learning rates were kept identical.

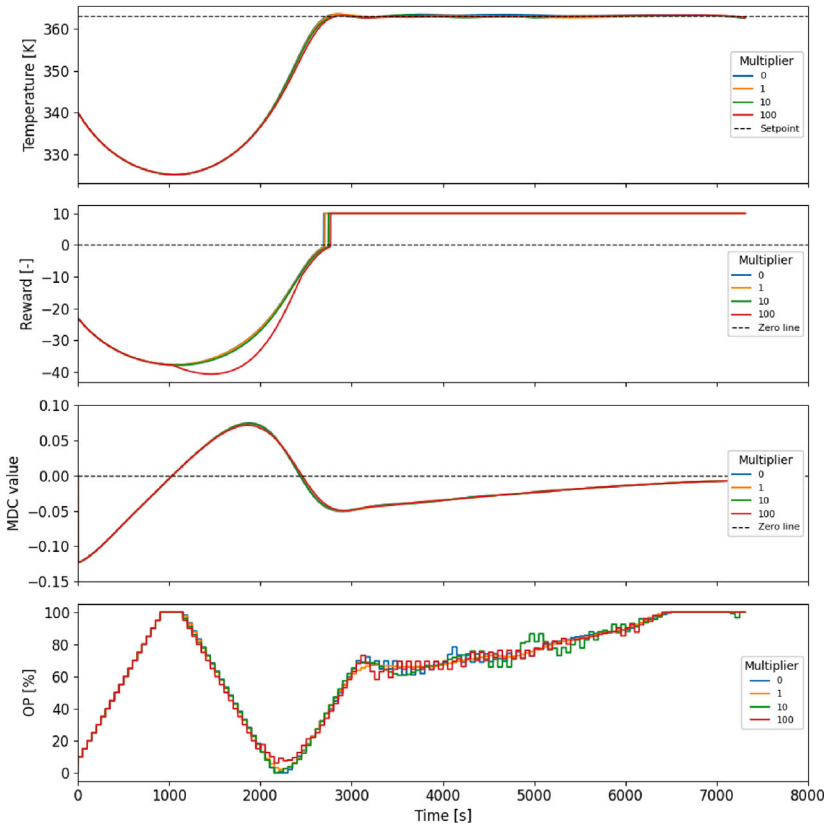


Fig. 7. Temperature, reward, MDC value, and control action plots for different penalty factor values using DDPG.

Table 2
Quantitative performance measures for different penalty weight factors using TD3.

λ	Max MDC	IAE	Runaway indicated time [s]	Time to reach SP [s]
0	0.0725	40 066	71.08	1393.82
1	0.0743	40 555	70.50	1366.52
10	0.0718	41 608	71.87	1399.06
100	0.0733	41 333	71.07	1392.52
750	0.0508	44 919	84.58	1610.32
1000	0.0489	48 796	88.47	1734.85
1250	0.0349	60 164	124.14	2201.04
2000	0.0288	82 022	189.04	3646.81
3000	0.00004	363 544	29.02	–
10 000	0.00004	492 747	43.09	–

Similarly to the TD3 case, the DDPG results for $\lambda = 0, 1, 10$, and 100 show nearly overlapping temperature trajectories, as presented in Fig. 7. The reward and MDC profiles remain very close, indicating that the MDC-based penalty has only a minor influence in this range.

For larger penalty factors, the effect of the MDC term becomes more visible, as shown in Fig. 8. As λ increases, the DDPG controller becomes more conservative. For moderate penalty weights, the controller can still reach the setpoint, although the temperature response becomes slower. For larger values, the heat-up is strongly delayed, and in some cases the temperature remains far below the setpoint during the investigated time horizon. The MDC trajectories show the same trend. At lower and intermediate penalty weights, the MDC still becomes positive, but the positive peaks decrease as the penalty weight increases. This shows that a larger MDC penalty helps the controller reduce runaway-indicated operation.

The valve profiles also support this observation. For smaller penalty weights, the valve opening increases quickly at the beginning and later approaches the upper bound. This gives faster heating, but it also leads to higher positive MDC peaks. For $\lambda = 10\,000$, the controller keeps the valve opening much lower for most of the simulation and increases it only slowly. As a result, the MDC values are reduced, but the controller cannot reach the setpoint.

The 10-element moving average of the episodic rewards for selected DDPG cases is shown in Fig. 9, and the curves are used to evaluate the learning trend and stability. The DDPG results indicate that larger λ values make the learning process more challenging, as the corresponding reward curves stabilize at lower values and show more frequent drops.

Quantitative measures for the DDPG controller are provided in Table 3, including the maximum MDC value, the integral of absolute error, the time spent in the runaway-indicated region, and the time required to reach the setpoint for each tested λ value. The results show that the penalty factor has a clear effect on the learned DDPG policy. For smaller λ values, up to $\lambda = 750$, the controller reaches the setpoint relatively quickly and achieves low IAE values. In this range, however, the maximum MDC values remain relatively high, which indicates that the MDC penalty has only a limited effect on the closed-loop behavior.

As λ increases further, the controller becomes more conservative. From $\lambda = 1000$, the maximum MDC value decreases clearly, while both the IAE and the time required to reach the setpoint increase. For the largest penalty weight, $\lambda = 10\,000$, the maximum MDC value is almost zero. These results show that stronger MDC penalization can reduce runaway-indicated operation, but excessive penalization leads to poor tracking performance.

3.1.3. *Investigation of NMPC performance*

In addition to the reinforcement learning controllers, an NMPC-based benchmark was implemented to compare the learned RL policies with a model-based predictive control strategy. The NMPC cost function included a temperature tracking term and an MDC-based penalty term to drive the reactor temperature toward the setpoint while avoiding runaway-indicated operation. Consistently with the RL

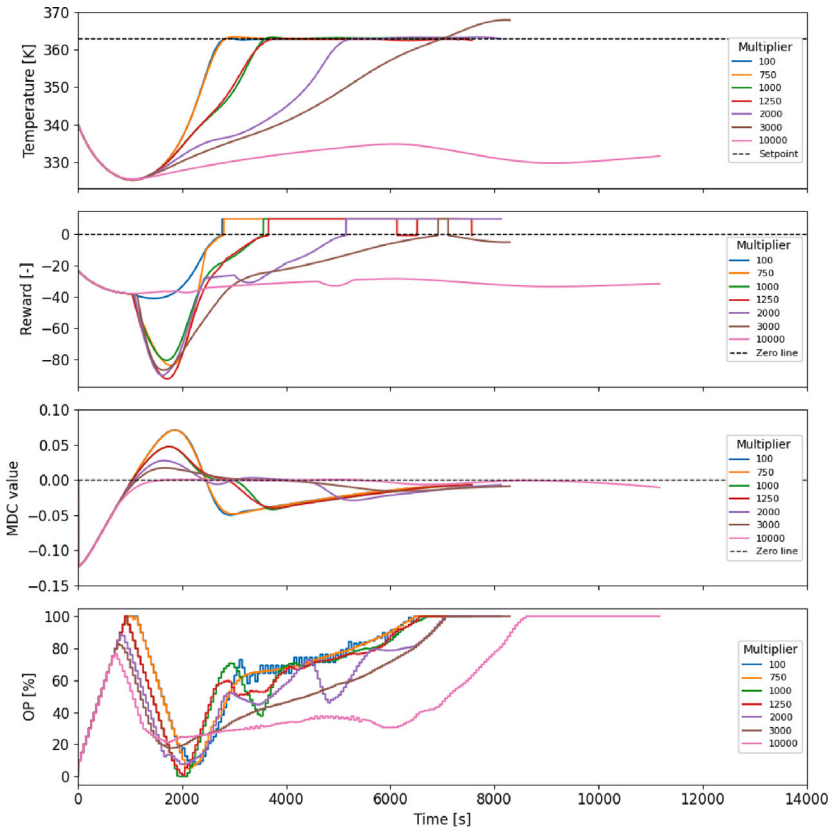


Fig. 8. Temperature, reward, MDC value, and control action plots for different penalty factor values using DDPG.

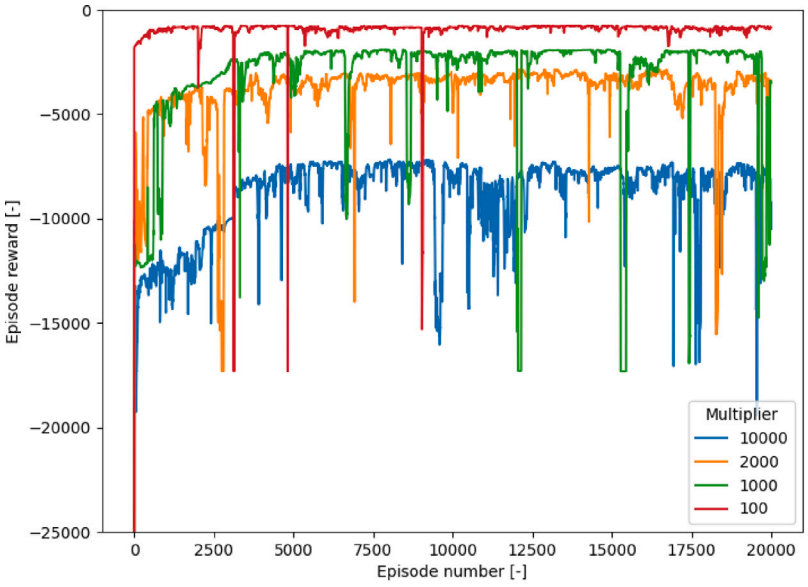


Fig. 9. Moving average of the episodic reward for selected penalty factor cases using DDPG.

reward formulation, the MDC penalty was applied only for positive MDC values.

To ensure a fair comparison, the NMPC setup was matched to the reinforcement learning framework wherever possible. The optimized variables were the future changes in the valve opening, corresponding to the RL action definition. Therefore, the valve opening was constrained between 0 and 100%, while each change in valve opening was limited to $[-5\%, 5\%]$, as in the RL controllers. The action interval was set to 50 s, identical to the RL controllers. The prediction horizon was set to 3000 s to capture the slower thermal dynamics and the possible development of thermal runaway in advance, while the control horizon was set to 750 s.

The trajectories obtained with the NMPC controller for the lower penalty-weight range are shown in Fig. 10. The reactor temperature first decreases and then gradually increases toward the setpoint. After reaching the setpoint region, the temperature shows visible oscillations

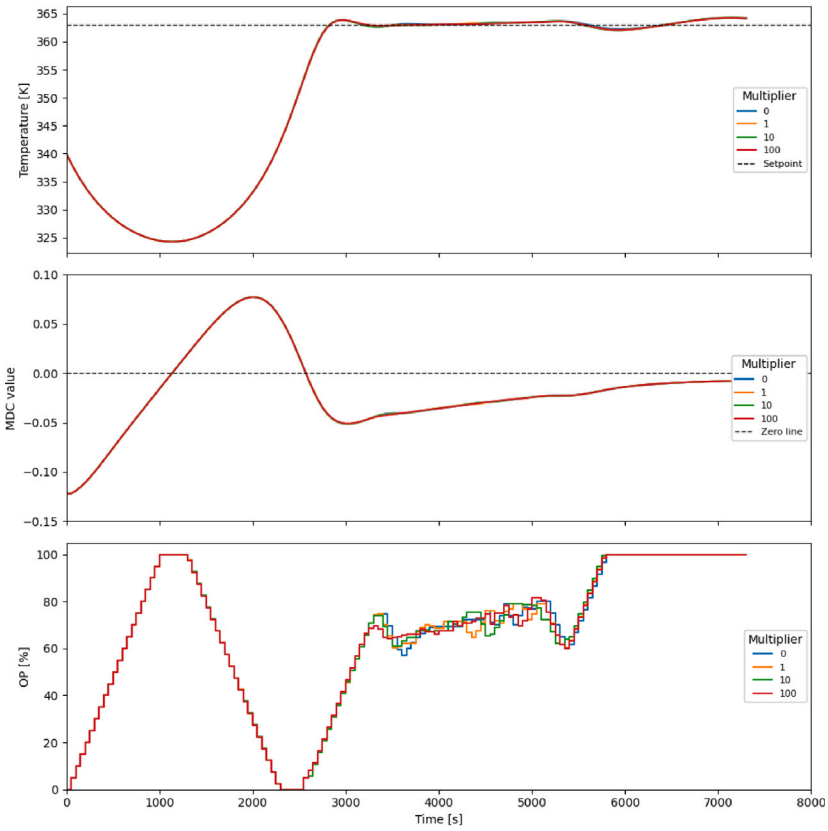


Fig. 10. Temperature, reward, MDC value, and control action plots for different penalty factor values using NMPC.

Table 3
Quantitative performance measures for different penalty weight factors using DDPG.

λ	Max MDC	IAE	Runaway indicated time [s]	Time to reach SP [s]
0	0.0743	41 016	70.55	1364.05
1	0.0737	41 189	70.97	1372.20
10	0.0735	41 582	71.00	1394.46
100	0.0711	41 629	72.02	1406.68
750	0.0708	41 518	71.96	1414.76
1000	0.0474	51 025	97.26	1870.45
1250	0.0474	51 347	97.89	1921.68
2000	0.0275	75 299	132.56	2920.59
3000	0.0172	103 127	130.32	3944.69
10 000	0.00036	343 186	118.66	–

around the reference value, indicating that the controller continues to correct the thermal response during the later phase of the simulation. The MDC trajectory becomes positive during the heating phase and remains above zero for a relatively long interval before returning to the safe region.

The NMPC trajectories for the higher penalty-weight range are shown in Fig. 11. For most λ values, the temperature trajectories are still similar: the temperature first decreases, then increases toward the setpoint, and stays close to the reference value. The most distinct behavior is observed for $\lambda = 10\,000$, where the temperature rise is slower and the setpoint is reached later. The MDC trajectories show that increasing the penalty weight reduces the positive MDC peak, but only to a limited extent. Even at larger λ values, the MDC still becomes positive.

The OP profiles also show the effect of the penalty weight. For lower values in this range, the valve opening first increases rapidly, then decreases close to the lower bound, and later increases again toward high values. For $\lambda = 10\,000$, the valve movement becomes more conservative, the opening is reduced earlier and increases more slowly afterwards. This explains the lower MDC peak, but also the slower temperature response.

The quantitative results with NMPC are summarized in Table 4. The table includes the maximum MDC value, the integral of absolute error (IAE), the time spent in the runaway-indicated region, and the time required to reach the setpoint for each tested λ value. The results show that the NMPC response changes only slightly for most penalty weights. For λ values between 0 and 3000, the maximum MDC remains close to 0.075, and the runaway-indicated time stays around 1450 s. This means that increasing the penalty weight in this range does not strongly reduce the runaway-indicated operation. At the same time, the IAE and the setpoint-reaching time increase slightly, showing a small deterioration in tracking performance.

The largest change can be observed for $\lambda = 10\,000$. In this case, the maximum MDC decreases to 0.0650, which indicates a lower runaway tendency. However, this improvement is obtained with a higher IAE and a longer time to reach the setpoint. Overall, the NMPC results show that larger MDC penalty weights can reduce the maximum MDC, but the effect is moderate and comes with slower temperature tracking.

3.1.4. Comparative evaluation of TD3, DDPG, and NMPC

The results of the TD3, DDPG, and NMPC controllers show the same general trade-off between temperature tracking and runaway avoidance. For all three approaches, small penalty weight factors lead to faster heat-up and lower tracking error, but the MDC values remain relatively high. As the penalty weight is increased, the controllers become more conservative, which reduces the maximum MDC value but also slows down the temperature response. Therefore, the choice of λ directly determines the balance between productivity and safety.

Among the reinforcement learning controllers, TD3 showed a clear response to the MDC-based penalty. At intermediate penalty weights, the maximum MDC decreased noticeably while the controller was

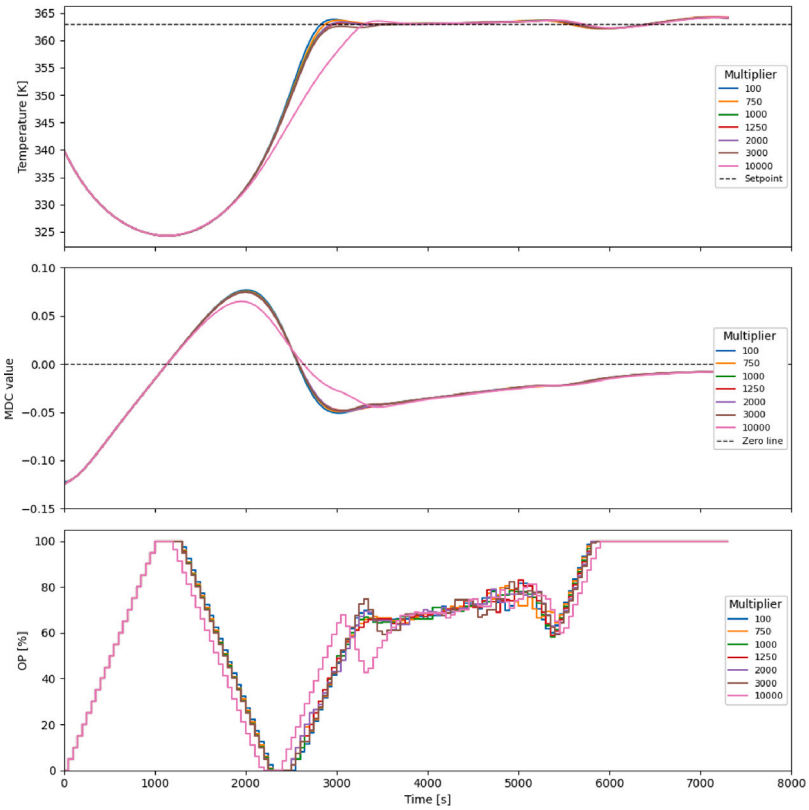


Fig. 11. Temperature, reward, MDC value, and control action plots for different penalty factor values using NMPC.

Table 4
Quantitative performance measures for different penalty weight factors using NMPC.

λ	Max MDC	IAE	Runaway indicated time [s]	Time to reach SP [s]
0	0.0772	84 880	1450	2800
1	0.0772	84 982	1450	2800
10	0.0770	84 927	1450	2800
100	0.0770	84 927	1450	2850
750	0.0757	85 322	1450	2850
1000	0.0752	85 391	1450	2900
1250	0.0747	85 480	1450	2950
2000	0.0747	85 472	1450	2900
3000	0.0747	85 747	1450	3000
10 000	0.0650	90 267	1500	3000

still able to reach the setpoint. However, at very large λ values, TD3 became too conservative and could no longer reach the setpoint within the simulation horizon. DDPG showed a similar trend, but the transition was more gradual. For small and moderate penalty weights, the DDPG trajectories remained close to the low-penalty cases, while larger penalty weights were required to strongly reduce the MDC peak.

The NMPC controller showed a different behavior. The maximum MDC changed only slightly for most penalty weights, and the runaway-indicated time remained almost constant over a wide range of λ values. A visible reduction in the maximum MDC was obtained only for the largest penalty weight. However, this also increased the IAE and delayed the setpoint reaching. This suggests that, in the tested formulation, the NMPC controller was less sensitive to the MDC penalty than the reinforcement learning controllers.

Overall, the variation of λ highlights how strongly the controller prioritizes runaway avoidance to temperature tracking and faster heat-up. This effect was most pronounced for the reinforcement learning-based controllers, particularly TD3, which exhibited a clear sensitivity to

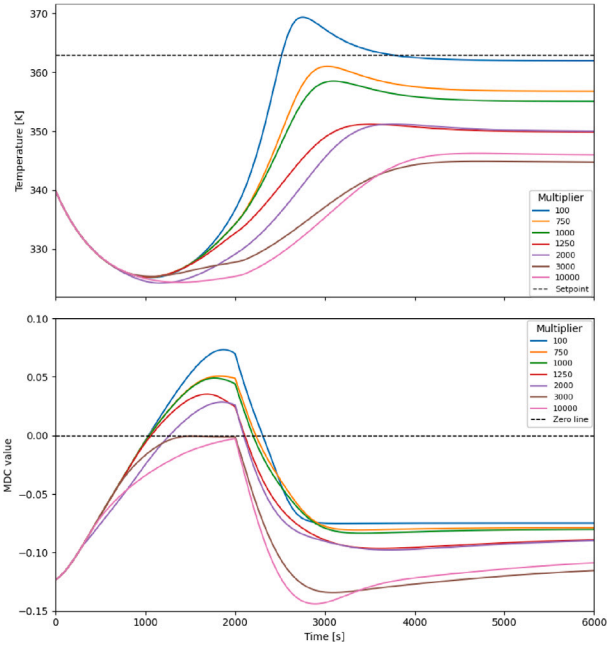


Fig. 12. Temperature, reward, and MDC value plots for different factor values with cooling failure.

the MDC penalty, while DDPG showed a more gradual transition. In contrast, the NMPC controller was less sensitive to changes in λ over the range, requiring larger penalty weights to achieve a noticeable reduction in the MDC. For the RL controllers, lower λ values favor faster setpoint reaching and higher productivity, while higher λ values lead to more conservative and safer operation. In practical

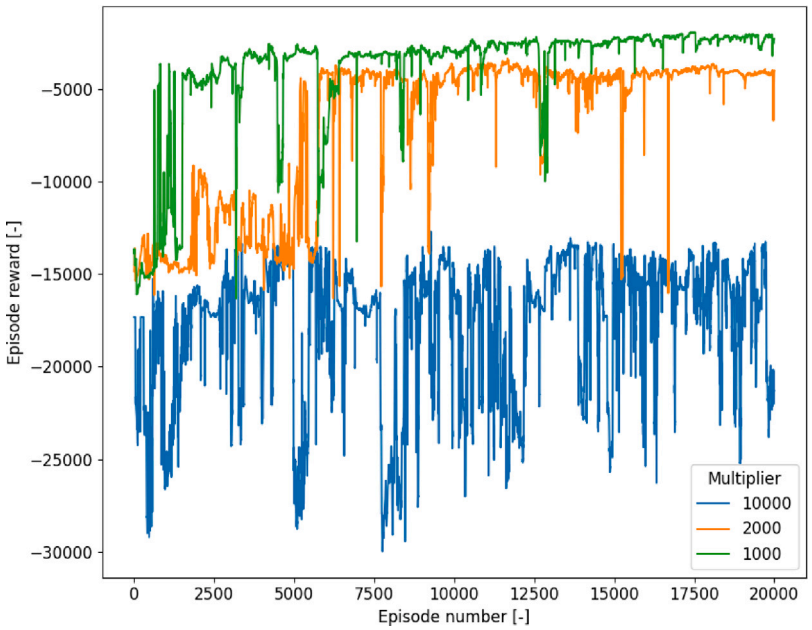


Fig. 13. Moving average of the episodic reward for $\lambda = 1000, 2000, 10\,000$ multiplier cases without concentration.

applications, this hyperparameter could be tuned systematically using optimization-based approaches, such as Bayesian optimization.

3.2. Cooling loss severity tests

Following that, we have performed tests to check the effect of feeding profile on thermal runaway potential and severity. To do this, we have set the valve actuator position of the jacket coolant to 0%, with the simulated operation continuing for a 90 s time period, where the agent keeps operating normally. In addition to the 90 s period, we also investigated longer failure durations of 180 s and 360 s to assess the effect of more severe disturbances, as shown in Appendix B. This was followed by a shutdown of the reagent feed inlet as an approximation of the operators having a delayed reaction to total coolant failure and shutting down the reactor. The often used solution of quickly draining the reactor in such cases was not modeled.

Fig. 12 shows that a penalty weight factor of 100 was not enough to prevent a spike in temperature, but penalty weight factors of 750 or 1000 are capable of preventing a temperature overshoot.

As illustrated in Fig. 5, the temperature profile differs only slightly compared to the quasi-ideal case with a penalty weight factor of 0 (or in this comparison 100). This means that in this case preventing potential thermal runaway does not require a major deviation from the optimal operation. Additionally, as the value of the factor is increased, the system goes closer to being asymptotic due to the cooling being sufficient to prevent underdamped process behavior.

3.3. MDC awareness without concentration

As shown by the previous tests, the agents trained with sufficiently large λ factors were able to prevent severe thermal runaway. However, these agents are based on the assumption that we are capable of measuring the concentrations inside the reactor. Usually concentrations are measured infrequently during the processing of a single batch, which means that with enough processed batches, a concentration predictor model can be designed. For future practical implementation, concentrations could be estimated using a soft sensor or state estimation methods such as a Kalman filter. This would allow the agent to indirectly access information about unmeasured states and improve its performance without direct concentration measurements. An alternative to that is to train the agent with only inputs that can be measured frequently. We have tested this by using the following modified state tuple format for training and operation, as shown in Eq. (18):

$$s = \{T_R, T_J, OP, e_t, e_{t-s}, (T_R - T_J)\} \tag{18}$$

The results from the training show that for low factor values, there is close to no difference in the way the agent learns, based on the reward functions. As the factor value increases, the training becomes completely incoherent, the agent loses the capability to learn an optimal operation. In Fig. 13, we show the moving average of the episodic rewards of the agents for 3 different penalty weight factors. It can be seen that at 2000 the agent still learns at a slower rate, but at 10 000 the agent is incapable of improving. It could be due to the fact that the MDC value uses concentration heavily, and the agent is incapable of learning something it cannot observe at all.

This increased reliance on concentration measurements is reflected in the temperature profiles. To illustrate the effect, we compared the profiles with and without concentrations included in the state variables in Fig. 14. The temperature profile comparison shows that both of the $\lambda = 1000$ cases closely resemble each other. The MDC value profiles show a small deviation at maximum MDC value and in the settling profile between 2000 and 4000 s. Since this difference is not significant in the temperature profiles, we consider it as a result of the training arriving at a different optima, rather than as a result of the concentration as network input. At $\lambda = 2000$ the temperature trendlines are very similar in shape, but there is a visible gap in the profiles. Without concentration, the agent reaches the temperature setpoint faster but with overshoot and deviating from the setpoint by the end of the feeding phase. In the MDC plot, it can be seen that both agents have a lower MDC maximum and during settling, both agents operate closely at $MDC = 0$. At $\lambda = 10\,000$, the concentration aware agent is capable of operating the process nearly along $MDC = 0$ while the MDC unaware agent is incapable of temperature control. This reinforces the finding in Fig. 13 that the agent was incapable of learning.

These finding highlight that if an RL agent implementation is done on a real life process, care must be taken to work with a state set that is properly selected for the task, as the effectiveness of a method can depend sharply on it. Additionally, the λ factor should be a hyperparameter that should be subject to basic optimization.

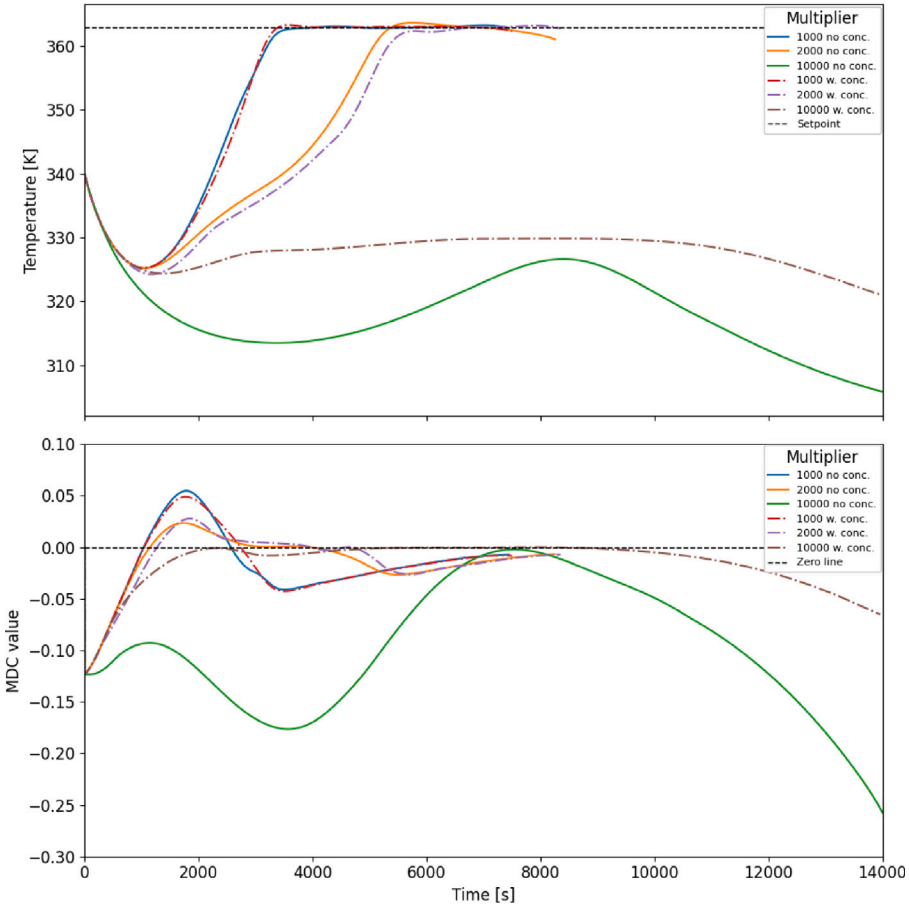


Fig. 14. Comparison of the temperature profiles and MDC value trajectories for $\lambda = 1000, 2000, 10\,000$ cases with and without concentrations in the state variables.

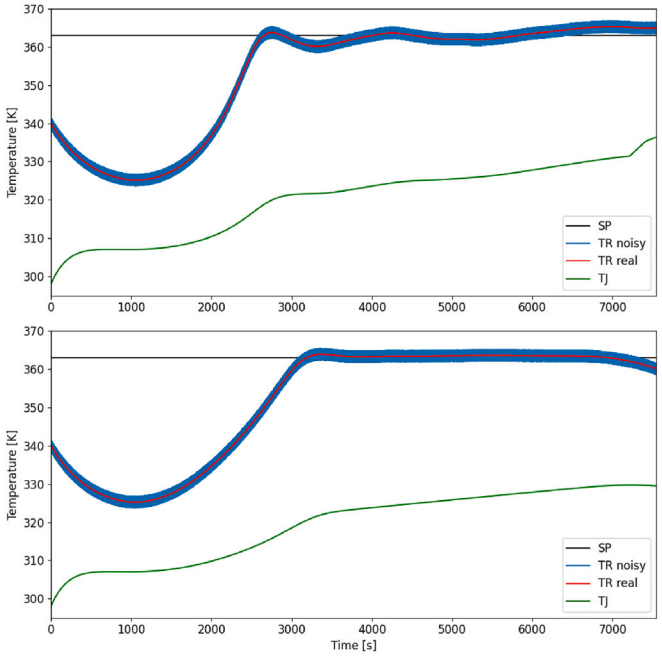


Fig. 15. Comparison of the temperature profiles of $\lambda = 0, 1000$ cases with noisy temperature measurements.

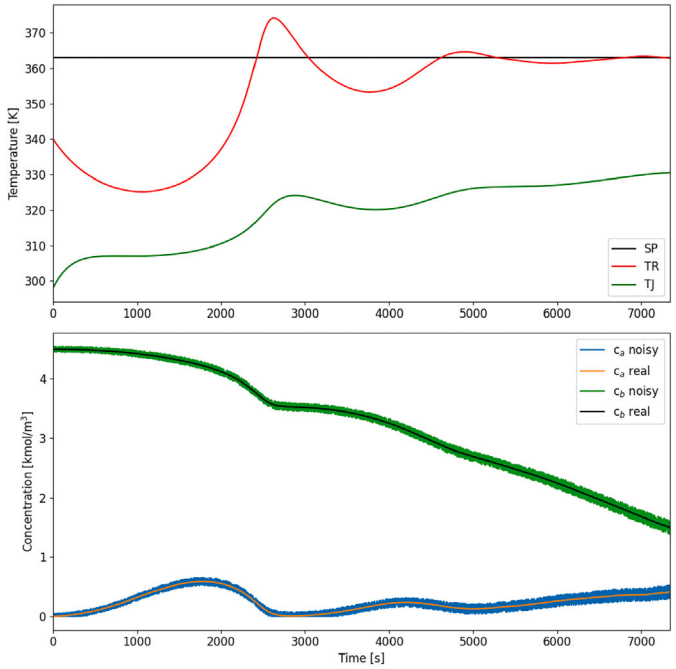


Fig. 16. Temperature and concentration trajectories for $\lambda = 0$ case with noisy concentration measurements.

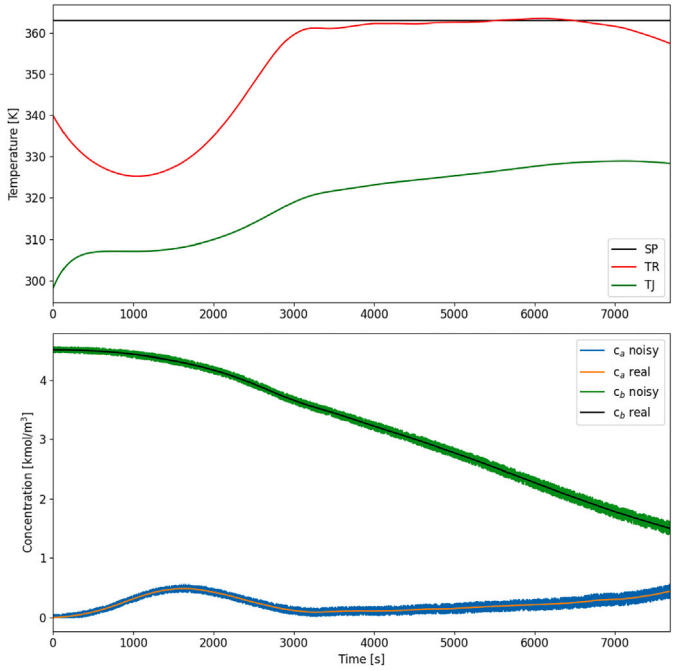


Fig. 17. Temperature and concentration trajectories for $\lambda = 1000$ case with noisy concentration measurements.

3.4. Uncertain measurement tests

As a last test of controller capabilities, we investigated how the various agents would behave if the reactor temperature and the concentration measurements were noisy. We have used the concentration aware agents with $\lambda = 0, 1000$ to compare the temperature profiles and whether runaway awareness makes the agent more sensitive or robust to sensor noise during operation. For concentration we assumed a single analyzer device for both concentrations with a $[-0.1; 0.1]$ range white noise, which was independent for the two concentrations. For the temperature, we used a $[-1.5; 1.5]$ range white noise.

First, the uncertain temperature case was tested, and the results are shown in Fig. 15. The upper figure shows the $\lambda = 0$ case, while the lower presents the $\lambda = 1000$ agent. It can be seen that the

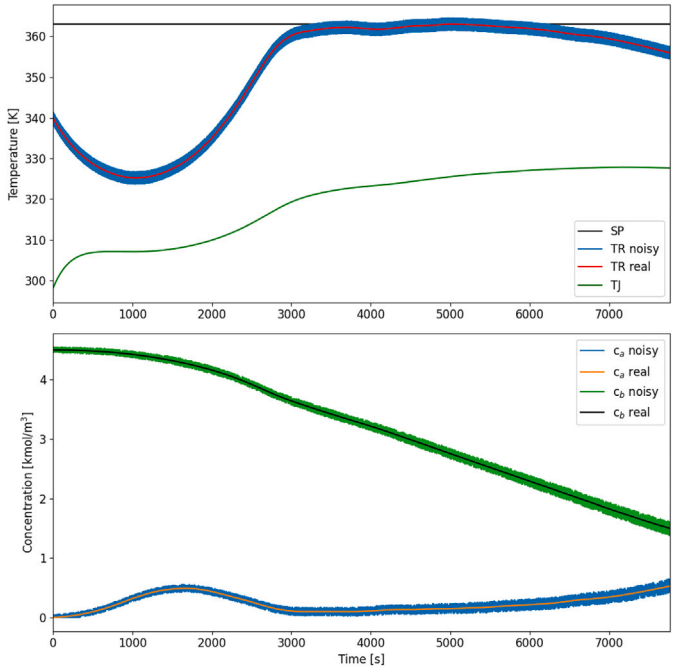


Fig. 18. Temperature and concentration trajectories for $\lambda = 1000$ case with noisy concentration and temperature measurements.

runaway aware agent maintains the temperature setpoint in a more stable manner, which could be due to the runaway aware agent already increasing the temperature more slowly. This creates a case where the temperature change due to reaction heat is slower and more closely matches the dynamics of the jacket cooling.

The second test was the use of noisy concentration measurements, the temperature and concentration profiles are shown in Figs. 16 and 17. For the $\lambda = 0$ case the agent is sensitive to concentration values, which can cause a significant thermal runaway at the beginning of the operation, however, the agent is capable of recovering that shows a degree of resilience. This runaway is caused by an accumulation of reagent A in the reactor, which ramps up the reaction rate sharply, and this process sustains itself, until reagent A is exhausted.

The $\lambda = 1000$ agent in Fig. 17 shows no runaway during operation, instead the agent does not reach the setpoint as fast as possible. The uncertain concentration values cause the agent to ramp down the reagent feeding, which causes achieving the setpoint later in the operation, but in a safe manner. This cautious behavior is a positive sign during real life noisy operation as the process operating personnel can make fine-tuning adjustments in a safe manner.

The last test was the combination of the uncertain temperature and concentration measurements. As the previous cases have shown that the $\lambda = 0$ agent underperforms when presented with noisy data, we will only show the results for the $\lambda = 1000$ case in Fig. 18. The temperature profile in the heating up phase shows strong similarities to the uncertain concentration case, with safe operation and the same temperature decrease at the end of the feeding phase, which is visible in all three $\lambda = 1000$ cases.

These results show that the inclusion of this runaway criterion has a positive impact on the control behavior of the trained RL agent. Our results are from a single case study to achieve a more complete assessment, multiple diverse case studies are required. However, the trained and presented agents show that it is a promising application direction.

4. Conclusion

The aim of our work was to analyze the effects of adding a thermal runaway awareness component to the reward function of an RL agent to create a potentially safer controller, while also utilizing the non-linear control capabilities of a neural network. We trained TD3 agents using the Modified Dynamic Condition runaway criterion and evaluated their control performance. Varying the MDC factor values created a good baseline for our further investigations by showing us what λ magnitudes are required to have an agent that balances fast reactor heatup, while also being safe from runaway. The cooling shutoff test showed the usefulness of the MDC value usage, and the modified state vector tests showed that to a moderate extent the rarely measured concentrations can be excluded. We believe that if the process controller agent is coupled with an estimation method for composition, then the concentration-aware agent could be utilized. As a final test, we have investigated the influence of the λ factor when noisy measurements are present in the process, as would be the case in a real life technology. The results show that the MDC augmented reward function improved the resulting trained agent, which could be beneficial when the controller is applied to a process in an industrial case.

Based on these tests, we believe that MDC augmentation could be helpful in future RL applications for the synthesis of an inherently safer process controller. The proposed approach showed promising results in the considered case study, but future work could include a more systematic assessment of its performance under parametric uncertainty, measurement noise during training, and parameter drift, as well as the investigation of additional case studies. Further studies on different processes and operation analysis (sensor defaulting to 0 or process drift) could help to gain a thorough picture on what the MDC method could improve the training of the agent. The impact of different failure severities (e.g., partial cooling failure) and the timing of failures (e.g., early/mid/late stages of the reaction) on control effectiveness could be considered for future work.

More advanced safety integration mechanisms, such as constraint-based reinforcement learning, safe RL frameworks, or supervisory safety layers, are also important and represent promising directions for

Table A.5
Parameters and initial conditions of the case study.

Parameter		Value	Unit
k_0	Pre-exponential factor	1.465×10^7	$\frac{m^3}{kmols}$
$\frac{E_A}{R_g}$	Activation energy	8500	K
ΔHr	Reaction heat parameter	-3.5×10^5	$\frac{kJ}{kmol}$
MAT	Maximum Allowable Temperature	373	K
UA_0	Initial heat transfer parameter	1.85	$\frac{kW}{K}$
V_0	Initial reagent volume	0.5	m^3
V_J	Jacket volume	0.41	m^3
ρc_p	Volumetric heat capacity in reactor	4800	$\frac{kJ}{m^3\ K}$
$(\rho c_p)_J$	Volumetric heat capacity in jacket	4183	$\frac{kJ}{m^3\ K}$

Table A.6
Operating parameters.

Parameter		Value	Unit
$n_{A,feed}$	Feed moles of A reagent	$n_{B,0}$	kmol
$n_{A,0}$	Initial moles of A reagent	0	kmol
$n_{B,0}$	Initial moles of B reagent	5	kmol
$c_{in,A}$	Feed concentration of A reagent	5	$\frac{kmol}{m^3}$
F_{max}	Maximum flow rate of reagent feed	1.4×10^{-4}	$\frac{m^3}{s}$
$F_{j,max}$	Maximum flow rate of coolant	1.1×10^{-3}	$\frac{m^3}{s}$
$T_{R,0}$	Initial reactor temperature	340	K
$T_{J,0}$	Initial jacket temperature	298	K
T_{in}	Reagent feed temperature	298	K
$T_{J,in}$	Coolant feed temperature	298	K

future research. In this sense, the proposed approach can be interpreted as a first step toward integrating process safety indicators into reinforcement learning, which could later be combined with such methods to achieve stronger safety guarantees.

CRediT authorship contribution statement

Balázs Fricz: Writing – original draft, Software, Methodology, Investigation, Conceptualization. **Kinga Szatmári:** Writing – review & editing, Writing – original draft, Software, Methodology, Conceptualization. **Sándor Németh:** Writing – review & editing, Supervision. **Lajos Nagy:** Writing – review & editing, Supervision. **Wenshuai Bai:** Writing – review & editing. **Alex Kummer:** Writing – review & editing, Supervision, Software, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This project was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences (B0/00439/25/6).

Prepared with the professional support of the Co-Operative Doctoral Program of the University Research Scholarship Program (grant number: 2024-2.1.2-EKÖP-KDP-2024-00017) of the Ministry of Culture and Innovation financed from the National Research, Development and Innovation Fund, Hungary.

Appendix A. Process model of the reactor

The reaction rate (r_R $\left[\frac{kmol}{m^3\ s}\right]$) is expressed with a Arrhenius-type equation:

$$r_R = kc_A c_B = k_0 \exp\left(\frac{-E_A}{R_g T_R}\right) c_A c_B \tag{A.1}$$

where k is the reaction rate constant $\left[\frac{m^3\ s}{kmol}\right]$, c_A and c_B is the concentration of A and B reagents $\left[\frac{kmol}{m^3}\right]$, k_0 and E_A are the pre-exponential factor $\left[\frac{m^3\ s}{kmol}\right]$ and the activation energy $\left[\frac{kJ}{kmol}\right]$ respectively, R_g is the gas constant $\left[\frac{kJ}{kmol\ K}\right]$ and T_R is temperature of the reactor [K].

The following differential equations describe the dynamic behavior of the system.

$$\frac{dV}{dt} = F \tag{A.2}$$

$$\frac{dn_A}{dt} = Fc_{in,A} - Vr_R \tag{A.3}$$

$$\frac{dn_B}{dt} = -\frac{dn_C}{dt} = -Vr_R \tag{A.4}$$

$$\frac{dT_R}{dt} = \frac{F\rho c_p(T_{in} - T_R) + (-\Delta H_r)Vr_R - UA(T_R - T_J)}{V\rho c_p} \tag{A.5}$$

$$\frac{dT_J}{dt} = \frac{F_J(\rho c_p)_J(T_{J,in} - T_J) + UA(T_R - T_J)}{(V\rho c_p)_J} \tag{A.6}$$

$$UA = UA_0\left(1 + \frac{V_{dos}}{V_0}\right) \tag{A.7}$$

where t is time [s], V is liquid volume [m^3], V_{dos} is dosed volume [m^3], F is feeding rate $\left[\frac{m^3}{s}\right]$, n_i is the mole amount of the ith component [kmol], T_R and T_J are the reactor and jacket temperatures respectively [K], ρ is density $\left[\frac{kg}{m^3}\right]$, c_p is the heat capacity $\left[\frac{kJ}{kg\ K}\right]$, ΔH_r is reaction heat $\left[\frac{kJ}{kmol}\right]$, UA is the heat transfer parameter $\left[\frac{kW}{K}\right]$.

The considered control valves have linear characteristics, so the following equations (Eq. (A.8)–(A.9)) describe the inlet flow rates.

$$F = F_{max}\frac{V1_{pos}}{100} \tag{A.8}$$

$$F_j = F_{j,max}\frac{V2_{pos}}{100} \tag{A.9}$$

The kinetic, material and reactor geometry parameters are presented in Table A.5, and operating parameters are presented in Table A.6.

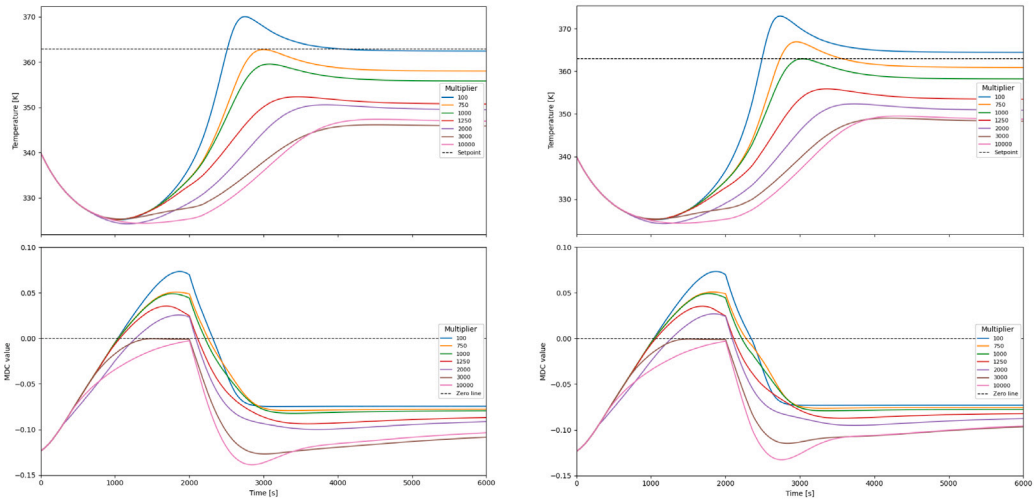
Appendix B. Cooling loss severity tests

Additional simulation results are presented for extended coolant actuator failure durations of 180 s and 360 s, complementing the 90 s failure scenario discussed in the main text. These cases represent more severe disturbance conditions and the simulation setup is identical to that described in the main text.

For longer failure durations (Fig. B.19), the differences between the control policies become more pronounced. Lower penalty values result in higher temperature peaks and prolonged positive MDC regions, indicating increased vulnerability to runaway conditions under extended cooling failure. In contrast, higher λ values lead to more conservative control actions, reflected in smoother temperature trajectories and reduced MDC peaks.

Data availability

Data will be made available on request.



(a) 180 s cooling failure. (b) 360 s cooling failure.

Fig. B.19. Temperature, reward, and MDC value plots for extended cooling failure durations.

References

Ballard, N., Farajzadehahary, K., Hamzehlou, S., Mori, U., Asua, J.M., 2024. Reinforcement learning for the optimization and online control of emulsion polymerization reactors: Particle morphology. Comput. Chem. Eng. 187, 108739.

Barto, A.G., 2021. Reinforcement learning: An introduction. By Richard's Sutton. SIAM Rev. 6 (2), 423.

Dakkoune, A., Vernières-Hassimi, L., Leveneur, S., Lefebvre, D., Estel, L., 2019. Analysis of thermal runaway events in French chemical industry. J. Loss Prev. Process. Ind. 62, 103938.

Fujimoto, S., Hoof, H., Meger, D., 2018. Addressing function approximation error in actor-critic methods. In: International Conference on Machine Learning. PMLR, pp. 1587–1596.

Haarnoja, T., Zhou, A., Abbeel, P., Levine, S., 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International Conference on Machine Learning. Pmlr, pp. 1861–1870.

Hangos, K.M., Lakner, R., Gerzson, M., 2001. Intelligent Control Systems: An Introduction with Examples. Springer.

Hong, X., Shou, Z., Chen, W., Liao, Z., Sun, J., Yang, Y., Wang, J., Yang, Y., 2024. A reinforcement learning-based temperature control of fluidized bed reactor in gas-phase polyethylene process. Comput. Chem. Eng. 183, 108588.

Joshi, T., Makker, S., Kodamana, H., Kandath, H., 2021. Twin actor twin delayed deep deterministic policy gradient (TATD3) learning for batch process control. Comput. Chem. Eng. 155, 107527.

Kavitha, K., 2019. The importance of intelligent control systems. Int. J. Hum. Comput. Stud. 1 (1), 11–13.

Kummer, A., Varga, T., 2019. Completion of thermal runaway criteria: Two new criteria to define runaway limits. Chem. Eng. Sci. 196, 277–290.

Kummer, A., Varga, T., 2021. What do we know already about reactor runaway?–A review. Process. Saf. Environ. Prot. 147, 460–476.

Kummer, A., Varga, T., Nagy, L., 2020. Semi-batch reactor control with NMPC avoiding thermal runaway. Comput. Chem. Eng. 134, 106694.

Li, J., Yu, T., Yang, B., 2021. A data-driven output voltage control of solid oxide fuel cell using multi-agent deep reinforcement learning. Appl. Energy 304, 117541.

Lin, R., Chen, J., Xie, L., Su, H., 2023. Accelerating reinforcement learning with case-based model-assisted experience augmentation for process control. Neural Netw. 158, 197–215.

Ma, L., Zhao, H., Pan, S., 2024. Research on temperature control of proton exchange membrane electrolysis cell based on MO-TD3. IET Renew. Power Gener. 18 (9–10), 1597–1610.

Ma, Y., Zhu, W., Benton, M.G., Romagnoli, J., 2019. Continuous control of a polymerization system with deep reinforcement learning. J. Process Control 75, 40–47.

Mendiola-Rodriguez, T.A., Ricardez-Sandoval, L.A., 2022. Robust control for anaerobic digestion systems of Tequila vinasses under uncertainty: A deep deterministic policy gradient algorithm. Digit. Chem. Eng. 3, 100023.

Moerland, T.M., Broekens, J., Plaat, A., M. Jonker, C., 2023. Model-based reinforcement learning: A survey. Found. Trends Mach. Learn. 16 (1), 1–118.

Nian, R., Liu, J., Huang, B., 2020. A review on reinforcement learning: Introduction and applications in industrial process control. Comput. Chem. Eng. 139, 106886.

Panjapornpon, C., Chinchalongporn, P., Bardeeniz, S., Makkayatorn, R., Wongpunnawat, W., 2022. Reinforcement learning control with deep deterministic policy gradient algorithm for multivariable pH process. Processes 10 (12), 2514.

Park, J., Choi, W., Kim, D.I., El Park, H., Lee, J.M., 2025. Real-world implementation of offline reinforcement learning for process control in industrial dividing wall column. Comput. Chem. Eng. 109383.

Rajasekhar, N., Radhakrishnan, T.K., Mohamed, S.N., 2024. Reinforcement learning based temperature control of a fermentation bioreactor for ethanol production. Biotechnol. Bioeng. 121 (10), 3114–3127.

Sass, A., Kummer, A., Abonyi, J., 2022. Multi-agent reinforcement learning-based exploration of optimal operation strategies of semi-batch reactors. Comput. Chem. Eng. 162, 107819.

Shi, X., Li, Y., Sun, B., Xu, H., Yang, C., Zhu, H., 2020. Optimizing zinc electrowinning processes with current switching via deep deterministic policy gradient learning. Neurocomputing 380, 190–200.

Stoessel, F., 2021. Thermal Safety of Chemical Processes: Risk Assessment and Process Design. John Wiley & Sons.

Westerterp, K.R., Molga, E., 2006. Safety and runaway prevention in batch and semibatch reactors—A review. Chem. Eng. Res. Des. 84 (7), 543–552.

Yang, Y.-N., Jin, J., Zhu, L.-T., Zhou, Y.-N., Luo, Z.-H., 2024. Runaway criteria for predicting the thermal behavior of chemical reactors. Curr. Opin. Chem. Eng. 43, 100986.

Yifei, Y., Lakshminarayanan, S., 2023. Multi-agent reinforcement learning for process control: Exploring the intersection between fields of reinforcement learning, control theory, and game theory. Can. J. Chem. Eng. 101 (11), 6227–6239.

Yoo, H., Kim, B., Kim, J.W., Lee, J.H., 2021. Reinforcement learning based optimal control of batch processes using Monte-Carlo deep deterministic policy gradient with phase segmentation. Comput. Chem. Eng. 144, 107133.

Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., Wu, Y., 2022. The surprising effectiveness of ppo in cooperative multi-agent games. Adv. Neural Inf. Process. Syst. 35, 24611–24624.

Zhang, Z., Wu, Z., Rincon, D., Garcia, C., Christofides, P.D., 2019. Operational safety of chemical processes via safeness-index based MPC: Two large-scale case studies. Comput. Chem. Eng. 125, 204–215.