Received: 8 April 2026 | Revised: 3 May 2026 | Accepted: 21 May 2026

DOI: 10.1002/aic.70479

RESEARCH ARTICLE

Process Systems Engineering

AIChE JOURNAL

Reinforcement learning-based control of nonlinear systems: Economic optimization and operational safety

Xiaodong Cui [1] | Arthur Khodaverdian [1] | Panagiotis D. Christofides [1,2]

[1]Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, California, USA

[2]Department of Electrical and Computer Engineering, University of California, Los Angeles, California, USA

Correspondence
Panagiotis D. Christofides, Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, CA 90095-1592, USA.
Email: pdc@seas.ucla.edu

Funding information
National Science Foundation, Grant/Award Number: CBET-2227241

Abstract

This work develops a reinforcement learning (RL)-based control framework for nonlinear systems that optimizes economic performance while ensuring operational safety. The proposed method combines a real-time RL feedback policy with an online supervisory mechanism that enforces prescribed safety and closed-loop stability conditions, using a Lyapunov-based economic model predictive controller (LEMPC) as a backup controller when needed. To reduce backup reliance, the RL policy is trained in a constraint-informed environment with penalties on rejected actions. The control framework is applied to a nonlinear chemical process with an economic objective, operational safety considerations, and bounded disturbances. Simulations show recovery of unsafe initial states, improved economic performance relative to LEMPC and imitation learning, and much lower online computational burden than LEMPC.

KEYWORDS
economic optimization, model predictive control (MPC), operational safety, process control, reinforcement learning (RL)

1 | INTRODUCTION

Economic performance and operational safety are both central objectives in modern chemical process operation.[1–3] In industrial practice, plants are expected not only to maintain product quality, satisfy environmental regulations, and preserve stable operation, but also to reduce energy consumption, maximize throughput, improve raw-material utilization, and enhance profitability under changing market conditions. These goals are often complicated by uncertainty in feed composition, reaction kinetics, heat-transfer characteristics, and unmeasured disturbances. As a result, process operation is rarely a pure, constant setpoint-tracking problem. Instead, many practical applications require the controller to make decisions that directly balance economics, operability, and constraint satisfaction over time. At the same time, violations of process constraints, such as temperature, concentration, or pressure limits, can lead to off-specification products, accelerated equipment degradation, or even hazardous operating conditions.[3,4] These requirements are often formalized through safe operating limits (SOLs), which define an admissible operating envelope for the process.[5] Because economically favorable operation often lies close to active constraints, advanced control systems must optimize performance while maintaining sufficient safety margins in the presence of disturbances and model uncertainty.[3] This simultaneous need for profitability and risk-aware operation makes the joint treatment of economics and operational safety a fundamental challenge in nonlinear process control.

Model predictive control (MPC) provides a widely used framework for constrained multivariable control because it can explicitly account for state and input constraints while optimizing predicted closed-loop behavior.[2,6,7] Its model-based and optimization-based structure has made it especially attractive for chemical process applications, where interactions among variables, actuator limitations, and operating constraints play a major role in controller closed-loop performance. Building on conventional tracking MPC, economic MPC (EMPC) replaces the tracking objective with an economic stage cost and can deliver improved steady-state and transient performance by directly optimizing process economics.[2] In addition, safety-oriented MPC formulations, such as Safeness Index-based MPC, incorporate an explicit measure of proximity to unsafe operating regions and

enforce safety through hard constraints or soft penalties.[8] These developments make MPC-based methods appealing for applications where both profitability and safety must be considered in a unified framework. Despite these advantages, however, MPC-based approaches may face practical limitations in nonlinear real-time applications. Their implementation relies on an explicit process model and repeated online optimization, so computational cost can increase significantly with system nonlinearity, prediction horizon, and problem size.[9–11] Moreover, iterative numerical optimization may lead to economically myopic decisions through local minima[12] as well as infeasibility, numerical issues, or excessive solve times in practice.[11,13] These limitations motivate the search for alternative control architectures that preserve the ability to address economics and constraints while reducing online computational burden.

Reinforcement learning (RL) has emerged as a promising alternative for nonlinear control because it learns a feedback policy that maps measured states or outputs directly to control actions.[14–20] Once trained, an RL controller can be implemented online with very low computational cost, requiring only a function evaluation rather than the repeated solution of an optimization problem.[21–23] RL also offers the ability to optimize long-term performance directly and can reduce reliance on highly accurate first-principles models, which makes it attractive for nonlinear economic process control and other applications with fast sampling requirements or partially known dynamics.[10,11,24] In principle, this makes RL an attractive candidate for replacing or approximating computationally intensive model-based controllers during online implementation; however, standard RL methods do not inherently guarantee closed-loop stability or satisfaction of hard safety and operating constraints, and they may produce unsafe or infeasible actions during either training or deployment.[25–29] These shortcomings are particularly serious in safety-critical process systems, where even brief excursions outside the admissible operating region may be unacceptable. Thus, although RL offers major potential benefits in terms of online speed and long-horizon optimization, its practical adoption in chemical process control requires architectures that explicitly account for safety, stability, and fallback operation.

Motivated by these considerations, this work develops a reinforcement learning-based control framework for nonlinear systems that explicitly addresses both economic optimization and operational safety. The proposed framework combines a learned RL policy with an online supervisory mechanism that checks whether the candidate control action satisfies prescribed stability and safety conditions before implementation. At each sampling instant, the RL action is applied only if the required conditions are satisfied; otherwise, a backup model predictive control action is used. In this way, the framework seeks to retain the low online computational burden and long-term optimization capability of RL while improving reliability and constraint satisfaction in closed-loop operation. To further reduce backup reliance and improve policy quality, constraint considerations are incorporated during training through a constraint-informed learning strategy. The proposed approach is applicable to nonlinear systems under both economic and safety requirements, and it is demonstrated using a nonlinear chemical process example with bounded disturbances. The resulting control law provides guaranteed closed-loop stability together with recovery to the safe operating region, while achieving improved economic performance and significantly lower online computational cost than optimization-based control alone.

2 | PRELIMINARIES

2.1 | Notation

The actual state and the nominal predicted state are denoted by x and $\tilde{x}$, respectively. The transpose of a vector x, the set of real numbers, set difference, functions, and piecewise-constant functions with period Δ are denoted by $x^\top$, $\mathbb{R}$, $\Omega_1 \backslash \Omega_2$, $f(\cdot)$, and $S(\Delta)$, respectively, where both f and S are arbitrary notations. The initial time is denoted by t_0, and arbitrary sampling instants are denoted by t_k, with $t_{k+1} = t_k + \Delta$. For a given state $x(t_k)$ and input u, $\tilde{x}(t|t_k)$ denotes the nominal predicted state trajectory generated from $x(t_k)$. In the RL setting, s denotes the state, a denotes the action proposed by a policy, and π denotes a policy mapping s to a.

2.2 | Class of systems

This work considers nonlinear first-order ordinary differential equation (ODE) systems of the form

$$\dot{x} = F(x,u) + w(x,u,t) \tag{1}$$

where $x = [x_1, x_2, ..., x_n]^\top \in \mathbb{R}^n$ denotes the state vector collecting the process variables of interest and $u = [u_1, u_2, ..., u_m]^\top \in \mathbb{R}^m$ denotes the control input vector, with $u \in U$ and $U \subset \mathbb{R}^m$ the admissible input set. The admissible input set U is assumed to be nonempty and compact, i.e., U is a closed and bounded subset of $\mathbb{R}^m$. This assumption reflects physical actuator limits. The state is assumed to be measured at the sampling instants t_k. The function $F(x,u)$ describes the nominal process dynamics and is assumed to be sufficiently smooth on $D \times U$, where $D \subset \mathbb{R}^n$ is a domain containing the origin, while $w(x,u,t)$ accounts for bounded model mismatch and disturbances present in the real process, with $\|w(x,u,t)\| \le W_{max}$ for all $(x,u,t) \in D \times U \times \mathbb{R}$. Without loss of generality, the system is written in deviation-variable form such that the origin is an equilibrium of the nominal open-loop model, i.e., $F(0,0) = 0$.

2.3 | Stabilizability assumption

The assumptions used to ensure stabilizability are collectively referred to as the stabilizability assumption. These assumptions are stated with respect to the nominal model $\dot{x} = F(x,u)$, i.e., the system in Equation (1) with $w \equiv 0$.

Assumption 1. There exists a sufficiently smooth explicit feedback control law (the reference stabilizing controller, or simply the reference controller) $\Phi : D \to U$ and a sufficiently smooth Lyapunov function $V(x)$ such

that, for the closed-loop nominal system under continuous-time control with control actions given by $u = \Phi(x)$, there exist constants $c_1, c_2, c_3, c_4 > 0$ for which the following inequalities hold for all $x \in D$:

$$c_1|x|^2 \le V(x) \le c_2|x|^2 \tag{2a}$$

$$\frac{\partial V(x)}{\partial x} F(x, \Phi(x)) \le -c_3|x|^2 \tag{2b}$$

$$\left|\frac{\partial V(x)}{\partial x}\right| \le c_4|x| \tag{2c}$$

We denote level sets of the Lyapunov function within this open region as $\Omega_\eta := \{x \in D | V(x) \le \eta\}$ where η is a positive constant. Furthermore, for any $\rho > 0$, since F and V are smooth, U is bounded, and Ω_ρ is compact, there exist positive constants M_F, L_x, and L'_x such that for all $x, x' \in \Omega_\rho$ and $u \in U$:

$$|F(x', u) - F(x, u)| \le L_x|x - x'| \tag{3a}$$

$$|F(x, u)| \le M_F \tag{3b}$$

$$\left|\frac{\partial V(x)}{\partial x} F(x, u) - \frac{\partial V(x')}{\partial x'} F(x', u)\right| \le L'_x|x - x'| \tag{3c}$$

2.4 | Characterization of unsafe regions

A Safeness index[3] is introduced as a scalar function $S : D \to \mathbb{R}$ of the process state, where larger values of $S(x)$ indicate less safe operating conditions. Given a prescribed threshold S_{th}, the unsafe region is characterized as

$$\mathcal{X}_u := \{x \in D : \ S(x) > S_{\text{th}}\} \tag{4}$$

and the corresponding safe operating region is defined by

$$\mathcal{X}_s := \{x \in D : \ S(x) \le S_{\text{th}}\} = D \backslash \mathcal{X}_u. \tag{5}$$

Therefore, the safety index provides a convenient state-based characterization of safe and unsafe operating regions without requiring the unsafe set to have a particular geometric form.

Assumption 2. There exists a set $\mathcal{X}_{s,0} \subset D$ such that $0 \in \mathcal{X}_{s,0} \subset \mathcal{X}_s$. In other words, the origin is contained in a nontrivial neighborhood that lies entirely inside the safe operating region.

2.5 | Safeness index-based LEMPC

The following Safeness Index-based LEMPC formulation is used to optimize the economic objective while accounting for the Safeness requirement.[8] Let $\Omega_{\rho_e} := \{x \in D | V(x) \le \rho_e\}$, where $0 < \rho_e < \rho$ is chosen such that, if the measured process state satisfies $x(t_k) \in \Omega_{\rho_e}$ at a sampling time t_k, then the measured state at the next sampling time remains in Ω_ρ. Based on this set, define the inner safe operating region as $\mathcal{X}_p := \Omega_{\rho_e} \cap \mathcal{X}_s$, which collects the states satisfying both the inner Lyapunov bound and the Safeness requirement. At each sampling instant t_k, the controller computes a piecewise-constant input trajectory $u(\cdot) \in S(\Delta)$ over the prediction horizon $N\Delta$ by solving

$$\max_{u(t) \in S(\Delta)} \quad \int_{t_k}^{t_k + N\Delta} L_e(\tilde{x}(\tau), u(\tau))\, d\tau \tag{6a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F(\tilde{x}(t), u(t)) \tag{6b}$$

$$u(t) \in U, \quad \forall t \in [t_k, t_k + N\Delta) \tag{6c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{6d}$$

$$\tilde{x}(t) \in \mathcal{X}_p, \quad \forall t \in [t_k, t_k + N\Delta), \quad \text{if } x(t_k) \in \mathcal{X}_p \tag{6e}$$

$$\frac{\partial V(x(t_k))}{\partial x} F(x(t_k), u(t_k)) \le \frac{\partial V(x(t_k))}{\partial x} F(x(t_k), \Phi(x(t_k))), \quad \text{if } x(t_k) \in \Omega_\rho \backslash \mathcal{X}_p \tag{6f}$$

In Equation (6), $L_e(x, u)$ is the economic stage objective and $\tilde{x}(t)$ is the predicted state trajectory. The constraint in Equation (6e) requires the predicted trajectory to remain in the inner safe operating region $\mathcal{X}_p$ over the prediction horizon whenever the current measured state belongs to $\mathcal{X}_p$. The constraint in Equation (6f) imposes a Lyapunov-based decrease condition when the measured state is outside $\mathcal{X}_p$ but still inside Ω_ρ, including both the case where the state is safe but outside Ω_{ρ_e} and the case where it lies in the unsafe region $\mathcal{X}_u \cap \Omega_\rho$. After solving Equation (6), only the first control action is applied over $[t_k, t_{k+1})$.

2.6 | Reinforcement/control

In contrast to MPC, which computes the control action online by solving an optimization problem at each sampling instant, RL-based control seeks to learn a policy π that maps the observed state to a control action directly. At each sampling instant, the agent observes a state s, applies an action $a = \pi(s)$, receives a reward $r(s, a)$, and transitions to a successor state s'. The resulting experience tuples are stored in a replay buffer, $\mathcal{D} := \{(s_i, a_i, r_i, s'_i)\}_{i=1}^N$, which can be used to train the policy and associated value functions. In this formulation, the i subscript denotes the index of the sample within the replay buffer.

The objective in RL is to learn a policy that maximizes the expected cumulative reward. A central concept is the value function, which satisfies the Bellman relation:

$$V^\pi(s) = r(s, \pi(s)) + \gamma V^\pi(s') \tag{7}$$

where $0 < \gamma < 1$ is the discount factor. Many RL methods also learn an action-value function $Q(s, a)$ to evaluate state-action pairs and improve the policy.

For continuous-state and continuous-action control problems, value-based methods such as Q-learning become difficult to apply directly. To handle such problems, actor-critic methods have been developed, where one approximator represents the policy and another approximates the action-value function. Among these methods, deep deterministic policy gradient (DDPG) is a representative algorithm for deterministic continuous control; however, DDPG may produce overestimated Q-values during learning, which can lead to poor policy updates. To mitigate this issue, Twin Delayed Deep Deterministic Policy Gradient (TD3) introduces two critic networks, delayed policy updates, and target policy smoothing. These modifications improve learning stability and control performance in continuous-action settings.[30] Therefore, TD3 is adopted in this work as the RL algorithm for policy learning.

3 | SAFE ECONOMIC RL-BASED CONTROL FRAMEWORK DESIGN

3.1 | Economic reinforcement learning-based controller

We consider the nonlinear system Equation (1) with state measurements available at sampling instants $\{t_k\}_{k\geq 0}$ and admissible control set U. For a given trajectory, the state at time t_k is denoted by $s_k \triangleq s(t_k)$. A parameterized policy (actor) π_θ proposes a candidate control action:

$$u_{\mathrm{RL}}(t_k) := \pi_\theta(s_k) \tag{8}$$

where θ collects the trainable parameters. Unlike tracking-based RL formulations, where the reward is typically designed to penalize deviation from a prescribed setpoint, the proposed controller is developed in an economic fashion. Specifically, the reward function $r_e(s,u)$ is chosen to reflect the economic stage objective L_e in Equation (6a), so that the learned policy is encouraged to improve process economics directly, such as increasing production rate or profit, rather than merely driving the state to a fixed operating point.

> *Remark* 1. For clarity, the Lyapunov function $V(x)$ used for stability analysis and enforcement is distinct from the RL value function in Equation (7). Throughout this section, $V(x)$ always denotes the Lyapunov function from Section 2.3.

3.2 | Safe online implementation

In this study, the economic RL policy is implemented in a proposed shielded manner so that the applied input is feasible in U and satisfies the prescribed safety and stability condition. At each sampling instant t_k, the actor proposes $u_{\mathrm{RL}}(t_k)$ according to Equation (8), and the hard input constraint is enforced through the projection

$$\bar{u}_{\mathrm{RL}}(t_k) := \mathrm{Proj}_U(u_{\mathrm{RL}}(t_k)) \tag{9}$$

For $x(t_k) \in \Omega_\rho$, let $\mathcal{C}_p(x(t_k), u(t_k))$ denote the prescribed condition at t_k, namely, if $x(t_k) \in \mathcal{X}_p$, then

$$\tilde{x}(t|t_k) \in \mathcal{X}_p, \quad \forall t \in [t_k, t_{k+1}] \tag{10}$$

whereas if $x(t_k) \in \Omega_\rho \backslash \mathcal{X}_p$, then

$$\frac{\partial V(x(t_k))}{\partial x} F(x(t_k), u(t_k)) \leq \frac{\partial V(x(t_k))}{\partial x} F(x(t_k), \Phi(x(t_k))) \tag{11}$$

The projected RL action is accepted only if $\mathcal{C}_p(x(t_k), \bar{u}_{\mathrm{RL}}(t_k))$ is satisfied. Otherwise, the backup action is selected hierarchically. The first choice is $\Phi_{\mathrm{LEMPC}}(x(t_k))$, provided that Equation (6) returns a feasible solution within the available sampling time; if not, $\Phi(x(t_k))$ is applied. Since Φ_{LEMPC} is constructed from Equation (6), it satisfies the same prescribed condition by construction whenever such a feasible solution is available.

This distinction is important. If the additional inner-region safety requirement that $x(t_k) \in \mathcal{X}_p$ implies $\tilde{x}(t|t_k) \in \mathcal{X}_p$ over the next sampling interval were not imposed, then the underlying LEMPC problem would admit a feasible solution on Ω_ρ, because the reference controller $\Phi(x)$ provides a feasible candidate input sequence[8]; however, once this prescribed safety requirement is enforced, recursive feasibility of Equation (6) is no longer automatic because $\mathcal{X}_p$ need not be forward invariant. Therefore, $\Phi(x)$ is not introduced as a controller that is required to satisfy the same prescribed condition pointwise at every sampling instant. Rather, it serves as a stabilizing controller to ensure that the implemented policy remains well defined even when the backup LEMPC is infeasible or when a feasible solution cannot be returned within the available sampling time. Thus, the final fallback $\Phi(x)$ should be interpreted as preserving the stability guarantee, rather than as enforcing forward invariance of $\mathcal{X}_p$. Therefore, the implemented input is given by

$$u(t_k) = \begin{cases} \bar{u}_{\mathrm{RL}}(t_k), & \text{if } \mathcal{C}_p(x(t_k), \bar{u}_{\mathrm{RL}}(t_k)) \\ \Phi_{\mathrm{LEMPC}}(x(t_k)), & \text{if} \mathcal{C}_p(x(t_k), \bar{u}_{\mathrm{RL}}(t_k)) \\ & \text{is not satisfied and Eq.}(6) \text{ is available} \\ \Phi(x(t_k)), & \text{otherwise} \end{cases} \tag{12}$$

and is held constant on $[t_k, t_{k+1})$. The safe online implementation is summarized in Algorithm 1. The resulting feasibility, safety, and stability properties are analyzed in Section 3.4.

> *Remark* 2. When the projected RL action violates the prescribed safety or stability condition, we do not further project it onto the set of inputs satisfying $\mathcal{C}_p$. Such a projection would only select an input close to $u_{\mathrm{RL}}(t_k)$ under a chosen metric and would not account for the economic objective. Therefore, when $\mathcal{C}_p$ is violated, the backup action is instead generated by Φ_{LEMPC}, and ϕ is only used when Φ_{LEMPC} is not available.

Algorithm 1 **Safe online implementation of the economic RL policy**

Input: Current state $x(t_k) \in \Omega_\rho$, RL policy π_θ, admissible input set U, backup controllers Φ_{LEMPC} and Φ

Compute RL action proposal $u_{\text{RL}}(t_k) \leftarrow \pi_\theta(s_k)$;

Project the proposed action onto U: $\bar{u}_{\text{RL}}(t_k) \leftarrow \text{Proj}_U(u_{\text{RL}}(t_k))$;

if $C_\rho(x(t_k), \bar{u}_{\text{RL}}(t_k))$ *is satisfied* **then**

  Set implemented input $u(t_k) \leftarrow \bar{u}_{\text{RL}}(t_k)$;

else

  Solve the backup LEMPC of Eq. (6) at $x(t_k)$;

  if *a feasible solution is returned within the available sampling time* **then**

    Set implemented input $u(t_k) \leftarrow \Phi_{\text{LEMPC}}(x(t_k))$;

  else

    Set implemented input $u(t_k) \leftarrow \Phi(x(t_k))$;

Apply the sample-and-hold input $u(t) = u(t_k)$ for all $t \in [t_k, t_{k+1})$;

3.3 | Constraint-informed two-stage RL training

Since the learned policy is deployed under the shield, the RL agent is also trained in a shielded environment in which the constraint-enforcement mechanism is treated as part of the environment.[31] As illustrated in Figure 1, the environment is no longer just the process; instead, it consists of the process together with the enforcer that checks the proposed action and determines the implemented input. Specifically, at each sampling instant t_k, the actor proposes the raw action $a_k = u_{\text{RL}}(t_k) = \pi_\theta(s_k)$, which is first projected into the admissible input set as $\bar{u}_{\text{RL}}(t_k) = \text{Proj}_U(u_{\text{RL}}(t_k))$, and the implemented input is then selected according to Equation (12). Therefore, the successor state s'_k is generated by the true process under the implemented input $u(t_k)$, held constant over $[t_k, t_{k+1})$, rather than necessarily under the raw RL proposal itself. In this way, the agent is trained against the actual closed-loop behavior induced by the shield.

This shielded training setup may otherwise encourage excessive reliance on shield intervention, since a rejected RL proposal can still lead to a favorable transition after being replaced by a non-RL implemented input. This creates an issue during learning: Although the rejected RL proposal is not implemented in the process and therefore does not compromise closed-loop safety, the replay buffer may still associate the raw proposal with the reward and successor state generated by the backup action. Without an intervention penalty, the agent may therefore learn to rely on backup controller correction rather than improve the admissibility of its own proposed actions. To reduce such reliance, the training reward is modified from the implemented-input reward $r_e(s_k, u(t_k))$ by penalizing any intervention of the shield. Let $\mathbb{I}_k = 0$ if $u(t_k) = \bar{u}_{\text{RL}}(t_k)$, and $\mathbb{I}_k = 1$ otherwise. The reward used for training is then

$$\bar{r}_k = (1 - (1-\beta)\mathbb{I}_k)\, r_e(s_k, u(t_k)) \tag{13}$$

with $\beta \in (0, 1)$, so that the reward is reduced whenever the implemented input is not the RL action. The coefficient β controls the trade-off between discouraging inadmissible RL proposals and allowing economic-performance improvement: Smaller values impose a stronger

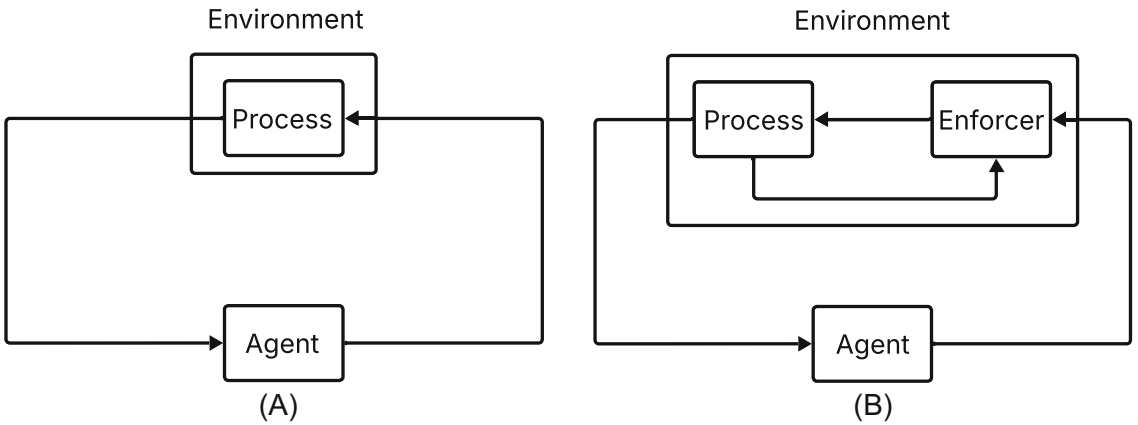

FIGURE 1 Comparison of agent-environment interaction in conventional RL and the proposed constraint-informed RL framework. (A) Conventional RL, (B) Constraint-informed RL.

penalty on backup usage and encourage the actor to generate admissible actions, whereas larger values relax the penalty and may improve economic exploration but can also increase reliance on backup controller intervention. Accordingly, the replay buffer stores the transition

$$(s_k, a_k, r_k, s'_k) = (s_k, u_{\mathrm{RL}}(t_k), \bar{r}_k, s'_k) \tag{14}$$

where the stored action is the raw RL proposal, while the reward and successor state are generated by the implemented input after constraint enforcement.

To balance admissibility and performance improvement, a two-stage training strategy is adopted. In Stage 1, a relatively small penalty coefficient $\beta_1 < 1$ is used together with more aggressive training hyperparameters so that backup usage is penalized more strongly and the policy is driven toward generating admissible actions with limited reliance on the backup controller, while still maintaining acceptable closed-loop economic performance. The objective of this stage is to obtain a reliable initial policy that can be deployed with little intervention from the backup layer. In Stage 2, the best policy obtained from Stage 1 is used to initialize further training, and the penalty on backup usage is progressively relaxed so that the policy can be further fine-tuned toward improved economic performance. Throughout this stage, policy updates are accepted only when the candidate policy achieves improved closed-loop performance while preserving sufficiently low reliance on the backup controller. In this way, the second stage refines performance in a warm-started manner while retaining the admissibility properties established in Stage 1. The overall constraint-informed two-stage RL training procedure is summarized in Algorithm 2.

Remark 3. To bias RL toward proposing feasible actions, one may penalize backup usage either multiplicatively as in Equation (13) or additively by subtracting a constant penalty whenever the backup is invoked, e.g., $\bar{r}_k = r_e(s_k, u(t_k)) - \lambda\, \mathbb{I}_k$ with $\lambda > 0$ (see,[32]). Both choices encourage policies that reduce infeasible control actions from RL.

3.4 | Feasibility, stability and safety analysis

We now analyze the feasibility, stability, and safety properties of the proposed RL-based control framework. The main feasibility issue in the present setting concerns the backup LEMPC optimization embedded in the hierarchical implementation of Equation (12). In fact, if the inner-region requirement in Equation (6e) were replaced by the weaker condition that $\tilde{x}(t) \in \Omega_{\rho_e}$ for all $t \in [t_k, t_k + N\Delta)$ whenever $x(t_k) \in \mathcal{X}_p$, then the

Algorithm 2 **Constraint-informed two-stage RL training**

Input: Initial policy π_θ, replay buffer $\mathcal{D}$, Stage 1 coefficient β_1, initial Stage 2 coefficient β_2 with $\beta_1 < \beta_2 < 1$
Output: Best policy $\pi_{\theta^\star}$
Initialize $\pi_{\theta^\star} \leftarrow \pi_\theta$, $J^\star \leftarrow -\infty$;
Stage 1: Set $\beta \leftarrow \beta_1$;
for *each training iteration* **do**
  Observe s_k, compute $a_k \leftarrow \pi_\theta(s_k)$, and determine $u(t_k)$ from Eq. (12);
  Apply $u(t) = u(t_k)$ on $[t_k, t_{k+1})$ and observe s'_k;
  Set $\mathbb{I}_k \leftarrow \mathbf{1}_{\{u(t_k) \neq \bar{u}_{\mathrm{RL}}(t_k)\}}$;
  Set $\bar{r}_k \leftarrow (1 - (1-\beta)\mathbb{I}_k)\; r_e(s_k, u(t_k))$;
  Store $(s_k, a_k, \bar{r}_k, s'_k)$ in $\mathcal{D}$ and update policy/value parameters;
  Periodically evaluate π_θ; if acceptable and improved, set $\pi_{\theta^\star} \leftarrow \pi_\theta$, $J^\star \leftarrow J(\pi_\theta)$;
Stage 2: Set $\pi_\theta \leftarrow \pi_{\theta^\star}$;
while $\beta_2 < 1$ **do**
  Set $\beta \leftarrow \beta_2$ and $\pi_\theta \leftarrow \pi_{\theta^\star}$;
  for *each training iteration* **do**
    Observe s_k, compute $a_k \leftarrow \pi_\theta(s_k)$, and determine $u(t_k)$ from Eq. (12);
    Apply $u(t) = u(t_k)$ on $[t_k, t_{k+1})$ and observe s'_k;
    Set $\mathbb{I}_k \leftarrow \mathbf{1}_{\{u(t_k) \neq \bar{u}_{\mathrm{RL}}(t_k)\}}$;
    Set $\bar{r}_k \leftarrow (1 - (1-\beta)\mathbb{I}_k)\; r_e(s_k, u(t_k))$;
    Store $(s_k, a_k, \bar{r}_k, s'_k)$ in $\mathcal{D}$ and update policy/value parameters;
    Periodically evaluate π_θ; if acceptable and improved, set $\pi_{\theta^\star} \leftarrow \pi_\theta$, $J^\star \leftarrow J(\pi_\theta)$;
  Increase β_2 while keeping $\beta_2 < 1$;
return $\pi_{\theta^\star}$;

standard candidate-sequence argument based on the reference controller $\Phi(x)$ would recover feasibility of the backup optimization; see, for example.[2] Moreover, the stability and safety analysis developed below would still go through under this weaker formulation, because the inner-region part of the proof only uses the fact that the nominal predicted trajectory satisfies $V(\tilde{x}(t)) \le \rho_e$ over the relevant sampling interval.

In the present work, however, we retain the stronger requirement in Equation (6e), namely that $\tilde{x}(t) \in \mathcal{X}_p = \Omega_{\rho_e} \cap \mathcal{X}_s$, because we wish to enforce the safety requirement explicitly over the prediction horizon rather than impose only Lyapunov-region invariance. Once this stronger horizon-wise safety requirement is imposed, recursive feasibility of Equation (6e) is no longer automatic, since $\mathcal{X}_p$ need not be forward invariant.

For this reason, the analysis below focuses on the implemented switching law Equation (12), rather than on the recursive feasibility of the backup optimization alone. Specifically, whenever the RL proposal is rejected, the controller first attempts to apply the backup LEMPC action, and falls back to $\Phi(x(t_k))$ if a feasible LEMPC solution is unavailable or cannot be returned within the available sampling period Δ. We first show that, under suitable conditions, the resulting closed-loop trajectory remains in the Lyapunov region Ω_ρ for all time. We then use this invariance property together with the outer-region Lyapunov decrease condition to establish finite-time recovery from the unsafe region $\mathcal{X}_u \cap \Omega_\rho$ to the safe region $\mathcal{X}_s$.

Theorem 1. *Consider the true process Equation (1) in closed loop under the switching law Equation (12) implemented in a sample-and-hold fashion with sampling period $\Delta > 0$ (i.e., $u(t) = u(t_k)$ for all $t \in [t_k, t_k + \Delta)$). Define Lyapunov level sets $\rho > \rho_e > \rho_{\min} > \rho_s > 0$, where*

$$\rho_{\min} := \sup\{V(x(t_k+\Delta)) \mid x(t_k) \in \Omega_{\rho_s},\ w \in \mathcal{W}\} \tag{15}$$

Assume there exists $\epsilon_w > 0$ such that

$$-\frac{c_3}{c_2}\rho_s + L'_x(M_F + W_{\max})\Delta + c_4\left(\sqrt{\frac{\rho}{c_1}} + (M_F + W_{\max})\Delta\right)W_{\max} \le -\epsilon_w \tag{16}$$

and define

$$\Delta_\rho := c_4\left(\sqrt{\frac{\rho_e}{c_1}} + \frac{W_{\max}}{L_x}\left(e^{L_x\Delta} - 1\right)\right)(M_F + W_{\max})\Delta \tag{17}$$

We assume $\rho_e + \Delta_\rho < \rho$. Then, for any initial condition $x(t_0) \in \Omega_\rho$ and any admissible mismatch $w \in \mathcal{W}$, the closed-loop trajectory under Equation (12) satisfies $x(t) \in \Omega_\rho$ for all $t \ge t_0$.

Proof. It suffices to show that, for any t_k, $x(t_k) \in \Omega_\rho$ implies $x(t) \in \Omega_\rho$ for all $t \in [t_k, t_k + \Delta)$. Then $x(t_{k+1}) \in \Omega_\rho$, and the result for all $t \ge t_0$ follows by induction. Fix any t_k such that $x(t_k) \in \Omega_\rho$. Suppose, by contradiction, that there exists $t^* \in [t_k, t_k + \Delta)$ such that $x(t^*) \notin \Omega_\rho$. Define

$$\tau_1 := \inf\{t \in [t_k, t_k + \Delta) \mid V(x(t)) > \rho\} \tag{18}$$

Then $\tau_1 \in (t_k, t_k + \Delta)$, $V(x(\tau_1)) = \rho$, and $V(x(t)) \le \rho$ for all $t \in [t_k, \tau_1]$; hence $x(t) \in \Omega_\rho$ on $[t_k, \tau_1]$.

For any $t \in [t_k, \tau_1]$, by Equation (1) and the sample-and-hold implementation in Equation (12),

$$\begin{aligned}\frac{d}{dt}V(x(t)) &= \frac{\partial V(x(t))}{\partial x}(F(x(t), u(t_k)) + w(x(t), u(t_k), t)) \\ &= \frac{\partial V(x(t))}{\partial x}F(x(t), u(t_k)) - \frac{\partial V(x(t_k))}{\partial x}F(x(t_k), u(t_k)) \\ &\quad + \frac{\partial V(x(t_k))}{\partial x}F(x(t_k), u(t_k)) + \frac{\partial V(x(t))}{\partial x}w(x(t), u(t_k), t) \\ &\le L'_x\|x(t) - x(t_k)\| + \frac{\partial V(x(t_k))}{\partial x}F(x(t_k), u(t_k)) + c_4\|x(t)\|W_{\max}\end{aligned} \tag{19}$$

where Equations (3c) and (2c) and $\|w(x,u,t)\| \le W_{\max}$ from Equation (1) have been used. Also, integrating Equation (1) over $[t_k, t]$, and using Equation (3b), gives

$$\begin{aligned}\|x(t) - x(t_k)\| &\le \int_{t_k}^{t} (\|F(x(\tau), u(t_k))\| + \|w(x(\tau), u(t_k), \tau)\|)\, d\tau \\ &\le \int_{t_k}^{t} (M_F + W_{\max})\, d\tau = (M_F + W_{\max})(t - t_k) \\ &\le (M_F + W_{\max})\Delta\end{aligned} \tag{20}$$

Case 1. If $x(t_k) \in \Omega_\rho \setminus \Omega_{\rho_e}$, then by the outer-region condition in Equation (11) and Equations (2b) and (2a),

$$\begin{aligned}\frac{\partial V(x(t_k))}{\partial x}F(x(t_k), u(t_k)) &\le \frac{\partial V(x(t_k))}{\partial x}F(x(t_k), \Phi(x(t_k))) \\ &\le -c_3\|x(t_k)\|^2 \le -\frac{c_3}{c_2}V(x(t_k)) \le -\frac{c_3}{c_2}\rho_e < -\frac{c_3}{c_2}\rho_s\end{aligned} \tag{21}$$

Moreover, since $V(x(t)) \le \rho$ on $[t_k, \tau_1]$, Equations (2a) and (20) imply

$$\|x(t)\| \le \|x(t_k)\| + \|x(t) - x(t_k)\| \le \sqrt{\frac{\rho}{c_1}} + (M_F + W_{\max})\Delta, \quad \forall t \in [t_k, \tau_1] \tag{22}$$

Substituting Equations (21) and (22) into Equation (19), for all $t \in [t_k, \tau_1]$,

$$\begin{aligned}\frac{d}{dt}V(x(t)) &< L'_x(M_F + W_{\max})\Delta - \frac{c_3}{c_2}\rho_s + c_4\left(\sqrt{\frac{\rho}{c_1}} + (M_F + W_{\max})\Delta\right) \\ &\quad W_{\max} \le -\epsilon_w\end{aligned} \tag{23}$$

where the last inequality follows from Equation (16). Integrating Equation (23) over $[t_k, \tau_1]$ gives

$$V(x(\tau_1)) < V(x(t_k)) - \epsilon_w(\tau_1 - t_k) < V(x(t_k)) \le \rho \tag{24}$$

which contradicts $V(x(\tau_1)) = \rho$. ∎

Case 2. If $x(t_k) \in \Omega_{\rho_e}$, let $\tilde{x}(\cdot)$ be the nominal trajectory in the inner-region condition of Equation (10). Then $V(\tilde{x}(t)) \le \rho_e$ for all $t \in [t_k, t_k + \Delta)$. Define $e(t) := x(t) - \tilde{x}(t)$, with $e(t_k) = 0$, so

$$\dot{e}(t) = F(x(t), u(t_k)) - F(\tilde{x}(t), u(t_k)) + w(x(t), u(t_k), t) \tag{25}$$

Hence, by Equation (3a) and $\|w(x,u,t)\| \le W_{\max}$:

$$\|\dot{e}(t)\| \le \|F(x(t), u(t_k)) - F(\tilde{x}(t), u(t_k))\| + \|w(x(t), u(t_k), t)\| \le L_x \|e(t)\| + W_{\max} \tag{26}$$

Integrating the differential inequality in Equation (26) together with $e(t_k) = 0$ gives

$$\|e(t)\| \le \frac{W_{\max}}{L_x}\left(e^{L_x(t-t_k)} - 1\right), \quad \forall t \in [t_k, t_k + \Delta) \tag{27}$$

Also, $V(\tilde{x}(t)) \le \rho_e$ and Equation (2a) imply $c_1 \|\tilde{x}(t)\|^2 \le V(\tilde{x}(t)) \le \rho_e$, so

$$\|x(t)\| \le \|\tilde{x}(t)\| + \|e(t)\| \le \sqrt{\frac{\rho_e}{c_1}} + \frac{W_{\max}}{L_x}\left(e^{L_x \Delta} - 1\right), \quad \forall t \in [t_k, \tau_1] \tag{28}$$

Moreover, for $t \in [t_k, \tau_1]$, by Equation (1),

$$\begin{aligned} \frac{d}{dt}V(x(t)) &= \frac{\partial V(x(t))}{\partial x}(F(x(t), u(t_k)) + w(x(t), u(t_k), t)) \\ &\le \left|\frac{\partial V(x(t))}{\partial x}\right|(\|F(x(t), u(t_k))\| + \|w(x(t), u(t_k), t)\|) \\ &\le c_4 \|x(t)\|(M_F + W_{\max}) \end{aligned} \tag{29}$$

where Equations (2c) and (3b) and $\|w(x,u,t)\| \le W_{\max}$ have been used. Substituting Equation (28) into Equation (29) yields

$$\frac{d}{dt}V(x(t)) \le c_4\left(\sqrt{\frac{\rho_e}{c_1}} + \frac{W_{\max}}{L_x}\left(e^{L_x \Delta} - 1\right)\right)(M_F + W_{\max}), \quad \forall t \in [t_k, \tau_1] \tag{30}$$

Integrating Equation (30) over $[t_k, \tau_1]$, one obtains

$$\begin{aligned} V(x(\tau_1)) &= V(x(t_k)) + \int_{t_k}^{\tau_1} \frac{d}{dt}V(x(t))\, dt \\ &\le V(x(t_k)) + \int_{t_k}^{\tau_1} c_4\left(\sqrt{\frac{\rho_e}{c_1}} + \frac{W_{\max}}{L_x}\left(e^{L_x \Delta} - 1\right)\right)(M_F + W_{\max})\, dt \\ &= V(x(t_k)) + c_4\left(\sqrt{\frac{\rho_e}{c_1}} + \frac{W_{\max}}{L_x}\left(e^{L_x \Delta} - 1\right)\right)(M_F + W_{\max})(\tau_1 - t_k) \\ &\le \rho_e + c_4\left(\sqrt{\frac{\rho_e}{c_1}} + \frac{W_{\max}}{L_x}\left(e^{L_x \Delta} - 1\right)\right)(M_F + W_{\max})\Delta = \rho_e \\ &\quad + \Delta_\rho < \rho \end{aligned} \tag{31}$$

where the second inequality uses $V(x(t_k)) \le \rho_e$ and $\tau_1 - t_k \le \Delta$, and the last inequality follows from the assumption $\rho_e + \Delta_\rho < \rho$. This contradicts $V(x(\tau_1)) = \rho$.

Hence no such τ_1 exists, so $x(t) \in \Omega_\rho$ for all $t \in [t_k, t_k + \Delta)$. Since t_k is arbitrary, the result follows by induction. ∎

The result of Theorem 1 establishes forward invariance of Ω_ρ under the control law given by Equation (12). Building on the discussion in Theorem 1, the next theorem further characterizes the safety behavior of the closed-loop system by showing that, if the state starts in the unsafe region $\mathcal{X}_u \cap \Omega_\rho$, it must enter the safe region $\mathcal{X}_s$ within a finite time.

Theorem 2. *Let $t_e \ge t_0$ satisfy $x(t_e) \in \mathcal{X}_u \cap \Omega_\rho$. Assume $\Omega_{\rho_s} \subset \mathcal{X}_{s,0}$, where $\mathcal{X}_{s,0}$ is the safe neighborhood in Assumption 2, and let*

$$t_{\max} := t_e + \Delta + \frac{\rho - \rho_s}{\epsilon_w} \tag{32}$$

If the assumptions of Theorem 1 hold, then there exists $t_s \in [t_e, t_{\max}]$ such that $x(t_s) \in \mathcal{X}_s$.

Proof. Let t_k be the first sampling instant such that $t_k \ge t_e$. Then $t_k \le t_e + \Delta$. If there exists $t \in [t_e, t_k]$ such that $x(t) \in \mathcal{X}_s$, then the conclusion holds with $t_s = t$. It therefore remains to consider the case $x(t) \in \mathcal{X}_u$ for all $t \in [t_e, t_k]$, which implies $x(t_k) \in \mathcal{X}_u$.

Since $\mathcal{X}_p \subset \mathcal{X}_s$, one has $x(t_k) \notin \mathcal{X}_p$. Moreover, $\Omega_{\rho_s} \subset \mathcal{X}_{s,0} \subset \mathcal{X}_s$ implies $x(t_k) \notin \Omega_{\rho_s}$. Suppose, by contradiction, that the trajectory does not enter $\mathcal{X}_s$ on $[t_k, t_{\max}]$, i.e., $x(t) \in \mathcal{X}_u$ for all $t \in [t_k, t_{\max}]$. Then every sampling instant $t_j \in [t_k, t_{\max}]$ satisfies $x(t_j) \in \mathcal{X}_u \subset \Omega_\rho \setminus \mathcal{X}_p$, and hence $x(t_j) \notin \Omega_{\rho_s}$. Therefore, on each sampling interval $[t_j, t_{j+1}) \subset [t_k, t_{\max}]$, the case 1 of Theorem 1 applies, so $\frac{d}{dt}V(x(t)) \le -\epsilon_w$. Applying this successively over all such intervals yields

$$V(x(t)) \le V(x(t_k)) - \epsilon_w(t - t_k) \le \rho - \epsilon_w(t - t_k), \quad \forall t \in [t_k, t_{\max}] \tag{33}$$

where the last inequality follows from $x(t_k) \in \Omega_\rho$.

Now define $\hat{t} := t_k + (\rho - \rho_s)/\epsilon_w$. Since $t_k \le t_e + \Delta$, one has $\hat{t} \le t_{\max}$. Evaluating Equation (33) at $t = \hat{t}$ gives

$$V(x(\hat{t})) \le \rho - \epsilon_w(\hat{t} - t_k) = \rho - \epsilon_w \frac{\rho - \rho_s}{\epsilon_w} = \rho_s \tag{34}$$

Hence $x(\hat{t}) \in \Omega_{\rho_s} \subset \mathcal{X}_{s,0} \subset \mathcal{X}_s$, contradicting the assumption that $x(t) \in \mathcal{X}_u$ for all $t \in [t_k, t_{\max}]$. Therefore, there exists $t_s \in [t_k, \hat{t}] \subset [t_e, t_{\max}]$ such that $x(t_s) \in \mathcal{X}_s$. ∎

Theorems 1 and 2 also show how the sampling period Δ affects the admissible choices of ρ_e and ρ_s for a fixed outer set Ω_ρ. In particular, Δ appears in the positive terms of Equation (16) through $L'_x(M_F + W_{\max})\Delta$ and $c_4\left(\sqrt{\rho/c_1} + (M_F + W_{\max})\Delta\right)W_{\max}$, so a smaller Δ reduces the sample-and-hold contribution in Equation (16). Thus, for the same fixed ρ, one can admit a smaller ρ_s while still satisfying Equation (16); this is useful because the requirement $\Omega_{\rho_s} \subset \mathcal{X}_{s,0}$ in

Theorem 2 is easier to realize for a smaller level set Ω_{ρ_s}. Likewise, since Δ_ρ in Equation (17) decreases with Δ, the condition $\rho_e + \Delta_\rho < \rho$ can be satisfied with a larger ρ_e for fixed ρ, and therefore with a larger inner set Ω_{ρ_e}. Hence, for a fixed Ω_ρ, a smaller feasible sampling period enlarges the admissible range of ρ_e and ρ_s: ρ_e can be taken larger and ρ_s can be taken smaller under the same inequalities. This is particularly relevant for the proposed RL-based controller, because the learned policy can be evaluated online with much lower computational burden than repeatedly solving the backup optimization problem. As a result, the RL-based controller can support a smaller feasible Δ.

Remark 4. The finite-time recovery result in Theorem 2 should be distinguished from forward invariance of the safe set $\mathcal{X}_s$. The proposed switching law does not require the final fallback controller $\Phi(x)$ to render $\mathcal{X}_s$ or $\mathcal{X}_p$ forward invariant. Rather, $\Phi(x)$ is used as a stabilizing fallback to preserve operation within the outer Lyapunov region Ω_ρ when the LEMPC backup is unavailable. The recovery statement relies on the outer Lyapunov-decrease condition and the inclusion $\Omega_{\rho_s} \subset \mathcal{X}_{s,0} \subset \mathcal{X}_s$; it does not imply monotonic decrease of the Safeness index $S(x)$ or forward invariance of $\mathcal{X}_s$ after entry.

4 | APPLICATION TO A CHEMICAL PROCESS EXAMPLE

4.1 | Process description

We consider a well-mixed, non-isothermal continuous stirred-tank reactor (CSTR) in which a single irreversible, exothermic, second-order liquid-phase reaction takes place, $A \to B$. The feed stream contains pure reactant A with inlet concentration C_{A0} and inlet temperature T_0 and enters the reactor at constant volumetric flow rate F. Heat is added to or removed from the reactor at a rate $\dot{Q}$. The reactor is assumed perfectly mixed with constant liquid volume V_L, density ρ_L, and heat capacity C_p.

The states are the reactant concentration C_A and the reactor temperature T, which evolve according to standard CSTR mass and energy balances:

TABLE 1 Parameter values of the CSTR model.

Parameter	Value	Parameter	Value
C_{As}	$1.2\,\mathrm{kmol\,m^{-3}}$	C_{A0s}	$4.0\,\mathrm{kmol\,m^{-3}}$
C_p	$0.231\,\mathrm{kJkg^{-1}K^{-1}}$	ΔH	$-11{,}500\,\mathrm{kJkmol^{-1}}$
E	$5.0\times10^{4}\,\mathrm{kJkmol^{-1}}$	F	$5\,\mathrm{m^3h^{-1}}$
k_0	$8.46\times10^{6}\,\mathrm{m^3kmol^{-1}h^{-1}}$	$\dot{Q}_s$	$0.0\,\mathrm{kJh^{-1}}$
R	$8.314\,\mathrm{kJkmol^{-1}K^{-1}}$	ρ_L	$1.0\times10^{3}\,\mathrm{kgm^{-3}}$
T_0	300.0K	T_{0s}	300.0K
T_s	438.0K	V_L	$1\,\mathrm{m^3}$

$$\frac{dC_A}{dt} = \frac{F}{V_L}(C_{A0} - C_A) - k_0 \exp\left[-\frac{E}{RT}\right] C_A^2 \tag{35a}$$

$$\frac{dT}{dt} = \frac{F}{V_L}(T_0 - T) - \frac{\Delta H}{\rho_L C_p} k_0 \exp\left[-\frac{E}{RT}\right] C_A^2 + \frac{\dot{Q}}{\rho_L C_p V_L} \tag{35b}$$

where ΔH, k_0, E, and R denote the heat of reaction, pre-exponential factor, activation energy, and ideal-gas constant, respectively. Parameter values are listed in Table 1. The parameter set admits three steady states; throughout this work, we operate around the open-loop asymptotically stable steady state $[C_{As}, T_s]$ with associated steady-state inputs $\left[C_{A0s}, \dot{Q}_s\right]$.

4.2 | Control problem

To formulate this problem, we work in deviation variables around the open-loop asymptotically stable steady state introduced in Section 4.1, so that the steady state corresponds to the origin. Specifically, $x := [\,C_A - C_{As},\, T - T_s\,]^\top \in \mathbb{R}^2$ and $u := \left[\,C_{A0} - C_{A0s},\, \dot{Q} - \dot{Q}_s\,\right]^\top \in \mathbb{R}^2$, where u_1 denotes the deviation in inlet concentration and u_2 denotes the deviation in heat input/removal rate. The manipulated inputs are subject to the bounds $|u_1| \le 3.5\,\mathrm{kmol\,m^{-3}}$ and $|u_2| \le 5\times10^{5}\,\mathrm{kJh^{-1}}$. Under this formulation, the control objective is to maximize the production of product B over the operating window $t_p = 1.0$h. Thus, the stage economic objective is defined as

$$L_e(x) := k_0 \exp\left[-\frac{E}{RT}\right] C_A^2 \tag{36}$$

In addition, we impose the following resource constraint over a fixed operating period $t_p = 1.0$h:

$$\frac{1}{t_p}\int_0^{t_p} u_1(\tau)\, d\tau = 0 \tag{37}$$

The reference controller $\phi = [\phi_1, \phi_2]$ in Section 2.3 is designed as follows. The inlet-concentration action u_1 (denoted ϕ_1) is chosen to satisfy the input bounds and the material constraint Equation (37); Specifically, ϕ_1 is set as $0\,\mathrm{kmol/m^3}$ and clipped to the feasible region. Given any admissible ϕ_1, the heat-rate action u_2 (denoted ϕ_2) is computed via a scalar Sontag law.[33] Specifically, Equation (35) can be written as the form of $\dot{x} = f(x) + g_1(x)u_1 + g_2(x)u_2$. Define $a(x) := L_f V(x) + L_{g_1} V(x)\phi_1$ and $b(x) := L_{g_2} V(x)$, where $L_f V(x) := \frac{\partial V(x)}{\partial x} f(x)$ and $L_{g_i} V(x) := \frac{\partial V(x)}{\partial x} g_i(x)$. Then we set[33]:

$$\phi_2(x) := \begin{cases} -\dfrac{a(x) + \sqrt{a(x)^2 + b(x)^4}}{b(x)}, & b(x) \neq 0 \\ 0, & b(x) = 0 \end{cases} \tag{38}$$

followed by saturation of ϕ_2 to $[u_{2,\min}, u_{2,\max}]$. We use a quadratic Lyapunov function $V(x) = x^\top P x$ with

$$P = \begin{bmatrix} 1{,}060 & 22 \\ 22 & 0.52 \end{bmatrix} \tag{39}$$

Based on closed-loop simulations under ϕ, we select $\rho = 368$ and $\rho_e = 320$ in this study.

In terms of safety, we would like to avoid operation at excessively high reactor temperatures. To characterize unsafe operation, we define the Safeness index for this CSTR as

$$S(x) = \frac{ax_1 + bx_2}{\max\{ax_1 + bx_2 : V(x) \le \rho\}} \tag{40}$$

where a and b are weighting constants. In this study, we set $a = b = 1$. Since the temperature deviation x_2 is several orders of magnitude larger than the concentration deviation x_1, the Safeness index is dominated by temperature, and larger values of $S(x)$ correspond to less safe operating conditions. Over the stability region Ω_ρ, the normalizing constant is $\max\{x_1 + x_2 : V(x) \le \rho\} = 74.46$, so $S(x) \in [-1, 1]$. We set the Safeness threshold to $S_{\text{th}} = 0.6$, which limits the reactor temperature deviation to approximately $x_2 \le 47$ K.

To test the robustness of various designed controllers, the closed-loop simulations are carried out under additive bounded disturbances w with zero-mean Gaussian distribution and standard deviations $(\sigma_1, \sigma_2) = (1.5, 15)$, clipped to satisfy $|w_1| \le 3$ and $|w_2| \le 30$.

Remark 5. In this work, neither the RL discussed in Section 4.3 nor the FNN discussed in Section 4.4.2 is trained with explicit disturbance information; both are trained using the nominal process model, and the evaluations performed during offline training are also based on the nominal model. Extending the learning design to explicitly account for disturbances, e.g., through offset-free approaches, is outside the scope of this study.

Remark 6. In this study, the sampling period is fixed at $\Delta = 0.01$ h for all controllers to enable a fair closed-loop comparison under the same update frequency. In practice, however, the feasible sampling period may vary across controllers because their online control-decision times can be different. In particular, controllers that require more online computation may need a larger sampling period, whereas an RL controller can often be implemented with a smaller one.

4.3 | Safe economic RL-based control framework design

For the CSTR considered in this work, the safe economic RL-based control framework shown in Section 3 is implemented by using the LEMPC in Section 2.5 and Sontag law ϕ as the backup controllers and training the RL policy in the shielded environment of Section 3.3. The design details of the LEMPC are shown in Section 4.4.1.

The RL state is chosen as $s_k = [x(t_k)^\top, l_{CA0}(t_k), t_k/t_p]^\top$, where $l_{CA0}(t_k) := \int_0^{t_k} u_1(\tau)\, d\tau$ denotes the accumulated inlet-concentration usage up to time t_k. The state $x(t_k)$ is augmented with $l_{CA0}(t_k)$ and t_k/t_p because both quantities influence the control action decision through the material constraint and the remaining time in the operating period. The actor outputs a normalized action $a_k \in [-1, 1]^2$, which is mapped to a physical input proposal $u_{\text{RL}}(t_k) \in U$. As in Equation (14), the replay buffer stores the raw action proposal, while the state transition and reward are generated by the implemented input $u(t_k)$. The reward is chosen to be the same as the economic stage objective, i.e., $r_e = L_e$. As discussed in Section 2.6, TD3 is adopted as the underlying reinforcement-learning algorithm.

4.3.1 | RL training

Following Section 3.3, the constraint-informed 2-stage RL training is used. During training, initial states are sampled over the whole Ω_ρ, including unsafe states, and the reward is penalized whenever the backup controller is used.

In Stage 1, training starts from scratch with an initial random-exploration phase and periodic policy evaluation every 1000 training steps up to a total of 4×10^5 steps. Several fixed values of β_1 are tested to identify a suitable penalty level for learning a highly admissible initial policy with low backup usage. During each training run, we save the RL policy that achieves the highest cumulative reward among the evaluated policies whose backup usage is below 20%. As shown in Figure 2, $\beta_1 = 1$ leads to persistently high backup usage, whereas

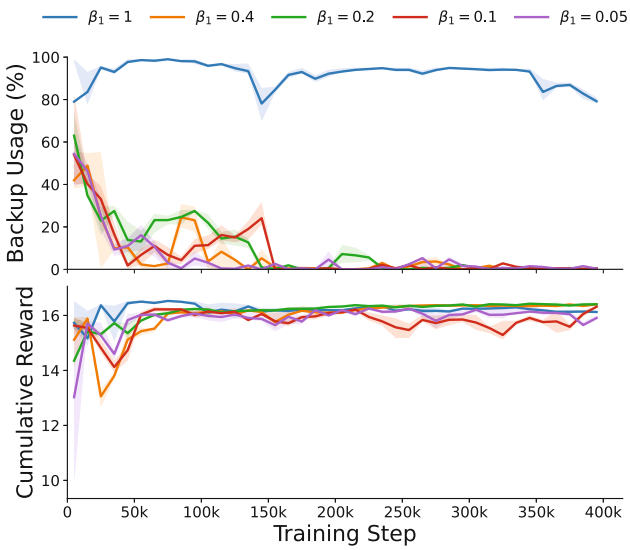

FIGURE 2 Stage 1 training performance for different fixed values of β_1. The upper panel shows backup usage, and the lower panel shows cumulative reward. In each panel, the solid curves represent the mean over evaluations within each 10^4-step interval, and the shaded regions indicate the 25%–75% range.

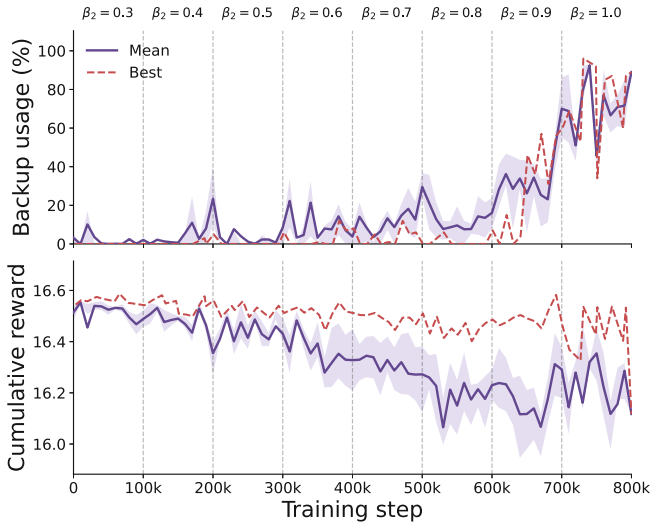

FIGURE 3 Stage 2 training performance as β_2 is gradually increased. The upper panel shows backup usage, and the lower panel shows cumulative reward. In each panel, the solid curves represent the mean over evaluations within each 10^4-step rollback interval, the shaded regions indicate the 25%–75% range, and the dashed curves denote the best cumulative reward found within each rollback interval and its corresponding backup usage.

smaller values reduce backup usage substantially without severe loss in cumulative reward. Among these low-intervention cases, $\beta_1 = 0.2$ gives one of the highest cumulative rewards while satisfying the backup-usage requirement, and the selected checkpoint also achieves zero backup usage. Therefore, Stage 1 is carried out with $\beta_1 = 0.2$.

In Stage 2, training is initialized from the best Stage 1 checkpoint, and β_2 is increased from 0.3 to 1.0 in increments of 0.1. At each penalty level, training is carried out for 10^5 steps, the replay buffer is reset, and policy evaluation is performed every 10^3 steps. To keep training near the good Stage 1 solution, the agent is rolled back every 10^4 steps to the best previously saved constrained snapshot, where "best" refers to the evaluated policy with the highest cumulative reward among those whose backup usage is below 50%. In addition, both the overall best policy and the best constrained policy with backup usage below 50% are tracked, with the latter used for rollback and model selection. The final selected RL policy achieves a cumulative reward of 16.59 with zero backup usage.

The corresponding training behavior is shown in Figure 3. For small values of β_2, the low-backup behavior learned in Stage 1 is largely preserved. As β_2 increases, backup usage becomes more pronounced, especially in the later stages, indicating that the weaker intervention penalty allows exploration farther away from the original low-backup solution. The cumulative reward does not show a clear monotonic improvement as β_2 increases, and the best-performing candidates in later stages are generally associated with larger backup usage. Overall, Stage 2 serves as a warm-started local continuation procedure that seeks further economic improvement while preserving the admissible behavior learned in Stage 1.

The TD3 hyperparameters used in the two stages are listed in Table 2. Compared with Stage 1, Stage 2 uses smaller learning rates and lower exploration noise so that the policy can be refined more conservatively around the admissible behavior obtained in Stage 1.

TABLE 2 TD3-based RL hyperparameters for Stage 1 and Stage 2.

Hyperparameter	Stage 1	Stage 2
Replay buffer size	10^6	10^6
Mini-batch size	256	256
Discount factor γ	0.99	0.99
Target update τ	0.005	0.005
Policy delay	2	2
Actor learning rate	3×10^{-4}	1×10^{-4}
Critic learning rate	3×10^{-4}	1×10^{-4}
Exploration noise	0.10	0.02
Target policy smoothing noise	0.2	0.05
Target policy smoothing clip	0.5	0.2

Remark 7. During RL training, or during LEMPC data generation for Section 4.4.2, if the LEMPC optimization fails, the current episode is terminated, and the entire trajectory is discarded rather than added to the replay buffer or supervised dataset. In online implementation, one may still apply the stabilizing controller $\Phi(x)$ after such a failure so that the control action remains well defined. During training and data generation, however, we do not retain these trajectories because $\Phi(x)$ is introduced for stability recovery rather than for economic optimization. Therefore, the resulting transitions are not representative of the intended economically optimized shielded-policy behavior, and including them would bias the dataset toward fallback behavior that is not the target of learning.

Remark 8. Although the final RL policy selected in this study achieves zero backup usage in the evaluation (no disturbance), this is not a necessary outcome of the proposed training framework. In general, the final selected RL policy may still exhibit nonzero backup usage if that policy provides the most favorable balance between cumulative reward and admissibility. Further discussion is given in Section 4.3.2.

4.3.2 | Closed-loop performance

We first highlight the benefit of the safe online implementation by comparing the proposed shielded RL controller with the corresponding unshielded RL controller under the same bounded disturbances. RL 1 denotes the best-performing policy selected from the training

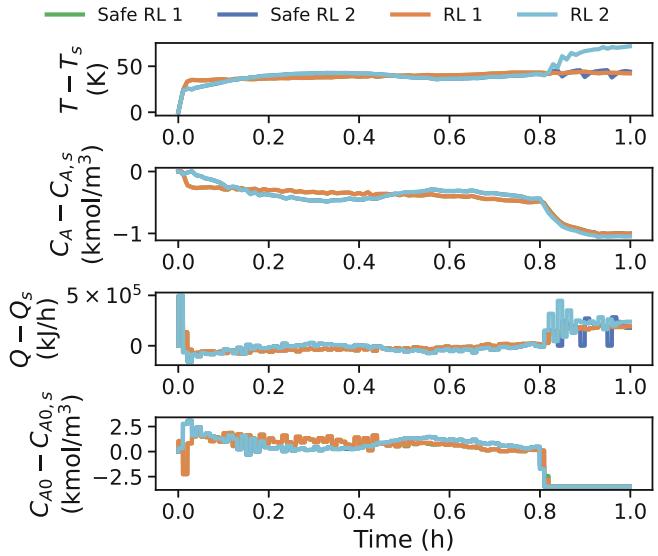

FIGURE 4 Closed-loop dynamic responses under bounded disturbances for the proposed safe RL-based controller and the nominal RL-based controller with initial state $x(0) = [0,0]^\top$.

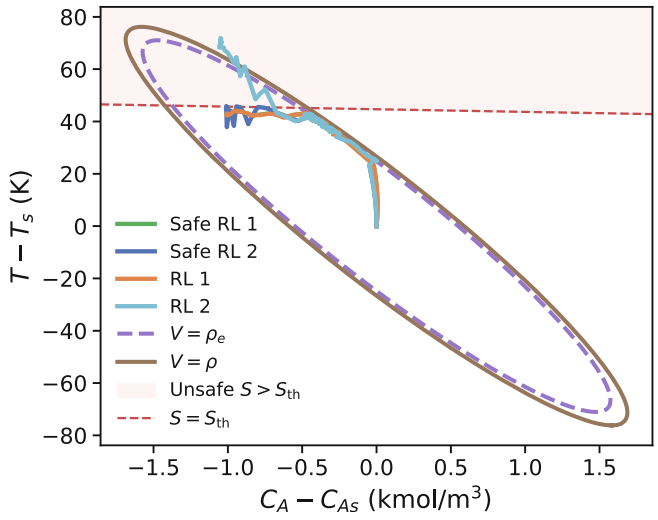

FIGURE 5 State-space trajectories under bounded disturbances for the proposed safe RL-based controller and the nominal RL-based controller with initial state $x(0) = [0,0]^\top$.

procedure above. As shown in Figures 4 and 5, the difference between Safe RL 1 and RL 1 is relatively small, which indicates that when the learned policy already proposes highly admissible actions, the backup controller is only minimally involved and the safe online implementation is nearly non-intrusive.

However, even though the constraint-informed training can efficiently reduce reliance on the backup controller, it does not eliminate the need for backup during online implementation. To illustrate this point more clearly, we also include another learned policy, denoted RL 2, obtained using the same training method but with higher reliance on the backup controller. In this case, if the backup layer is removed and the RL policy is implemented directly, the closed-loop behavior becomes unsafe. As shown in Figure 4, RL 2 leads to a pronounced temperature rise near the end of the operating period together with corresponding deviations in the manipulated inputs. The state-space plot in Figure 5 further shows that the RL 2 trajectory enters the unsafe region $S > S_{th}$ and even leaves the prescribed stability region Ω_ρ. By contrast, the corresponding Safe RL 2 trajectory remains inside the admissible operating region because the backup controller intervenes whenever the RL proposal fails the online acceptance test.

Therefore, these results show that the backup controller remains important even after constraint-informed training. When the learned policy is reliable, as in RL 1, the safe online implementation is nearly non-intrusive. When the learned policy is less reliable, as in RL 2, the backup controller is essential for preventing unsafe operation and loss of stability.

To further illustrate the effect of the safe online implementation, we also test several initial states outside $\mathcal{X}_p$. The resulting closed-loop state-space responses are shown in Figures 6 and 7, and the corresponding Safeness index trajectories are shown in Figure 8. Figure 6 shows the locations of the closed-loop states at several representative time instants, while Figure 7 shows the corresponding full trajectories. Starting from initial conditions outside $\mathcal{X}_p$, including unsafe initial states with $S > S_{th}$, the trajectories are first driven quickly away from the unsafe region and back into the admissible operating region. This behavior is also reflected in Figure 8, where the Safeness index for the initially unsafe cases decreases rapidly below the threshold S_{th}, indicating fast recovery of safe operation. After this transient recovery, the state trajectories move toward nearly the same region in the state space, and the corresponding Safeness index trajectories merge to nearly the same level. This indicates that, although the initial states are different and some are initially unsafe, the proposed controller first restores safety and then steers the process toward a common economically favorable operating region.

4.4 | Benchmark with different controllers

4.4.1 | Safe LEMPC design

To implement the backup controller and provide an optimization-based benchmark, we design a safe LEMPC based on the formulation in Section 2.5. For efficient online computation, the objective function and constraints are implemented in CasADi, and the resulting nonlinear program is solved using SciPy's SLSQP solver with tolerance 10^{-10}, finite-difference step size 2×10^{-4}, and a maximum of 500 iterations. Two prediction horizon lengths are considered. A baseline Safeness index-based LEMPC with $N = 10$ is used for closed-loop performance comparison, while a short-horizon Safeness index-based LEMPC with $N = 2$ is used in the shield as a computationally efficient backup controller.

4.4.2 | Imitation learning

Since imitation learning can also produce a direct state-to-action feedback law and thereby significantly reduce the online computation time

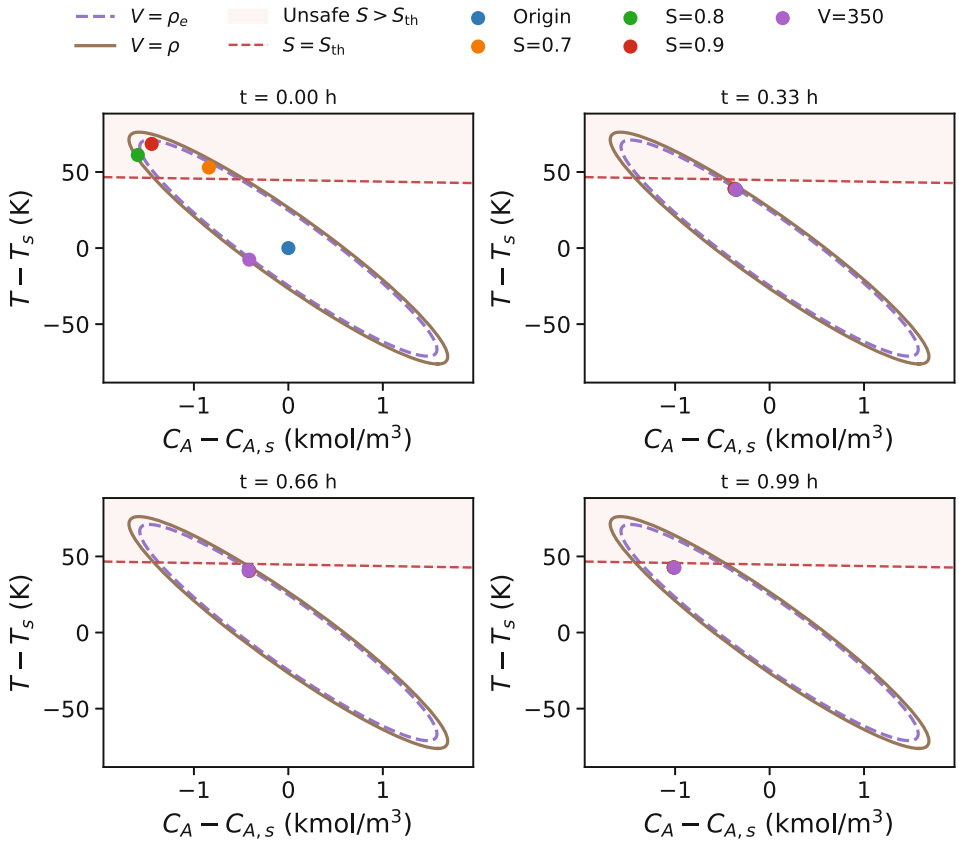

FIGURE 6 State-space snapshots at different time instants under the proposed safe RL-based controller from initial states at origin and outside $\mathcal{X}_p$.

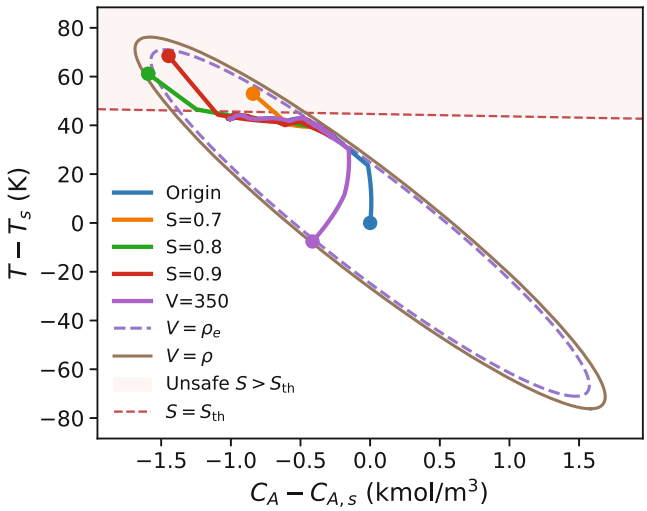

FIGURE 7 Full state-space trajectories under the proposed safe RL-based controller from initial states at origin and outside $\mathcal{X}_p$.

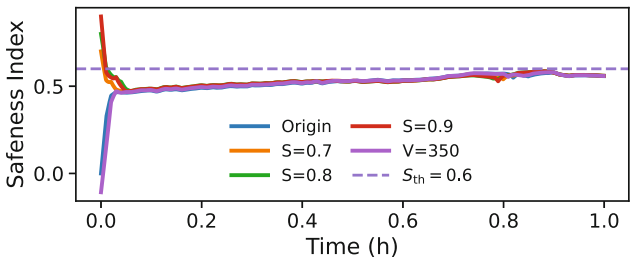

FIGURE 8 Safeness index under the proposed safe RL-based controller from initial states at origin and outside $\mathcal{X}_p$.

relative to repeatedly solving the LEMPC optimization problem, it serves as a natural comparison candidate for the proposed RL-based controller. Therefore, we train an FNN via imitation learning to approximate the baseline LEMPC policy. To this end, we generate a supervised dataset by running the baseline LEMPC in closed-loop simulations from diverse initial conditions sampled from Lyapunov rings of $V(x)=x^\top Px$ over $V\in[0,\rho]$ with $N_{\text{ring}}=100$. In total, we generate 10^3 trajectories with 10^5 data points.

At each sampling instant t_k, the FNN input is $s_k=[x(t_k)^\top, I_{CA0}(t_k), t_k/t_p]^\top$, and the output is the implemented control action $u(t_k)$. The inputs are normalized using min-max scaling, and the outputs are normalized using `tanh` activation. The dataset is split into training, validation, and test sets in the ratio 70%/15%/15%. The FNN is trained using Adam to minimize the mean-squared error in the bounded action space with early stopping. The hyperparameters are selected by Bayesian optimization to minimize the mean normalized MSE on the validation set, yielding the following optimal values: Learning rate 4.84×10^{-3}, batch size 64, width 128, depth 2, and dropout 0.0144.

As shown in Figure 9, the test-set prediction errors for both u_1 and u_2 are strongly concentrated near zero. In both subfigures, most of the histogram mass lies in the small-error region, and the cumulative distribution rises rapidly near the origin, indicating that a large fraction of the FNN predictions are very close to the corresponding LEMPC actions. Only a relatively small portion of the samples falls in the larger-error region. Moreover, the approximation for u_2 appears slightly tighter than that for u_1, which contains a higher count of high relative error outliers. One possible reason is that u_1 is also influenced by the material constraint in Equation (37), so its value depends not only on the current process state but also more sensitively on the accumulated inlet-concentration usage and the remaining time in the operating period. This makes the input-output mapping for u_1 less predictable and therefore harder to approximate accurately by a static FNN. Overall, Figure 9 indicates that the trained FNN reproduces the dominant input-output behavior of the baseline LEMPC sufficiently well to serve as a meaningful imitation-learning benchmark.

Remark 9. In this work, the FNN is trained to minimize MSE so as to mimic the LEMPC input-output mapping. If one also wishes to reduce backup usage during deployment, additional penalty terms can be introduced to discourage infeasible actions from FNN; however, doing so typically trades off imitation fidelity and may degrade how closely the FNN reproduces LEMPC actions.

4.4.3 | Comparison of different control approaches

To evaluate the closed-loop performance of the proposed RL-based controller, we compare it with the LEMPC and FNN

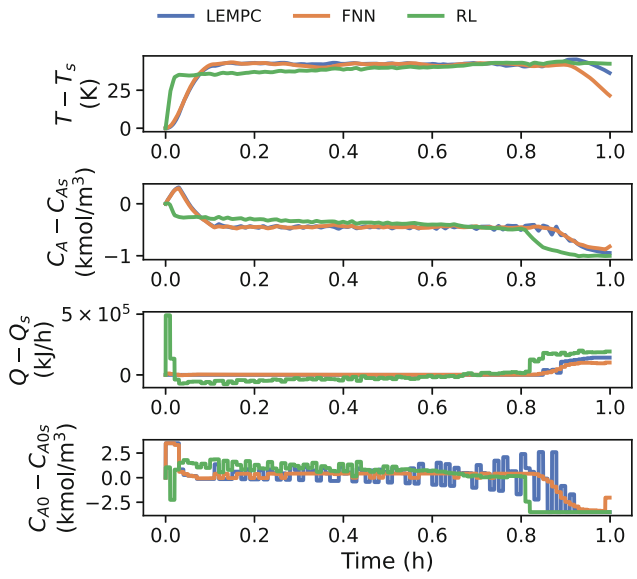

FIGURE 10 Closed-loop dynamic responses under bounded disturbances for LEMPC, imitation learning (FNN), and the proposed RL-based controller with initial states at $x(0) = [0,0]^\top$.

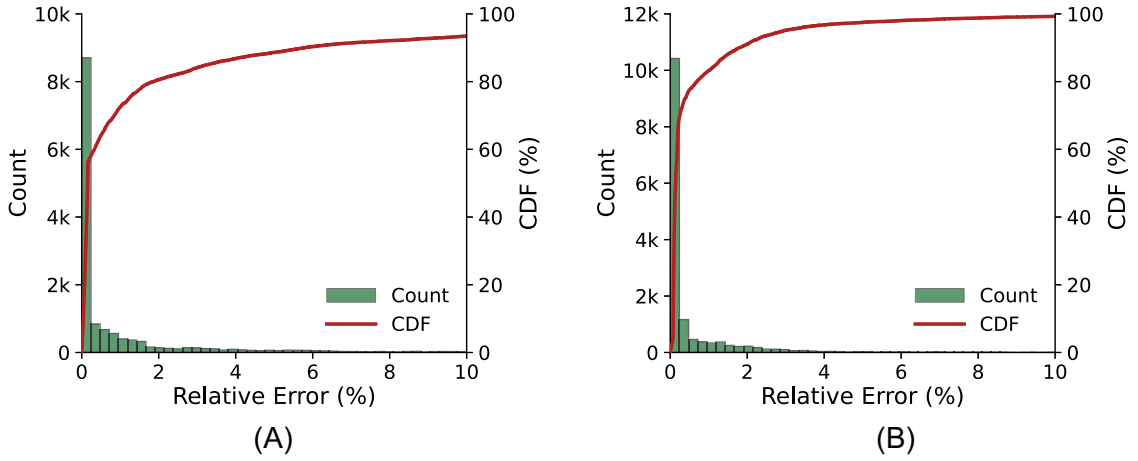

FIGURE 9 Test set relative prediction error distributions and cumulative distributions for the FNN approximation of the LEMPC actions u_1 and u_2. (A) u_1, (B) u_2.

Controller	Cumulative reward	Improvement	Backup	t_{max} (s)	t_{ave} (s)
LEMPC	16.64	19.59%	—	6.182	0.233
FNN	16.51	18.66%	25%	1.027	0.017
RL	16.75	20.39%	6%	1.079	0.016

Note: The improvement is computed relative to the open-loop cumulative reward $J_{op} = 13.91$. t_{ave} and t_{max} denote the average and maximum computation times for one control action, respectively.

TABLE 3 Performance comparison of different controllers.

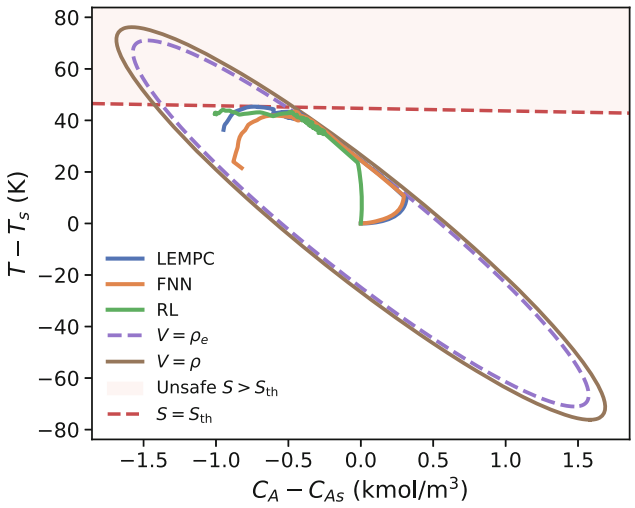

FIGURE 11 State-space trajectories under bounded disturbances for LEMPC, imitation learning (FNN), and the proposed RL-based controller with initial states at $x(0) = [0,0]^\top$.

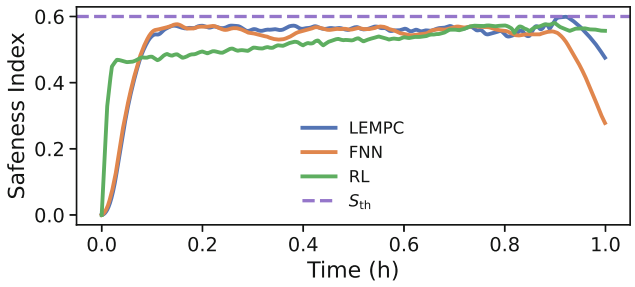

FIGURE 12 Safeness index under bounded disturbances for LEMPC, imitation learning (FNN), and the proposed RL-based controller with initial states at $x(0) = [0,0]^\top$.

controllers under the same operating constraints. To quantify the economic benefit of each controller, we use as a baseline the cumulative reward of the open-loop process operated at the steady-state input $u \equiv 0$ under disturbances, which is $J_{op} = 13.91$. The improvement of a controller is then computed relative to this open-loop value.

Table 3 summarizes the closed-loop cumulative reward, improvement relative to the open-loop baseline, backup usage, and online computation time for the three controllers. The proposed RL-based controller achieves the highest cumulative reward, 16.75, corresponding to an improvement of 20.39%, which is higher than that of the LEMPC (16.64, 19.59%) and the FNN controller (16.51, 18.66%). In addition, the RL controller requires backup only 6% of the time, which is substantially less than the FNN controller (25%). In terms of online computation, the RL controller is significantly faster than the LEMPC, with $t_{ave} = 0.016$ s and $t_{max} = 1.079$ s, compared with $t_{ave} = 0.233$ s and $t_{max} = 6.182$ s for the LEMPC. Compared with the FNN controller, the RL controller achieves better economic performance with essentially the same online computation time.

Figures 10 and 11 show that, despite bounded disturbances, all three controllers drive the reactor quickly from the initial steady state to an operating region that improves the economic objective. Due to the high production rate, the concentration decreases a lot. A higher temperature due to the larger heat input accelerates the production rate. All trajectories remain inside the Lyapunov function level set Ω_ρ, consistent with the stability guarantees, and stay close to the safe-region boundary for much of the operating time. The main difference appears near the end of the operating time, where the material constraint on u_1 forces a terminal correction in both inputs. Figure 12 shows that all three controllers operate under the allowable safeness index requirements during the operating time. Overall, the RL controller preserves safe bounded operation while achieving the best economic performance with low online computational cost.

5 | CONCLUSION

This work developed a safe economic RL-based control framework for nonlinear systems using supervisory backup controllers to enforce prescribed safety and stability conditions. The proposed method achieves both economic improvement and operational safety through a constraint-informed two-stage training strategy and a safe online implementation mechanism. Theoretical results established forward invariance of the admissible operating region with respect to closed-loop stability and demonstrated recovery from unsafe operation. Using a CSTR example, the implementation of the proposed method was clarified, the significance of the backup controllers was demonstrated, and trajectories starting from initial states outside $\mathcal{X}_p$ were successfully driven back into the safe operating region. Compared with the benchmark controllers, the proposed RL controller achieved the best economic performance while maintaining safe operation and substantially reducing online computational burden relative to LEMPC. These results demonstrate that the proposed framework provides an effective way to integrate the low online cost of RL with model-based safety and stability guarantees for nonlinear process control.

AUTHOR CONTRIBUTIONS

Xiaodong Cui: conceptualization; methodology; software; investigation; writing – original draft. **Arthur Khodaverdian:** conceptualization; methodology; investigation; writing – original draft. **Panagiotis D. Christofides:** conceptualization; methodology; investigation; writing – review and editing.

ACKNOWLEDGMENTS

Financial support from the National Science Foundation, CBET-2227241, is gratefully acknowledged.

FUNDING INFORMATION

This work was supported by National Science Foundation, CBET-2227241.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

ORCID

Panagiotis D. Christofides https://orcid.org/0000-0002-8772-4348

REFERENCES

1. Rangaiah G, Kariwala V. *Plantwide Control: Recent Developments and Applications*. Wiley; 2012.
2. Ellis M, Durand H, Christofides PD. A tutorial review of economic model predictive control methods. *J Process Control*. 2014;24:1156-1178.
3. Albalawi F, Durand H, Christofides PD. Process operational safety via model predictive control: Recent results and future research directions. *Comput Chem Eng*. 2018;114:171-190.
4. Lees F. *Lees' Loss Prevention in the Process Industries: Hazard Identification, Assessment and Control*. Butterworth-Heinemann; 2012.
5. Stauffer T, Chastain-Knight D. Do not let your safe operating limits leave you S-O-L (out of luck). *Process Saf Prog*. 2021;40:e12163.
6. Qin SJ, Badgwell TA. A survey of industrial model predictive control technology. *Control Eng Pract*. 2003;11:733-764.
7. Mhaskar P, El-Farra NH, Christofides PD. Stabilization of nonlinear systems with state and control constraints using Lyapunov-based predictive control. *Syst Control Lett*. 2006;55:650-659.
8. Albalawi F, Durand H, Christofides PD. Process operational safety using model predictive control based on a process Safeness Index. *Comput Chem Eng*. 2017;104:76-88.
9. Chen WH, Ballance DJ, O'Reilly J. Model predictive control of nonlinear systems: Computational burden and stability. *IEE Proc Control Theory Appl*. 2000;147:387-394.
10. Khodaverdian A, Cui X, Christofides PD. Utilizing reinforcement learning in feedback control of nonlinear processes with stability guarantees. *Digit Chem Eng*. 2025;17:100277.
11. Cui X, Khodaverdian A, Christofides PD. Robust reinforcement learning for nonlinear process control with stability guarantees. *Digit Chem Eng*. 2026;18:100291.
12. Liu S, Liu J. Economic model predictive control with extended horizon. *Automatica*. 2016;73:180-192.
13. Löfberg J. Oops! I cannot do it again: Testing for recursive feasibility in MPC. *Automatica*. 2012;48:550-555.
14. Kaelbling LP, Littman ML, Moore AW. Reinforcement learning: A survey. *J Artif Intell Res*. 1996;4:237-285.
15. Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *Proceedings of International Conference on Machine Learning*; 2018:1861-1870.
16. Sutton RS, Barto AG. *Reinforcement Learning: An Introduction*. Vol 1. MIT Press Cambridge; 1998.
17. Han M, Yin X. Deep Neural Koopman Operator-Based Economic Model Predictive Control of Shipboard Carbon Capture System. *IEEE Trans Control Syst Technol*. 2025;33:2064-2079.
18. Yao J, Han M, Yin X. Lyapunov-based distributed reinforcement learning control with stability guarantee. *Comput Chem Eng*. 2025;195:108979.
19. Zhu X, Wang Y, Wu Z. Reinforcement learning for optimal control of stochastic nonlinear systems. *AICHE J*. 2025;71:e18840.
20. Wang Y, Zhu X, Wu Z. A tutorial review of policy iteration methods in reinforcement learning for nonlinear optimal control. *Digit Chem Eng*. 2025;15:100231.
21. Faria RDR, Capron BDO, Secchi AR, de Souza MB Jr. Where reinforcement learning meets process control: Review and guidelines. *Processes*. 2022;10:2311.
22. Levine S, Kumar A, Tucker G, Fu J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. arXiv preprint arXiv:200501643 2020.
23. Elmaz F, Di Caprio U, Wu M, et al. Reinforcement learning-based approach for optimizing solvent-switch processes. *Comput Chem Eng*. 2023;176:108310.
24. Ernst D, Glavic M, Capitanescu F, Wehenkel L. Reinforcement learning versus model predictive control: a comparison on a power system problem. *IEEE Trans Syst Man Cybern*. 2008;39:517-529.
25. Garcia J, Fernández F. A comprehensive survey on safe reinforcement learning. *J Mach Learn Res*. 2015;16:1437-1480.
26. Chow Y, Nachum O, Duenez-Guzman E, Ghavamzadeh M. A Lyapunov-based approach to safe reinforcement learning. *Adv Neural Inf Proces Syst*. 2018;31.
27. Dulac-Arnold G, Mankowitz D, Hester T. Challenges of real-world reinforcement learning. arXiv preprint arXiv:190412901 2019.
28. Berkenkamp F, Turchetta M, Schoellig A, Krause A. Safe model-based reinforcement learning with stability guarantees. *Adv Neural Inf Proces Syst*. 2017;30.
29. Osinenko P, Dobriborsci D, Aumer W. Reinforcement learning with guarantees: a review. *IFAC-PapersOnLine*. 2022;55:123-128.
30. Fujimoto S, Hoof H, Meger D. Addressing function approximation error in actor-critic methods. *Proceedings of International Conference on Machine Learning*; 2018:1587-1596.
31. Kasaura K, Miura S, Kozuno T, Yonetani R, Hoshino K, Hosoe Y. Benchmarking actor-critic deep reinforcement learning algorithms for robotics control with action constraints. *IEEE Rob Autom Lett*. 2023;8(8):4449-4456.
32. Hassanpour H, Corbett B, Mhaskar P. A practical reinforcement learning control design for nonlinear systems with input and output constraints. *Comput Chem Eng*. 2025;201:109248.
33. Lin Y, Sontag ED. A universal formula for stabilization with bounded controls. *Syst Control Lett*. 1991;16:393-397.

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

How to cite this article: Cui X, Khodaverdian A, Christofides PD. Reinforcement learning-based control of nonlinear systems: Economic optimization and operational safety. *AIChE J*. 2026;72(9):e70479. doi:10.1002/aic.70479