Systems & Control Transactions

Research Article - Peer Reviewed Conference Proceeding
ESCAPE 35 - European Symposium on Computer Aided Process Engineering
Ghent, Belgium. 6-9 July 2025
Jan F.M. Van Impe, Grégoire Léonard, Satyajeet S. Bhonsale,
Monika E. Polańska, Filip Logist (Eds.)

PSE PRESS

Safe Reinforcement Learning with Lyapunov-Based Constraints for Control of an Unstable Reactor

José R. Torraca Neto[a*], Bruno D. O. Capron[a], Argimiro R. Secchi[a,b] and Antonio d.R. Chanona[c]

[a] Universidade Federal do Rio de Janeiro, Chemical and Biochemical Process Engineering – EQ, Rio de Janeiro, RJ, Brazil
[b] Universidade Federal do Rio de Janeiro, Chemical Engineering Program – COPPE, Rio de Janeiro, RJ, Brazil
[c] Imperial College London, Sargent Centre for Process Systems Engineering, London, United Kingdom
* Corresponding Author: joseneto@eq.ufrj.br.

ABSTRACT

This work presents a Lyapunov-based framework for safe reinforcement learning (RL) applied to the control of an unstable reactor. The proposed method imposes stability constraints on the value and Q-functions through a Lyapunov candidate function defined as the negative of these functions, $L(s)=-V(s)$ and $L(s,a)=-Q(s,a)$. Constraints enforce positivity of the Lyapunov candidate function and non-positive time derivatives, promoting monotonic behavior aligned with Lyapunov stability conditions. The framework was tested on both on-policy (PPO) and off-policy (SAC, TD3, and DDPG) RL algorithms, with performance evaluated against their baseline versions and a nonlinear Model Predictive Controller (NMPC). Results showed that stability constraints significantly improved control performance across all tested algorithms, yielding consistently higher cumulative rewards, reduced overshoot, and decreased variability. Derivative-based constraints successfully mitigated abrupt changes and oscillatory behavior in the Q and value functions, especially evident in PPO, DDPG, and TD3. This work demonstrates that Lyapunov-based constraints are an effective tool for improving the safety and stability of RL algorithms in safety-critical applications. The proposed approach avoids the complexity of auxiliary optimization and offers a computationally efficient solution for enhancing the safety and stability of RL algorithms in safety-critical control applications.

Keywords: Lyapunov functions, process control, safety-critical systems, unstable dynamics

INTRODUCTION

Reinforcement Learning (RL) has shown great potential for solving complex control tasks. However, its application in safety-critical systems, such as unstable reactors, remains limited due to the absence of closed-loop stability guarantees and inherent mechanisms for enforcing hard constraints, unlike traditional Model Predictive Control (MPC) formulations [1]. These limitations pose significant challenges in process control, where stability and constraint satisfaction are of prime importance.

Safe RL aims to address these challenges by incorporating mechanisms that enforce stability and safety constraints during both training and deployment. Unlike conventional control methods, which explicitly encode these properties, RL optimizes policies directly from data, often without structured guarantees of stability, particularly in model-free RL approaches. To overcome these challenges, this work introduces a Lyapunov-based framework to enforce closed-loop stability by directly constraining the value function, without relying on explicit system models.

It is important to distinguish between safety constraint enforcement, which ensures the system operates within predefined safety bounds, and closed-loop stability guarantees, which ensure that the system converges to a stable/stabilized operating point over time. While both aspects are critical for safe control, our approach focuses on the latter by leveraging Lyapunov functions to directly enforce stability during policy learning.

Lyapunov functions provide a mathematical framework for enforcing stability by ensuring that the system's "energy" decreases over time. However, designing such functions for RL controllers is challenging and often requires detailed system-specific knowledge [2]. For instance, Brunke et al. [3] emphasized the importance of

developing RL algorithms that maintain safety during training and deployment, especially in uncertain environments where modelling errors or disturbances can compromise stability. Moreover, Gu et al. [4] highlighted that constructing Lyapunov constraints that remain valid across varying conditions remains an open challenge, particularly in high-dimensional and stochastic settings.

Recent advancements in safe RL have highlighted the potential of integrating stability guarantees directly into the learning process. For example, methods leveraging Lyapunov functions [5] define invariant state domains that ensure stability under neural network control policies. This approach provides a framework for guaranteeing system convergence to an equilibrium point while managing nonlinearities and disturbances. Similarly, Osinenko et al. [6] proposed an RL framework incorporating Lyapunov-like constraints, which utilize a control Lyapunov function (CLF) as a basis to establish semi-global stability guarantees through an online sample-and-hold optimization scheme, demonstrating its efficacy in maintaining closed-loop stability during learning. Another notable advancement involves using control invariant sets (CIS) [7], which streamline training by embedding stability guarantees into the learning process through invariant set constraints.

Despite these advancements, significant gaps remain. Lyapunov-based methods often require identifying suitable Lyapunov functions or barrier functions, which can be computationally expensive and challenging for highly nonlinear systems. Furthermore, these approaches frequently rely on predefined invariant domains or approximations that may lack generalization across diverse operational conditions. CIS-based methods, while efficient in improving sample efficiency and providing robust stability guarantees, require precise state reset mechanisms and careful integration of invariant constraints into both offline and online phases, which may not always translate effectively to complex control environments.

Building on these foundations, this work introduces a Lyapunov-based framework that simplifies the enforcement of closed-loop stability by directly constraining the value function. Unlike existing methods that rely on auxiliary cost objectives, this approach integrates stability constraints into the reinforcement learning process, reducing computational overhead and improving scalability.

Conventional RL implementations typically fall into two main categories: on-policy and off-policy algorithms. On-policy methods, such as Proximal Policy Optimization (PPO), update the policy using data sampled sequentially from the current policy, allowing for consistent tracking of changes in the value function over time. This sequential sampling simplifies the computation of time derivatives, making on-policy algorithms more stable under policy updates. However, these methods require significant amounts of data due to the need for fresh samples in each update cycle. Conversely, off-policy methods, such as Soft Actor-Critic (SAC), Twin Delayed Deep Deterministic Policy Gradient (TD3), and Deep Deterministic Policy Gradient (DDPG), use experience replay buffers that enable random sampling of past experiences. This improves sample efficiency but complicates the estimation of time derivatives due to the lack of temporal ordering in the samples, making the learning process more prone to instability.

This work proposes to enforce closed-loop stability across on-policy and off-policy RL algorithms using Lyapunov-based constraints. Specifically, the Lyapunov candidate functions were defined explicitly as the negatives of the value (on-policy) and Q-functions (off-policy), $L(s) = -V(s)$ and $L(s,a) = -Q(s,a)$. Stability is enforced by penalizing positive values of the original functions, thus ensuring $V(s), Q(s,a) < 0$. Additionally, penalties on time derivatives encourage monotonic increases in these original functions, which translates directly to monotonic decreases in the Lyapunov functions, satisfying the Lyapunov stability condition ($dL/dt \leq 0$). These constraints promote that the resulting learned policies consistently meet stability requirements. The main contributions of this work are as follows:

- A unified Lyapunov-based stability constraint framework applicable to both on-policy and off-policy RL algorithms.
- A comparative analysis of time derivative-based and positivity constraints and their impact on training stability.
- An evaluation of the proposed approach to control an unstable reactor, demonstrating improved safety and performance compared to baseline methods.

The remainder of this paper is structured as follows: Section 2 presents the framework and implementation details. Section 3 provides experimental results and discussion. Section 4 concludes with a summary and future research directions.

METHODOLOGY

Framework and Environment Design

This work utilizes PC-Gym [8], an open-source framework for developing RL environments in chemical process control systems, for implementing the environment. PC-Gym provides robust tools for simulating process control problems, including customizable constraints, disturbance generation, and reward function design. It offers compatibility with RL algorithms like PPO, SAC, DDPG, and TD3, enabling standardized

benchmarking for process systems engineering problems.

The environment was constructed to simulate the dynamics of a Continuous Stirred Tank Reactor (CSTR), a nonlinear system often used as a benchmark in process systems engineering due to its complex dynamics and control challenges. The CSTR involves an irreversible reaction that converts species A into species B, with cooling provided by water circulating through a coiled tube. Two coupled nonlinear ordinary differential equations govern the dynamics:

$$\frac{dC_A}{dt} = \frac{q}{V}(C_{A,f} - C_A) - kC_A e^{\frac{-E_A}{RT}} \quad (1)$$

$$\frac{dT}{dt} = \frac{q}{V}(T_f - T) - \frac{\Delta H_R}{\rho C_p} kC_A e^{\frac{-E_A}{RT}} + \frac{UA}{\rho C_p V}(T_c - T) \quad (2)$$

Here, C_A and T represent the concentration of reactant A and reactor temperature, respectively, while T_c is the cooling water temperature, serving as the control input. Additional model parameters include feed concentration ($C_{A,f}$), feed temperature (T_f), volumetric flow rate (q), reactor volume (V), reaction rate constant (k), activation energy (E_A), universal gas constant (R), reaction enthalpy (ΔH_R), fluid density (ρ), heat capacity (C_p), and heat transfer coefficient-area product (UA). The nonlinearities, particularly the exponential dependence of reaction rate on temperature, pose significant control challenges, including potential instability and multiple steady states. Simulation parameters and constraints were designed to realistically represent process control challenges:

Action Space: The manipulated variable T_c was defined as a continuous range between 250 K and 350 K, normalized to [0, 1] during training.

Observation Space: The state variables C_A, T, and setpoint temperature T_{SP} were normalized to [0,1] based on their physical bounds ([0.0, 200 K, 300 K] to [1.0, 600 K, 400 K]). Additionally, Gaussian measurement noise proportional to 0.1% of each observed state value was introduced during training.

Reward Function: Negative squared error between normalized T and T_{SP}.

Dynamic Setpoint Strategy: The setpoint dynamically transitioned midway through the episode to evaluate the RL algorithms' adaptability.

Disturbance Generation: Time-indexed disturbances in feed concentration ($C_{A,f}$) and inlet temperature (T_f) were dynamically generated to emulate real-world variability. These disturbances were sampled from predefined bounds and applied over multiple episodes to test robustness.

The implementation includes a steady-state solver to initialize the system state variables, ensuring valid initial conditions under varying disturbances. The reward function was designed so that the difference between the normalized reactor temperature and the setpoint temperature was squared and scaled, ensuring that the reward signal directly incentivized accurate tracking of the setpoint while maintaining interpretability.

The setpoint strategy was implemented dynamically, splitting the episode into two phases. During the first half of each episode, the setpoint was initialized to the steady-state temperature derived from the system's initial conditions. In the second half, the setpoint transitioned to a fixed value (e.g., 340 K). This approach tests the RL agent's ability to adapt to changing operational goals while maintaining system stability.

Safe RL Implementation

The proposed stability enforcement mechanism is inspired by Lyapunov principles, focusing explicitly on the positive definiteness and monotonic decrease conditions required for stability. Given our reward function defined as the negative squared difference between the current state and desired setpoint, we define Lyapunov candidate functions as L(s) = −V(s) and L(s,a) = −Q(s,a) to maintain a consistent formulation where L > 0 and dL/dt ≤ 0 indicate stable behavior. The stability constraints, consistently applied across PPO, SAC, DDPG, and TD3 algorithms, are:

Positive Definiteness Constraint: Penalizing positive values of the original value/Q-functions (V, Q) to promote them remains negative, thereby enforcing L > 0.

Time Derivative Constraint: Penalizing decreases in the original value/Q-functions between consecutive time steps, ensuring their monotonic increase. This promotes a non-positive time derivative (dL/dt ≤ 0), satisfying the Lyapunov stability criteria.

These constraints were applied dynamically during training, with penalties weighted by hyperparameters λ_{pos} (positive definiteness) and λ_{der} (time derivative constraint) —additionally, the algorithms logged mean values and time derivatives of value/Q functions for interpretability and transparency.

Safe RL Algorithm Implementations

1. DDPG and TD3:

These off-policy methods relied on replay buffers for experience sampling. The critic loss incorporated stability constraints as follows:

$$\text{critic\_loss} = \text{MSE\_loss}(\text{curr\_Q}, \text{target\_Q}) + \lambda_{pos} \cdot \max(0, \text{curr\_Q}) + \lambda_{der} \cdot \max(0, \text{prev\_Q} - \text{curr\_Q}) \quad (3)$$

where λ_{pos} and λ_{der} are penalty weights. Here:

curr_Q: The instantaneous Q-values estimated by the critic for the current batch of samples.

prev_Q: The Q-values for the same batch are computed during the previous gradient step in the current training iteration.

TD3 further used twin critics and applied constraints

to the minimum of the critics' Q-values for robustness. Noise models for exploration were algorithm-specific:

DDPG: Ornstein-Uhlenbeck noise was used, introducing temporal correlations in action exploration.

TD3: Normal noise was employed along with target policy noise and noise clipping for stable updates.

2. SAC:

The entropy-regularized objective was preserved in the SAC implementation to balance exploration and exploitation while incorporating stability constraints into the Q-function updates. The critic loss was computed as:

$$\text{critic\_loss} = 0.5 \sum_i \text{MSE\_loss}(\text{curr\_Q}_i, \text{target\_Q}) + \lambda_{pos} \cdot \max(0, \text{curr\_Q}) + \lambda_{der} \cdot \max(0, \text{prev\_Q} - \text{curr\_Q}) \quad (4)$$

where i indexes over multiple critics.

The actor loss was computed using the minimum Q-value across critics and included the entropy regularization term:

$$\text{actor\_loss} = \mathbb{E}[\alpha \ \log \pi(a|s) - \text{minQ(s, a)}] \quad (5)$$

where α is the entropy coefficient, and $\pi(a|s)$) is the stochastic policy. This promotes safe exploration while optimizing the policy.

3. PPO:

PPO, an on-policy method, inherently stabilizes training using a clipped surrogate objective. Additional constraints on the value function $V(s)$ were introduced:

$$\text{loss} = \text{policy\_loss} + \beta_{vf} \cdot \text{value\_loss} + \lambda_{ent} \cdot \text{entropy\_loss} + \lambda_{pos} \cdot \max(0, -\text{V(s)}) + \lambda_{der} \cdot \max(0, \text{V(s)} - \text{prev\_V}) \quad (6)$$

where β_{vf} and λ_{ent} are coefficients for the value and entropy losses. Value functions were dynamically monitored for compliance with these constraints.

Hyperparameter Optimization

This work employed Optuna [9], a hyperparameter optimization framework, to tune both common and algorithm-specific hyperparameters across all reinforcement learning (RL) algorithms. Optuna utilizes the Tree-structured Parzen Estimator (TPE) as its default optimization algorithm, enabling efficient exploration of the hyperparameter space.

A unified approach was used for tuning common hyperparameters across all algorithms, including:

- Learning rate (α),
- Discount factor (γ),
- Replay buffer size (off-policy only),
- Batch size,
- Start step for training,
- Target network update factor (τ).

Algorithm-specific optimizations included:

DDPG: Ornstein-Uhlenbeck action noise parameters (μ, σ, θ, dt).

TD3: Target policy noise parameters (σ, noise clip).

PPO: Epoch count, Generalized Advantage Estimation (GAE) parameters (λ), clip range, and entropy coefficient.

SAC: Entropy coefficient and seed for reproducibility.

For all Safe RL implementations, additional stability-specific hyperparameters were tuned:

- λ_{pos}: Weight for the positive definiteness penalty.
- λ_{der}: Weight for the time derivative penalty.

Evaluation

To establish a performance benchmark, the baseline RL cases and a nonlinear Model Predictive Controller (NMPC) were implemented [8]. This controller leveraged full knowledge of the system dynamics and perfect state estimation to solve the optimal control problem at each time step. The NMPC formulation incorporated features absent in the RL-based controllers, such as state tracking and normalized control effort terms for consistent scaling across variables. Additionally, the system dynamics were discretized using orthogonal collocation to achieve high accuracy. This oracle served as the performance ceiling, representing near-optimal control strategies under perfect conditions, allowing us to quantitatively assess the efficacy of the RL algorithms.

Each RL policy (PPO, SAC, DDPG, and TD3) was evaluated across 20 episodes, each initialized with identical initial conditions and setpoint profiles that differed from those encountered during training. The primary variation among these episodes was introduced through progressively increasing levels of state measurement noise, linearly scaled from 0.0% to 0.5%. Performance metrics, including cumulative rewards and control effort, were collected to assess each policy's robustness to measurement noise. The collected data were aggregated to compute median cumulative rewards and mean value/Q functions. Since the reward was defined as the negative squared difference between the controlled states and setpoints, it offers a meaningful comparison to the NMPC oracle. Additionally, the evaluation tracked the evolution of the value/Q functions and their time derivatives for each policy. Reward distributions were visualized using histograms computed from cumulative rewards across evaluation episodes.

RESULTS AND DISCUSSION

Figures 1 and 2 present the median performance of RL algorithms (SAC, PPO, DDPG, and TD3) compared to the NMPC oracle across 20 unseen evaluation episodes.

Figure 1 corresponds to the algorithms without stability constraints, while Figure 2 shows their performance under stability constraints.

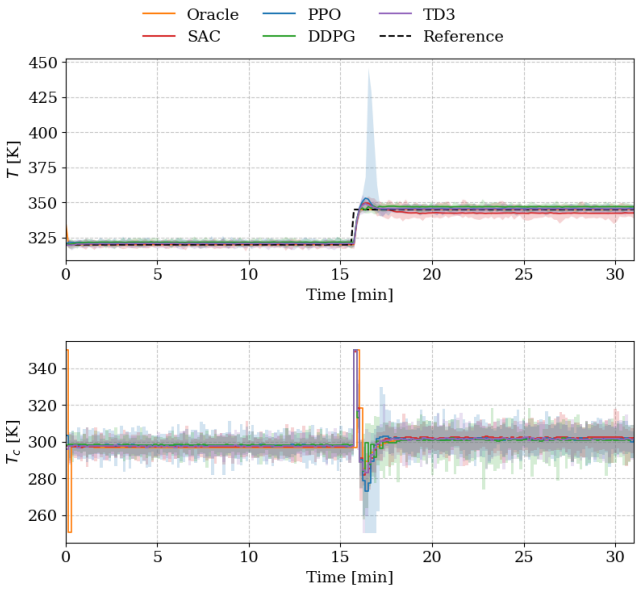

Figure 1. Performance of RL algorithms without stability constraints compared to the NMPC oracle (shaded areas: min-max, solid lines: median). Plots show (top) reactor temperature (T) and (bottom) control input T_c.

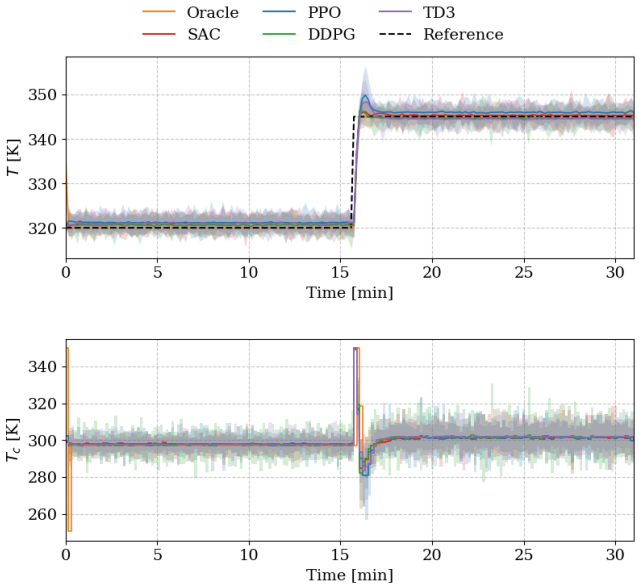

Figure 2. Performance of RL algorithms with stability constraints compared to the NMPC oracle (shaded areas: min-max, solid lines: median). Plots show (top) reactor temperature (T) and (bottom) control input T_c.

Under the baseline scenario (Figure 1), PPO and SAC exhibit significant overshoot and oscillations in reactor temperature following the setpoint change at 15 min, with PPO showing the largest transient deviation, indicating less stability. Coolant temperature (T_c) shows significant fluctuations, particularly for PPO.

In contrast, the safe RL scenario (Figure 2) demonstrates improved stability across all RL algorithms tested. Overshoot and variability are significantly reduced, while manipulated variable responses (T_c) also present smaller transient fluctuations, especially PPO.

The reward distribution (Figure 3) highlights the performance differences between safe and baseline RL algorithms. The safe RL strategies (SAC, PPO, and DDPG) generally achieve cumulative rewards that are more concentrated and shifted towards higher values compared to their baseline counterparts. Baseline methods such as SAC and DDPG display wider distributions with lower reward values, indicating greater variability and less consistent performance.

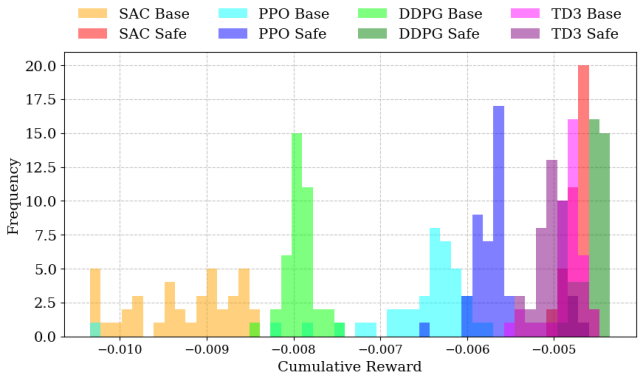

Figure 3. Distribution of cumulative rewards comparing baseline and safe RL strategies (SAC, PPO, DDPG, and TD3).

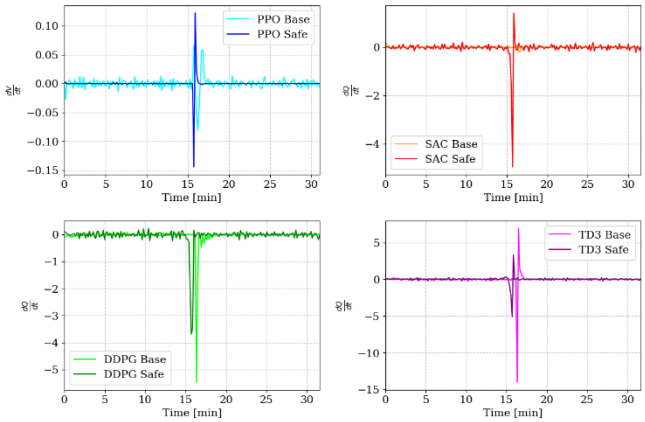

Figure 4. Comparison of time derivatives of Q-function (SAC, DDPG, TD3) or value function (PPO) for baseline and safe RL implementations.

Figure 4 shows the time derivatives of the Q/value functions for SAC, TD3, DDPG, and PPO, comparing baseline and safe implementations. Standard deviations are omitted for clarity. Safe PPO exhibits the most significant reduction in fluctuations, with near-zero oscillations. Safe implementations for DDPG and TD3 demonstrate reduced spikes, indicating improved stability.

However, safe PPO and SAC display larger transition spikes, likely due to value/Q function penalization on the setpoint change.

The spikes during setpoint transitions could be softened by "freezing" or temporarily relaxing constraint penalties during these events. Increasing training episodes or fine-tuning constraints may also further enhance stability across all algorithms.

CONCLUSION

This work introduced a Lyapunov-based framework for stabilizing RL algorithms by directly constraining value and Q-functions. The proposed approach significantly improved the control performance of an unstable reactor when compared to baseline methods. Results demonstrated consistently higher cumulative rewards across all tested algorithms, indicating a reduction in setpoint tracking errors. Furthermore, safe RL substantially reduced overshoot and control variability, especially in SAC, PPO, and DDPG.

Time derivative-based penalties were critical for achieving monotonic decreases in the defined Lyapunov functions (L=−V and L=−Q), thereby promoting Lyapunov stability criteria. This improvement was particularly evident in DDPG and TD3, with reduced transient spikes. In contrast, PPO showed the lowest overall oscillations but still displayed higher transient spikes, which could be due to the value function penalization on the setpoint change.

The results demonstrate that the proposed Lyapunov-based framework effectively enhances stability in RL-based control of unstable systems. Future work should explore dynamic penalty tuning, mechanisms to reduce transient spikes during disturbances (e.g., freezing penalties), and application to more complex environments to broaden its applicability.

ACKNOWLEDGEMENTS

Jose Rodrigues Torraca Neto reports financial support provided by Coordination for the Improvement of Higher Education Personnel (CAPES), Finance Code 001. Argimiro R. Secchi reports a relationship with CNPq that includes funding grants (303587/2020-2). Other 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.

REFERENCES

1. Badgwell TA, Lee JH, Liu KH. Reinforcement learning – overview of recent progress and implications for process control. In: 13th International Symposium on Process Systems Engineering (PSE 2018). Eds: Eden MR, Ierapetritou MG, Towler GP. Vol 44, pp 71–85. Elsevier (2018) https://doi.org/10.1016/B978-0-444-64241-7.50008-2
2. Annaswamy AM. Adaptive control and intersections with reinforcement learning. Annu Rev Control Robot Autom Syst 6:65–93 (2023) https://doi.org/10.1146/annurev-control-062922-090153
3. Brunke L, Greeff M, Hall AW, Yuan Z, Zhou S, Panerati J, Schoellig AP. Safe learning in robotics: from learning-based control to safe reinforcement learning. Annu Rev Control Robot Autom Syst 5:411–444 (2022) https://doi.org/10.1146/annurev-control-042920-020211
4. Gu S, Yang L, Du Y, Chen G, Walter F, Wang J, Knoll A. A review of safe reinforcement learning: methods, theories, and applications. IEEE Trans Pattern Anal Mach Intell 46:11216-11235 (2024) https://doi.org/10.1109/TPAMI.2024.3457538
5. Gu S, Kumar R. Robust optimal safe and stability guaranteeing reinforcement learning control for quadcopter. arXiv (2024) https://arxiv.org/abs/2412.14003
6. Osinenko P, Beckenbach L, Göhrt T, Streif S. A reinforcement learning method with closed-loop stability guarantee. IFAC-PapersOnLine 53(2):8043–8048 (2020) https://doi.org/10.1016/j.ifacol.2020.12.2237
7. Bo S, Agyeman BT, Yin X, Liu J. Control invariant set enhanced safe reinforcement learning: improved sampling efficiency, guaranteed stability and robustness. arXiv (2023) https://arxiv.org/abs/2305.15602
8. Bloor M, Torraca J, Sandoval IO, Ahmed A, White M, Mercangöz M, Tsay C, Del Rio Chanona EA, Mowbray M. PC-Gym: benchmark environments for process control problems. arXiv (2024) https://arxiv.org/abs/2410.22093
9. Akiba T, Sano S, Yanase T, Ohta T, Koyama M. Optuna: a next-generation hyperparameter optimization framework. arXiv (2019) https://arxiv.org/abs/1907.10902