Neurocomputing 638 (2025) 130041

Contents lists available at ScienceDirect

Neurocomputing

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

NLBAC: A neural ODE-based algorithm for state-wise stable and safe reinforcement learning

Liqun Zhao [a,*], Keyan Miao [a], Hongpeng Cao [b], Konstantinos Gatsis [c], Antonis Papachristodoulou [a]

[a] Department of Engineering Science, University of Oxford, Oxford, OX1 3PJ, United Kingdom
[b] School of Engineering and Design, Technical University of Munich, Munich, 85748, Germany
[c] School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, United Kingdom

ARTICLE INFO

Communicated by J. Xu

Keywords:
Reinforcement learning
State constraints
Control barrier function
Neural certificates
Policy iteration
Neural ordinary differential equations
Augmented lagrangian method

ABSTRACT

Ensuring safety and stability is critical when using reinforcement learning (RL) to control safety-critical systems. However, model-free RL algorithms usually suffer from low sample efficiency, and employing widely-used methods like dual ascent to solve constrained RL problems may be challenging due to their sensitivity to hyperparameters. To address these difficulties, in this work, we first propose an augmented Lagrangian-based method to maintain safety and stability through state-wise control Lyapunov function (CLF) and pre-defined control barrier function (CBFs) constraints in non-constrained Markov decision process (non-CMDP) settings. To handle tasks without pre-defined CBFs, we extend this method by training a barrier certificate jointly with the control policy, supported by theoretical guarantees to ensure monotonically improved control performance. Moreover, we investigate the issue of infeasibility arising from the presence of multiple state-wise constraints. A practical algorithm, Neural ordinary differential equations-based Lyapunov-Barrier Actor-Critic (NLBAC), is further designed by integrating the proposed method with the Soft Actor-Critic (SAC) and leveraging neural ordinary differential equations (NODEs) for system modeling. Comparisons with baselines and ablation experiments demonstrate that our algorithm achieves superior performance in terms of safety and driving the system towards the desired state with higher sample efficiency.

1. Introduction

Random explorations, inherent in most reinforcement learning (RL) algorithms, may lead to actions with catastrophic consequences. Thus, ensuring safety in RL has become a focal point in recent research [1,2]. Many existing safe RL approaches use constrained Markov decision process (CMDP) settings [3–5], and various techniques have been proposed, including trust region policy optimization [6,7], and primal–dual update [8–10]. However, these approaches enforce safety constraints as cumulative costs along trajectories over time, while many real-world applications require enforcing safety at each individual state [11]. State-wise safety is particularly important in safety-critical applications such as autonomous driving and robotics, where even a single violation can result in irreversible consequences [12]. For example, in autonomous driving tasks, an RL-controlled self-driving car may prioritize maximizing the reward over ensuring safety by driving at high speeds, and without efficient safety constraints, this can greatly increase the risk of collisions. Importantly, unlike the cumulative safety constraints used in CMDPs, which allow temporary violations as long as the cumulative safety cost remains within a threshold, collision avoidance must be enforced at every time step to ensure safe operation.

Other examples include robotic manipulation, where enforcing state-wise constraints is essential for preventing damage to hardware, ensuring human safety, and maintaining operational constraints. A robot handling fragile objects must release them only when placed on a stable surface; otherwise, improper release may cause irreversible damage [11]. Additionally, in drone navigation, preventing unsafe conditions such as sudden altitude drops necessitates the strict enforcement of instantaneous safety constraints.

Despite the necessity of enforcing safety constraints at every individual state, many existing constrained RL methods rely on the dual ascent update, which is primarily applied to ensure constraint satisfaction for CMDPs. Furthermore, this approach may require sophisticated hyperparameter tuning and a long training process to obtain satisfactory results, mainly because of its sensitivity to hyperparameters like

* Corresponding author.
E-mail addresses: liqun.zhao@eng.ox.ac.uk (L. Zhao), keyan.miao@eng.ox.ac.uk (K. Miao), cao.hongpeng@tum.de (H. Cao), k.gatsis@soton.ac.uk (K. Gatsis), antonis@eng.ox.ac.uk (A. Papachristodoulou).

https://doi.org/10.1016/j.neucom.2025.130041
Received 10 September 2024; Received in revised form 7 February 2025; Accepted 15 March 2025
Available online 26 March 2025

learning rates [13]. This may result in many safety violations during learning.

An increasing focus has emerged on integrating control approaches with RL to help ensure safety [14]. Concepts like safe set algorithm (SSA) [15], constraint decay function (CDF) [16], Hamilton–Jacobi–Bellman(HJB) equation [17], model predictive controller [18], and control barrier function (CBF) [19–25] have been employed to enable safety in RL. However, in many previous studies [19,20,26], a physics-based control-affine nominal model of the system is required. Also, requiring the pre-defined CBF to be affine with respect to the state [19], or using the continuous version of the CBF despite computing control actions in discrete time [20,23], could potentially limit the applicability of these algorithms.

Previous studies employing neural barrier certificates [27–30] have demonstrated promising results in reducing safety violations, and such methods are effective in tasks involving adversarial attacks, which require a robust training process [31], as well as in tasks demanding probabilistic safety guarantees due to model uncertainties [32]. However, many of them require system models, precise labeling of safe and unsafe states, and the condition that time derivatives should be negative to learn neural barrier certificates, while safety labels are usually given by human demonstrations [27,28] that can be rough in practice, or by certain functions [33] with additional computation for a separate safe policy and relevant conditions. In [32], extra computation is required for distributional RL, while in [29], the barrier function condition is defined based on the expected value, and thus some unsafe states can still be visited.

In addition, maintaining stability is significant in many tasks, and the Lyapunov framework has found application in learning [34,35]. In [36,37], Lyapunov functions are used in model-free RL to help guarantee stability, though facing the potential problem of having relatively low sample efficiency. Taken together, ensuring both state-wise safety and stability with RL-based controllers is essential for achieving reliable performance in safety-critical systems. Most existing approaches in constrained RL focus on either safety or stability in isolation. However, purely safety-focused RL methods can lead to overly conservative policies [2], resulting in lower rewards, extended trajectories, and potentially suboptimal task completion. Conversely, stability-focused RL methods risk violating safety constraints, which can have severe consequences in safety-critical applications. A long-standing challenge in combining both aspects is the issue of infeasibility caused by imposing both safety and stability constraints simultaneously [38], and addressing this challenge efficiently is crucial for enabling more reliable RL applications across a wide range of tasks.

Sample efficiency is another critical aspect that needs to be considered in RL. Conventional model-free RL algorithms often exhibit lower sample efficiency compared to model-based approaches. Moreover, reliable models of systems can offer valuable guidance for maintaining both safety and stability in controlled agents. However, when employing neural networks to directly approximate the transition, there is an implicit assumption that all data is collected with the same discretization step. Consequently, all predictions can only be made with that discretization step as well. Inspired by the fact that many physical systems are modeled by differential equations, neural ordinary differential equations (NODEs) [39] can be leveraged to approximate the real dynamics [40]. However, these earlier studies integrating NODEs with RL did not take safety and stability of the controlled system into consideration, and the tendency for approximation errors to propagate remains.

Some previous studies [41,42] investigate state-wise constraints by adapting common CMDPs to special subclasses of CMDPs. However, these methods are still developed within a CMDP framework, limiting constraints to be formulated solely based on the instantaneous cost of the state–action–transition tuple. To address the limitations of existing methods, this work departs from CMDP-based RL approaches by directly incorporating state-wise constraints like discrete-time CBFs with high relative degree, which requires multiple future states, and neural Lyapunov and barrier certificates with theoretical guarantees in non-CMDP settings. Compared to other CBF-based methods that often rely on specific assumptions about the constraints' form, such as linearity for quadratic programming solvers [19,20], our method accommodates tasks with diverse dynamics and pre-defined CBFs without such restrictions. Importantly, the infeasibility issue that arises from applying multiple state-wise constraints simultaneously is explicitly investigated in this work. The schematic of the proposed algorithm is shown in Fig. 1. Detailed contributions made in this work are:

1. With control Lyapunov function (CLF) and pre-defined CBFs, we propose a method to achieve state-wise stable and safe RL in non-CMDP settings by using the augmented Lagrangian method (ALM). Also, we investigate the infeasible states from which the controlled system's approach towards the desired state, driven by CLF, will cause immediate safety violations due to the task setting, and then design an additional RL-based backup controller to achieve and maintain safety while encouraging exploration at such infeasible states.
2. For tasks without pre-defined CBFs, we first prove that the cumulative discounted value of a specifically defined signal is a barrier certificate. We then propose a policy update method whose constraint is such a barrier certificate which is easy to train in a principled way without requiring additional expert demonstrations. We further theoretically prove that the set of states where safety is maintained monotonically expands if the control policy is updated according to the proposed method because of the barrier certificate, and show the monotonic improvement of the control performance in terms of the cumulative discounted reward, which will finally converge to an optimal solution under certain assumptions. Moreover, the infeasibility caused by the addition of CLF is also studied, and we show that the controlled system will learn to avoid those infeasible states as training proceeds.
3. We provide a practical implementation called Neural ordinary differential equations-based Lyapunov-Barrier Actor-Critic (NLBAC) by combining the proposed method with Soft Actor-Critic (SAC) [43] where policy, value, Lyapunov and barrier (if no pre-defined CBFs are available) functions are approximated by multi-layer neural networks. This algorithm alleviates the problem of large error propagation that is challenging for model-based RL by applying NODEs for modeling and using a shorter prediction horizon compared to those in many previous model-based RL studies. We further evaluate our algorithm in four widely-used tasks. The results indicate that NLBAC achieves satisfactory performance in reducing safety violations and driving the controlled system towards the destination with higher sample efficiency. Ablation studies show better performance of the ALM for non-CMDP constrained RL problems compared to the widely-used dual ascent update.

Compared to our previous work [26], this paper removes the need for a nominal model, and we make changes to the update of Lagrangian multipliers to avoid extra hyperparameter tuning. The QP-based backup controller in [26] is replaced with an RL-based backup controller when pre-defined CBFs are used, making the algorithm directly applicable to tasks with nonconvex constraints. Also, different from [26] and our workshop paper [44], an integration with neural barrier certificates is provided here, together with theoretical analysis. We also present comparisons to more baselines in more tasks, and ablation studies comparing ALM to the dual ascent update are also included.

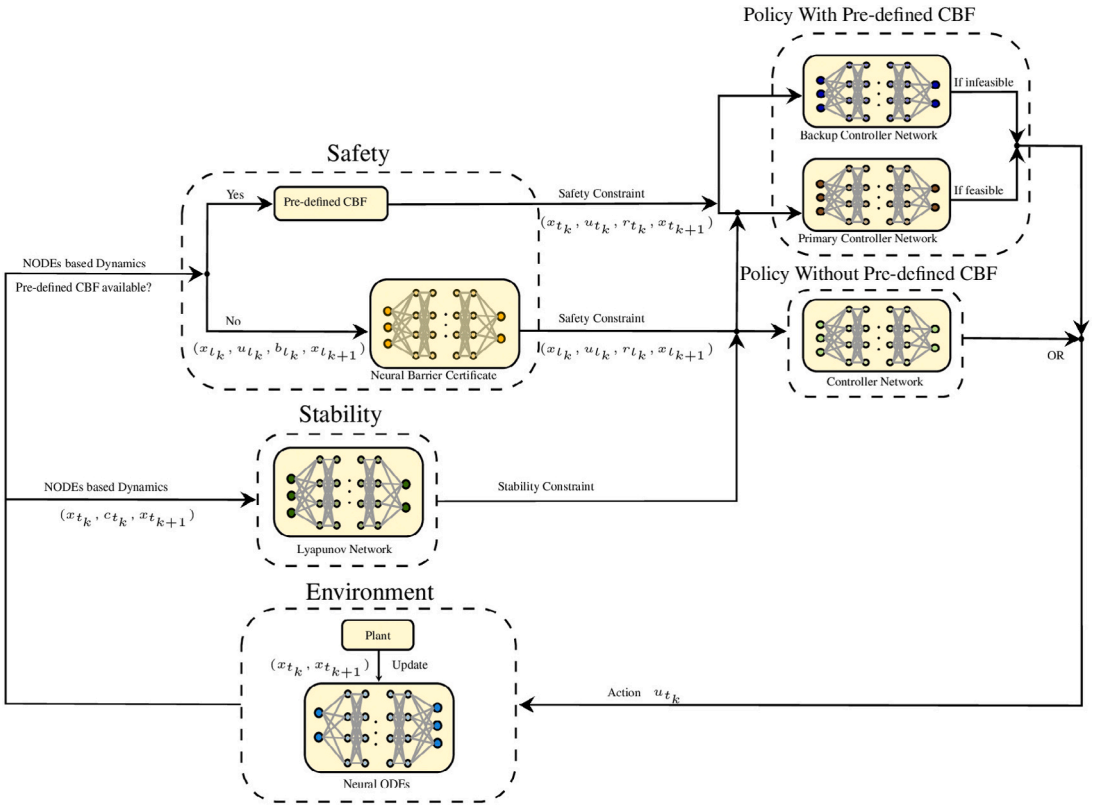


Fig. 1. Schematic of the proposed NLBAC algorithm, which includes a Lyapunov network for stability, pre-defined CBFs or neural barrier certificates for safety, and neural ODEs for modeling. The algorithm operates as follows: If pre-defined CBF is available, both safety and stability constraints are applied during the training of the primary controller network, while the backup controller network is trained with only the safety constraint to handle the issue of infeasibility. If no pre-defined CBF is available, the safety constraint is imposed by using a neural barrier certificate, and the policy without pre-defined CBF is trained together with the stability constraint. The output junction denotes that action signals are computed by one of the two policies, depending on the availability of pre-defined CBF.

2. Preliminaries and problem statement

2.1. Preliminaries

2.1.1. Markov decision process

Consider a Markov decision process (MDP) defined by the tuple $\mathcal{M} = (\mathcal{X}, \mathcal{U}, \mathcal{T}, r, \gamma)$. $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{U} \subset \mathbb{R}^m$ are state and control input spaces which are both bounded, and $x_{t_k} \in \mathcal{X}$ is the state at time t_k, $u_{t_k} \in \mathcal{U}$ is the control signal at time t_k. The reward is denoted as r, and γ is the discount factor. $\mathcal{T}$ denotes the dynamics. For a system with dynamics

$$\dot{x}_t = f(x_t, u_t), \tag{1}$$

by fixing the control signal to u_{t_k} between consecutive times t_k and t_{k+1}, we have:

$$x_{t_{k+1}} = x_{t_k} + \int_{t_k}^{t_{k+1}} f(x_\chi, u_{t_k}) d\chi, \tag{2}$$

following (1).

2.1.2. Safety

Here we first define the following subspaces [45].

- $\mathcal{X}_{\text{vio}}$: Set of states where safety violation occurs.
- $\mathcal{X}_{\text{irrec}}$: Set of states $x \notin \mathcal{X}_{\text{vio}}$, but under any controller, the trajectory starting from $x_{t_0} = x$ will enter $\mathcal{X}_{\text{vio}}$ at least once with a positive probability, i.e., $\forall \pi \in \Pi$, $\exists k > 0$, such that $Pr(x_{t_k} \in \mathcal{X}_{\text{vio}}) > 0$, where Π denotes the set of all policies.
- $\mathcal{X}_{\text{unsafe}} = \mathcal{X}_{\text{vio}} \cup \mathcal{X}_{\text{irrec}}$ is the unsafe set, and $\mathcal{X}_{\text{safe}} = \mathcal{X} \backslash \mathcal{X}_{\text{unsafe}}$ is safe set.

Remark 1. For any $x_{t_0} = x \in \mathcal{X}_{\text{safe}}$, there exists controller π, such that $\forall k \geq 0$, $Pr(x_{t_k} \in \mathcal{X}_{\text{vio}}) = 0$, which means any trajectory starting from $x \in \mathcal{X}_{\text{safe}}$ will enter $\mathcal{X}_{\text{vio}}$ and cause safety violations with probability 0, and then it is said that safety is maintained for state x under such control policy π. Furthermore, a policy that maintains safety for all $x \in \mathcal{X}_{\text{safe}}$ is denoted as $\pi^s \in \Pi^s$, where Π^s is the set of controllers under which $\forall k \geq 0$, $Pr(x_{t_k} \in \mathcal{X}_{\text{vio}}) = 0$ for any trajectory starting from any $x_{t_0} \in \mathcal{X}_{\text{safe}}$.

With m distinct state-wise safety constraints, if given pre-defined functions $h_i, i \in [1, \dots, m]$ whose super-level sets

$$C_{i,0} = \{x_{t_k} \in \mathcal{X} | h_i(x_{t_k}) \geq 0\} \tag{3}$$

satisfy $\bigcap_{i=1}^m C_{i,0} \subset \mathcal{X}_{\text{safe}}$, the system is then safe if $\bigcap_{i=1}^m C_{i,0}$ is forward invariant. Since RL generates the control signal discretely, here discrete-time CBFs [46] are used and a list of functions can be defined:

$$\begin{aligned}
&\Phi_{i,0}(x_{t_k}) := h_i(x_{t_k}) \\
&\Phi_{i,1}(x_{t_k}) := \Delta\Phi_{i,0}(x_{t_k}, u_{t_k}) + \kappa_{i,1}(\Phi_{i,0}(x_{t_k})) \\
&\quad\vdots \\
&\Phi_{i,r_i}(x_{t_k}) := \Delta\Phi_{i,r_i-1}(x_{t_k}, u_{t_k}) + \kappa_{i,r_i}(\Phi_{i,r_i-1}(x_{t_k}))
\end{aligned} \tag{4}$$

where r_i is the relative degree [47,48], $\Delta\Phi_{i,j}(x_{t_k}, u_{t_k}) := \Phi_{i,j}(x_{t_{k+1}}) - \Phi_{i,j}(x_{t_k})$, $j = 0, 1, \dots, r_i - 1$, and $\kappa_{i,j}(\cdot)$ are class $\mathcal{K}$ functions satisfying $\kappa_{i,j}(x) < x$, where $j = 1, \dots, r_i$, for the ith safety constraints.

Definition 1 (*Discrete-time Control Barrier Function [46]*)**.** For the system (2), the pre-defined function $h_i : \mathbb{R}^n \to \mathbb{R}$ is called a discrete-time CBF of relative degree r_i if there exists $\Phi_{i,j}(x_{t_k})$, $j = 0, 1, \dots, r_i$ defined by (4) and $C_{i,j}$ which are super-level sets of $\Phi_{i,j}(x_{t_k})$ defined similarly to

(3), $j = 0, 1, \dots, r_i - 1$ such that for all $x_{t_k} \in \bigcap_{j=0}^{r_i-1} C_{i,j}$,

$$\boldsymbol{\Phi}_{i,r_i}(x_{t_k}) \geq 0. \tag{5}$$

Given the existence of a CBF, a controller satisfying (5) will make the set $\bigcap_{j=0}^{r-1} C_{i,j}$ forward invariant. Consequently, safety is maintained if there exists a controller such that $\forall i \in [1, \dots, m]$, (5) holds for all $x_{t_k} \in \bigcap_{i=1}^{m} \bigcap_{j=0}^{r_i-1} C_{i,j}$.

2.1.3. Stability

For stabilization tasks, the system state is required to eventually reach a goal state. As in [26], define an additional cost signal as $c(x_{t_k}, u_{t_k}) = \left\| x_{t_{k+1}} - x_{\text{desired}} \right\|$ where $x_{t_{k+1}}$ is the next state following (2), and x_{desired} is the desired state (goal). Furthermore, define the cost function under the controller π as $c_\pi(x_{t_k}) = \mathbb{E}_{u_{t_k} \sim \pi} c(x_{t_k}, u_{t_k}) = \mathbb{E}_{u_{t_k} \sim \pi}[\left\| x_{t_{k+1}} - x_{\text{desired}} \right\|]$. Based on [36], we provide the following definition.

Definition 2. The system is stable in mean cost if the condition $\lim_{t_k \to \infty} \mathbb{E}_{x_{t_k} \sim \upsilon}\left[c_\pi(x_{t_k})\right] = 0$ holds, where υ is the state distribution at time t_k under π.

Define $L^\pi(x_{t_k}) = \mathbb{E}_{\tau \sim \pi}\left[\sum_{i=0}^{\infty} \gamma_c^i c_\pi(x_{t_{k+i}})\right]$ as the value function of the cost at x_{t_k}, where γ_c is the discount factor, and $\tau = \{x_{t_k}, x_{t_{k+1}}, x_{t_{k+2}}, \dots\}$ is a trajectory under controller π starting from x_{t_k}. [26] shows that the system is stable in mean cost if there exist a positive constant β and a controller π, such that

$$\mathbb{E}_{x_{t_k} \sim \mu_\pi}\left[L^\pi(x_{t_{k+1}}) - L^\pi(x_{t_k})\right] \leq -\beta \mathbb{E}_{x_{t_k} \sim \mu_\pi}\left[L^\pi(x_{t_k})\right]. \tag{6}$$

Here $\mu_\pi(x)$ is the sampling distribution. Such L^π is called exponentially stabilizing CLF, abbreviated CLF hereafter.

2.1.4. Neural ordinary differential equations (NODEs)

In many cases, the precise system dynamics or even a necessary nominal model are unknown. Neural networks are important tools for estimating dynamics, and there has been a rise in considering neural network hidden layers as states [49,50]. Chen et al. [39] expanded on this notion by introducing an ODE to model continuous dynamics $\dot{\hat{x}}_{t_k} = \mathcal{F}_\psi\left(t_k, \hat{x}_{t_k}, u_{t_k}\right)$ where $\hat{x}_{t_0} = x_{t_0}$ and $\mathcal{F}_\psi$ is a neural network with parameter ψ. NODEs allow simple incorporation of prior information (such as control-affine property, where $\mathcal{F}_\psi = f_{\psi_1}(t_k, \hat{x}_{t_k}) + g_{\psi_2}(t_k, \hat{x}_{t_k}) u_{t_k}$). By assuming a constant control signal, i.e., $u_\chi = u_{t_k}, \chi \in [t_k, t_{k+1})$, we have:

$$\hat{x}_{t_{k+1}} = \hat{x}_{t_k} + \int_{t_k}^{t_{k+1}} \mathcal{F}_\psi\left(\chi, \hat{x}_\chi, u_{t_k}\right) d\chi \tag{7}$$

which can be used to estimate (2).

2.2. Definition of the stable and safe control problem

Building upon the previous subsection, here we define:

Problem 1. For a system described by (2), given a desired goal state $x_{\text{desired}} \in \mathcal{X}_{\text{safe}}$, a set $\mathcal{X}_0$ which is the set of x_{t_0} denoting initial states such that $\mathcal{X}_0 \subset \mathcal{X}_{\text{safe}}$, find a controller π producing sequence $\{u_{t_k}\}_{k \geq 0}$ such that the state sequence $\{x_{t_k}\}_{k \geq 0}$ following (2) satisfies:

1. **Safety**: $\forall k \geq 0,\ Pr(x_{t_k} \in \mathcal{X}_{\text{vio}}) = 0$.
2. **Stability:** $\lim_{t_k \to \infty} \mathbb{E}_{x_{t_k} \sim \upsilon}\left[c_\pi(x_{t_k})\right] = 0$.

Remark 2. In this paper, we only consider cases where $\mathcal{X}_{\text{safe}}$ is a connected set and where there exists at least one control policy that enables the system to reach x_{desired} from x_{t_0} without passing through $\mathcal{X}_{\text{vio}}$. We exclude tasks where no such path exists, particularly when x_{t_0} and x_{desired} belong to different parts of $\mathcal{X}_{\text{safe}}$ that are completely separated by $\mathcal{X}_{\text{vio}}$.

3. Training with pre-defined control barrier functions

3.1. Controller update via ALM

When pre-defined CBFs are available, to achieve state-wise safety and stability (approaching the desired goal state at every timestep), a more stringent state-wise version of CBF constraints (5) and CLF constraint (6) is proposed:

$$\begin{aligned} &-\boldsymbol{\Phi}_{i,r_i}(x_{t_k}) \leq 0, && \forall x_{t_k} \in \mathcal{X}, \forall i \in [1, \dots, m], \\ &L^{\pi_p}(x_{t_{k+1}}) - L^{\pi_p}(x_{t_k}) + \beta L^{\pi_p}(x_{t_k}) \leq 0, && \forall x_{t_k} \in \mathcal{X}, \end{aligned} \tag{8}$$

where $\{x_{t_{k+1}}, x_{t_{k+2}}, \dots, x_{t_{k+r_i}}\}$ are future states following a primary controller π_p according to (2), and thus (8) are constraints that π_p should satisfy. It is noteworthy that the CBF constraints are also imposed when $x_{t_k} \in \mathcal{X}_{\text{unsafe}}$, aiming to obtain a control policy that is able to drive the agent from $\mathcal{X}_{\text{unsafe}}$ to $\mathcal{X}_{\text{safe}}$ and then maintain safety. Although the dual ascent update has been widely used in previous studies, this method is sensitive to hyperparameters and can lead to lengthy and possibly oscillating training processes with many constraint violations. To quickly find feasible solutions under state-wise constraints, here we first convert inequality constraints (8) to equality ones as

$$\begin{aligned} &ReLU\left(-\boldsymbol{\Phi}_{i,r_i}(x_{t_k})\right) = 0, && \forall x_{t_k} \in \mathcal{X}, \forall i \in [1, \dots, m], \\ &ReLU\left(L^{\pi_p}(x_{t_{k+1}}) - L^{\pi_p}(x_{t_k}) + \beta L^{\pi_p}(x_{t_k})\right) = 0, && \forall x_{t_k} \in \mathcal{X}. \end{aligned} \tag{9}$$

Therefore, we obtain the constrained optimization problem:

$$\begin{aligned} \min_{\pi_p}\ & \mathbb{E}_{x_{t_k}}[-V^{\pi_p}(x_{t_k})] \\ s.t.\ & \sum_{x_{t_k} \in \mathcal{X}} ReLU\left(-\boldsymbol{\Phi}_{i,r_i}(x_{t_k})\right) = 0 \quad \forall i \in [1, \dots, m], \\ & \sum_{x_{t_k} \in \mathcal{X}} ReLU\left(L^{\pi_p}(x_{t_{k+1}}) - L^{\pi_p}(x_{t_k}) + \beta L^{\pi_p}(x_{t_k})\right) = 0, \end{aligned} \tag{10}$$

where $V^{\pi_p}(x_{t_k}) = \mathbb{E}_{\tau \sim \pi_p}\left[\sum_{i=0}^{\infty} \gamma^i r(x_{t_{k+i}})\right]$ is the value function under the primary policy π_p. The commonly used dual ascent update relies on the application of a Lagrangian function, however, here we enhance the Lagrangian function by incorporating an additional squared term, resulting in an augmented Lagrangian function. This squared term helps accelerate convergence and reduce sensitivity to hyperparameters, mitigating issues such as oscillation during training. Based on this augmented Lagrangian function, the augmented Lagrangian method can be effectively utilized to handle complex constraints, leading to improved performance in constrained reinforcement learning tasks. To elaborate, denote the Lagrangian multipliers for CBF and CLF constraints in Problem (10) as λ_i and ζ, respectively. Additionally, let $c > 0$ represent a penalty parameter that increases monotonically with the number of updates and approaches infinity as the updates progress. Using these parameters, the augmented Lagrangian function is formulated as follows:

$$\begin{aligned} \mathcal{L}_c(\pi_p, \lambda_i, \zeta) = {} & \mathbb{E}_{x_{t_k}}\left[-V^{\pi_p}(x_{t_k})\right] + \sum_{i=1}^{m} \lambda_i \left(\sum_{x_{t_k} \in \mathcal{X}} ReLU\left(-\boldsymbol{\Phi}_{i,r_i}(x_{t_k})\right) \right) \\ & + \sum_{i=1}^{m} \frac{c}{2} \left(\sum_{x_{t_k} \in \mathcal{X}} ReLU\left(-\boldsymbol{\Phi}_{i,r_i}(x_{t_k})\right) \right)^2 \\ & + \zeta \left(\sum_{x_{t_k} \in \mathcal{X}} ReLU\left(L^{\pi_p}(x_{t_{k+1}}) - L^{\pi_p}(x_{t_k}) + \beta L^{\pi_p}(x_{t_k})\right) \right) \\ & + \frac{c}{2} \left(\sum_{x_{t_k} \in \mathcal{X}} ReLU\left(L^{\pi_p}(x_{t_{k+1}}) - L^{\pi_p}(x_{t_k}) + \beta L^{\pi_p}(x_{t_k})\right) \right)^2. \end{aligned} \tag{11}$$

In this equation, the first term on the right-hand side corresponds to the objective function of Problem (10). The second term enforces the

safety constraints based on CBFs, as described in Problem (10), with the third term being its corresponding squared penalty term with the penalty parameter c. Also, m represents the number of CBF constraints. Similarly, the fourth term imposes the stability constraints derived from the CLF, and the fifth term is its corresponding squared penalty term. With function (11), the augmented Lagrangian method can be applied to solve the state-wise constrained RL problem in non-CMDP settings, as shown in Algorithm 1. Furthermore, considering practical needs, such as approximating necessary functions and reducing the computational burden, a practical implementation with more detailed descriptions is provided in Section 5.

Algorithm 1 State-Wise Stable and Safe RL via ALM

Input: Initialized policy π_p^0, Lagrangian multipliers λ_i^0 and ζ^0, penalty parameter c^0, factor $\rho_c \in (1,\infty)$, maximum number of iterations J.

1: **for** $j = 0, 1, \ldots, J$ **do**
2: $\pi_p^j \leftarrow \text{argmin}_{\pi_p} \mathcal{L}_{c^j}(\pi_p, \lambda_i^j, \zeta^j)$
3: $\lambda_i^{j+1} \leftarrow \lambda_i^j + c^j \left(\sum_{x_{t_k}\in\mathcal{X}} ReLU\left(-\Phi_{i,r_i}(x_{t_k})\right)\right)$
4: $\zeta^{j+1} \leftarrow \zeta^j + c^j \left(\sum_{x_{t_k}\in\mathcal{X}} ReLU\left(L^{\pi_p^j}(x_{t_{k+1}}) - L^{\pi_p^j}(x_{t_k}) + \beta L^{\pi_p^j}(x_{t_k})\right)\right)$
5: $c^{j+1} \leftarrow \rho_c c^j$
6: **end for**

Output: π^J

Remark 3. Different from [42], which solves problems in a special subclass of CMDPs, the method proposed here can directly handle state-wise constraints in non-CMDP settings via ALM, which allows for constraints that are not solely based on the instantaneous cost of the state–action–transition tuple. Also, by updating the Lagrangian multipliers following Lines 3 and 4, this method eliminates the need to tune a learning rate for updating the Lagrangian multipliers, as required in [26,42]. Moreover, while CBF and CLF constraints are obtained in a model-based manner here, this method can still be applied to model-free RL or other tasks with trajectory-based constraints by simply changing the constraints used, and thus has broad applicability.

3.2. Backup controller design

When state-wise CLF and CBF constraints are considered simultaneously, there may exist some states where the controlled system's approach towards the goal x_{desired}, driven by the CLF constraint, will cause immediate safety violations. Therefore, the following definition is given:

Definition 3. $\mathcal{X}_{\text{infeasible}}$ is the set of states where there exists no controller such that the agent can further approach towards the goal x_{desired} according to the CLF constraint without causing immediate safety violations.

At $x_{t_k} \in \mathcal{X}_{\text{infeasible}}$, once the controller satisfies the CLF constraint and drives the controlled agent to approach towards x_{desired}, safety violations will happen, which means the pre-defined CBF constraint is violated. At such states, priority should be given to safety constraints. Rather than using a QP-based controller or other manually specified stochastic policy, here we use an additional RL-based backup controller. This approach can handle nonconvex constraints, and for a primary RL-based controller π_p trained with CBF and CLF constraints, the corresponding RL-based backup controller π_b can be designed to be the solution of the following constrained optimization problem:

$$\begin{aligned} &\min_{\pi_b} \mathbb{E}_{x_{t_k},u_{t_k}\sim\pi_b}[-V^{\pi_p}(x_{t_{k+1}})] \\ &s.t. \sum_{x_{t_k}\in\mathcal{X}} ReLU\left(-\Phi^b_{i,r_i}(x_{t_k})\right) = 0 \ \forall i \in [1,\ldots,m]. \end{aligned} \tag{12}$$

The definition of Φ^b_{i,r_i} is the same as that of Φ_{i,r_i}, except that the states $\{x_{t_{k+1}}, x_{t_{k+2}}, \ldots, x_{t_{k+r_i}}\}$ used in the objective function and constraints are future states following the backup controller π_b. The objective function in Problem (12) reflects the intention of only using the backup controller π_b for one step and then switching back to the primary controller π_p while maximizing the cumulative discounted reward. The augmented Lagrangian function for updating π_b is given as

$$\begin{aligned} \mathcal{L}_{c_b}(\pi_b, \lambda_{b,i}) = &\mathbb{E}_{x_{t_k},u_{t_k}\sim\pi_b}\left[-V^{\pi_p}(x_{t_{k+1}})\right] \\ &+ \sum_{i=1}^{m} \lambda_{b,i} \left(\sum_{x_{t_k}\in\mathcal{X}} ReLU\left(-\Phi^b_{i,r_i}(x_{t_k})\right)\right) \\ &+ \sum_{i=1}^{m} \frac{c_b}{2} \left(\sum_{x_{t_k}\in\mathcal{X}} ReLU\left(-\Phi^b_{i,r_i}(x_{t_k})\right)\right)^2, \end{aligned} \tag{13}$$

where $\lambda_{b,i}$ and c_b are Lagrangian multipliers for CBF constraints and the penalty parameter to update π_b.

When $x_{t_k} \in \mathcal{X}_{\text{infeasible}}$, the control signals generated may impede the system's swift approaching towards x_{desired} when CBF constraint is satisfied instead. For instance, the agent is forbidden to cross an obstacle by the CBF constraints while the CLF constraint requires it to do so to reach x_{desired}. The agent therefore gets stuck close to the obstacle (see Fig. 2), which is easy to detect. Based on the aforementioned analysis, the policy π_b is activated when the agent is detected to be trapped for some time. The primary controller is reinstated when the agent moves a certain distance away from the trapped position, or the time threshold for using the backup controller is exceeded. These conditions can be encoded in the implementation easily. Moreover, when integrated with SAC, an additional advantage of using the proposed RL-based backup controller is that SAC can encourage exploration by simply tuning a hyperparameter that sets the required lower bound of entropy, denoted as $\mathcal{H}_b$. The exploration and exploitation inherent in SAC with CBFs as constraints allow the system to escape trapped positions through diverse actions, without significantly reducing the rewards obtained. The detailed implementation is provided in Section 5.

4. Training with learned barrier certificate

It is common for useful pre-defined CBFs to be unavailable before training. In this section, we present a convenient way to extend the proposed method to tasks without pre-defined CBFs by using neural barrier certificates, and provide corresponding theoretical analysis.

4.1. Neural barrier certificates

We investigate a barrier certificate rather than a composite Lyapunov barrier function because, by decoupling Lyapunov and barrier certificates, each can be plug-and-played in new tasks with different goals or different safety constraints. A previous study [51] showed that the value function of a fixed policy that tries to maintain safety greedily can be modified to be a CBF to separate $\mathcal{X}_{\text{safe}}$ and $\mathcal{X}_{\text{unsafe}}$. However, here we need to update the policy jointly with a learned barrier certificate to satisfy both state-wise stability and safety constraints in an online manner. Therefore, firstly, we provide the following definition:

Definition 4. Three sets dependent on the control policy π are defined:

- $\mathcal{X}^{\pi}_{\text{irrec}}$: Set of states $x \notin \mathcal{X}_{\text{vio}}$, but under the controller π, the trajectory starting from $x_{t_0} = x$ will enter $\mathcal{X}_{\text{vio}}$ at least once with a positive probability, i.e., $\exists k \geq 0$, such that $Pr(x_{t_k} \in \mathcal{X}_{\text{vio}}) > 0$ under the controller π.
- $\mathcal{X}^{\pi}_{\text{unsafe}} = \mathcal{X}_{\text{vio}} \cup \mathcal{X}^{\pi}_{\text{irrec}}$ is the unsafe set under π, and $\mathcal{X}^{\pi}_{\text{safe}} = \mathcal{X} \backslash \mathcal{X}^{\pi}_{\text{unsafe}}$ is the safe set under π, i.e., for every $x_{t_0} = x \in \mathcal{X}^{\pi}_{\text{safe}}$, under the controller π, for all $k \geq 0$, $Pr(x_{t_k} \in \mathcal{X}_{\text{vio}}) = 0$, namely the safety is maintained under π.

Later, $\mathcal{X}^{\pi}_{\text{safe}}$ will be considered instead since π is changing during learning. For further analysis, the following assumptions are provided:

Assumption 1. The reward signal obtained at any timestep is upper bounded by r_{max} and lower bounded by r_{min}, respectively.

Assumption 2. There exists an upper bound N such that all trajectories starting from $x_{t_0} \in \mathcal{X}^{\pi}_{\text{irrec}}$ will enter the set $\mathcal{X}_{\text{vio}}$ at least once within N steps under the controller π.

Assumption 1 is widely used in many existing RL research to set an upper bound for the value function. Assumption 2 requires that a safety violation will happen reasonably soon after entering $\mathcal{X}^{\pi}_{\text{irrec}}$. Further, the definition of a barrier certificate is listed below:

Definition 5. H^{π} is a barrier certificate of the policy π if following properties hold:

- $\forall x_{t_k} \in \mathcal{X}^{\pi}_{\text{unsafe}}$, $H^{\pi}(x_{t_k}) < 0$;
- $\forall x_{t_k} \in \mathcal{X}^{\pi}_{\text{safe}}$, $H^{\pi}(x_{t_k}) \geq 0$;
- $\forall x_{t_k} \in \mathcal{X}^{\pi}_{\text{safe}}$, $H^{\pi}(x_{t_{k+1}}) - H^{\pi}(x_{t_k}) \geq -\epsilon H^{\pi}(x_{t_k})$ where $\epsilon \in (0,1)$ and $x_{t_{k+1}}$ is the next state following the controller π.

Then, based on Assumption 2 and Definition 5, we provide the following lemma:

Lemma 1. *Introduce an additional barrier signal as*

$$b(x_{t_k}) = \begin{cases} d & \text{if } x \notin \mathcal{X}_{vio}, \\ D & \text{if } x \in \mathcal{X}_{vio}, \end{cases} \tag{14}$$

where $d = 0$ and $D < 0$ are constants. Under Assumption 2, the function $H^{\pi}(x_{t_k}) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{i=0}^{\infty} \gamma^i b(x_{t_{k+i}})\right]$, where $\gamma \in (0,1)$ is the discount factor and $\tau = \{x_{t_k}, x_{t_{k+1}}, x_{t_{k+2}}, \dots\}$ is a trajectory under controller π starting from x_{t_k}, is a barrier certificate of policy π.

Proof. When $x_{t_k} \in \mathcal{X}^{\pi}_{\text{safe}}$, by Definition 4, under the controller π, for all $i \geq 0$, $Pr(x_{t_{k+i}} \in \mathcal{X}_{\text{vio}}) = 0$. Therefore, $H^{\pi}(x_{t_k}) = H^{\pi}(x_{t_{k+1}}) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{i=0}^{\infty} \gamma^i d\right] = 0$ given $d = 0$, and hence the second property in Definition 5 is satisfied. When $x_{t_k} \in \mathcal{X}^{\pi}_{\text{unsafe}}$, according to Definition 4, there exists $n \geq 0$ such that $Pr(x_{t_{k+n}} \in \mathcal{X}_{\text{vio}}) > 0$ under the controller π, and therefore $\mathbb{E}_{x_{t_{k+n}}}\left[b(x_{t_{k+n}})\right] < 0$. Furthermore, under Assumption 2, we have $n \leq N$. Thus, we have:

$$\begin{aligned} H^{\pi}(x_{t_k}) =& \mathbb{E}_{\tau\sim\pi}\left[\sum_{i=0}^{\infty} \gamma^i b(x_{t_{k+i}})\right] \leq \gamma^n \mathbb{E}_{x_{t_{k+n}}}\left[b(x_{t_{k+n}})\right] \\ \leq& \gamma^N \mathbb{E}_{x_{t_{k+n}}}\left[b(x_{t_{k+n}})\right] < 0, \end{aligned}$$

since $\gamma \in (0,1)$. Therefore, the first property in Definition 5 is satisfied. Also, when $x_{t_k} \in \mathcal{X}^{\pi}_{\text{safe}}$, by Definition 4, $x_{t_{k+1}} \in \mathcal{X}^{\pi}_{\text{safe}}$, and thus $H^{\pi}(x_{t_{k+1}}) - H^{\pi}(x_{t_k}) = 0 - 0 = -\epsilon H^{\pi}(x_{t_k})$. Therefore, the third property in Definition 5 is satisfied. In sum, all three properties for being a barrier certificate are satisfied, and thus, the function $H^{\pi}(x_{t_k})$ is a barrier certificate of policy π. □

Remark 4. It is noteworthy that due to the setting $d = 0$, which indicates $\forall x_{t_k} \in \mathcal{X}^{\pi}_{\text{safe}}$, $H^{\pi}(x_{t_k}) = 0$, the range of ϵ in the third condition of Definition 5 can also be changed to $\epsilon \geq 1$.

Common methods in previous studies using barrier functions require the forward invariance of a pre-defined safe set, which can be inaccurate and conservative in practice, or rely on an extra safe policy and system dynamics to train a barrier certificate by imposing additional conditions on derivatives. Different from these prior works, this method uses a barrier certificate that, by design, has a more negative value when more severe safety violations occur, and 0 when safety is maintained at every subsequent timestep under the current policy π. Also, this certificate can be directly trained, as a value function, following standard RL algorithms in a model-free, convenient, and principled way.

4.2. Theoretical analysis

A lemma showing the convergence of the barrier certificate of a fixed policy π is presented below.

Lemma 2. *For a fixed control policy π, consider a mapping $H^0 : \mathcal{X} \to \mathcal{R}$ and the Bellman operator:*

$$\mathcal{B}^{\pi}[H(x_{t_k})] = b(x_{t_k}) + \gamma \sum_{u_{t_k}} \pi(u_{t_k}|x_{t_k}) H(x_{t_{k+1}}) \tag{15}$$

where $x_{t_{k+1}}$ is the next state following x_{t_k} and u_{t_k} according to the deterministic system dynamics, by defining q as the number of updates and $H^{q+1} = \mathcal{B}^{\pi}[H^q]$, the sequence H^q will converge to the barrier certificate H^{π} when $q \to \infty$.

Proof. For any two mappings H and H', define the maximum norm as:

$$\|H - H'\|_{\infty} = \max_{x_{t_k}\in\mathcal{X}} |H(x_{t_k}) - H'(x_{t_k})|$$

Then,

$$\begin{aligned} \|\mathcal{B}^{\pi}[H] - \mathcal{B}^{\pi}[H']\|_{\infty} &= \max_{x_{t_k}\in\mathcal{X}} |\mathcal{B}^{\pi}[H(x_{t_k})] - \mathcal{B}^{\pi}[H'(x_{t_k})]| \\ &= \max_{x_{t_k}\in\mathcal{X}} |\gamma \sum_{u_{t_k}} \pi(u_{t_k}|x_{t_k})[H(x_{t_{k+1}}) - H'(x_{t_{k+1}})]| \\ &\leq \gamma \max_{x_{t_k}\in\mathcal{X}} \sum_{u_{t_k}} \pi(u_{t_k}|x_{t_k}) \|H - H'\|_{\infty}. \end{aligned}$$

Therefore, $\|\mathcal{B}^{\pi}[H] - \mathcal{B}^{\pi}[H']\|_{\infty} \leq \gamma \|H - H'\|_{\infty}$. Since $\gamma \in (0,1)$, the Bellman operator $\mathcal{B}^{\pi}$ is a γ contraction mapping. According to [52, Proposition A.26], it can be concluded that $H^q(x_{t_k})$ will converge to $H^{\pi}(x_{t_k})$ for $x_{t_k} \in \mathcal{X}$ when $q \to \infty$. □

Denote π_n to be the control policy obtained at nth time, then we need to solve the following constrained optimization problem, for all $x_{t_k} \in \mathcal{X}$, to obtain the policy π_{n+1}:

$$\pi_{n+1} = \arg\min_{\pi\in\Pi} \; -\mathbb{E}_{u_{t_k}\sim\pi} V^{\pi_n}(x_{t_{k+1}}) \tag{16a}$$

$$s.t. \; -H^{\pi_n}(x_{t_{k+1}}) + H^{\pi_n}(x_{t_k}) - \epsilon H^{\pi_n}(x_{t_k}) \leq 0 \;\; \forall x_{t_k} \in \mathcal{X}, \tag{16b}$$

where $x_{t_{k+1}}$ is the next state following x_{t_k} and u_{t_k} according to the system dynamics. Different from previous studies [16] applying region-wise policy improvement that firstly separate the whole state space $\mathcal{X}$ into different regions according to certain functions, e.g., constraint decay function (CDF), and then solve different optimization problems on different regions with the safety being enhanced by minimizing safety-related objective functions, in our proposed method, the safety is required and achieved by applying the barrier certificate as a constraint of a constrained optimization problem defined and should be solved for each $x_{t_k} \in \mathcal{X}$. This proposed method avoids dividing $\mathcal{X}$ into different regions based on values output by a certain neural network trained during the learning process to approximate functions like the CDF, thereby circumventing additional errors, such as misclassifications of states, that occur when the neural network's approximation is inaccurate.

Then, we prove that by solving Problem (16) iteratively, under certain assumptions, the set $\mathcal{X}^{\pi_n}_{\text{safe}}$ will converge to $\mathcal{X}_{\text{safe}}$, and V^{π_n} will converge to an optimal solution on $\mathcal{X}_{\text{safe}}$. The additional assumption required is given as follows:

Assumption 3. For any state $x_{t_k} \in \mathcal{X}$ where there exists at least one policy that satisfies the constraint (16b), the solution π_{n+1} obtained by solving Problem (16) via constrained optimization methods will satisfy the constraint (16b) at x_{t_k}. For states x_{t_k} where no policy that satisfies the constraint (16b) exists, the policy π_{n+1} satisfies $\pi_{n+1}(x_{t_k}) = \arg\max_{\pi\in\Pi} H^{\pi_n}(x_{t_{k+1}})$ where $x_{t_{k+1}}$ is the next state following π.

Assumption 3 means that once at least one control policy exists to satisfy the constraint (16b) at x_{t_k}, such policy (policies) will be searched for, and the policy π_{n+1} returned by solving Problem (16) will be one of them. $\forall x_{t_k} \in \mathcal{X}_{\text{safe}}^{\pi_n}$, π_n satisfies (16b) and at least π_{n+1} can be set equal to π_n. For $\mathcal{X}_{\text{unsafe}}^{\pi_n}$, there may exist some states $x_{t_k} \in \mathcal{X}_{\text{unsafe}}^{\pi_n}$ such that no policy satisfying (16b) exists. At such states, π_{n+1} is the policy under which the next state $x_{t_{k+1}}$ has the maximum value $H^{\pi_n}(x_{t_{k+1}})$.

Then, we provide the following theorem:

Theorem 1. *Suppose that initial policy π_0 satisfies $\mathcal{X}_{safe}^{\pi_0} \neq \emptyset$, and denote the control policy obtained by iteratively solving Problem (16) as $\{\pi_n\}_{n>0}$. Under Assumptions 1 and 3, when $\epsilon \geq 1$, the safe set under the obtained policy monotonically expands, i.e., $\mathcal{X}_{safe}^{\pi_n} \subseteq \mathcal{X}_{safe}^{\pi_{n+1}}$, and will converge to $\mathcal{X}_{safe}$, i.e., $\lim_{n\to\infty} \mathcal{X}_{safe}^{\pi_n} = \mathcal{X}_{safe}$. Additionally, for the objective function V^{π_n}, $V^{\pi_n}(x_{t_k}) \leq V^{\pi_{n+1}}(x_{t_k})$ holds for all $x_{t_k} \in \mathcal{X}_{safe}^{\pi_n}$.*

Proof. Firstly we show that the set $\mathcal{X}_{\text{safe}}^{\pi_n}$ monotonically expands.

For all $x_{t_k} \in \mathcal{X}_{\text{safe}}^{\pi_n}$, the condition in Problem (16) requires $H^{\pi_n}(x_{t_{k+1}}) \geq (1-\epsilon)H^{\pi_n}(x_{t_k}) = 0$ for $x_{t_{k+1}}$ following the newly obtained controller π_{n+1}. Therefore, $x_{t_{k+1}} \in \mathcal{X}_{\text{safe}}^{\pi_n}$, and similarly, $\forall x_{t_0} \in \mathcal{X}_{\text{safe}}^{\pi_n}$, the trajectory $\{x_{t_0}, x_{t_1}, x_{t_2}, \dots\}$ under the policy π_{n+1} remains in $\mathcal{X}_{\text{safe}}^{\pi_n}$ without entering $\mathcal{X}_{\text{vio}}$ with a positive probability. Thus, $x_{t_0} \in \mathcal{X}_{\text{safe}}^{\pi_{n+1}}$ since $H^{\pi_{n+1}}(x_{t_0}) = \mathbb{E}_{\tau\sim\pi_{n+1}}\left[\sum_{i=0}^{\infty}\gamma^i b(x_{t_{k+i}})\right] = \mathbb{E}_{\tau\sim\pi_{n+1}}\left[\sum_{i=0}^{\infty}\gamma^i d\right] = 0$.

When $\epsilon \geq 1$, if $\exists x_{t_k} \in \mathcal{X}_{\text{irrec}}^{\pi_n}$, such that under the new policy π_{n+1}, $H^{\pi_n}(x_{t_{k+1}}) = \max_{\pi\in\Pi} H^{\pi_n}(x_{t_{k+1}}^{\pi}) = 0$, where $x_{t_{k+1}}$ follows the policy π_{n+1} and $x_{t_{k+1}}^{\pi}$ follows any other policy π, then, according to the previous analysis, the following states $\{x_{t_{k+2}}, x_{t_{k+3}}, x_{t_{k+4}}, \dots\}$ will stay within $\mathcal{X}_{\text{safe}}^{\pi_n}$ under π_{n+1}, and therefore $x_{t_k} \in \mathcal{X}_{\text{safe}}^{\pi_{n+1}}$ since $H^{\pi_{n+1}}(x_{t_k}) = 0$ given $b(x_{t_k}) = 0$.

Based on the aforementioned analysis, given $\mathcal{X}_{\text{safe}}^{\pi_0} \neq \emptyset$, it can be concluded that the set $\mathcal{X}_{\text{safe}}^{\pi_n}$ monotonically expands, i.e., $\mathcal{X}_{\text{safe}}^{\pi_n} \subseteq \mathcal{X}_{\text{safe}}^{\pi_{n+1}}$.

Secondly, we show $\lim_{n\to\infty} \mathcal{X}_{\text{safe}}^{\pi_n} = \mathcal{X}_{\text{safe}}$[1]. Since $\{\mathcal{X}_{\text{safe}}^{\pi_n}\}_{n\geq 0}$ is an increasing sequence of sets, i.e., $\mathcal{X}_{\text{safe}}^{\pi_n} \subseteq \mathcal{X}_{\text{safe}}^{\pi_{n+1}}$ for each n, then $\lim_{n\to\infty} \mathcal{X}_{\text{safe}}^{\pi_n} = \cup_{n=1}^{\infty}\mathcal{X}_{\text{safe}}^{\pi_n}$. Because $\forall x_{t_k} \in \cup_{n=1}^{\infty}\mathcal{X}_{\text{safe}}^{\pi_n}$, there exists a number $Z \in \mathbb{Z}^+$ specific to x_{t_k} such that $\forall i \geq Z$, $x_{t_k} \in \mathcal{X}_{\text{safe}}^{\pi_i}$. Hence, there exist control policies maintaining safety for x_{t_k}, which indicates $x_{t_k} \in \mathcal{X}_{\text{safe}}$, and therefore $\cup_{n=1}^{\infty}\mathcal{X}_{\text{safe}}^{\pi_n} \subseteq \mathcal{X}_{\text{safe}}$. Also, $\forall x_{t_k} \notin \cup_{n=1}^{\infty}\mathcal{X}_{\text{safe}}^{\pi_n}$, assume there exists at least a control policy under which safety can be maintained for x_{t_k}, according to Assumption 3 which implies that the search is performed on the whole policy space, there exists a positive number $M \in \mathbb{Z}^+$ such that under the new policy π_{M+1}, $H^{\pi_M}(x_{t_{k+1}}) = \max_{\pi\in\Pi} H^{\pi_M}(x_{t_{k+1}}^{\pi}) = 0$, where $x_{t_{k+1}}$ follows the policy π_{M+1} and $x_{t_{k+1}}^{\pi}$ follows any other policy π, then, according to the previous analysis, $H^{\pi_{M+1}}(x_{t_k}) = 0$ and $x_{t_k} \in \mathcal{X}_{\text{safe}}^{\pi_{M+1}}$, which contradicts with the condition that $x_{t_k} \notin \cup_{n=1}^{\infty}\mathcal{X}_{\text{safe}}^{\pi_n}$. Hence, there exists no control policy π to achieve $H^{\pi}(x_{t_k}) = 0$, which indicates either $x_{t_k} \in \mathcal{X}_{\text{vio}}$, or $\forall\pi$, the trajectory starting from x_{t_k} will enter $\mathcal{X}_{\text{vio}}$ with a positive probability. Thus, by definition, $x_{t_k} \in \mathcal{X}_{\text{vio}} \cup \mathcal{X}_{\text{irrec}} = \mathcal{X}_{\text{unsafe}}$. Since $\mathcal{X}_{\text{safe}} = \mathcal{X}\backslash\mathcal{X}_{\text{unsafe}}$, we have $x_{t_k} \notin \mathcal{X}_{\text{safe}}$. Because $\forall x_{t_k} \notin \cup_{n=1}^{\infty}\mathcal{X}_{\text{safe}}^{\pi_n}$, we have $x_{t_k} \notin \mathcal{X}_{\text{safe}}$, equivalently we have $\forall x_{t_k} \in \mathcal{X}_{\text{safe}}$, $x_{t_k} \in \cup_{n=1}^{\infty}\mathcal{X}_{\text{safe}}^{\pi_n}$, namely $\mathcal{X}_{\text{safe}} \subseteq \cup_{n=1}^{\infty}\mathcal{X}_{\text{safe}}^{\pi_n}$. Given $\cup_{n=1}^{\infty}\mathcal{X}_{\text{safe}}^{\pi_n} \subseteq \mathcal{X}_{\text{safe}}$ and $\mathcal{X}_{\text{safe}} \subseteq \cup_{n=1}^{\infty}\mathcal{X}_{\text{safe}}^{\pi_n}$, we have $\mathcal{X}_{\text{safe}} = \cup_{n=1}^{\infty}\mathcal{X}_{\text{safe}}^{\pi_n}$, and thus $\lim_{n\to\infty} \mathcal{X}_{\text{safe}}^{\pi_n} = \mathcal{X}_{\text{safe}}$.

Furthermore, for any policy π_n, $\forall x_{t_k} \in \mathcal{X}_{\text{safe}}^{\pi_n}$, according to the objective function in Problem (16), we have:

$$\mathbb{E}_{u_{t_k}\sim\pi_{n+1}} V^{\pi_n}(x_{t_{k+1}}) \geq \mathbb{E}_{u_{t_k}\sim\pi_n} V^{\pi_n}(x'_{t_{k+1}})$$

where $x_{t_{k+1}}$ is the next state following π_{n+1} and $x'_{t_{k+1}}$ is the next state following π_n. Such π_{n+1} must exist because at least π_{n+1} can be chosen to be equal to π_n. Therefore,

$$\begin{aligned}
V^{\pi_n}(x_{t_k}) &= r(x_{t_k}) + \gamma\mathbb{E}_{u_{t_k}\sim\pi_n} V^{\pi_n}(x'_{t_{k+1}}) \\
&\leq r(x_{t_k}) + \gamma\mathbb{E}_{u_{t_k}\sim\pi_{n+1}} V^{\pi_n}(x_{t_{k+1}}) \\
&\leq r(x_{t_k}) + \mathbb{E}_{u_{t_k}\sim\pi_{n+1}}[\gamma r(x_{t_{k+1}}) + \gamma^2\mathbb{E}_{u_{t_{k+1}}\sim\pi_{n+1}} V^{\pi_n}(x_{t_{k+2}})] \\
&\leq r(x_{t_k}) + \mathbb{E}_{\pi_{n+1}}[\gamma r(x_{t_{k+1}}) + \gamma^2 r(x_{t_{k+2}}) + \dots] \\
&= V^{\pi_{n+1}}(x_{t_k})
\end{aligned}$$

Thus, $\forall x_{t_k} \in \mathcal{X}_{\text{safe}}^{\pi_n}$, $V^{\pi_n}(x_{t_k}) \leq V^{\pi_{n+1}}(x_{t_k})$. □

Based on Theorem 1, a proposition can be given as follows:

Proposition 1. *If there exists $Z \in \mathbb{Z}^+$ such that $\forall i \geq Z$, $\mathcal{X}_{safe}^{\pi_i} = \mathcal{X}_{safe}^{\pi_Z}$, then, the objective function V^{π_n} will converge, i.e., $\lim_{n\to\infty} V_{safe}^{\pi_n} = V^*$, where $V^*(x_{t_k}) \geq V^{\pi}(x_{t_k})$, $\forall\pi \in \Pi^s$ and $\forall x_{t_k} \in \mathcal{X}_{safe}$.*

Proof. According to Theorem 1, $\forall i \geq Z$, $\mathcal{X}_{\text{safe}}^{\pi_i} = \lim_{n\to\infty}\mathcal{X}_{\text{safe}}^{\pi_n} = \mathcal{X}_{\text{safe}}$, and thus when $n \to \infty$, V^{π_n} is optimized on the set $\mathcal{X}_{\text{safe}}$. By Theorem 1, $V^{\pi_n}(x_{t_k}) \leq V^{\pi_{n+1}}(x_{t_k})$ for all $x_{t_k} \in \mathcal{X}_{\text{safe}}$. Given the reward signal is upper bounded, i.e., $r(x_{t_k}) \leq r_{max}$ for all $x_{t_k} \in \mathcal{X}$, $V^{\pi_n}(x_{t_k}) \leq \frac{r_{max}}{1-\gamma}$ is also upper bounded, and thus V^{π_n} will converge to a V^* with the corresponding policy π^*. At convergence, it follows that for all $\pi \in \Pi^s$ and $x_{t_k} \in \mathcal{X}_{\text{safe}}$,

$$\mathbb{E}_{u_{t_k}\sim\pi^*} V^*(x_{t_{k+1}}^*) \geq \mathbb{E}_{u_{t_k}\sim\pi} V^*(x_{t_{k+1}}^{\pi})$$

where $x_{t_{k+1}}^*$ is the next state following π^* and $x_{t_{k+1}}^{\pi}$ is the next state following π. Hence,

$$\begin{aligned}
V^*(x_{t_k}) &= r(x_{t_k}) + \gamma\mathbb{E}_{u_{t_k}\sim\pi^*} V^*(x_{t_{k+1}}^*) \\
&\geq r(x_{t_k}) + \gamma\mathbb{E}_{u_{t_k}\sim\pi} V^*(x_{t_{k+1}}^{\pi}) \\
&\geq r(x_{t_k}) + \mathbb{E}_{u_{t_k}\sim\pi}[\gamma r(x_{t_{k+1}}^{\pi}) + \gamma^2\mathbb{E}_{u_{t_{k+1}}\sim\pi} V^*(x_{t_{k+2}}^{\pi})] \\
&\geq r(x_{t_k}) + \mathbb{E}_{\pi}[\gamma r(x_{t_{k+1}}^{\pi}) + \gamma^2 r(x_{t_{k+2}}^{\pi}) + \dots] \\
&= V^{\pi}(x_{t_k})
\end{aligned}$$

for all $x_{t_k} \in \mathcal{X}_{\text{safe}}$, which means the converged V^* has a larger or equal value compared to the value function of any other policy $\pi \in \Pi^s$, i.e., $V^*(x_{t_k}) \geq V^{\pi}(x_{t_k})$, $\forall\pi \in \Pi^s$ and $\forall x_{t_k} \in \mathcal{X}_{\text{safe}}$. □

Remark 5. In practical implementation, it is usually difficult to find the optimizer of Problem (16) on the whole policy set at each iteration, and therefore gradient-based methods are commonly applied, which means π_{n+1} is usually found in a neighborhood of π_n. Thus, a locally optimal solution, whose safe set is a subset of $\mathcal{X}_{\text{safe}}$ and value function is only locally optimal, is usually obtained in real implementation which is described in Section 5.

4.3. Combination with the CLF constraint

Similar to the previous section, CLF constraint can be added to Problem (16) to have a new constrained optimization problem, which has equality constraints and is listed below, to additionally maintain stability:

$$\begin{aligned}
\pi_{n+1} = \arg\min_{\pi}\ & -\mathbb{E}_{u_{t_k}\sim\pi} V^{\pi_n}(x_{t_{k+1}}) \\
s.t.\ & \sum_{x_{t_k}\in\mathcal{X}} ReLU\left(-H^{\pi_n}(x_{t_{k+1}}) + H^{\pi_n}(x_{t_k}) - \epsilon H^{\pi_n}(x_{t_k})\right) = 0, \\
& \sum_{x_{t_k}\in\mathcal{X}} ReLU\left(L^{\pi_n}(x_{t_{k+1}}) - L^{\pi_n}(x_{t_k}) + \beta L^{\pi_n}(x_{t_k})\right) = 0,
\end{aligned} \tag{17}$$

where $x_{t_{k+1}}$ is the next state following x_{t_k} and u_{t_k} according to the system dynamics. Moreover, adding the CLF constraint offers an extra benefit that, as the controller is updated to drive the agent towards the desired state x_{desired}, certain states near x_{desired} may have their safety easily maintained by the updated controller that directly drives the agent to reach x_{desired}. Then, the safe set will monotonically expand,

[1] Here, we consider cases where each update results in a nonzero-measure expansion of the identified safe set if new safe states are found.

as stated in Theorem 1. Problem (17) will still be solved by using the augmented Lagrangian method. However, still, there may exist $x_{t_k} \in \mathcal{X}_{\text{infeasible}}$ that we hope to avoid. Therefore, we introduce the following assumption:

Assumption 4. $\forall n \geq 0$ and $\forall x_{t_k} \in \mathcal{X}_{\text{infeasible}}$, the probability that the policy π_{n+1} obtained by solving Problem (17) via the augmented Lagrangian method satisfies the CLF constraint is positive.

This assumption assumes that when the agent is within $\mathcal{X}_{\text{infeasible}}$ where no feasible solution exists to simultaneously maintain safety and drive the agent closer to the goal according to the CLF constraint, for each iteration, there is a positive probability that the augmented Lagrangian method outputs a control policy that satisfies the CLF constraint, thereby driving the agent closer to the goal and causing immediate safety violations. Then, a proposition is given:

Proposition 2. *Suppose $\exists Z \in \mathbb{Z}^+$ such that $x_{t_0} \in \mathcal{X}^{\pi_Z}_{safe}$, then, under Assumption 4, when $n \to \infty$, for any trajectory that starts from x_{t_0} and is driven by π_n obtained by solving Problem (17) iteratively, $Pr(x_{t_k} \in \mathcal{X}_{infeasible}) = 0$, $\forall k \geq 0$.*

Proof. Since $x_{t_0} \in \mathcal{X}^{\pi_i}_{\text{safe}}$, $\forall i \geq Z$, when $n \to \infty$, $x_{t_0} \in \mathcal{X}^{\pi_n}_{\text{safe}}$ and $H^{\pi_n}(x_{t_0}) = 0$. Because $\forall n \geq 0$ and $\forall x_{t'_k} \in \mathcal{X}_{\text{infeasible}}$, $Pr(x_{t'_{k+1}} \in \mathcal{X}_{\text{vio}}) > 0$ according to Assumption 4 and the definition of $\mathcal{X}_{\text{infeasible}}$, it can be concluded that

$$H^{\pi_{n+1}}(x_{t'_k}) = b(x_{t'_k}) + \gamma \sum_{u_{t_k}} \pi_{n+1}(u_{t_k}|x_{t'_k}) H^{\pi_{n+1}}(x_{t'_{k+1}}) < 0$$

since $Pr(H^{\pi_{n+1}}(x_{t'_{k+1}}) < 0) > 0$ given $Pr(x_{t'_{k+1}} \in \mathcal{X}_{\text{vio}}) > 0$. Because $\forall n \geq Z$, $H^{\pi_n}(x_{t_0}) = 0 > H^{\pi_n}(x_{t'_k})$ for any $x_{t'_k} \in \mathcal{X}_{\text{infeasible}}$, any trajectory starts from x_{t_0} will not enter $\mathcal{X}_{\text{infeasible}}$ with a positive probability under π_n when $n \to \infty$ due to the neural barrier certificate. □

According to Proposition 2, the controlled system will be prevented from visiting $\mathcal{X}_{\text{infeasible}}$ due to the neural barrier certificate. Therefore, under such circumstances, the backup controller proposed in the previous section where pre-defined CBFs are used can be not necessary.

5. Practical implementation with SAC and NODEs

In this section, we provide a practical implementation, which is called Neural ordinary differential equations-based Lyapunov-Barrier Actor-Critic (NLBAC) by combining the proposed method with Soft Actor-Critic (SAC) and applying neural ordinary differential equations (NODEs) for system modeling. Specifically, depending on whether pre-defined CBFs are available, two versions of the algorithm NLBAC are given separately with individual pseudocodes.

If useful pre-defined CBFs are given prior to the training process, then, according to (4), future states $\{x_{t_{k+1}}, x_{t_{k+2}}, \dots, x_{t_{k+r_i}}\}$ under the controller π_p, where r_i is the relative degree, are required. When a learned barrier certificate is used, $x_{t_{k+1}}$ is also required to construct Problem (17). Therefore, predictions must be made in a real implementation. Without nominal models, instead of conventional neural networks, we choose NODEs to directly approximate the dynamics of the system, since NODEs offer adaptability to discretization steps due to their inherent capability to learn the continuous form of system dynamics which are usually differential equations. To train the NODE model, we use state sequences $X = \{x_{t_k}, x_{t_{k+1}}, \dots, x_{t_{k+h}}\}$ and corresponding control sequences $U = \{u_{t_k}, u_{t_{k+1}}, \dots, u_{t_{k+h-1}}\}$ whose elements are collected during the system's interaction with the environment and then stored in the replay buffer which saves all data, including states x_{t_k}, action signals u_{t_k}, reward signals r_{t_k}, cost signals c_{t_k} and barrier signals b_{t_k} (if used). The model then computes the approximated state sequence $\{\hat{x}_{t_{k+1}}, \hat{x}_{t_{k+2}}, \dots, \hat{x}_{t_{k+h}}\}$ based on (7). The loss function used to train the NODE model is:

$$\ell = \frac{1}{h} \sum_{i=1}^{h} |x_{t_{k+i}} - \hat{x}_{t_{k+i}}| \tag{18}$$

Comparisons showing NODEs have better performance in modeling compared to conventional neural networks are provided on our GitHub page at the link shown in Section 6.

Instead of using the learned system dynamics to generate imaginary rollouts to help train value functions, we utilize the output of the learned NODEs to construct CLF and CBFs for clearly guiding the controller to satisfy stability and safety constraints quickly. Therefore, the prediction horizon equals the relative degree r_i or 1, which alleviates the problem of large error propagation since r_i is usually not large.

In the implementation, we construct constrained optimization problems (10) and (17) based on the batch $\mathcal{D}$ sampled from the replay buffer, which is common practice in RL. Here we first consider the case where pre-defined CBFs are available. Denote the parameters of the primal RL-based control policy π_p and action-value networks $Q_i^{\pi_p}, i = 1, 2$ by θ_p and ϕ_i, and use L_ν, referred to as the Lyapunov network, to approximate L^{π_p} with parameters ν. By applying the Soft Actor-Critic (SAC) (other popular RL algorithms, such as Deep Deterministic Policy Gradient [53] and Proximal Policy Optimization [54] could also be applied) and the sampled data, the objective function of Problem (10) is rewritten as:

$$\begin{aligned} -R^{\theta_p} = -\mathbb{E}_{x_{t_k}\sim\mathcal{D},\xi\sim\mathcal{N}}\big[& \min_{j=1,2} Q_{\phi_j}(x_{t_k}, \tilde{u}_{\theta_p}(x_{t_k}, \xi)) \\ & - \alpha_p \log \pi_{\theta_p}(\tilde{u}_{\theta_p}(x_{t_k}, \xi)|x_{t_k})\big], \end{aligned} \tag{19}$$

where $\xi \sim \mathcal{N}(0, I)$ and $\tilde{u}_{\theta_p}$ denotes the action generated by the controller π_{θ_p}. The loss functions $J_{Q^{\pi_p}}(Q_{\phi_i})$, $J_{\alpha_p}(\alpha_p)$ and $J_{L^{\pi_p}}(L_\nu)$ used to update $Q_{\phi_i}, i = 1, 2$, coefficient α_p, and Lyapunov network L_ν are given as:

$$\begin{aligned} J_{Q^{\pi_p}}(Q_{\phi_i}) = \mathbb{E}_{(x_{t_k}, u_{t_k}, r_{t_k}, x_{t_{k+1}})\sim\mathcal{D},\xi\sim\mathcal{N}}\Big[\big[r_{t_k} + \gamma\big(& \min_{j=1,2} Q_{targ,\phi_j} \\ & (x_{t_{k+1}}, \tilde{u}_{\theta_p}(x_{t_{k+1}}, \xi)) \\ & - \alpha_p \log \pi_{\theta_p}(\tilde{u}_{\theta_p}(x_{t_{k+1}}, \xi)|x_{t_{k+1}})\big) - Q_{\phi_i}(x_{t_k}, u_{t_k})\big]^2\Big], \end{aligned} \tag{20}$$

$$J_{\alpha_p}(\alpha_p) = -\alpha_p \times \mathbb{E}_{x_{t_k}\sim\mathcal{D},\xi\sim\mathcal{N}}\big[\log \pi_{\theta_p}(\tilde{u}_{\theta_p}(x_{t_k}, \xi)|x_{t_k}) + \mathcal{H}\big], \tag{21}$$

$$J_{L^{\pi_p}}(L_\nu) = \mathbb{E}_{(x_{t_k}, c_{t_k}, x_{t_{k+1}})\sim\mathcal{D}}\Big[\big[c_{t_k} + \gamma_c L_{targ,\nu}(x_{t_{k+1}}) - L_\nu(x_{t_k})\big]^2\Big]. \tag{22}$$

Here L_ν can also be designed as an action-value network in the implementation. Furthermore, a surrogate augmented Lagrangian function based on neural networks and sampled data is given:

$$\begin{aligned} \mathcal{L}_c^{surr}(\theta_p, \lambda_i, \zeta) = -R^{\theta_p} & + \sum_{i=1}^{m} \lambda_i \left(\sum_{x_{t_k}\in D} ReLU\big(-\boldsymbol{\Phi}_{i,r_i}(x_{t_k})\big) \right) \\ & + \sum_{i=1}^{m} \frac{c}{2} \left(\sum_{x_{t_k}\in D} ReLU\big(-\boldsymbol{\Phi}_{i,r_i}(x_{t_k})\big) \right)^2 \\ & + \zeta \left(\sum_{x_{t_k}\in D} ReLU\big(L_\nu(\hat{x}_{t_{k+1}}) - L_\nu(x_{t_k}) + \beta L_\nu(x_{t_k})\big) \right) \\ & + \frac{c}{2} \left(\sum_{x_{t_k}\in D} ReLU\big(L_\nu(\hat{x}_{t_{k+1}}) - L_\nu(x_{t_k}) + \beta L_\nu(x_{t_k})\big) \right)^2. \end{aligned} \tag{23}$$

where $\{\hat{x}_{t_{k+1}}, \hat{x}_{t_{k+2}}, \dots, \hat{x}_{t_{k+r_i}}\}$ are predicted future states, output by the NODE model, following the controller π_{θ_p}. To reduce the computational burden of the typical augmented Lagrangian method, which involves repeated minimization of $\mathcal{L}_c^{surr}$ over θ_p by RL with λ_i, ζ, and c that are also updated, we use a different timescale method to update action-value networks (including the Lyapunov network) and the policy network with different learning rates. Delayed update [12] is also applied, which is shown in Algorithm 2.

With respect to the backup controller π_b which is parameterized by θ_b, by applying SAC, the objective function of Problem (12) is rewritten as:

$$-R^{\theta_b} = -\mathbb{E}_{x_{t_k}\sim\mathcal{D},\xi\sim\mathcal{N}}\big[\min_{j=1,2} Q_{\phi_j}(x_{t_k}, \tilde{u}_{\theta_b}(x_{t_k}, \xi)) - \alpha_b \log \pi_{\theta_b}(\tilde{u}_{\theta_b}(x_{t_k}, \xi)|x_{t_k})\big].$$

The loss function to update the coefficient α_b is:

$$J_{\alpha_b}(\alpha_b) = -\alpha_b \times \mathbb{E}_{x_{t_k}\sim D,\xi\sim\mathcal{N}}\left[\log \pi_{\theta_b}(\tilde{u}_{\theta_b}(x_{t_k},\xi)|x_{t_k}) + \mathcal{H}_b\right] \tag{24}$$

where $\mathcal{H}_b$ can be tuned to encourage exploration. Similarly, a surrogate augmented Lagrangian function for updating θ_b can be given:

$$\begin{aligned}\mathcal{L}^{surr}_{c_b}(\theta_b,\lambda_{b,i}) = -R^{\theta_b} &+ \sum_{i=1}^{m}\lambda_{b,i}\left(\sum_{x_{t_k}\in D} ReLU\left(-\Phi^b_{i,r_i}(x_{t_k})\right)\right)\\ &+ \sum_{i=1}^{m}\frac{c_b}{2}\left(\sum_{x_{t_k}\in D} ReLU\left(-\Phi^b_{i,r_i}(x_{t_k})\right)\right)^2.\end{aligned} \tag{25}$$

With these elements, the complete algorithm NLBAC, when pre-defined CBFs are available, is provided in Algorithm 2.

Algorithm 2 Neural Ordinary Differential Equations-based Lyapunov-Barrier Actor-Critic (NLBAC) with Pre-defined CBFs

Input: Maximum number of iterations N, initialized NODE parameterized by ψ, action-value networks Q_{ϕ_i}, Lyapunov network L_ν, primary controller network π_{θ_p}, α_p, Lagrangian multipliers λ_i and ζ, backup controller network π_{θ_b}, α_b, Lagrangian multipliers $\lambda_{b,i}$, penalty parameter c and c_b, factor $\rho_c \in (1,\infty)$, learning rates $\eta_1 > \eta_2$, η_3, delay steps n_m, n_L, and n_b.

for $n = 0, 1, \ldots, N$ **do**
  if $n \bmod n_m = 0$ **then**
    $\psi \leftarrow \psi - \eta_3 \nabla_\psi \ell$
  end if $\nu \leftarrow \nu - \eta_1 \nabla_\nu J_L(L_\nu)$;
  $\phi_i \leftarrow \phi_i - \eta_1 \nabla_{\phi_i} J_{Q^{\pi_p}}(Q_{\phi_i})$ for $i \in \{1,2\}$;
  $\nu \leftarrow \nu - \eta_1 \nabla_\nu J_{L^{\pi_p}}(L_\nu)$
  $\theta_p \leftarrow \theta_p - \eta_2 \nabla_{\theta_p} \mathcal{L}^{surr}_c(\theta_p,\lambda_i,\zeta)$;
  $\alpha_p \leftarrow \alpha_p - \eta_2 \nabla_{\alpha_p} J_{\alpha_p}(\alpha_p)$
  if $n \bmod n_L = 0$ **then**
    $\lambda_i \leftarrow \lambda_i + c\left(\sum_{x_{t_k}\in D} ReLU\left(-\Phi_{i,r_i}(x_{t_k})\right)\right)$;
    $\zeta \leftarrow \zeta + c\left(\sum_{x_{t_k}\in D} ReLU\left(L_\nu(\hat{x}_{t_{k+1}}) - L_\nu(x_{t_k}) + \beta L_\nu(x_{t_k})\right)\right)$
  end if
  $c \leftarrow \rho_c \times c$
  if $n \bmod n_b = 0$ **then**
    $\theta_b \leftarrow \theta_b - \eta_2 \nabla_{\theta_b} \mathcal{L}^{surr}_{c_b}(\theta_b,\lambda_{b,i})$;
    $\alpha_b \leftarrow \alpha_b - \eta_2 \nabla_{\alpha_b} J_{\alpha_b}(\alpha_b)$
    if $n \bmod (n_b \times n_L) = 0$ **then**
      $\lambda_{b,i} \leftarrow \lambda_{b,i} + c_b\left(\sum_{x_{t_k}\in D} ReLU\left(-\Phi^b_{i,r_i}(x_{t_k})\right)\right)$
    end if
    $c_b \leftarrow \rho_c \times c_b$
  end if
  if Backup controller should be used according to the conditions specific to the task **then**
    $u_{t_k} \sim \pi_{\theta_b}(u_{t_k}|x_{t_k})$ and apply u_{t_k}
  else
    $u_{t_k} \sim \pi_\theta(u_{t_k}|x_{t_k})$ and apply u_{t_k}, and store the pair $(x_{t_k}, u_{t_k}, r_{t_k}, c_{t_k}, x_{t_{k+1}})$ in the replay buffer
  end if
end for

Output: π_{θ_p}, π_{θ_b}, $Q_{\phi_i}, i = 1, 2$, and L_ν

Then we consider the case where the learned barrier certificate is used. With a slight abuse of notation, denote the parameter of the RL-based controller by θ, and define the parameters of the action-value networks and the Lyapunov network similarly to the previous part where pre-defined CBFs are used. The barrier certificate H^π is parameterized by ω as H_ω, and in this implementation, H_ω is designed as an action-value function and trained by minimizing the TD error in SAC for better sample efficiency. Therefore, the constraint based on the barrier certificate in Problem (17) can be rewritten as $\sum_{x_{t_k}\in D} ReLU\left(-H_\omega(\hat{x}_{t_{k+1}},\hat{u}_{t_{k+1}}) + H_\omega(x_{t_k},u_{t_k}) - \epsilon H_\omega(x_{t_k},u_{t_k})\right) = 0$, where $\hat{x}_{t_{k+1}}$ is the next state predicted by the NODE given x_{t_k} and $u_{t_k} = \pi_\theta(x_{t_k})$, and $\hat{u}_{t_{k+1}} = \pi_\theta(\hat{x}_{t_{k+1}})$. Here, the loss function $J_{H^\pi}(H_\omega)$ to update H_ω is:

$$\begin{aligned}J_{H^\pi}(H_\omega) = \mathbb{E}_{(x_{t_k},u_{t_k},b_{t_k},x_{t_{k+1}})\sim D}\Big[\big[b_{t_k} &+ \gamma H_{targ,\omega}(x_{t_{k+1}},u_{t_{k+1}})\\ &- H_\omega(x_{t_k},u_{t_k})\big]^2\Big],\end{aligned} \tag{26}$$

where $u_{t_{k+1}} = \pi_\theta(x_{t_{k+1}})$. The objective function of Problem (17) is given as:

$$\begin{aligned}-R^\theta_{\text{certificate}} = -\mathbb{E}_{x_{t_k}\sim D,\xi\sim\mathcal{N}}\big[&\min_{j=1,2} Q_{\phi_j}(x_{t_k},\tilde{u}_\theta(x_{t_k},\xi))\\ &- \alpha \log \pi_\theta(\tilde{u}_\theta(x_{t_k},\xi)|x_{t_k})\big],\end{aligned} \tag{27}$$

and a surrogate augmented Lagrangian function is further provided as follows:

$$\begin{aligned}&\mathcal{L}^{surr}_{c,\text{certificate}}(\theta,\lambda_i,\zeta) = -R^\theta_{\text{certificate}}\\ &+ \sum_{i=1}^{m}\lambda_i\left(\sum_{x_{t_k}\in D} ReLU\left(-H_\omega(\hat{x}_{t_{k+1}},\hat{u}_{t_{k+1}}) + H_\omega(x_{t_k},u_{t_k}) - \epsilon H_\omega(x_{t_k},u_{t_k})\right)\right)\\ &+ \sum_{i=1}^{m}\frac{c}{2}\left(\sum_{x_{t_k}\in D} ReLU\left(-H_\omega(\hat{x}_{t_{k+1}},\hat{u}_{t_{k+1}}) + H_\omega(x_{t_k},u_{t_k}) - \epsilon H_\omega(x_{t_k},u_{t_k})\right)\right)2\\ &+ \zeta\left(\sum_{x_{t_k}\in D} ReLU\left(L_\nu(\hat{x}_{t_{k+1}}) - L_\nu(x_{t_k}) + \beta L_\nu(x_{t_k})\right)\right)\\ &+ \frac{c}{2}\left(\sum_{x_{t_k}\in D} ReLU\left(L_\nu(\hat{x}_{t_{k+1}}) - L_\nu(x_{t_k}) + \beta L_\nu(x_{t_k})\right)\right)^2.\end{aligned} \tag{28}$$

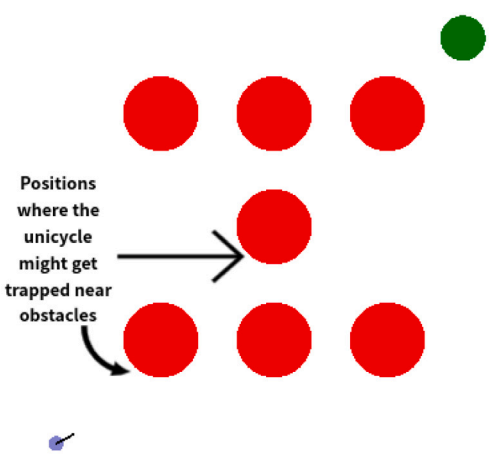
Then, the algorithm NLBAC, when a learned barrier certificate is used, is shown in Algorithm 3.

Algorithm 3 Neural Ordinary Differential Equations-based Lyapunov-Barrier Actor-Critic (NLBAC) with Learned Barrier Certificate

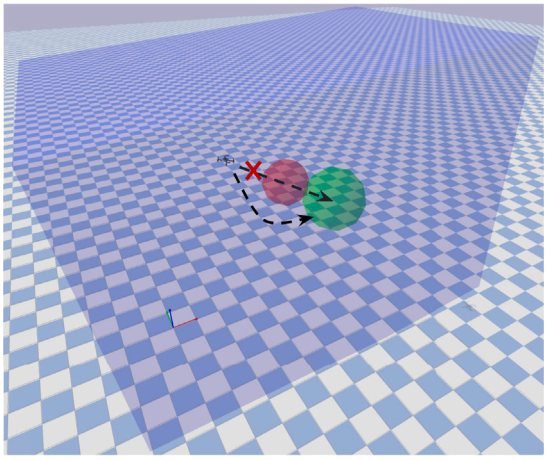
Input: Maximum number of iterations N, initialized NODE parameterized by ψ, action-value networks Q_{ϕ_i}, barrier certificate H_ω, Lyapunov network L_ν, primary controller network π_θ, α, Lagrangian multipliers λ_i and ζ, penalty parameter c, factor $\rho_c \in (1,\infty)$, learning rates $\eta_1 > \eta_2$, η_3, delay steps n_m, n_L.

for $n = 0, 1, \ldots, N$ **do**
  if $n \bmod n_m = 0$ **then**
    $\psi \leftarrow \psi - \eta_3 \nabla_\psi \ell$
  end if $\nu \leftarrow \nu - \eta_1 \nabla_\nu J_L(L_\nu)$;
  $\phi_i \leftarrow \phi_i - \eta_1 \nabla_{\phi_i} J_{Q^\pi}(Q_{\phi_i})$ for $i \in \{1,2\}$;
  $\nu \leftarrow \nu - \eta_1 \nabla_\nu J_{L^\pi}(L_\nu)$;
  $\omega \leftarrow \omega - \eta_1 \nabla_\omega J_{H^\pi}(H_\omega)$
  $\theta \leftarrow \theta - \eta_2 \nabla_\theta \mathcal{L}^{surr}_{c,\text{certificate}}(\theta,\lambda_i,\zeta)$;
  $\alpha \leftarrow \alpha - \eta_2 \nabla_\alpha J_\alpha(\alpha)$
  if $n \bmod n_L = 0$ **then**
    $\lambda_i \leftarrow \lambda_i + c\left(\sum_{x_{t_k}\in D} ReLU\left(-H_\omega(\hat{x}_{t_{k+1}},\hat{u}_{t_{k+1}}) + H_\omega(x_{t_k},u_{t_k}) - \epsilon H_\omega(x_{t_k},u_{t_k})\right)\right)$;
    $\zeta \leftarrow \zeta + c\left(\sum_{x_{t_k}\in D} ReLU\left(L_\nu(\hat{x}_{t_{k+1}}) - L_\nu(x_{t_k}) + \beta L_\nu(x_{t_k})\right)\right)$
  end if
  $c \leftarrow \rho_c \times c$
  $u_{t_k} \sim \pi_\theta(u_{t_k}|x_{t_k})$ and apply u_{t_k}, and store the pair $(x_{t_k}, u_{t_k}, r_{t_k}, c_{t_k}, b_{t_k}, x_{t_{k+1}})$ in the replay buffer
end for

Output: π_θ, $Q_{\phi_i}, i = 1, 2$, and L_ν



(a) The unicycle environment. The blue point is the controlled unicycle, the green circle is the destination (desired state), and the red circles are obstacles to avoid. Some positions where the unicycle can get trapped due to the infeasibility are shown



(b) The quadrotor environment. The green ball represents the destination that the controlled quadrotor should reach, while the red ball is an obstacle that needs to be avoided. Additionally, the quadrotor is required to remain within the blue rectangular area for safety.

Fig. 2. "Unicycle" and "Quadrotor" tasks.

6. Experiments

In this section, we evaluate the proposed NLBAC algorithm (with either pre-defined CBFs or neural barrier certificates) on four tasks to answer the following research questions:

1. Does the NLBAC algorithm result in a decreasing number of safety violations during training and achieve zero or near-zero safety violations by the end of training?
2. Does the NLBAC algorithm effectively drive the controlled agent towards the desired state?
3. Does the NLBAC algorithm exhibit better sample efficiency compared to other baseline methods?

Baselines. Baselines compared here are CMDP-based or developed to solve state-wise constraints, and range from model-free to model-based approaches. CMDP baselines include MBPPO-Lagrangian [55], CPO [6], PPO-Lagrangian and TRPO-Lagrangian [56]. SAC-RCBF [20] is the baseline for hard safety constraints with CBFs. Additionally, LAC [36] is used as the model-free baseline combining CLF for stability. These baselines encompass diverse approaches, and are representative works in the field of constrained RL.

Environments. We conduct experiments on four widely-used tasks[2] for safe model-based RL studies [19,20,23,57,58]. These tasks are carefully selected and ordered to progressively demonstrate the capabilities of the proposed NLBAC algorithm. The progression moves from lower-dimensional systems (e.g., Unicycle) to higher-dimensional systems (e.g., Car Following), from static obstacle environments (e.g., Unicycle) to dynamic obstacle environments (e.g., Car Following and PVTOL), and from systems with lower relative degrees (e.g., Unicycle and Car Following) to those with higher relative degrees (e.g., PVTOL). Additionally, the tasks transition from simpler gym-based environments to more sophisticated physics-engine-based scenarios (e.g., Quadrotor), providing a comprehensive evaluation of the algorithm's adaptability and effectiveness. Details of these four tasks are provided below:

Unicycle: This task is designed to evaluate the proposed algorithm in a simpler environment with a low-dimensional state space and pre-defined CBFs of relative degree one. Both versions of the algorithm (with pre-defined CBFs and with neural barrier certificates) are tested with static obstacles. In this experiment setup, a unicycle is tasked with reaching the designated location, i.e., the destination, while avoiding collisions with obstacles, as shown in Fig. 2(a). The real dynamics of the system is:

$$\dot{x}_{t_k} = \begin{bmatrix} \cos(\theta_{t_k}) & 0 \\ \sin(\theta_{t_k}) & 0 \\ 0 & 1.0 \end{bmatrix} (u_{t_k} + u_{d,t_k}).$$

Here $x_{t_k} = [x_{1t_k}, x_{2t_k}, \theta_{t_k}]^T$ is the state where x_{1t_k} and x_{2t_k} represent the X-coordinate and Y-coordinate of the unicycle and θ_{t_k} is the orientation at the timestep t_k. The control signal $u_{t_k} = [v_{t_k}, \omega_{t_k}]^T$ comprises linear velocity v_{t_k} and angular velocity ω_{t_k}. The time interval used in this experiment is 0.02 s. $u_{d,t_k} = -0.1[\cos(\theta_{t_k}), 0]^T$. As to NODE-based models, the input of network $\hat{f}$ is the state with the dimension of 3, and the output dimension is 3; the input of network $\hat{g}$ is the state with the dimension of 3, and the output dimension is 6 (which is then be resized as $[3, 2]$). Then, a point at a distance l_p ahead of the unicycle is considered to establish safety constraints for collision-free navigation, and the function $p : \mathbb{R}^3 \to \mathbb{R}^2$ is defined as:

$$p(x_{t_k}) = \begin{bmatrix} x_{1t_k} \\ x_{2t_k} \end{bmatrix} + l_p \begin{bmatrix} \cos(\theta_{t_k}) \\ \sin(\theta_{t_k}) \end{bmatrix}.$$

The reward signal is formulated as $-K_1(v_t - v_s)^2 + K_2 \Delta d$, where v_s represents the predefined velocity, Δd denotes the reduction in the distance between the unicycle and the destination in two consecutive timesteps, and K_1 and K_2 are coefficients set to 0.1 and 30, respectively. An additional reward of 500 will be given if the unicycle reaches a small neighborhood of the destination. The cost signal is given by $\left\| p(x_{t_{k+1}}) - p(x_{\text{desired}}) \right\|$ where $p(x_{\text{desired}}) = [x_{1\text{desired}}, x_{2\text{desired}}]^T$ denotes the position of the desired location. Pre-defined CBFs, if provided, are $h_i(x_{t_k}) = \frac{1}{2}\left((p(x_{t_k}) - p_{\text{obs}_i})^2 - \delta^2\right)$ where p_{obs_i} represents the position of the ith obstacle, and δ is the minimum required distance between the unicycle and obstacles. Hence, the relative degree is 1 and the planning horizon for NODEs predictions is 1. When no pre-defined CBFs are available, the neural barrier certificate will be learned jointly with the controller by using additional barrier signals with $D = -20$.

When pre-defined CBFs are used, the inability to satisfy safety and stability constraints simultaneously at state $x_{t_k} \in \mathcal{X}_{\text{infeasible}}$ can cause the unicycle to become trapped near obstacles. In such cases, the

[2] Codes are on our GitHub page: https://github.com/LiqunZhao/Neural-ordinary-differential-equations-based-Lyapunov-Barrier-Actor-Critic-NLBAC.

backup controller takes over from the primary controller. The primary controller is reinstated when the unicycle moves a long distance away from the trapped position, or when the predefined time threshold for using the backup controller is exceeded.

Car Following: This task is designed to evaluate the proposed algorithm in an environment with a higher-dimensional state space and pre-defined CBFs of relative degree two. The version of the algorithm with pre-defined CBFs is tested with dynamic obstacles. This environment simulates a chain of five cars following each other on a straight road. The objective is to control the acceleration of the 4th car to maintain a desired distance from the 3rd car while avoiding collisions with other cars. The real dynamics of all cars except the 4th one is given by:

$$\dot{x}_{t_k,i} = \begin{bmatrix} v_{t_k,i} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 + d_i \end{bmatrix} a_{t_k,i} \qquad \forall i \in \{1,2,3,5\}.$$

Each state of the system is denoted as $x_{t_k,i} = [p_{t_k,i}, v_{t_k,i}]^T$, indicating the position $p_{t_k,i}$ and velocity $v_{t_k,i}$ of the ith car at the timestep t_k, $d_i = 0.1$. The time interval used in this experiment is 0.02 s. The predefined velocity of the 1st car is $v_s - 4\sin(t_k)$, where $v_s = 3.0$. Its acceleration is given as $a_{t_k,1} = k_v(v_s - 4\sin(t_k) - v_{t_k,1})$ where $k_v = 4.0$. Accelerations of Car 2 and 3 are given by:

$$a_{t_k,i} = \begin{cases} k_v(v_s - v_{t_k,i}) - k_b(p_{t_k,i-1} - p_{t_k,i}), & |p_{t_k,i-1} - p_{t_k,i}| < 6.5 \\ k_v(v_s - v_{t_k,i}), & otherwise, \end{cases}$$

where $k_b = 20.0$ and $i = 2, 3$. The acceleration of the 5th car is:

$$a_{t_k,5} = \begin{cases} k_v(v_s - v_{t_k,5}) - k_b(p_{t_k,3} - p_{t_k,5}), & |p_{t_k,3} - p_{t_k,5}| < 13.0 \\ k_v(v_s - v_{t_k,5}), & otherwise. \end{cases}$$

The model of the 4th car is as follows:

$$\dot{x}_{t_k,4} = \begin{bmatrix} v_{t_k,4} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1.0 \end{bmatrix} u_{t_k},$$

where u_{t_k} is the acceleration of the 4th car, and also the control signal generated by the controller at the timestep t_k.

The reward signal is defined to minimize the overall control effort, and an additional reward of 2.0 is granted during timesteps when $d_{t_k} = p_{t_k,3} - p_{t_k,4}$, which is the distance between the 3rd and 4th car, falls within $[9.0, 10.0]$. This range is defined as the desired region for d_{t_k}. Thus, the cost signal is determined as $\left\| d_{t_{k+1}} - d_{\text{desired}} \right\|$, where $d_{\text{desired}} = 9.5$. CBFs are defined as $h_1(x_{t_k}) = p_{t_k,3} - p_{t_k,4} - \delta$ and $h_2(x_{t_k}) = p_{t_k,4} - p_{t_k,5} - \delta$, with δ being the minimum required distance between the cars. Hence, the relative degree is 2 and the planning horizon for making predictions using NODE is 2.

When we use NODEs to model this system, we assume that we do not have the priori information that this system is control-affine. The input of network $\mathcal{F}$ is $\left(t_k, \hat{x}_{t_k}, u_{t_k}\right)$ with the dimension of 12, and the output dimension is 10. When a safety constraint is violated if two types of constraints cannot be satisfied simultaneously, the 4th car might be in close proximity to the 5th car in order to make d_{t_k} be within $[9.0, 10.0]$. In such cases, the backup controller is activated. The primary controller is reactivated when the 4th car moves beyond the dangerous area, namely out of the vicinity of the 5th car, or when the predetermined time threshold for utilizing the backup controller is exceeded.

Planar Vertical Take-Off and Landing (PVTOL): This task evaluates the proposed algorithm in an environment with both static and dynamic obstacles, more complex system dynamics, and pre-defined CBFs of relative degree three. Both versions of the algorithm (with pre-defined CBFs and with neural barrier certificates) are tested in scenarios with static obstacles. In this experiment, a quadcopter is required to reach a destination while avoiding obstacles and keeping within a specified range along the y-axis and a specific distance from a safety pilot along the x-axis. The real dynamics of the system is:

$$\dot{x}_{t_k} = \begin{bmatrix} v_{1t_k} \\ v_{2t_k} \\ 0 \\ -\sin(\theta_{t_k}) \times f_{t_k} \\ \cos(\theta_{t_k}) \times f_{t_k} - 1.0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} u_{t_k}.$$

Here $x_{t_k} = [x_{1t_k}, x_{2t_k}, \theta_{t_k}, v_{1t_k}, v_{2t_k}, f_{t_k}]^T$ is the state where x_{1t_k} and x_{2t_k} represent the X-coordinate and Y-coordinate of the quadcopter and θ_{t_k} is the orientation at the timestep t_k. v_{1t_k}, v_{2t_k} are velocities along X and Y axis, and f_{t_k} is the thrust. The control signal $u_{t_k} = [u_{ft_k}, \omega_{t_k}]^T$ includes the derivative of the thrust and angular velocity. The time interval used in this experiment is 0.02 s.

The reward signal is defined to minimize the distance from the destination, and an additional reward of 1500 will be given if the quadcopter reaches a small neighborhood of the destination. The cost signal is the current distance from the destination, and the safety operator tracks the quadcopter via a proportional controller along the x-axis. When pre-defined CBFs are used, collision avoidance and confinement within specific ranges along the x-axis and y-axis are ensured by those fixed CBFs, following a similar approach to the previous two environments. The relative degree, and therefore the planning horizon for NODEs predictions, is 3. When no pre-defined CBFs are available, the neural barrier certificate will be learned jointly with the controller by using additional barrier signals with $D = -0.1$.

For NODE-based models, the input of network $\hat{f}$ is the state of dimension 6, and the output dimension is 6; the input of network $\hat{g}$ is the state with the dimension of 6, and the output dimension is 12 (which is then be resized as $[6, 2]$). The conditions for activating the backup controller resemble those in the previous two tasks.

Quadrotor: This task evaluates the proposed algorithm in a sophisticated physics-engine-based environment. The version of the algorithm using neural barrier certificates is tested. This is a stabilization task which is modified from safety-control-gym [58]. In this task, the 2D quadrotor needs to approach towards and finally arrive at a desired destination, while keeping both the vertical and horizontal positions within a pre-defined range, and avoiding the collision with an obstacle shown in Fig. 2(b). A more detailed description, including the state, action, and system dynamics, can be found in [58]. The reward signal is designed to minimize the distance between the current position and the goal position, which is also defined to be the cost signal. An additional reward of 250 is given when the quadrotor reaches the goal. In this task, no pre-defined CBFs are provided, and therefore, a neural barrier certificate will be learned jointly with the control policy. We define the barrier signal as $D_1 = -1.0$ when the quadrotor's position is not within the pre-defined range, and $D_2 = -10.0$ when the quadrotor collides with the obstacle.

For the NODE used to model the system dynamics, we also assume that we do not have any prior information. Different from previous tasks, here states and actions are normalized before being used as the input of the NODE, and outputs of the NODE are denormalized before being used as predictions. The input dimension of the NODE is 8, and the output dimension is 6.

Compute Resources. Our code runs on Intel Platinum 8628 (Cascade Lake), 2.90 GHz CPUs, and NVIDIA Tesla V100 GPUs.

Experimental results and comparisons. Simulations were conducted for the tasks where pre-defined CBFs are available ("Unicycle", "Car Following", and "PVTOL"); the results are depicted in Fig. 3. Since in these tasks, high rewards are granted when the system approaches and achieves the desired state (destination), we use the cumulative reward as a measure, where a higher cumulative reward can indicate better convergence to the destination. In the experiments, our framework demonstrates superior performance, yielding consistently higher cumulative rewards compared to baselines, which suggests that the controlled agents, under our framework, can more efficiently approach

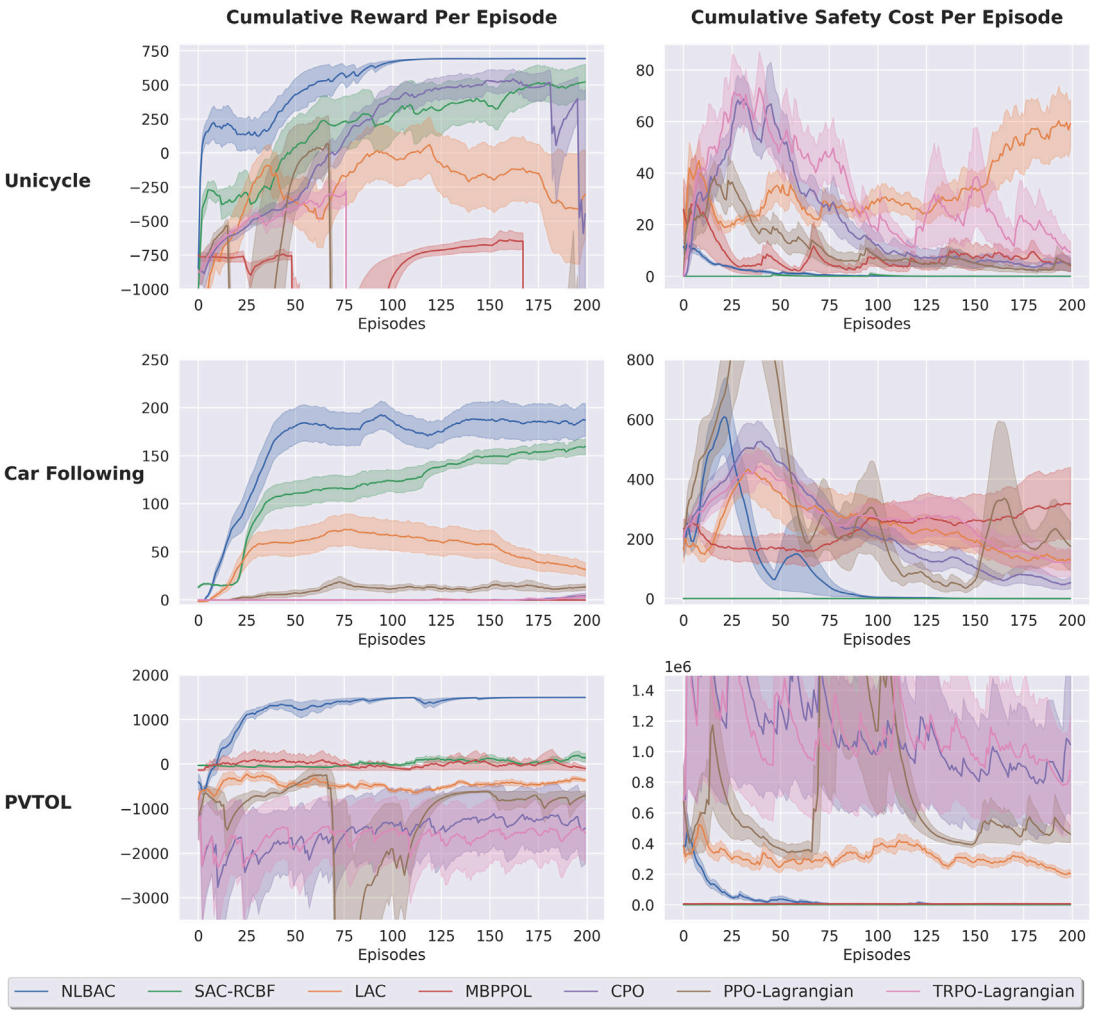


Fig. 3. The cumulative reward and cumulative safety cost of each episode in "Unicycle", "Car Following" and "PVTOL" tasks are compared between the proposed NLBAC (with pre-defined CBFs and in blue) and baselines. Each curve illustrates the average across ten experiments employing different random seeds, with the shaded area denoting the standard error.

and reach their desired states (destinations) in much fewer episodes. Moreover, the fluctuations observed in the cumulative rewards are notably smaller than those exhibited by other baselines. This indicates that the systems can quickly recover to their previous high reward levels after experiencing deterioration. This is attributed to the CLF, which explicitly makes the algorithm goal-oriented. Also, compared to the model-free LAC methods with Lyapunov certificates, our algorithm utilizes NODEs for effective modeling, and utilizes state-wise CLF constraints instead of expectation-based constraints, resulting in better performance.

Concerning safety, our framework exhibits much fewer violations compared to other baselines except for SAC-RCBF. By using pre-defined CBFs to enforce state-wise safety constraints, our methods result in substantially fewer safety violations compared to CMDP-based model-free baselines. The model-based baseline MBPPO-Lagrangian did not perform well in our experiments. This could be due to its reliance on CMDP-based constraints and extensive use of imaginary rollouts to update parameters, which may introduce error aggregation. When compared to SAC-RCBF, our NLBAC framework achieves higher cumulative rewards with the assistance of CLF, ultimately reaching zero or near-zero safety violations without requiring a good nominal model. Additionally, our framework avoids the need to solve a QP, making it directly applicable to nonconvex settings, and enabling the integration of techniques like neural barrier certificates.

Furthermore, our algorithm requires the fewest interactions with the environment to achieve a good performance with regard to reward and safety, which means it has higher sample efficiency. This suggests its potential applicability in scenarios where data availability may be limited.

We also conduct evaluations of NLBAC using neural barrier certificates in the "Unicycle", "PVTOL" and "Quadrotor" tasks, with the results presented in Fig. 4. Similarly, even without good nominal models or pre-defined CBFs, NLBAC with neural barrier certificate achieves satisfactory performance compared to other baselines. However, since the neural barrier certificate is trained jointly with the control policy from scratch, while the fixed pre-defined CBFs may be well-defined, more safety violations may be observed at the beginning of the training process if a neural barrier certificate is used.

To better show the comparisons between different algorithms, here we present some statistics in Table 1. These statistics are obtained by calculating the average value of the cumulative rewards and cumulative safety costs of the last 50 episodes run by each algorithm, and the result shows that our method achieves the highest average cumulative rewards, and zero or near-zero average cumulative safety costs.

To provide a clearer understanding of the computational complexity of the proposed algorithm, we calculated the training times of NLBAC and the baseline algorithms across all four tasks, ensuring that each algorithm was trained for the same maximum number of episodes. For example, in the Unicycle, Car Following, and PVTOL tasks, every algorithm was trained for 200 episodes, while in the Quadrotor task, each algorithm was trained for 1100 episodes. These statistics are

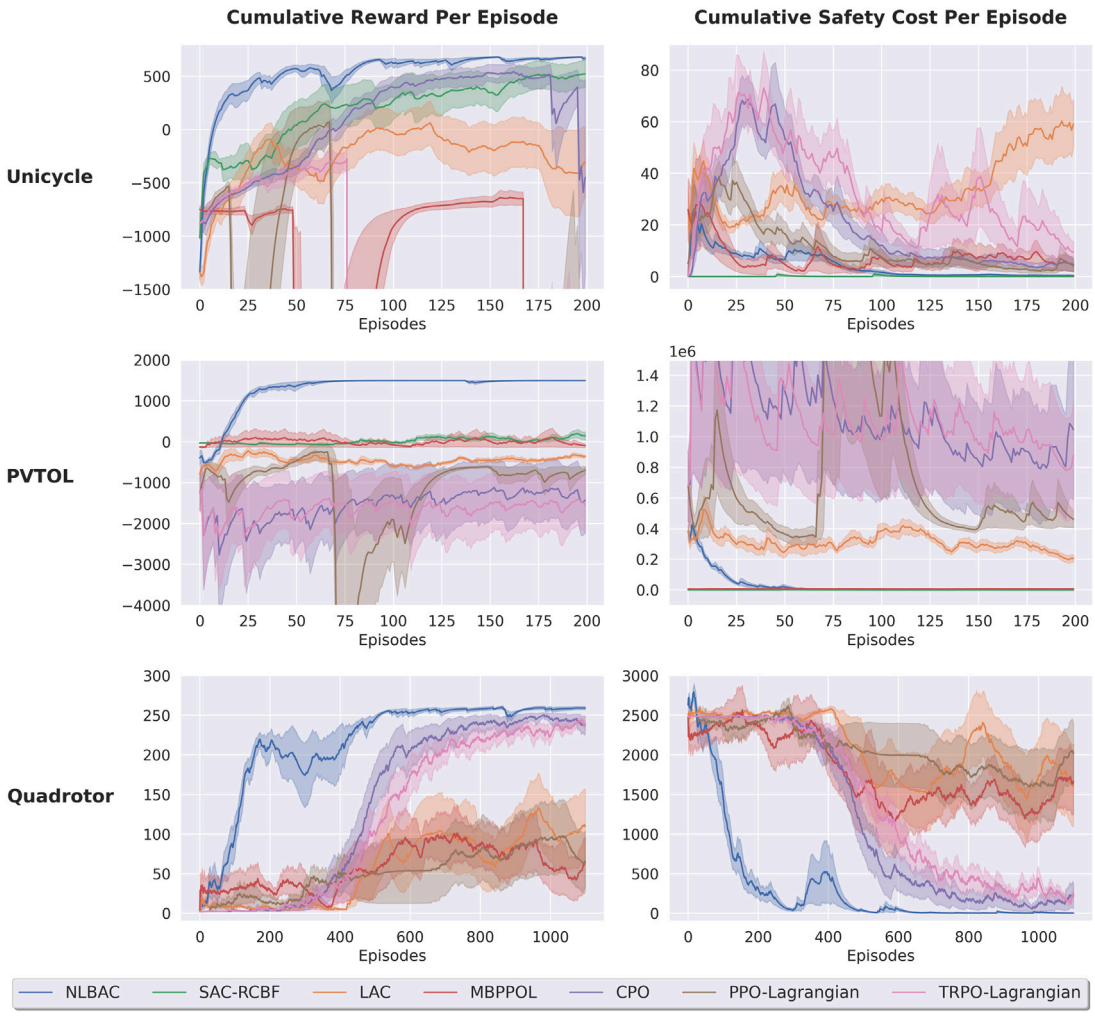


Fig. 4. The cumulative reward and cumulative safety cost of each episode in "Unicycle", "PVTOL" and "Quadrotor" tasks are compared between the proposed NLBAC (with neural barrier certificate and in blue) and baselines. For "Unicycle" and "PVTOL", each curve illustrates the average across ten experiments employing different random seeds, with the shaded area denoting the standard error. For "Quadrotor", each curve illustrates the average across five experiments employing different random seeds, and the baseline SAC-RCBF is not performed due to the lack of a nominal model.

Table 1
Average cumulative reward and average cumulative safety cost on tasks.

Environments	NLBAC (ours)		SAC-RCBF		LAC		MBPPOL		CPO		PPO-Lagrangian		TRPO-Lagrangian	
	ACR	ACSC	ACR	ACSC	ACR	ACSC	ACR	ACSC	ACR	ACSC	ACR	ACSC	ACR	ACSC
Unicycle	**692.59** (pre-defined CBFs) **663.71** (learned barrier certificate)	**0.00** 0.51	489.80	**0.00**	−261.06	53.99	−198 574.01	6.61	221.08	4.99	−17 658.89	3.48	−48 140.71	15.72
Car following	**185.31** (pre-defined CBFs)	**0.00**	155.26	**0.00**	38.21	137.52	−0.22	295.46	1.76	63.21	13.50	233.39	1.39	140.53
PVTOL	**1498.47** (pre-defined CBFs) **1498.60** (learned barrier certificate)	**0.66** **0.28**	82.31	4.54	−403.53	259 310.51	−27.09	6309.82	−1286.91	921 072.58	−800.51	508 446.27	−1529.78	939 415.86
Quadrotor	**259.34** (learned barrier certificate)	**0.09**	N.A.	N.A.	109.29	1624.35	90.08	1835.45	239.28	207.69	60.74	2083.70	239.70	204.88

ACR and ACSC mean Average Cumulative Reward and Average Cumulative Safety Cost, respectively. The best results have been marked in bold.
The state spaces $\mathcal{X}$ of "Unicycle", "Car Following" and "PVTOL" tasks are set to be large, and therefore some baselines result in large negative ACRs and substantial ACSCs when the controlled agent is driven significantly far from the desired state and violates the position-based safety constraints severely.

summarized in Table 2. As can be seen, NLBAC consistently demonstrates a competitive balance between computational efficiency and performance. Specifically, compared to SAC-RCBF, NLBAC achieves significantly shorter training times across all tasks, reflecting its superior efficiency. This is especially notable in tasks like PVTOL, where SAC-RCBF required over 26 h, while NLBAC completed training in less than 3 h. This is possibly due to the computational overhead of its differentiable QP layer for safety. While model-free baselines (e.g., PPO-Lagrangian and CPO) require less training time with the same maximum number of episodes, they generally underperform in cumulative rewards and safety costs. Furthermore, NLBAC achieves markedly better safety and performance metrics than MBPPOL and LAC, despite requiring comparable or slightly longer training times, which justifies its computational cost.

Next, the control signals generated by the controller trained with NLBAC are plotted to demonstrate its effectiveness. For this purpose, we chose the Car Following task. As shown in Fig. 5, the applied control signal is always within the allowed boundary of −3 to 3. The distance between the third and fourth cars is roughly maintained within 9 to 10, which is the desired region, while the distances between the third and

Table 2
Running times of different algorithms on tasks.

Environments	NLBAC	SAC-RCBF	LAC	MBPPOL	CPO	PPO-Lagrangian	TRPO-Lagrangian
Unicycle (200 episodes)	2 h 3 min (pre-defined CBFs) 2 h 58 min (learned barrier certificate)	9 h 43 min	1 h 45 min	3 h 1 min	17 min 49 s	20 min 50 s	16 min 56 s
Car following (200 episodes)	2 h 43 min (pre-defined CBFs)	5 h 30 min	49 min	1 h 13 min	6 min 32 s	5 min 50 s	6 min 49 s
PVTOL (200 episodes)	2 h 52 min (pre-defined CBFs) 2 h 16 min (learned barrier certificate)	26 h 53 min	2 h 35 min	1 h 54 min	20 min 17 s	19 min 34 s	21 min 58 s
Quadrotor (1100 episodes)	40 min 8 s (learned barrier certificate)	N.A.	20 min 20 s	2 h 18 min	6 min 11 s	10 min 11 s	6 min 31 s

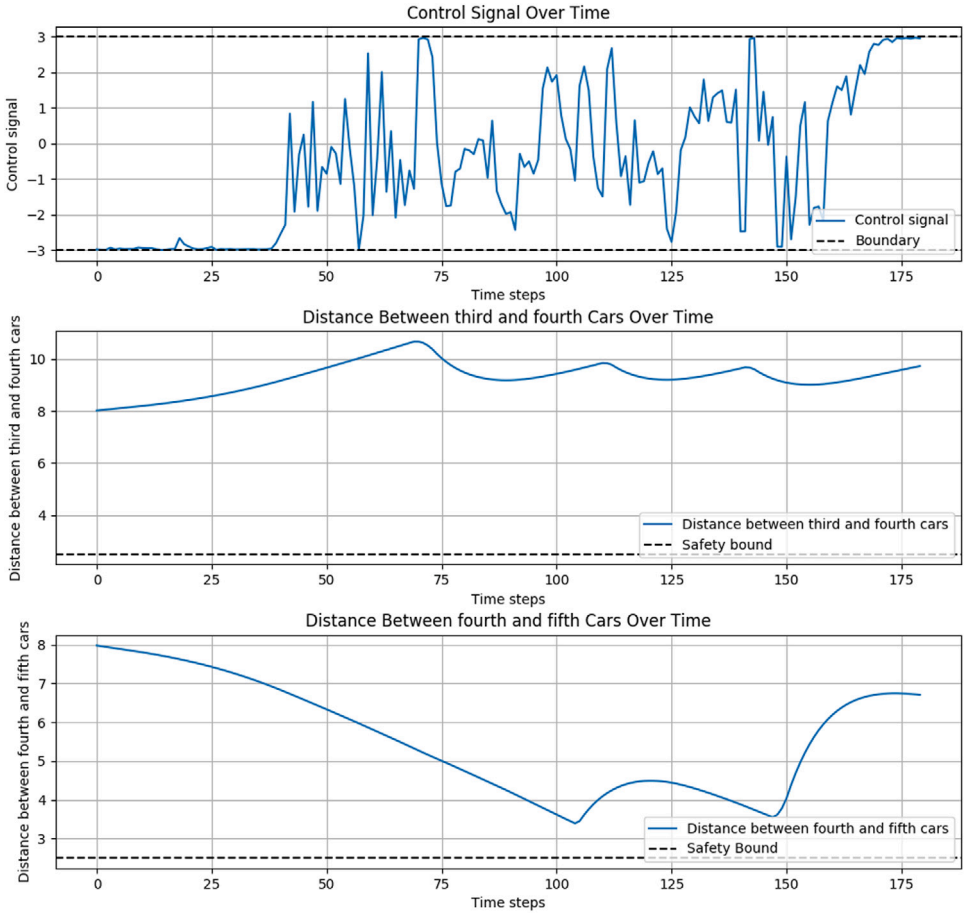


Fig. 5. Control signals generated during deployment for the "Car Following" task.

fourth cars and the fourth and fifth cars remain greater than the minimum safety threshold of 2.5. These results demonstrate the algorithm's capacity to effectively maintain both the desired inter-vehicle distances and safety constraints during deployment.

Sensitivity Analysis. Robustness is key for the successful application of RL algorithms to uncertain safety-critical tasks. Here, we evaluate the robustness of control policies after being trained with NLBAC in the presence of external disturbance and parameter uncertainties. The sensitivity analysis was conducted on two tasks: Unicycle and PVTOL. The following modifications were made to the original settings introduced earlier:

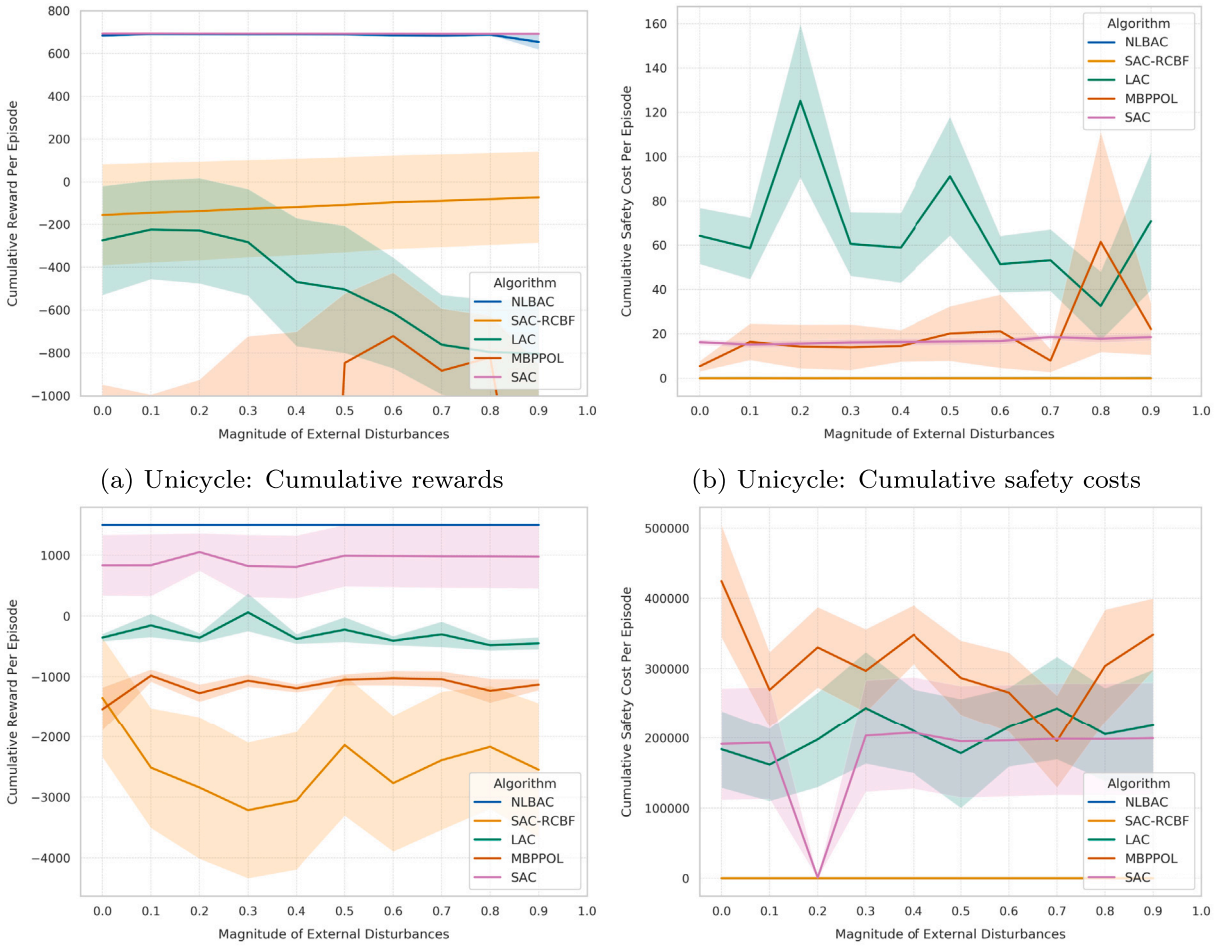
- Unicycle: a disturbance term $u_{d t_k} = -\alpha[\cos(\theta_{t_k}), 0]^T$ is applied, aiming to represent the effect of a sloped surface. The disturbance coefficient α is systematically varied from 0 to 0.9 in increments of 0.1.
- PVTOL: Horizontal wind disturbances are introduced into the system dynamics in the form of a force term F_x, with increments from 0 to 0.9 in steps of 0.1.

For parameter uncertainties:

- Unicycle: a scaling factor β was applied to the linear velocity terms in the dynamics, modifying the equations to:

$$\dot{x}_{t_k} = \begin{bmatrix} \beta\cos(\theta_{t_k}) & 0 \\ \beta\sin(\theta_{t_k}) & 0 \\ 0 & 1.0 \end{bmatrix} (u_{t_k} + u_{d,t_k}),$$

  where $u_{d t_k} = -0.1[\cos(\theta_{t_k}), 0]^T$. The scaling factor β is varied from 0.6 to 1.5 in increments of 0.1.
- PVTOL: Parameter uncertainty is introduced by applying a scaling factor δ to the angular velocity term in the dynamics. The modified control input matrix for angular velocity was scaled by



(a) Unicycle: Cumulative rewards

(b) Unicycle: Cumulative safety costs

(c) PVTOL: Cumulative rewards

(d) PVTOL: Cumulative safety costs

Fig. 6. The effects of external disturbances during deployment on the cumulative reward and cumulative safety cost in the "Unicycle" and "PVTOL" tasks. In the cumulative safety cost plots (right column), the curve representing NLBAC overlaps with that of SAC-RCBF, making only the SAC-RCBF curve visible. In subfigure (d), an upper limit has been applied to standard errors to enhance clarity by reducing excessively large shaded areas.

δ:

$$\dot{x}_{t_k} = \begin{bmatrix} v_{1t_k} \\ v_{2t_k} \\ 0 \\ -\sin(\theta_{t_k}) \times f_{t_k} \\ \cos(\theta_{t_k}) \times f_{t_k} - 1.0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & \delta \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} u_{t_k}.$$

δ is incremented from 0.6 to 1.5 in steps of 0.1, and the system's performance was assessed accordingly.

The performance of the proposed NLBAC algorithm under external disturbances and parameter uncertainties during deployment after training is presented in Figs. 6 and 7, respectively. We include SAC as an additional baseline for comparison.

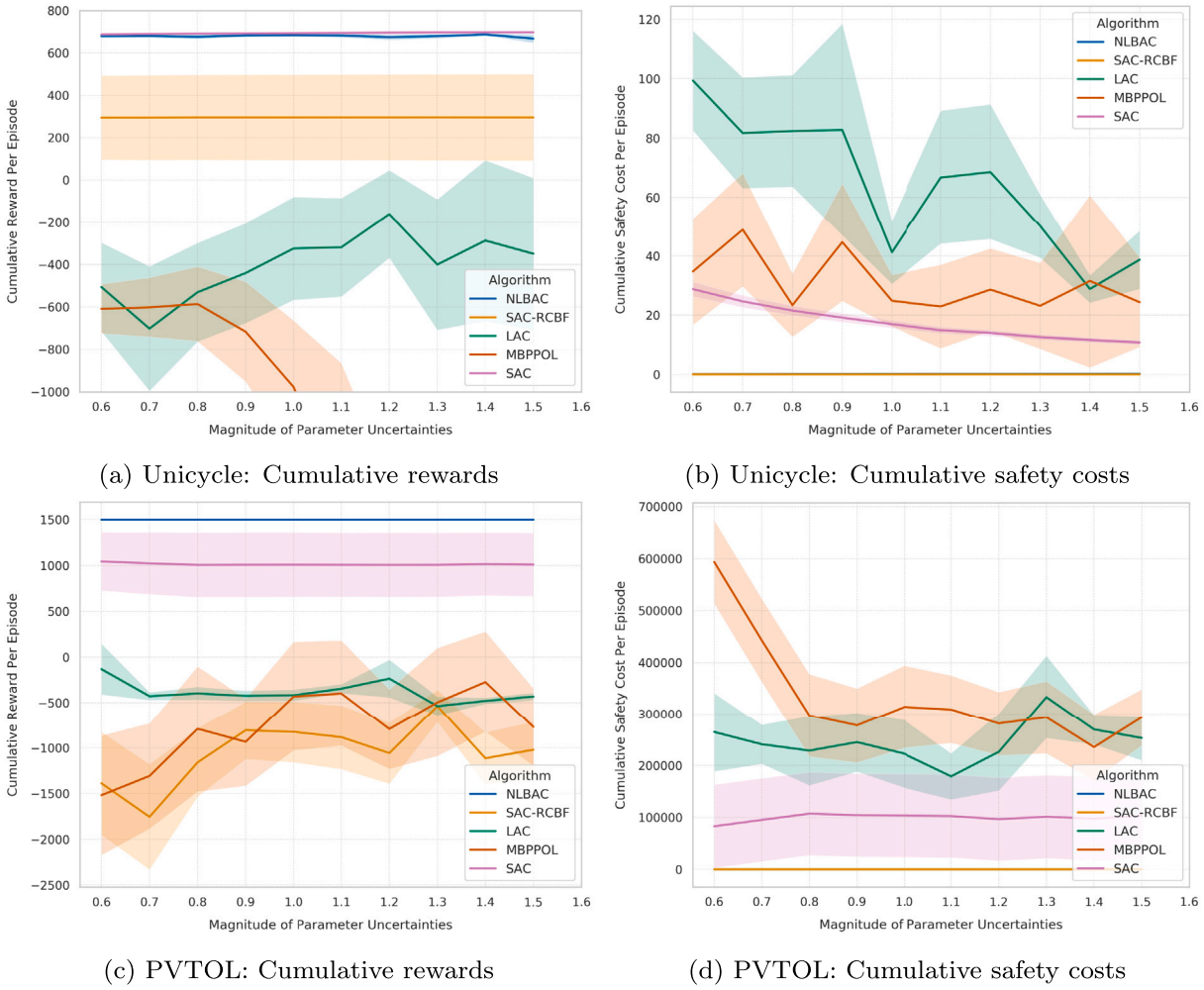
As shown in Fig. 6, during deployment after the training process is complete, the proposed algorithm NLBAC demonstrates robust performance under external disturbances of varying magnitudes. Both cumulative rewards and safety costs of the NLBAC algorithm remain at superior levels compared to other baselines on both tasks, achieving high cumulative rewards while maintaining zero or near-zero cumulative safety violations (the curve for cumulative safety costs of NLBAC overlaps with the curve of SAC-RCBF). Importantly, NLBAC achieves these results without significant fluctuations or performance deterioration.

SAC achieves good cumulative rewards but fails to maintain low cumulative safety costs, rendering it less suitable for safety-critical applications. On the other hand, SAC-RCBF demonstrates robust safety performance during deployment under various external disturbances, maintaining near-zero safety violations. However, it cannot achieve high cumulative rewards, even with prior knowledge of the system dynamics.

Similar to the analysis of external disturbances, as shown in Fig. 7, during deployment, when parameter updates are finalized, NLBAC demonstrates robust performance under parameter uncertainties of varying magnitude. Both the cumulative rewards and safety costs of the NLBAC algorithm remain at favorable levels compared to other baselines. Notably, in the Unicycle task, the cumulative rewards obtained by controllers trained with NLBAC show some noticeable degradation. However, overall, NLBAC achieves strong performance without significant fluctuations, demonstrating robustness to parameter uncertainties during deployment.

While SAC achieves high cumulative rewards, it fails to maintain low cumulative safety costs. Furthermore, MBPPOL exhibits significant performance variation under different system parameters, showing its sensitivity to parameter uncertainties.

These results demonstrate the ability of NLBAC to handle external disturbances and parameter uncertainties during deployment while



(a) Unicycle: Cumulative rewards

(b) Unicycle: Cumulative safety costs

(c) PVTOL: Cumulative rewards

(d) PVTOL: Cumulative safety costs

Fig. 7. The effects of parameter uncertainties during deployment on the cumulative reward and cumulative safety cost in the "Unicycle" and "PVTOL" tasks. In the cumulative safety cost plots (right column), the curve representing NLBAC overlaps with that of SAC-RCBF, making only the SAC-RCBF curve visible. In subfigure (d), an upper limit has been applied to standard errors to enhance clarity by reducing excessively large shaded areas.

achieving reliable performance in terms of both cumulative rewards and cumulative safety costs.

Ablation Studies We also compare results obtained by our NLBAC algorithm with those obtained by using the commonly-used dual ascent update. All other parameters remain consistent for comparison, and the results are depicted in Fig. 8. In the "unicycle" environment, five different fixed learning rates are used. In the "Car Following" and "PVTOL" environments, linear decay and exponential decay are used respectively to dynamically adjust the learning rates for more extensive comparisons. The results indicate that the dual ascent update results in much more safety violations compared to ALM, which eventually achieves high rewards, while introducing the least safety violations throughout most of the training process. This suggests that more sophisticated parameter tuning for the learning rate, including the selection of initial values and decay types, and perhaps other hyperparameters, may be necessary when utilizing the dual ascent update, whereas ALM demonstrates superior performance.

7. Conclusions and future work

This paper presents the NLBAC algorithm, designed to guarantee both state-wise safety and stability for controlled systems by leveraging the Augmented Lagrangian Method and Neural ODEs. The algorithm supports tasks with either pre-defined Control Barrier Functions or learned barrier certificates, and addresses the issue of infeasibility when multiple state-wise constraints are imposed. Experimental results demonstrate that NLBAC achieves higher cumulative rewards, fewer safety violations, and improved sample efficiency compared to baselines.

This research can have potential long-term impact on the relevant fields. The use of neural ODEs enhances model precision, improves adaptability to different discretization steps, and mitigates error propagation, contributing to good system modeling in the field of model-based RL. Additionally, our investigation into the infeasibility of enforcing multiple state-wise constraints simultaneously introduces an effective alternative to composite Lyapunov-Barrier functions, enabling principled constraint handling in RL without requiring additional labeled data. Moreover, our results demonstrate that replacing dual ascent updates with the Augmented Lagrangian Method efficiently reduces constraint violations and improves learning efficiency, making it a valuable and practical tool for constrained RL. These insights collectively contribute to the development of state-wise stable and safe RL algorithms for future applications.

While the algorithm's effectiveness is validated across several tasks, the discrepancies between real dynamics and NODE-based models should be further minimized. Potential solutions include ensemble NODE models, and using Gaussian dynamic models, which can be trained using maximum likelihood. We plan to address this question in future work for solving more diverse tasks.

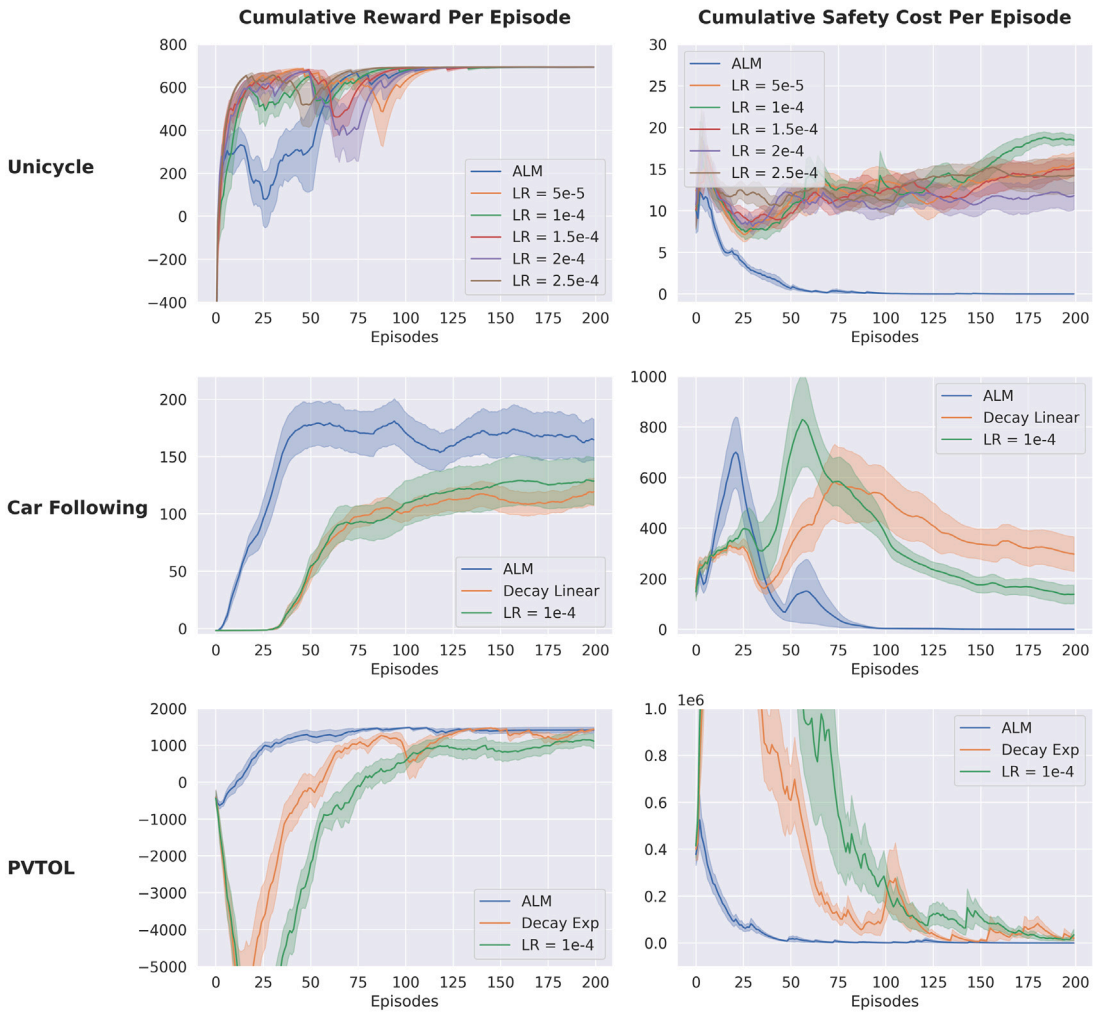


Fig. 8. Results of the ablation study. Each curve shows the average across fifteen experiments using different random seeds.

CRediT authorship contribution statement

Liqun Zhao: Writing – review & editing, Writing – original draft, Software, Methodology, Data curation, Conceptualization. **Keyan Miao:** Software, Methodology. **Hongpeng Cao:** Visualization, Software. **Konstantinos Gatsis:** Writing – review & editing, Validation, Supervision, Conceptualization. **Antonis Papachristodoulou:** Writing – review & editing, Validation, Supervision, Project administration, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Liqun Zhao reports statistical analysis was provided by Technical University of Munich. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

We would like to acknowledge Dr Han Wang for his insightful suggestions. Keyan Miao was funded by the Engineering and Physical Sciences Research Council grant EP/T517811/1. Antonis Papachristodoulou was funded in part by the Engineering and Physical Sciences Research Council grants EP/Y014073/1 and EP/X031470/1.

Data availability

I have shared the link to my code in the attached file.

References

[1] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, A. Knoll, A review of safe reinforcement learning: Methods, theories, and applications, IEEE Trans. Pattern Anal. Mach. Intell. 46 (12) (2024) 11216–11235, http://dx.doi.org/10.1109/TPAMI.2024.3457538.

[2] L. Brunke, M. Greeff, A. Hall, Z. Yuan, S. Zhou, J. Panerati, A. Schoellig, Safe learning in robotics: From learning-based control to safe reinforcement learning, Annu. Rev. Control. Robot. Auton. Syst. 5 (1) (2022) 411–444, http://dx.doi.org/10.1146/annurev-control-042920-020211.

[3] E. Altman, Constrained Markov Decision Processes, Routledge, New York, 2021, http://dx.doi.org/10.1201/9781315140223.

[4] A. Schlaginhaufen, M. Kamgarpour, Identifiability and generalizability in constrained inverse reinforcement learning, in: Proceedings of the 40th International Conference on Machine Learning, Vol. 202, PMLR, Honolulu, Hawaii, USA, 2023, pp. 30224–30251.

[5] Y. Yao, Z. Liu, Z. Cen, P. Huang, T. Zhang, W. Yu, D. Zhao, Gradient shaping for multi-constraint safe reinforcement learning, in: Proceedings of the 6th Annual Learning for Dynamics & Control Conference, Vol. 242, PMLR, 2024, pp. 25–39.

[6] J. Achiam, D. Held, A. Tamar, P. Abbeel, Constrained policy optimization, in: Proceedings of the 34th International Conference on Machine Learning, Vol. 70, PMLR, Sydney, Australia, 2017, pp. 22–31.

[7] T. Yang, J. Rosca, K. Narasimhan, P. Ramadge, Projection-based constrained policy optimization, 2020, http://dx.doi.org/10.48550/arXiv.2010.03152, arxiv preprint, arXiv:2010.03152.

[8] C. Tessler, D. Mankowitz, S. Mannor, Reward constrained policy optimization, 2018, http://dx.doi.org/10.48550/arXiv.1805.11074, arxiv preprint, arXiv:1805.11074.

[9] S. Paternain, L. Chamon, M. Calvo-Fullana, A. Ribeiro, Constrained reinforcement learning has zero duality gap, in: Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., 2019.

[10] S. Gu, J. Kuba, Y. Chen, Y. Du, L. Yang, A. Knoll, Y. Yang, Safe multi-agent reinforcement learning for multi-robot control, Artificial Intelligence 319 (2023) 103905, http://dx.doi.org/10.1016/j.artint.2023.103905.

[11] W. Zhao, T. He, R. Chen, T. Wei, C. Liu, State-wise safe reinforcement learning: A survey, in: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, International Joint Conferences on Artificial Intelligence Organization, Macau, SAR China, 2023, pp. 6814–6822, http://dx.doi.org/10.24963/ijcai.2023/763.

[12] H. Ma, C. Liu, S. Li, S. Zheng, J. Chen, Joint synthesis of safety certificate and safe control policy using constrained reinforcement learning, in: Proceedings of the 4th Annual Learning for Dynamics and Control Conference, Vol. 168, PMLR, Stanford, California, USA, 2022, pp. 97–109.

[13] W. Chen, J. Onyejizu, L. Vu, L. Hoang, D. Subramanian, K. Kar, S. Mishra, S. Paternain, Adaptive primal-dual method for safe reinforcement learning, 2024, http://dx.doi.org/10.48550/arXiv.2402.00355, arxiv preprint arXiv:2402.00355.

[14] L. Hewing, K. Wabersich, M. Menner, M. Zeilinger, Learning-based model predictive control: Toward safe learning in control, Annu. Rev. Control. Robot. Auton. Syst. 3 (1) (2020) 269–296, http://dx.doi.org/10.1146/annurev-control-090419-075625.

[15] W. Zhao, T. He, C. Liu, Model-free safe control for zero-violation reinforcement learning, in: 5th Annual Conference on Robot Learning, Vol. 164, PMLR, 2021, pp. 784–793.

[16] Y. Yang, Z. Zheng, S. Li, J. Duan, J. Liu, X. Zhan, Y.-Q. Zhang, Feasible policy iteration, 2023, http://dx.doi.org/10.48550/arXiv.2304.08845, arxiv preprint arXiv:2304.08845.

[17] J. Tan, S. Xue, H. Li, H. Cao, D. Li, Safe stabilization control for interconnected virtual-real systems via model-based reinforcement learning, in: 2024 14th Asian Control Conference, ASCC, 2024, pp. 605–610.

[18] H. Tian, H. Hamedmoghadam, R. Shorten, P. Ferraro, Reinforcement learning with adaptive regularization for safe control of critical systems, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., 2024.

[19] R. Cheng, G. Orosz, R. Murray, J. Burdick, End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 3387–3395, http://dx.doi.org/10.1609/aaai.v33i01.33013387.

[20] Y. Emam, G. Notomista, P. Glotfelter, Z. Kira, M. Egerstedt, Safe reinforcement learning using robust control barrier functions, IEEE Robot. Autom. Lett. (2022) 1–8, http://dx.doi.org/10.1109/LRA.2022.3216996.

[21] A. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, P. Tabuada, Control barrier functions: Theory and applications, in: 2019 18th European Control Conference, ECC, IEEE, Naples, Italy, 2019, pp. 3420–3431, http://dx.doi.org/10.23919/ECC.2019.8796030.

[22] S. Liu, L. Liu, Z. Yu, Safe reinforcement learning for affine nonlinear systems with state constraints and input saturation using control barrier functions, Neurocomputing 518 (2023) 562–576, http://dx.doi.org/10.1016/j.neucom.2022.11.006.

[23] A. Chriat, C. Sun, On the optimality, stability, and feasibility of control barrier functions: An adaptive learning-based approach, IEEE Robot. Autom. Lett. 8 (11) (2023) 7865–7872, http://dx.doi.org/10.1109/LRA.2023.3322088.

[24] S. Liu, L. Liu, Z. Yu, Safe reinforcement learning for discrete-time fully cooperative games with partial state and control constraints using control barrier functions, Neurocomputing 517 (2023) 118–132, http://dx.doi.org/10.1016/j.neucom.2022.10.058.

[25] S. Liu, L. Liu, Z. Yu, Safe robust multi-agent reinforcement learning with neural control barrier functions and safety attention mechanism, Inform. Sci. 690 (2025) 121567, http://dx.doi.org/10.1016/j.ins.2024.121567.

[26] L. Zhao, K. Gatsis, A. Papachristodoulou, Stable and safe reinforcement learning via a Barrier-Lyapunov Actor-Critic Approach, in: 2023 62nd IEEE Conference on Decision and Control, CDC, IEEE, Singapore, 2023, pp. 1320–1325, http://dx.doi.org/10.1109/CDC49753.2023.10383742.

[27] C. Dawson, Z. Qin, S. Gao, C. Fan, Safe nonlinear control using robust neural Lyapunov-barrier functions, in: Proceedings of the 5th Conference on Robot Learning, Vol. 164, PMLR, London, UK, 2021, pp. 1724–1735.

[28] Y. Yang, Y. Jiang, Y. Liu, J. Chen, S. Li, Model-free safe reinforcement learning through neural barrier certificate, IEEE Robot. Autom. Lett. 8 (3) (2023) 1295–1302, http://dx.doi.org/10.1109/LRA.2023.3238656.

[29] D. Du, S. Han, N. Qi, H. Ammar, J. Wang, W. Pan, Reinforcement learning for safe robot control using control Lyapunov barrier functions, in: 2023 IEEE International Conference on Robotics and Automation, ICRA, IEEE, London, UK, 2023, pp. 9442–9448, http://dx.doi.org/10.1109/ICRA48891.2023.10160991.

[30] Y. Wang, S. Zhan, R. Jiao, Z. Wang, W. Jin, Z. Yang, Z. Wang, C. Huang, Q. Zhu, Enforcing hard constraints with soft barriers: Safe reinforcement learning in unknown stochastic environments, in: Proceedings of the 40th International Conference on Machine Learning, Vol. 202, PMLR, Honolulu, Hawaii, USA, 2023, pp. 36593–36604.

[31] Z. Li, C. Hu, Y. Wang, Y. Yang, S. Li, Safe reinforcement learning with dual robustness, IEEE Trans. Pattern Anal. Mach. Intell. 46 (12) (2024) 10876–10890, http://dx.doi.org/10.1109/TPAMI.2024.3443916.

[32] D. Yu, W. Zou, Y. Yang, H. Ma, S. Li, Y. Yin, J. Chen, J. Duan, Safe model-based reinforcement learning with an uncertainty-aware reachability certificate, IEEE Trans. Autom. Sci. Eng. 21 (3) (2023) 4129–4142, http://dx.doi.org/10.1109/TASE.2023.3292388.

[33] Y. Yang, Y. Zhang, W. Zou, J. Chen, Y. Yin, S. Li, Synthesizing control barrier functions with feasible region iteration for safe reinforcement learning, IEEE Trans. Autom. Control 69 (4) (2023) 2713–2720, http://dx.doi.org/10.1109/TAC.2023.3336263.

[34] S. Zhang, Y. Xiu, G. Qu, C. Fan, Compositional neural certificates for networked dynamical systems, in: Proceedings of the 5th Annual Learning for Dynamics and Control Conference, Vol. 211, PMLR, Philadelphia, USA, 2023, pp. 272–285.

[35] Y. Chow, O. Nachum, A. Faust, E. Duenez-Guzman, M. Ghavamzadeh, Lyapunov-based safe policy optimization for continuous control, 2019, http://dx.doi.org/10.48550/arXiv.1901.10031, arxiv preprint arXiv:1901.10031.

[36] M. Han, L. Zhang, J. Wang, W. Pan, Actor-critic reinforcement learning for control with stability guarantee, IEEE Robot. Autom. Lett. 5 (4) (2020) 6217–6224, http://dx.doi.org/10.1109/LRA.2020.3011351.

[37] Y. Chang, S. Gao, Stabilizing neural control using self-learned almost Lyapunov critics, in: 2021 IEEE International Conference on Robotics and Automation, ICRA, IEEE, Xi'an, China, 2021, pp. 1803–1809, http://dx.doi.org/10.1109/ICRA48506.2021.9560886.

[38] M. Reis, P. Aguiar, P. Tabuada, Control barrier function-based quadratic programs introduce undesirable asymptotically stable equilibria, IEEE Control. Syst. Lett. 5 (2) (2020) 731–736, http://dx.doi.org/10.1109/LCSYS.2020.3004797.

[39] R. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud, Neural ordinary differential equations, in: Advances in Neural Information Processing Systems, Vol. 31, Curran Associates, Inc., Montreal, Canada, 2018.

[40] M. Alvarez, R. Roşca, C. Fălcuţescu, Dynode: Neural ordinary differential equations for dynamics modeling in continuous control, 2020, http://dx.doi.org/10.48550/arXiv.2009.04278, arxiv preprint, arXiv:2009.04278.

[41] W. Zhao, R. Chen, Y. Sun, T. Wei, C. Liu, State-wise constrained policy optimization, 2023, http://dx.doi.org/10.48550/arXiv.2306.12594, arxiv preprint, arXiv:2306.12594.

[42] J. Li, D. Fridovich-Keil, S. Sojoudi, C. Tomlin, Augmented Lagrangian method for instantaneously constrained reinforcement learning problems, in: 2021 60th IEEE Conference on Decision and Control, CDC, IEEE, Austin, TX, USA, 2021, pp. 2982–2989, http://dx.doi.org/10.1109/CDC45484.2021.9683088.

[43] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, S. Levine, Soft actor-critic algorithms and applications, 2018, http://dx.doi.org/10.48550/arXiv.1812.05905, arxiv preprint arXiv:1812.05905.

[44] L. Zhao, K. Miao, K. Gatsis, A. Papachristodoulou, Stable and safe human-aligned reinforcement learning through neural ordinary differential equations, 2024, http://dx.doi.org/10.48550/arXiv.2401.13148, arxiv preprint arXiv:2401.13148.

[45] G. Thomas, Y. Luo, T. Ma, Safe reinforcement learning by imagining the near future, in: Advances in Neural Information Processing Systems, Vol. 34, Curran Associates, Inc., 2021, pp. 13859–13869.

[46] Y. Xiong, D. Zhai, M. Tavakoli, Y. Xia, Discrete-time control barrier function: High-order case and adaptive case, IEEE Trans. Cybern. 53 (5) (2022) 3231–3239, http://dx.doi.org/10.1109/TCYB.2022.3170607.

[47] X. Tan, W.S. Cortez, D. Dimarogonas, High-order barrier functions: Robustness, safety, and performance-critical control, IEEE Trans. Autom. Control 67 (6) (2021) 3021–3028, http://dx.doi.org/10.1109/TAC.2021.3089639.

[48] W. Xiao, C. Belta, High-order control barrier functions, IEEE Trans. Autom. Control 67 (7) (2022) 3655–3662, http://dx.doi.org/10.1109/TAC.2021.3105491.

[49] K. Miao, K. Gatsis, How deep do we need: Accelerating training and inference of neural ODEs via control perspective, in: Proceedings of the 41st International Conference on Machine Learning, Proceedings of Machine Learning Research, 235, PMLR, 2024, pp. 35528–35545.

[50] K. Miao, K. Gatsis, Learning robust state observers using neural ODEs, in: Proceedings of The 5th Annual Learning for Dynamics and Control Conference, Proceedings of Machine Learning Research, 211, PMLR, 2023, pp. 208–219.

[51] D. Tan, F. Acero, R. McCarthy, D. Kanoulas, Z. Li, Value functions are control barrier functions: Verification of safe policies using control theory, 2023, http://dx.doi.org/10.48550/arXiv.2306.04026, arxiv preprint arXiv:2306.04026.

[52] D. Bertsekas, Nonlinear Programming, Athena Scientific, Belmont, Massachusetts, 2016.

[53] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, 2015, http://dx.doi.org/10.48550/arXiv.1509.02971, arxiv preprint arXiv:1509.02971.

[54] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, 2017, http://dx.doi.org/10.48550/arXiv.1707.06347, arxiv preprint arXiv:1707.06347.

[55] A. Jayant, S. Bhatnagar, Model-based safe deep reinforcement learning via a constrained proximal policy optimization algorithm, in: Advances in Neural Information Processing Systems, Vol. 35, Curran Associates, Inc., New Orleans, LA, USA, 2022, pp. 24432–24445.

[56] A. Ray, J. Achiam, D. Amodei, Benchmarking safe exploration in deep reinforcement learning, 2019.

[57] J. Wu, H. Zhang, Y. Vorobeychik, Verified safe reinforcement learning for neural network dynamic models, 2024, http://dx.doi.org/10.48550/arXiv.2405.15994, arxiv preprint arXiv:2405.15994.

[58] Z. Yuan, A. Hall, S. Zhou, L. Brunke, M. Greeff, J. Panerati, A. Schoellig, Safe-control-gym: A unified benchmark suite for safe learning-based control and reinforcement learning in robotics, IEEE Robot. Autom. Lett. 7 (4) (2022) 11142–11149, http://dx.doi.org/10.1109/LRA.2022.3196132.