Proceedings of Machine Learning Research vol 283:1–12, 2025 7th Annual Conference on Learning for Dynamics and Control

Safe Exploration in Reinforcement Learning: Training Backup Control Barrier Functions with Zero Training-Time Safety Violations

Pedram Rabiee PEDRAM.RABIEE@UKY.EDU
Amirsaeid Safari AMIRSAEID.SAFARI@UKY.EDU
Department of Mechanical and Aerospace Engineering, University of Kentucky

Editors: N. Ozay, L. Balzano, D. Panagou, A. Abate

Abstract

This paper introduces the Reinforcement Learning Backup Shield (RLBUS), a framework that guarantees safe exploration in reinforcement learning (RL) by incorporating Backup Control Barrier Functions (BCBFs). RLBUS synthesizes an implicit control forward invariant subset of the safe set using multiple backup policies, ensuring safety in the presence of input constraints. While traditional BCBFs often yield conservative control forward invariant sets due to the design of backup controllers, RLBUS addresses this limitation by leveraging model-free RL to train an additional backup policy, enlarging the identified forward invariant subset of the safe set. This approach enables safe exploration of larger regions of the state space with zero safety violations during training. The effectiveness of RLBUS is demonstrated on an inverted pendulum example, where the expanded invariant set facilitates safe exploration over a broader state space, enhancing performance without compromising safety.[1]

Keywords: Safe reinforcement learning, control barrier function, control invariant set

1. Introduction

Safe reinforcement learning (RL) has emerged as a key area of research for deploying RL to real-world applications while ensuring safe interactions with the environment. Many existing approaches in safe RL aim to develop a safe policy by the end of training (Haarnoja et al., 2018; Chow et al., 2018); however, these methods often fall short in ensuring safety throughout the training process. More recent methods have focused on reducing unsafe interactions during learning (Schreiter et al., 2015; Wachi et al., 2018; Achiam et al., 2017), yet achieving full safety guarantees over the entire training horizon remains challenging.

Model-based approaches have integrated Lyapunov stability analysis to enhance safety during online exploration (Koller et al., 2018; Berkenkamp et al., 2017; Wang et al., 2018). Despite effectively constraining system behavior, these methods lack a unified framework to simultaneously ensure safety and optimize performance. While some researchers have proposed switching between predefined backup controllers to ensure safety (Alshiekh et al., 2018; Wabersich and Zeilinger, 2018; Bastani et al., 2021), these approaches rely on statistical guarantees that accumulate errors over time, limiting their effectiveness for systems requiring long-term or infinite-horizon safety assurances.

A common approach to ensuring safety over an infinite horizon involves determining a policy that renders a specified safe set forward invariant (Blanchini, 1999). However, achieving controlled invariance of the safe set is challenging, particularly under input constraints, which often prevent

1. The source code of this work is available at: https://github.com/pedramrabiee/safe_rl

finding control inputs that preserve the forward invariance of the entire set. Methods like Hamilton-Jacobi reachability analysis (Mitchell et al., 2005) and sum-of-squares programming (Korda et al., 2014) provide computational approaches for determining a control forward-invariant subset of the safe set, but they can be computationally intensive for high-dimensional systems. Control barrier functions (CBFs) offer an alternative to traditional offline verification methods by constructing control-invariant sets that ensure online safety (Wieland and Allgöwer, 2007; Ames et al., 2016; Rabiee and Hoagg, 2024a; Safari and Hoagg, 2023). However, designing CBFs that account for input constraints remains a challenging task.

One approach to ensuring safety with actuator constraints is the Backup Control Barrier Function (BCBF), which involves predicting system trajectories over a finite horizon to obtain a control-forward invariant subset of the safe set. For instance, (Gurriet et al., 2020) determines this subset by constructing a CBF from finite-horizon predictions under a single backup control. Extending this approach, (Rabiee and Hoagg, 2023, 2025) introduced the soft-maximum/soft-minimum BCBF method, which leverages multiple backup controllers to further expand the forward-invariant subset. However, the resulting set remains conservative due to the design of the backup controllers

This paper introduces the Reinforcement Learning Backup Shield (RLBUS), a framework that guarantees safe exploration in RL by integrating Backup Control Barrier Functions (BCBFs). RLBUS begins with a conservative forward-invariant set constructed from multiple backup sets, ensuring safety under input constraints. It then employs model-free RL to iteratively train an additional backup policy. This optimization enlarges the forward invariant set, allowing RLBUS to explore a broader state space and enhance performance, all while guaranteeing zero safety violations throughout training. The effectiveness of RLBUS is demonstrated on an inverted pendulum example, where the expanded invariant set enables safe exploration over a broader state space and achieves improved performance without compromising safety[2].

2. Notation

Let $\eta : [0, \infty) \times \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable. Then, the partial Lie derivative of η with respect to x along the vector fields of $\psi : \mathbb{R}^n \to \mathbb{R}^{n\times\ell}$ is define as $L_\psi \eta(t, x) \triangleq \frac{\partial \eta(t,x)}{\partial x} \psi(x)$. Let int $\mathcal{A}$, bd $\mathcal{A}$ denote the interior and boundary of the set $\mathcal{A}$. For a positive integer n, let $\mathbb{I}[n]$ denote $\{1, 2, \ldots, n\}$, and $\mathbb{W}[n]$ denote $\{0, 1, \ldots, n\}$. Let $\mathcal{A}^\epsilon$ denote the ϵ-superlevel set of a set $\mathcal{A}$ defined by a function $h_A : \mathbb{R}^n \to \mathbb{R}$, as $\mathcal{A}^\epsilon \triangleq \{x \in \mathbb{R}^n : h_\mathcal{A}(x) \geq \epsilon\}$.

Let $\rho > 0$, and consider $\mathrm{softmin}_\rho : \mathbb{R}^N \to \mathbb{R}$ and $\mathrm{softmax}_\rho : \mathbb{R}^N \to \mathbb{R}$, which are defined by $\mathrm{softmin}_\rho(z_1, \cdots, z_N) \triangleq -\frac{1}{\rho} \log \sum_{i=1}^N e^{-\rho z_i}$ and $\mathrm{softmax}_\rho(z_1, \cdots, z_N) \triangleq \frac{1}{\rho} \log \sum_{i=1}^N e^{\rho z_i} - \frac{\log N}{\rho}$, respectively.

3. Problem Formulation

Consider

$$\dot{x}(t) = \tilde{f}_u(x(t)) \triangleq f(x(t)) + g(x(t))u(x(t)), \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the state, $x(0) = x_0 \in \mathbb{R}^n$ is the initial condition, $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are continuously differentiable on $\mathbb{R}^n$, $u : \mathbb{R}^n \to \mathcal{U} \subseteq \mathbb{R}^m$ is the control, which

2. Due to space limitations, the supplementary materials can be found at the end of the following version: https://arxiv.org/pdf/2312.07828

is locally Lipschitz continuous on $\mathbb{R}^n$ and $\mathcal{U}$ is defined as a compact and convex set of admissible controls, and $\tilde{f}_u : \mathbb{R}^n \to \mathbb{R}^n$ denotes the closed-loop dynamic under the control law u. We assume that the closed-loop dynamics $\tilde{f}_u$ is forward complete, meaning that the solution x to (1) exists for all $t \geq 0$.

Next, let $\phi_u : \mathbb{R}^n \times [0, \infty) \to \mathbb{R}^n$ defined by

$$\phi_u(x, \tau) \triangleq x + \int_0^\tau \tilde{f}_u(\phi_u(x, \sigma))\, \mathrm{d}\sigma, \tag{2}$$

which is the *flow map* of (1) under the control law u with initial condition x over a time period $\tau \in [0, \infty)$. The following definition is needed.

Definition 1 *(Rabiee and Hoagg, 2024b)[Definition 1]* A set $\mathcal{D} \subset \mathbb{R}^n$ is *control forward invariant* with respect to (1), if there exists a locally Lipschitz $\hat{u} : \mathcal{D} \to \mathcal{U}$ such that for all $x_0 \in \mathcal{D}$ and all $t \geq 0$, $\phi_u(x_0, t) \in \mathcal{D}$ with $u \equiv \hat{u}$.

Next, consider a continuously differentiable function $h_\mathrm{s}: \mathbb{R}^n \to \mathbb{R}$, and define the safe set

$$\mathcal{S}_\mathrm{s} \triangleq \{x \in \mathbb{R}^n : h_\mathrm{s}(x) \geq 0\}, \tag{3}$$

which consists of states that satisfy safety constraints. While $\mathcal{S}_\mathrm{s}$ characterizes safety requirements, it is not necessarily control forward invariant. To ensure safety, it is necessary to identify a control forward invariant subset of $\mathcal{S}_\mathrm{s}$ and to maintain the system's state within this set. However, if the identified subset is too small, it may negatively impact the performance of the task. The performance is typically characterized by how closely the safe policy follows the desired performance policy. Let $u_\mathrm{d} : \mathbb{R}^n \to \mathcal{U}$ be a performance policy designed to achieve control objectives without considering safety constraints. The key challenge is to identify a sufficiently large control forward invariant subset of $\mathcal{S}_\mathrm{s}$ that enables tracking u_d while guaranteeing safety.

To address this challenge, we first assume the existence of multiple, relatively small control forward invariant subsets of $\mathcal{S}_\mathrm{s}$. Formally, let ℓ be a positive integer and for each $j \in \mathbb{I}[\ell]$, let $h_{\mathrm{b}_j}: \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable and define the j-th *backup set*

$$\mathcal{S}_{\mathrm{b}_j} \triangleq \{x \in \mathbb{R}^n : h_{\mathrm{b}_j}(x) \geq 0\}, \tag{4}$$

and we assume $\mathcal{S}_{\mathrm{b}_j} \subseteq \mathcal{S}_\mathrm{s}$. Next, for each $j \in \mathbb{I}[\ell]$, let $u_{\mathrm{b}_j} : \mathbb{R}^n \to \mathcal{U}$ be the j-th backup policy and make the following assumption.

Assumption 1 For all $j \in \mathbb{I}[\ell]$, $x_0 \in \mathcal{S}_{\mathrm{b}_j}$ and $t \geq 0$, $\phi_{u_{\mathrm{b}_j}}(x_0, t) \in \mathcal{S}_{\mathrm{b}_j}$.

Assumption 1 implies that each backup policy u_{b_j} renders its corresponding backup set $\mathcal{S}_{\mathrm{b}_j}$ control forward invariant. The backup sets $\mathcal{S}_{\mathrm{b}_j}$ are small subsets of $\mathcal{S}_\mathrm{s}$ that provide infinite-horizon safety guarantees and can typically be constructed using Lyapunov stability theory as a region of attraction around the equilibrium points inside $\mathcal{S}_\mathrm{s}$.

To identify a control forward invariant subset of the safe set, a finite-time reach-avoid set problem is formulated. Let $T > 0$ be a time horizon and $\mathcal{U}^{[0,T]}$ denotes the set of measurable control functions mapping $[0, T]$ to the control space $\mathcal{U}$ and consider the value function $V: \mathbb{R}^n \to \mathbb{R}$ defined by

$$V(x) \triangleq \max_{\hat{u}(\cdot) \in \mathcal{U}^{[0,T]}} H(x, \hat{u}(\cdot)), \tag{5}$$

where

$$H(x, \hat{u}(\cdot)) \triangleq \min\left\{ \min_{\tau \in [0,T]} h_{\mathrm{s}}(\phi_{\hat{u}(\cdot)}(x, \tau)), \max_{j \in \mathbb{I}[\ell]} h_{\mathrm{b}_j}(\phi_{\hat{u}(\cdot)}(x, T)) \right\}, \tag{6}$$

and define

$$\mathcal{R} \triangleq \{x \in \mathbb{R}^n \colon V(x) \geq 0\}, \tag{7}$$

as the *finite-time safe backward image* of $\bar{\mathcal{S}}_{\mathrm{b}} \triangleq \cup_{i \in \mathbb{I}[\ell]} \mathcal{S}_{\mathrm{b}_i}$. The set $\mathcal{R}$ contains all states $x \in \mathcal{S}_{\mathrm{s}}$ for which a trajectory exists that (i) remains within the safe set $\mathcal{S}_{\mathrm{s}}$ throughout $[0, T]$, and (ii) reaches at least one backup set $\mathcal{S}_{\mathrm{b}_j}$ at time T. This formulation represents a finite-time reach-avoid problem, with $\bar{\mathcal{S}}_{\mathrm{b}}$ as the reachable set and $\mathbb{R}^n \setminus \mathcal{S}_{\mathrm{s}}$ as the avoid set, and the set $\mathcal{R} \subseteq \mathcal{S}_{\mathrm{s}}$ is control forward invariant by construction. The size of the identified control forward invariant subset of $\mathcal{S}_{\mathrm{s}}$ increases with the time horizon T, thereby potentially providing sufficient space for tracking u_{d} while preserving safety guarantees.

4. Proposed Method

Traditional Hamilton-Jacobi reachability analysis transforms the optimization problem (5) into a partial differential equation (PDE) using the Hamilton-Jacobi-Bellman principle, yielding a Hamilton-Jacobi-Isaacs equation with value function as a viscosity solution(Fisac et al., 2015). However, this approach suffers from the curse of dimensionality, as solving the PDE on a discretized state-space grid becomes computationally intractable with increasing state-space dimension. Moreover, given a specific task, obtaining the value function for all states is often unnecessary and computationally prohibitive.

To address these limitations, we propose to employ a learning-based method for approximating the value function. Consider a neural policy $u_\theta : \mathbb{R}^n \to \mathcal{U}$ parameterized by $\theta \in \Theta \subseteq \mathbb{R}^d$, where d represents the number of parameters. The objective is to train the neural policy u_θ to approximate the solution to (5), while ensuring that safety is maintained throughout all training and execution episodes.

For K training iterations, let

$$V^{(k)}(x) \triangleq H(x, u_{\theta^{(k)}}), \quad \mathcal{R}^{(k)} \triangleq \{x \in \mathbb{R}^n \colon V^{(k)}(x) \geq 0\}, \tag{8}$$

where $k \in \mathbb{I}[K]$ denotes the iteration number and $\theta^{(k)}$ denotes the parameters at iteration k. While $V^{(k)}$ approximates the value function V, the set $\mathcal{R}^{(k)}$ remains control forward invariant, as it consists of all states from which the flow of (1) under $u_{\theta^{(k)}}$ remains within $\mathcal{S}_{\mathrm{s}}$ over $t \in [0, T]$ and reaches $\bar{\mathcal{S}}_{\mathrm{b}}$ at time T.

Unlike HJ reachability analysis which requires sweeping the entire state space, our proposed learning-based approach collects data guided by the performance policy u_{d}. However, since u_{d} does not account for safety, we employ a minimum-intervention safety filtering approach: the system follows u_{d} unless safety is compromised, in which case it switches to the closest safe control to u_{d}. To this end, in Section 4.1, we propose a safety filtering approach based on backup control barrier functions (Gurriet et al., 2020; Rabiee and Hoagg, 2025) with the neural policy serving as a backup policy. Notably, while the neural policy may perform poorly at the start of training, our framework guarantees that this does not compromise safety. Using this safety filter, safety is maintained during training, and as the neural policy improves, the size of the safely explorable region grows. The training algorithm for the neural policy is detailed in Section 4.2.

4.1. Safe Neural Backup Policy

This subsection develops a safety-guaranteed framework for neural backup policies. We first introduce a neural policy architecture that ensures safety even during early training stages when the neural policy may be unreliable. Next, we adopt the approach from (Rabiee and Hoagg, 2025, 2023) to synthesize a barrier function using the neural backup policy and the designed backup policies $\{u_{\mathrm{b}_1}, \ldots, u_{\mathrm{b}_\ell}\}$. Our proposed method improves the computational efficiency of (Rabiee and Hoagg, 2025), and by leveraging the neural backup policy achieves a significantly larger forward invariant set compared to (Rabiee and Hoagg, 2025).

Let $\nu > 0$ and consider a continuously differentiable function $\xi : \mathbb{R} \to [0, 1]$ such that for all $x \in (-\infty, -\nu]$, $\xi(x) = 0$; for all $x \in [0, \infty)$, $\xi(x) = 1$; and ξ is strictly increasing on $x \in [-\nu, 0]$. Next, define $J(x) \triangleq \{j \in \mathbb{I}[\ell] : h_{\mathrm{b}_j}(x) \geq -\nu\}$, and make the following assumption.

Assumption 2 $J(x)$ is singleton.

Assumption 2 implies that for all $j_1, j_2 \in \mathbb{I}[\ell]$ with $j_1 \neq j_2$, $\mathcal{S}_{\mathrm{b}_{j_1}}^{-\nu} \cap \mathcal{S}_{\mathrm{b}_{j_2}}^{-\nu} = \emptyset$, ensuring that the $(-\nu)$-superlevel sets of the backup sets are disjoint.

Next, let $\pi_\theta\colon \mathbb{R}^n \to \mathcal{U}$ be a multi-layer perceptron (MLP) parameterized by $\theta \in \Theta$ with continuously differentiable activation functions (e.g., ELU, SiLU). Then, we consider the following structure for the neural policy u_θ.

$$u_\theta(x) = \begin{cases} u_{\mathrm{b}_{J(x)}}(x), & \text{if } x \in \mathcal{S}_{\mathrm{b}_{J(x)}}, \\ \xi(x) u_{\mathrm{b}_{J(x)}}(x) + [1 - \xi(x)]\pi_\theta(x), & \text{if } x \in \mathcal{S}_{\mathrm{b}_{J(x)}}^{-\nu} \setminus \mathcal{S}_{\mathrm{b}_{J(x)}}, \\ \pi_\theta(x), & \text{else.} \end{cases} \tag{9}$$

The neural policy u_θ uses the MLP network for the states in $\mathcal{S}_{\mathrm{s}} \setminus \mathcal{S}_{\mathrm{b}_{J(x)}}^{-\nu}$, and smoothly transitions to the backup control $u_{\mathrm{b}_{J(x)}}$ through the homotopy function ξ as trajectories approach $\mathcal{S}_{\mathrm{b}_{J(x)}}$. Note that Assumption 2 implies u_θ is well defined since $J(x)$ is unique. The following result is immediately followed by the definition of u_θ. All proofs are in the supplementary materials[3].

Proposition 1 Consider (1) where Assumptions 1 and 2 are satisfied. Let $x_0 \in \bar{\mathcal{S}}_{\mathrm{b}}$, then, for all $t \geq 0$, $\phi_u(x, t) \in \bar{\mathcal{S}}_{\mathrm{b}}$, with $u \equiv u_\theta$.

Let $u_{\mathrm{b}_{\ell+1}} \equiv u_\theta$ be the $(\ell + 1)$-th backup control. Let $\rho_1 > 0$ and consider $h_{\mathrm{b}_{\ell+1}} : \mathbb{R}^n \to \mathbb{R}$ defined by

$$h_{\mathrm{b}_{\ell+1}}(x) \triangleq \mathrm{softmax}_{\rho_1}(h_{\mathrm{b}_1}(x), h_{\mathrm{b}_2}(x), \ldots, h_{\mathrm{b}_\ell}(x)), \tag{10}$$

and define $\mathcal{S}_{\mathrm{b}_{\ell+1}} \triangleq \{x \in \mathbb{R}^n\colon h_{\mathrm{b}_{\ell+1}}(x) \geq 0\}$ as the $(\ell + 1)$-th backup set. The following result is the direct consequence of (Rabiee and Hoagg, 2025, Prop. 1).

Proposition 2 $\mathcal{S}_{\mathrm{b}_{\ell+1}} \subseteq \bar{\mathcal{S}}_{\mathrm{b}}$.

Proposition 2 implies that $\mathcal{S}_{\mathrm{b}_{\ell+1}}$ inner approximates $\bar{\mathcal{S}}_{\mathrm{b}}$, where ρ_1 determines the approximation accuracy. The next result which is the consequence of Proposition 1 and Proposition 2.

Proposition 3 Consider (1) where Assumptions 1 and 2 are satisfied. Let $x_0 \in \mathcal{S}_{\mathrm{b}_{\ell+1}}$, then, for all $t \geq 0$, $\phi_u(x, t) \in \bar{\mathcal{S}}_{\mathrm{b}}$, with $u \equiv u_\theta$.

3. https://arxiv.org/pdf/2312.07828

Proposition 3 states that any trajectory initialized in $\mathcal{S}_{\mathrm{b}_{\ell+1}}$ under the neural policy u_θ remains in $\bar{\mathcal{S}}_{\mathrm{b}}$ for all time.

Next, for all $j \in \mathbb{I}[\ell+1]$, let $\tilde{f}_{u_{\mathrm{b}_j}}: \mathbb{R}^n \to \mathbb{R}^n$ be defined by (1) where u is replaced by u_{b_j}, and let $\phi_{u_{\mathrm{b}_j}}: \mathbb{R}^n \times [0,\infty) \to \mathbb{R}^n$ be defined by (2) where ϕ_u and $\tilde{f}_u$ are replaced by $\phi_{u_{\mathrm{b}_j}}$ and $\tilde{f}_{u_{\mathrm{b}_j}}$. In the following, we adopt the soft-maximum/soft-minimum barrier function method from (Rabiee and Hoagg, 2025), modifying the safe control formulation in (17) and (18) to achieve a simpler control structure.

Consider $h_{*j}, h_*: \mathbb{R}^n \to \mathbb{R}$ and $\mathcal{S}_*$ defined by

$$h_{*j} \triangleq \min\left\{ h_{\mathrm{b}_j}(\phi_{u_{\mathrm{b}_j}}(x,T)), \min_{\tau \in [0,T]} h_{\mathrm{s}}(\phi_{u_{\mathrm{b}_j}}(x,\tau)) \right\}, \tag{11}$$

$$h_*(x) \triangleq \max_{j \in \mathbb{I}[\ell+1]} h_{*j}, \quad \mathcal{S}_* \triangleq \{x \in \mathbb{R}^n : h_*(x) \geq 0\}, \tag{12}$$

and it follows from (Rabiee and Hoagg, 2025, Prop. 8) that $\mathcal{S}_*$ is control forward invariant. $\mathcal{S}_*$ comprises all states for which there exists an index $\jmath \in \mathbb{I}[\ell+1]$ such that the trajectory under backup policy $u_{\mathrm{b}_\jmath}$ remains in $\mathcal{S}_{\mathrm{s}}$ throughout $[0,T]$ and reaches $\mathcal{S}_{\mathrm{b}_\jmath}$ at time T. However, neither h_{*j} nor h can serve directly as barrier functions: h_{*j} contains an embedded optimization problem and both functions lack differentiability. Instead, following (Rabiee and Hoagg, 2025), we use a continuous approximation that evaluates h_{s} along the backup policy trajectories at sampled time instances. Let N be a positive integer, and define $T_{\mathrm{s}} \triangleq T/N$. Let $\rho_2, \rho_3 > 0$ and for $j \in \mathbb{I}[\ell+1]$ consider $h_j, h : \mathbb{R}^n \to \mathbb{R}$ defined by

$$h_j(x) \triangleq \mathrm{softmin}_{\rho_2}(h_{\mathrm{s}}(\phi_{u_{\mathrm{b}_j}}(x,0)), h_{\mathrm{s}}(\phi_{u_{\mathrm{b}_j}}(x,T_{\mathrm{s}})), \ldots, h_{\mathrm{s}}(\phi_{u_{\mathrm{b}_j}}(x,NT_{\mathrm{s}})), h_{\mathrm{b}}(\phi_{u_{\mathrm{b}_j}}(x,NT_{\mathrm{s}}))), \tag{13}$$

$$h(x) \triangleq \mathrm{softmax}_{\rho_3}(h_1(x), \ldots, h_{\ell+1}(x)), \tag{14}$$

and define

$$\mathcal{S}_j \triangleq \{x \in \mathbb{R}^n : h_j(x) \geq 0\}, \quad \mathcal{S} \triangleq \{x \in \mathbb{R}^n : h(x) \geq 0\}.$$

Define $l_\phi \triangleq \max_{j \in \mathbb{I}[\ell+1]} \sup_{x \in \mathcal{S}_*} \|\tilde{f}_{u_{\mathrm{b}_j}}(x)\|_2$, and let l_{s} denotes the Lipschitz constant of h_{s} with respect to the two norm. Next, define $\epsilon_{\mathrm{s}} \triangleq \frac{1}{2} T_{\mathrm{s}} l_\phi l_{\mathrm{s}}$. The follwoing results are needed. The first result is similar to (Rabiee and Hoagg, 2025, Prop. 10) and shows that a level set of h is contained in $\mathcal{S}_*$. The second result relates $\mathcal{S}_{\ell+1}$ to $\mathcal{R}$, and is directly followed from (6) and (14).

Proposition 4 $\mathcal{S}^{\epsilon_{\mathrm{s}}} \subseteq \mathcal{S}_*$.

Proposition 5 $\mathcal{S}_{\ell+1}^{\epsilon_{\mathrm{s}}} \subseteq \mathcal{R}$.

Let $\alpha > 0$, $\epsilon \in [0, \max_{x \in \mathcal{S}} h(x))$, and consider $\beta: \mathbb{R}^n \to \mathbb{R}$ defined by

$$\beta(x) \triangleq L_f h(x) + \alpha(h(x) - \epsilon) + \max_{\hat{u} \in \mathcal{U}} L_g h(x) \hat{u}, \tag{15}$$

where for all $x \in \mathbb{R}^n$, $\beta(x)$ exists since $\mathcal{U}$ is not empty. Let $\kappa_h, \kappa_\beta > 0$ and consider $\gamma: \mathbb{R}^n \to \mathbb{R}$ defined by $\gamma(x) \triangleq \min\left\{ \frac{h(x)-\epsilon}{\kappa_h}, \frac{\beta(x)}{\kappa_\beta} \right\}$, and define $\Gamma \triangleq \{x \in \mathbb{R}^n : \gamma(x) \geq 0\}$. For all $x \in \Gamma$, define

$$u_*(x) \triangleq \operatorname*{argmin}_{\hat{u} \in \mathcal{U}} \|\hat{u} - u_{\mathrm{d}}(x)\|_2^2 \quad \text{subject to} \quad L_f h(x) + L_g h(x)\hat{u} + \alpha(h(x) - \epsilon) \geq 0. \tag{16}$$

It follows from (Rabiee and Hoagg, 2025, Prop. 6) that for all $x \in \Gamma$, the quadratic program (16) is feasible. Next for all $x \in \mathbb{R}^n$, define $I(x) \triangleq \{j \in \mathbb{I}[\ell+1] : h_j(x) \geq \epsilon\}$, and $\bar{\mathcal{S}} \triangleq \cup_{j \in \mathbb{I}[\ell+1]} \mathcal{S}_j^\epsilon$. Then, for all $x \in \bar{\mathcal{S}}$, define the augmented backup control

$$u_{\mathrm{a}}(x) \triangleq \frac{\sum_{j \in I(x)} [h_j(x) - \epsilon] u_{\mathrm{b}_j}(x)}{\sum_{j \in I(x)} [h_j(x) - \epsilon]}, \tag{17}$$

which is a weighted sum of the backup controls for which $h_j(x) > \epsilon$. Note that for all $x \in \operatorname{int} \bar{\mathcal{S}}$, $I(x)$ is not empty and thus, $u_{\mathrm{a}}(x)$ is well defined.

Consider a continuous function $\sigma : \mathbb{R} \to [0,1]$ such that for all $a \in (-\infty, 0]$, $\sigma(a) = 0$; for all $a \in [1, \infty)$, $\sigma(a) = 1$; and σ is strictly increasing on $a \in [0,1]$.

Finally, consider the control

$$u(x) = \begin{cases} [1 - \sigma(\gamma(x))] u_{\mathrm{a}}(x) + \sigma(\gamma(x)) u_*(x), & \text{if } x \in \Gamma, \\ u_{\mathrm{b}_q}(x), & \text{else,} \end{cases} \tag{18}$$

where $q : [0, \infty) \to \mathbb{I}[\ell+1]$ satisfies

$$\begin{cases} \dot{q} = 0, & \text{if } x \notin \operatorname{bd} \bar{\mathcal{S}}, \\ q^+ \in I(x), & \text{if } x \in \operatorname{bd} \bar{\mathcal{S}}, \end{cases} \tag{19}$$

where $q(0) \in \mathbb{I}[\ell+1]$ and q^+ denotes the value of q after the instantaneous change. It follows from (19) that q remains constant when $x \notin \bar{\mathcal{S}}$, and consequently, the same backup control u_{b_q} is used in (18) until state x reaches $\operatorname{bd} \bar{\mathcal{S}}$. Thus, backup control switching occurs only on $\operatorname{bd} \bar{\mathcal{S}}$. The controls (17) and (18) are modified versions of those in (Rabiee and Hoagg, 2025), offering a simpler formulation. While this modification leads to earlier switching to backup policies, the effect becomes negligible for larger values of ρ_2 and ρ_3, which is typical in practice.

The following theorem which is similar to (Rabiee and Hoagg, 2025, Theorem 2) establishes that the control law u defined by (13)–(19) is continuous and admissible, and renders $\mathcal{S}_*$ forward invariant.

Theorem 1 Consider (1), where Assumptions 1 and 2 are satisfied and u is given by (13)–(19). Then, the following conditions hold:

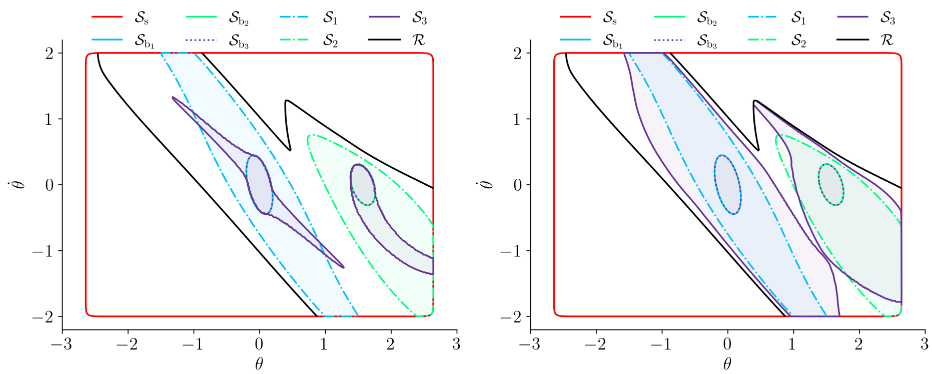
(a) u is continuous on $\mathbb{R}^n \setminus \operatorname{bd} \bar{\mathcal{S}}$.

(b) For all $x \in \mathbb{R}^n$, $u(x) \in \mathcal{U}$.

(c) Let $\epsilon \geq \epsilon_{\mathrm{s}}$ and $x_0 \in \mathcal{S}_*$. Then, for all $t \geq 0$, $x(t) \in \mathcal{S}_* \subseteq \mathcal{S}_{\mathrm{s}}$.

To summarize, we provided a structure for the neural policy to be used as a backup policy by (9) and introduced the backup set corresponding to this backup policy in (10). We also demonstrated how this learning-based backup policy can be used alongside other predesigned backup policies. Furthermore, we showed that even if the neural backup policy performs arbitrarily poorly at the beginning of training, the safety guarantees established by Theorem 1 remain valid.

The control (18) relies on computing the Lie derivatives $L_f h(x)$ and $L_g h(x)$, which are utilized in (15) and (16). In (Rabiee and Hoagg, 2025; Gurriet et al., 2020), these Lie derivatives are computed through sensitivity analysis, where a sensitivity matrix $Q \in \mathbb{R}^{n \times n}$ captures the evolution



(a) Episode 0 (b) Episode 100

Figure 1: Illustration of $\mathcal{S}_{\mathrm{s}}$, $\mathcal{S}_{\mathrm{b}_1}$, $\mathcal{S}_{\mathrm{b}_2}$, $\mathcal{S}_{\mathrm{b}_3}$, $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_2$, and $\mathcal{R}$.

of state trajectory variations with respect to initial conditions. This approach requires solving an augmented system of $n + n^2$ ordinary differential equations (ODEs). While this method ensures exact computation of the Lie derivatives, it introduces significant computational overhead due to the quadratic growth in the number of equations with respect to the state dimension.

To overcome this computational burden, this paper instead leverages the adjoint sensitivity method introduced by Chen et al. (2018). The adjoint method introduces an adjoint state $a(t) \in \mathbb{R}^n$ that evolves backward according to $da/dt = -a(t)^\top \partial \tilde{f}_{u_{\mathrm{b}_j}} / \partial x$. This approach offers significant computational advantages by reducing the dimensionality of the augmented system from $n + n^2$ to $2n$. Furthermore, unlike the forward sensitivity approach which requires storing the entire state trajectory, the adjoint method can reconstruct necessary forward states during the backward pass, achieving $O(1)$ memory complexity with respect to the integration time steps. This enables efficient computation of the Lie derivatives for real-time control applications.

Algorithm 1 in the Supplementary Materials[4] outlines the procedure for implementing the control defined in (13)–(19).

Remark 1 The desired control u_d in the quadratic program (16) can be replaced by a neural desired policy $u_{d_{\tilde{\theta}}} : \mathbb{R}^n \to \mathbb{R}^m$ parameterized by $\tilde{\theta} \in \Theta$. This creates a dual-network framework where $u_{d_{\tilde{\theta}}}$ and the neural backup policy u_θ are trained concurrently while maintaining safety guarantees. As $u_{d_{\tilde{\theta}}}$ guides state trajectories toward high-performance regions, it concentrates the training of u_θ in performance-critical areas, establishing a performance-oriented framework with guaranteed safety.

4.2. RLBUS: Reinforcement Learning Backup Control Barrier Function Shield

This section presents a method to train the neural policy u_θ using the reinforcement learning (RL) framework. We consider the standard RL setup with a finite-horizon deterministic Markov decision process (MDP) defined by tuple $(\mathcal{X}, \mathcal{U}, \gamma, r, \mu_0, H, f, g)$, where $\mathcal{X} \in \mathbb{R}^n$ is the state space, $\mathcal{U}$ is the action space, $r : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is the reward function, $\gamma \in (0, 1)$ is the discount factor, μ_0 is the distribution of the initial state x_0, $H > 0$ is the horizon, and f, g as in (1) represent the deterministic dynamics model. Let $\delta t > 0$, and note that the state transitions for the MDP are

4. https://arxiv.org/pdf/2312.07828

obtained by zero-order-holding (1) on the control u for δt. We note that this approximation does not affect safety guarantees. Provided δt is sufficiently small, and under some Lipschitz-continuity assumptions solving the constrained optimization in (16) at the required frequency ensures safety between triggering times, as highlighted in (Ames et al., 2016, Theorem 3). Specifically, with appropriate choices of δt and ϵ in (16), safety constraints are satisfied for all times.

Let μ^{u_θ} denote the distribution of states and actions induced by policy u_θ. Define the expected discounted return $\eta : \mathcal{U} \to \mathbb{R}$ as

$$\eta(u_\theta) = \mathbb{E}_{\mu^{u_\theta}}\left[\sum_{i=1}^{H} \gamma^{i-1} r(x(t), u_\theta(x(t)))\right]. \tag{20}$$

The policy u_θ can be trained to maximize $\eta(u_\theta)$ using standard RL methods. In what follows, we provide details on the reward structure and data collection process. For notational simplicity, let $x_{j,i}$ represent the i-th sampled point along the trajectory under the backup policy u_{b_j}, starting from state x, i.e., $x_{j,i} = \phi_{u_{\mathrm{b}_j}}(x, iT_{\mathrm{s}})$. During data collection at state x, for each $j \in \mathbb{I}[\ell+1]$:

(*1*) Generate the trajectory under backup policy u_{b_j} to obtain $\{x_{j,i}\}_{i=0}^{N}$

(*2*) Compute $h_j(x)$ from (14).

(*3*) Assign the reward
$$r_j(x_{j,i}, u_{\mathrm{b}_j}(x_{j,i})) \triangleq h_j(x), \quad \text{for all } i \in \mathbb{I}[N]. \tag{21}$$
The reward function (21) essentially assigns the same reward to all the state-action pairs along the trajectory under backup policy u_{b_j}.

(*4*) Add the set of tuples $\left\{(x_{j,i}, u_{\mathrm{b}_j}(x_{j,i}), x_{j,i+1}, r_j)\right\}_{i=0}^{N}$ to the replay buffer.

Note that for $j = \ell + 1$, corresponding to the neural backup policy u_θ, the reward function matches the expression in (6), with the exception that the maximum function is replaced by the soft maximum, and h_{s} is evaluated at sampled times. This makes the reward essentially equal to $h_{\ell+1}(x)$. This formulation directly inspired the reward function in (21). Although the trajectories under the neural backup policy u_θ alone could suffice for training the backup policy, we utilized the trajectories generated under all backup policies during safety filtering to augment the training dataset without incurring additional computation. Specifically, the reward for these trajectories is given by the value of h_j for $j \in \mathbb{I}[\ell]$. By leveraging this reward, the policy is encouraged to maximize h_j for the neural policy. Algorithm 2 in the Supplementary Materials outlines the RLBUS algorithm.

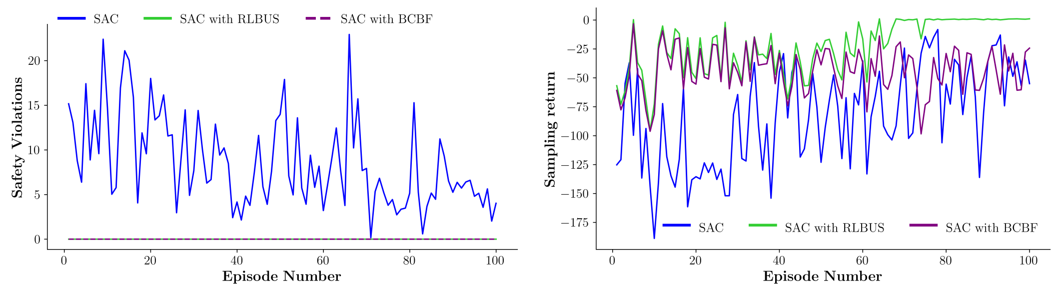
5. Numerical Results

Consider the inverted pendulum modeled by (1), where

$$f(x) = \begin{bmatrix} \dot\beta \\ \sin\beta \end{bmatrix}, \qquad g(x) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \qquad x = \begin{bmatrix} \beta \\ \dot\beta \end{bmatrix},$$

and β is the angle from the inverted equilibrium. Let $\bar u = 1.5$ and $\mathcal{U} = \{u \in \mathbb{R} \colon u \in [-\bar u, \bar u]\}$. The safe set $\mathcal{S}_{\mathrm{s}}$ is given by (3), where $h_{\mathrm{s}}(x) = 1 - \left\| \left[\begin{smallmatrix} \frac{1}{\pi - 0.5} & 0 \\ 0 & 0.5 \end{smallmatrix}\right] x \right\|_{100}$.

We consider two backup sets. Specifically, for $j \in \mathbb{I}[2]$, The backup safe set $\mathcal{S}_{\mathrm{b}_j}$ is given by (4), where $h_{\mathrm{b}_j}(x) = 0.02 - (x - x_{\mathrm{b}_j})^{\mathrm{T}} Pj (x - x_{\mathrm{b}_j})$, $P_1 = \left[\begin{smallmatrix} 0.625 & 0.125 \\ 0.125 & 0.125 \end{smallmatrix}\right]$, and $P2 = \left[\begin{smallmatrix} 0.650 & 0.150 \\ 0.150 & 0.240 \end{smallmatrix}\right]$. Next,



(*a*) Safety violation (*b*) Sampling Return

Figure 2: Safety violations and sampling returns of the performance agent across three scenarios: *(i)* SAC, *(ii)* SAC with BCBF, and *(iii)* SAC with RLBUS.

consider $u_{\mathrm{b}_j}(x) = \bar{u}\tanh\frac{1}{\bar{u}}K(x - x_{\mathrm{b}_j})$, where $x_{\mathrm{b}_1} \triangleq [0 \quad 0]^{\mathrm{T}}$, $x_{\mathrm{b}_2} \triangleq [\pi/2 \quad 0]^{\mathrm{T}}$. Lyapunov's direct method can be used to confirm that Assumption 1 is satisfied. The neural backup policy u_θ is trained using the approach described in Section 4, aiming to maximize the reward (21) to expand the identified control forward invariant subset of the safe set.

Figure 1 visualizes the sets $\mathcal{S}_{\mathrm{s}}$, $\mathcal{S}_{\mathrm{b}_1}$, $\mathcal{S}_{\mathrm{b}_2}$, $\mathcal{S}_{\mathrm{b}_3}$, $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$, and $\mathcal{R}$ at two different episodes. The set $\mathcal{R}$ is derived by solving (5) using the Hamilton-Jacobi-Bellman PDE, representing the largest finite-time safe backward image of $\bar{\mathcal{S}}_{\mathrm{b}}$. Figure 1(*a*) shows the initial state of the set $\mathcal{S}_3$ at the start of the first episode, while Figure 1(*b*) illustrates the expansion of $\mathcal{S}_3$ after 100 episodes of training the neural backup policy u_θ.

To evaluate the performance and safety violations of the proposed framework, we consider training a neural desired policy $u_{d_{\tilde{\theta}}} : \mathbb{R}^2 \to \mathcal{U}$ parameterized by $\tilde{\theta} \in \Theta$, trained as a RL agent to maximize the reward function $r_{\mathrm{p}} : \mathbb{R}^2 \times \mathcal{U} \to \mathbb{R}$ defined by $r_{\mathrm{p}}(\beta, \dot{\beta}, u) \triangleq -(\beta - 0.8)^2 - 0.1\dot{\beta}^2 - 0.001u^2$.

Both the neural policy u_θ and the neural desired policy $u_{d_{\tilde{\theta}}}$ are trained using the soft actor-critic method (Haarnoja et al., 2018).

The reward $r_{\mathrm{p}}(\beta, \dot{\beta}, u)$ is maximized at $u = 0$ and $x_{\mathrm{opt}} = [0.8 \quad 0]^T$. However, this state lies outside of $\mathcal{S}_1 \cup \mathcal{S}_2$, and achieving optimal performance requires expanding the forward invariant set. We evaluate the framework under three different scenarios: *(i)* SAC: training the neural desired policy without any safety constraints, *(ii)* SAC with BCBF: training the neural desired policy with a BCBF safety constraint using two backup policies u_{b_1} and u_{b_2}, and *(iii)* SAC with RLBUS: concurrently training the neural desired policy and a neural backup policy within the proposed framework with backup policies u_{b_1}, u_{b_2}, and u_θ.

Figure 2 compares safety violations and sampling returns across the three scenarios. Figure 2(*a*) shows that while scenario *(i)* exhibits multiple safety violations, scenarios *(ii)* and *(iii)* maintain safety throughout the training process. Additionally, Figure 2(*b*) demonstrates that scenarios *(iii)* achieve higher sampling returns compared to scenario *(ii)*. Initially, the x_{opt} lies outside $\bigcup_{j\in\mathbb{I}[3]} \mathcal{S}_j$, making it infeasible to reach. However, Figure 1 illustrates that as the forward invariant set expands, the optimal performance state becomes contained within $\mathcal{S}_3$. This expansion allows scenario *(iii)* to achieve higher sampling returns than scenario *(ii)*. Together, Figure 1 and Figure 2 demonstrate that the RLBUS expands the forward invariant set towards performance-optimal regions while maintaining safety throughout the training process.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017.

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8): 3861–3876, 2016.

Osbert Bastani, Shuo Li, and Anton Xu. Safe reinforcement learning via statistical model predictive shielding. In *Robotics: Science and Systems*, pages 1–13, 2021.

Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. *Advances in neural information processing systems*, 30, 2017.

Franco Blanchini. Set invariance in control. *Automatica*, 35(11):1747–1767, 1999.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *Journal of Machine Learning Research*, 18 (167):1–51, 2018.

Jaime F Fisac, Mo Chen, Claire J Tomlin, and S Shankar Sastry. Reach-avoid problems with time-varying dynamics, targets and constraints. In *Proceedings of the 18th international conference on hybrid systems: computation and control*, pages 11–20, 2015.

Thomas Gurriet, Mark Mote, Andrew Singletary, Petter Nilsson, Eric Feron, and Aaron D Ames. A scalable safety critical control framework for nonlinear systems. *IEEE Access*, 8:187249–187275, 2020.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Torsten Koller, Felix Berkenkamp, Matteo Turchetta, and Andreas Krause. Learning-based model predictive control for safe exploration. In *2018 IEEE conference on decision and control (CDC)*, pages 6059–6066. IEEE, 2018.

Milan Korda, Didier Henrion, and Colin N Jones. Convex computation of the maximum controlled invariant set for polynomial control systems. *SIAM Journal on Control and Optimization*, 52(5): 2944–2969, 2014.

Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on automatic control*, 50(7):947–957, 2005.

Pedram Rabiee and Jesse B. Hoagg. Soft-minimum barrier functions for safety-critical control subject to actuation constraints. In *2023 American Control Conference (ACC)*. IEEE, May 2023. doi: 10.23919/acc55779.2023.10156245. URL http://dx.doi.org/10.23919/ACC55779.2023.10156245.

Pedram Rabiee and Jesse B Hoagg. Composition of control barrier functions with differing relative degrees for safety under input constraints. In *2024 American Control Conference (ACC)*, pages 3692–3697. IEEE, 2024a.

Pedram Rabiee and Jesse B Hoagg. A closed-form control for safety under input constraints using a composition of control barrier functions. *arXiv preprint arXiv:2406.16874*, 2024b.

Pedram Rabiee and Jesse B Hoagg. Soft-minimum and soft-maximum barrier functions for safety with actuation constraints. *Automatica*, 171:111921, 2025.

Amirsaeid Safari and Jesse B Hoagg. Time-varying soft-maximum control barrier functions for safety in an a priori unknown environment. *arXiv preprint arXiv:2310.05261*, 2023.

Jens Schreiter, Duy Nguyen-Tuong, Mona Eberts, Bastian Bischoff, Heiner Markert, and Marc Toussaint. Safe exploration for active learning with gaussian processes. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part III 15*, pages 133–149. Springer, 2015.

Kim P Wabersich and Melanie N Zeilinger. Linear model predictive safety certification for learning-based control. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 7130–7135. IEEE, 2018.

Akifumi Wachi, Yanan Sui, Yisong Yue, and Masahiro Ono. Safe exploration and optimization of constrained mdps using gaussian processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Li Wang, Evangelos A Theodorou, and Magnus Egerstedt. Safe learning of quadrotor dynamics using barrier certificates. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2460–2465. IEEE, 2018.

Peter Wieland and Frank Allgöwer. Constructive safety using control barrier functions. *IFAC Proceedings Volumes*, 40(12):462–467, 2007.