Sampling-based Safe Reinforcement Learning for Nonlinear Dynamical Systems

Wesley A. Suttle
U.S. Army Research Laboratory

Vipul K. Sharma
Purdue University

Krishna C. Kosaraju
Clemson University

S. Sivaranjani
Purdue University

Ji Liu
Stony Brook University

Vijay Gupta
Purdue University

Brian M. Sadler
U.S. Army Research Laboratory

Abstract

We develop provably safe and convergent reinforcement learning (RL) algorithms for control of nonlinear dynamical systems, bridging the gap between the hard safety guarantees of control theory and the convergence guarantees of RL theory. Recent advances at the intersection of control and RL follow a two-stage, safety filter approach to enforcing hard safety constraints: model-free RL is used to learn a potentially unsafe controller, whose actions are projected onto safe sets prescribed, for example, by a control barrier function. Though safe, such approaches lose any convergence guarantees enjoyed by the underlying RL methods. In this paper, we develop a single-stage, sampling-based approach to hard constraint satisfaction that learns RL controllers enjoying classical convergence guarantees while satisfying hard safety constraints throughout training and deployment. We validate the efficacy of our approach in simulation, including safe control of a quadcopter in a challenging obstacle avoidance problem, and demonstrate that it outperforms existing benchmarks.

1 INTRODUCTION

Learning-based methods for safe control of physical systems have been gaining increasing attention (Brunke et al., 2022). RL is especially powerful for the control of systems where performance feedback in the form of a scalar reward is available, but the dynamics are unknown (Sutton and Barto, 2018). In such settings, RL methods can learn a controller maximizing reward through direct interaction with the environment. However, due to physical realities such as the need to guarantee safety, practical application of RL to control of physical systems requires constraints on the control policies throughout training (Garcıa and Fernández, 2015). While directly constraining the action space to a static, narrowly defined set of "safe" actions is frequently employed in practice, this can lead to learning highly suboptimal policies and more nuanced methods are therefore required. Furthermore, in most physical systems it is non-trivial to directly translate complex safety constraints on the states into allowable actions.

A variety of RL approaches to the problem of safe learning for control have been proposed in the literature (see Brunke et al. (2022) for a comprehensive survey), including RL methods for safety-focused problems formulated as constrained Markov decision processes (CMDPs) (Altman, 2021), methods for learning to achieve safety through stability (Berkenkamp et al., 2017), and projection-based – also known as "safety filter" – RL methods for maintaining hard safety constraints, typically achieved through the use of control barrier functions (CBFs) (Cheng et al., 2019). Though CMDP-based methods enjoy convergence guarantees, they encourage safety without guaranteeing it, and cannot provide guarantees for hard safety constraints commonly required in physical systems. Likewise, methods like Berkenkamp et al. (2017) do better by offering high-probability safety assurances, but stop short of guaranteeing safety. In systems where safety is critical, methods like Cheng et al. (2019) that prov-

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. Copyright 2024 by the author(s).

ably guarantee hard constraint satisfaction are necessary. However, the interaction between imposition of hard constraints and optimality of the resulting control policies is a subtle issue in RL. While projection-based safety-filter approaches (Wabersich et al., 2023) provably guarantee safety, the projection procedure undermines any convergence guarantees enjoyed by the underlying RL methods.

In this work, we develop a class of model-free policy gradient methods that maintain safety or other stability properties by sampling directly from the set of state-dependent safe actions. The key to our approach is that we consider *truncated* versions of commonly used stochastic policies, allowing us to sample directly from the safe action set at each state. This allows us to recover convergence guarantees by extending existing results for policy gradient methods to truncated policies. Our approach is applicable to a wide class of safety constraints including control barrier functions (CBFs), that enforce forward invariance of a set characterized by nonlinearly coupled states and actions (Ames et al., 2019, 2016), and reachability-type constraints (Wabersich et al., 2023). In addition to our theoretical results, we experimentally validate the practical utility of sampling-based safety-preservation methods by considering a special case: Beta policies with state-dependent, control barrier function (CBF)-constrained action sets. This novel approach extends the Beta policies in Chou et al. (2017) to the state-dependent action constraint setting. Finally, we train the resulting CBF-constrained Beta policies using PPO to solve a safety-constrained inverted pendulum problem as well as a quadcopter navigation and obstacle avoidance problem, and compare the latter to a safety filter-based benchmark.[1] These case studies illustrate that our method simultaneously guarantees safety throughout training and guarantees optimality, even where existing benchmarks fail.

1.1 Related Work

Safety and stability have seen a great deal of interest in recent years at the intersection of the RL and control communities (see Brunke et al. (2022); Garcıa and Fernández (2015) for overviews). We are interested in safety definitions that impose *hard constraints* on the states and control actions (rather than, e.g., those used in robust RL (Wiesemann et al., 2013; Aswani et al., 2013) or RL for CMDPs (Achiam et al., 2017; Paternain et al., 2019; Ma et al., 2021; Bai et al., 2022)). Model-based methods for guaranteeing stability using RL controllers in systems with known or learnable dynamics have been developed in Berkenkamp et al. (2017); Fazel et al. (2018); Zhang et al. (2021). Recently, techniques leveraging control barrier functions to maintain safety (Cheng et al., 2019) and dissipativity (Kosaraju et al., 2021) have been developed.

Our work lies in the model-free RL setting. The two dominant approaches in model-free RL are value function and policy gradient-based methods (Sutton and Barto, 2018). We focus on the latter in this paper. Since their origins early in the development of RL (Sutton et al., 2000; Borkar, 2005; Bhatnagar et al., 2009), policy gradient methods have become the model-free algorithms of choice for complex problems with continuous, high-dimensional state and action spaces (Lillicrap et al., 2015; Schulman et al., 2017; Haarnoja et al., 2018). Recent works have improved our understanding of gradient estimation procedures, global optimality properties, and convergence rates of these algorithms (Agarwal et al., 2020; Zhang et al., 2020; Suttle et al., 2023). Popular approaches for safety in model-free RL include using bounds resulting from Gaussian process models (Schreiter et al., 2015; Rasmussen, 2003; Sui et al., 2015), reward-shaping, constrained policy optimization (Achiam et al., 2017; Wachi et al., 2018), and teacher advice (Abbeel and Ng, 2004). Our work is most closely related to those approaches that use a hard safe set specification and constraints on control inputs, e.g., control barrier functions (Cheng et al., 2019; Fisac et al., 2018; Li et al., 2018; Kosaraju et al., 2021). In particular, our key contribution is a model-free safe RL algorithm with convergence guarantees and provable safety guarantees under hard constraints like CBFs, even during training.

2 PROBLEM SETTING

Consider a discounted MDP $(\mathcal{X}, \mathcal{U}, \mathcal{P}, r, \gamma)$, where $\mathcal{X} \subseteq \mathbb{R}^m$ is the state space, $\mathcal{U} \subseteq \mathbb{R}^n$ is the action space, $\mathcal{P}(\cdot|x, u)$ is the transition probability function given action u is taken in state x, $r : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor. The MDP, which can be used to model a wide array of discrete-time systems, proceeds as follows: at time k, the system is in state x_k; a control input u_k is applied to the system; a reward $r(x_k, u_k)$ is received; the system transitions into state x_{k+1} according to the distribution $P(\cdot|x_k, u_k)$. The goal in this problem formulation is to maximize the expected discounted reward, which we define in (3) below. Note that deterministic dynamics can be recovered by imposing that, for each $x \in \mathcal{X}, u \in \mathcal{U}$, there exists $x' \in \mathcal{X}$ such that $\mathcal{P}(x'|x, u) = 1$. This is useful for modeling discretizations of continuous-time control problems, for example. We assume throughout this paper that the dynamics are deterministic in this way, which is a common setting in safe control problems.

[1] Our implementation is publicly available at `https://github.com/sharma1256/cbf-constrained_ppo`.

Let $\mathcal{T} : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ represent the dynamics of the MDP, i.e., given state x and control input u, $\mathcal{T}(x, u)$ denotes the state the system transitions into when input u is applied while in state x.

Letting $\Delta(\mathcal{U})$ denote the set of all probability distributions over the set $\mathcal{U}$, a stochastic policy $\pi : \mathcal{X} \to \Delta(\mathcal{U})$ is a function mapping states to probability distributions over the action space $\mathcal{U}$. In other words, given a state x, an agent using policy π will choose a control action $u \in \mathcal{U}$ by sampling $u \sim \pi(\cdot|s)$. For our purposes it will be useful to consider policies π_θ parameterized by $\theta \in \Theta \subseteq \mathbb{R}^k$, for some $k \ll |\mathcal{X}| \cdot |\mathcal{U}| = m \cdot n$, where Θ is a compact set of permissible parameters.

Let $\mathcal{S} \subset \mathcal{X}$ denote some "safe" or stable set within which we wish to keep the system. Furthermore, let $\mathbb{P}(S)$ denote the powerset of a set S, and consider a set-valued function $C : \mathcal{X} \to \mathbb{P}(\mathcal{U})$ given by

$$C(x) = \{u \in \mathcal{U} \mid \mathcal{T}(x, u) \in \mathcal{S}\}. \tag{1}$$

Intuitively, $C(x)$ is the set of all control inputs which when applied at state x keep the system within the safe set at the next time step. We assume throughout that, for a given $x \in \mathcal{X}$, $C(x)$ is known. Since our primary focus is resolving the open problem of simultaneously guaranteeing convergence and hard safety constraint satisfaction, we leave the issue of learning or approximating $C(x)$ while maintaining these guarantees to future work. The general formulation can be used to accommodate a variety of notions of safety, including forward invariance, stability, and dissipativity enforced by, for example, CBFs and exponential CBFs (ECBFs), and control Lyapunov functions (CLFs) (see the supplementary material for an overview and Ames et al. (2019) for a comprehensive survey). As we will demonstrate in the case studies below, the use of our method in conjunction with (E)CBFs is particularly natural to provide guarantees in problems with hard safety constraints. To ensure that we can sample from $C(x)$ and integrals over $C(x)$ are well-defined, we make the following assumption. Let μ denote the Lebesgue measure.

Assumption 1. *There exist $m, M > 0$ such that $m \leq \mu(C(x)) \leq M$, for all $x \in \mathcal{X}$. Furthermore, $\cup_{x \in \mathcal{X}} C(x)$ is compact.*

Given a policy π_θ, consider the distribution $\pi_\theta^C(\cdot|x)$ obtained by truncating $\pi_\theta(\cdot|x)$ to the set $C(x)$. More precisely:

$$\pi_\theta^C(u|x) = \begin{cases} \frac{\pi_\theta(u|x)}{\pi_\theta(C(x)|x)} & u \in C(x) \\ 0 & u \notin C(x), \end{cases} \tag{2}$$

where $\pi_\theta(C(x)|x) = \int_{C(x)} \pi_\theta(u|x) du$. As long as we can check membership in $C(x)$ for any given $u \in \mathcal{U}$, and assuming that the volume of $C(x)$ is strictly positive, for all $x \in \mathcal{X}$, we can generate from this distribution by using rejection sampling, i.e. repeatedly sampling $u \sim \pi_\theta(\cdot|x)$ until we obtain $u \in C(x)$. Note that, depending on the structure of parametrized policies π_θ^C, if $C(x)$ has a particularly nice form, such as an interval or hyperrectangle, there may be more efficient methods than rejection sampling for sampling from the truncated distribution $\pi_\theta^C(\cdot|x)$ directly. We exploit this fact when leveraging Beta policies in the experimental results of Section 4 below.

With this setup in mind, and given a fixed start state x_0, we propose a policy gradient-based algorithm maximizing the objective function

$$J(\theta) = \mathbb{E}_{\pi_\theta^C}\left[\sum_{k=0}^{\infty} \gamma^k r(x_k, u_k) \;\Big|\; x_0\right], \tag{3}$$

the expected discounted reward under policy π_θ^C. Before proceeding with describing and analyzing the algorithm, we first need to identify conditions that ensure that, for each policy parameter θ, taking expectations with respect to π_θ^C is well-defined and thus meaningful. In order for (3) to be well-defined, we need to know that, for each policy parameter θ, the occupancy measure of the Markov chain induced by π_θ^C on $\mathcal{S}$ is irreducible and satisfies certain ergodicity conditions. Once these are proven, we will be justified in performing gradient ascent on the objective function (3).

3 THEORETICAL RESULTS

In this section we develop the theory underlying our sampling-based method for RL with hard safety constraints. Our key contributions include proving that (3) is well-defined (§3.1), obtaining gradient expressions for it from which we can sample (§3.2), and developing and establishing the convergence of a policy gradient algorithm for optimizing (3) (§3.3 and §3.4). All proofs are deferred to the supplementary material. It is important to note that, though we assumed the deterministic dynamics common to safe control in §2, all our theoretical results go through in the stochastic dynamics case under standard ergodicity assumptions. Our key theoretical contribution in what follows is to show that, even in the *deterministic* dynamics case, we can ensure that the objective is well-defined (§3.1) and obtain convergence (§3.4).

3.1 Discounted Return is Well-defined

First, we show that the objective (3) is well-defined when using truncated policies, even in continuous spaces systems with deterministic dynamics. Our key contribution in this setting is to ensure that, given reasonable conditions on the policies under consideration, important ergodicity properties of their induced Markov chains hold. This fact, established in Proposition 1 and Corollary 1, is nontrivial and its proof relies on a careful analysis of the propagation of probability mass through the transition dynamics and an interesting application of the Lebesgue-Radon-Nikodym Theorem (Folland, 1999, §3.2). As in the previous section, let $\mathcal{S} \subset \mathbb{R}^m$ denote the "safe" set within which the system remains so long as all control inputs are selected from $C(x)$. Though we leave open the possibility that $\mathcal{S}$ satisfies a more specific stability conditions rather than a generic notion of "safety", we will typically use the term "safe" for ease of presentation. Let μ denote Lebesgue measure. We make the following definition:

Definition 1. *The Markov chain $\{x_k\}_{k\in\mathbb{N}}$ induced by π_θ^C on $\mathcal{X}$ is μ-irreducible on $\mathcal{S}$ if, for any μ-measurable $\mathcal{B} \subset \mathcal{S}$, if $\mu(\mathcal{B}) > 0$, then $\sum_{k\in\mathbb{N}} P(x_k \in \mathcal{B} \mid x_0 = x) > 0$, for all $x \in \mathcal{S}$.*

This means that, for a Markov chain to be (μ-)irreducible on the safety set, all safe subsets with positive volume must be reachable from any initial safe state with positive probability. Notice that $\{x_k\}_{k\in\mathbb{N}}$ is in fact a Markov chain on the safe set $\mathcal{S}$ since, by the definition of $C(x)$, only those control inputs keeping the system within $\mathcal{S}$ are allowed. In the sequel, we will prove that, for each θ, under suitable conditions the Markov chain induced by π_θ^C on $\mathcal{S}$ is irreducible and the objective (3) is thus well-defined, which is a prerequisite for developing policy gradient methods based on it. See (Konda, 2002, §2.3) for details on irreducibility in this setting.

Given an element $x \in \mathcal{S}$ and dynamics $\mathcal{T}$, let $R(x) \subset \mathcal{S}$ consisting of all elements reachable in one step from x under $\mathcal{T}$. Furthermore, for $\mathcal{A} \subset \mathcal{S}$, define $R(\mathcal{A}) = \cup_{x\in\mathcal{A}} R(x)$. Also, given $\varepsilon > 0$ and $x \in \mathbb{R}^m$, let $B_\varepsilon(x)$ denote the open ball of radius ε centered at x. Finally, for $\mathcal{A} \subset \mathcal{S}$, define $\mathcal{T}_x^{-1}(\mathcal{A}) := \{u \in \mathcal{U} \mid T(x,u) \in \mathcal{A}\}$. Intuitively, $\mathcal{T}_x^{-1}(\mathcal{A})$ is the set of all control inputs that, when taken in state x, drive the system into $\mathcal{A}$. The following assumptions are needed in what follows.

Assumption 2. *For any $x \in \mathcal{S}$ and any μ-measurable set $\mathcal{A} \subset R(x)$, $\mu(\mathcal{A}) > 0$ if and only if $\mu(\mathcal{T}_x^{-1}(\mathcal{A})) > 0$.*

Assumption 2 ensures the system dynamics map positive volume subsets of control inputs to positive volume subsets of the state space and vice versa, which is important for our application of the Lebesgue-Radon-Nikodym Theorem in Proposition 1. It is satisfied by systems where control inputs have a measurable effect on each entry in the next state vector and thus encompasses a wide array of potentially nonlinear systems.

Assumption 3. *For any $\theta \in \Theta$, where Θ is the set of permissible policy parameters, for any element in the safe set $x \in \mathcal{S}$, and for any set $\mathcal{A} \subset C(x)$ satisfying $\mu(\mathcal{A}) > 0$, the policy $\pi_\theta^C(\cdot|x)$ assigns positive probability to $\mathcal{A}$, i.e. $\int_{\mathcal{A}} \pi_\theta^C(a|x)da > 0$.*

Assumption 3, which is standard in the RL literature, ensures that any set of allowable control inputs that has strictly positive volume will be sampled from with strictly positive probability.

Assumption 4. *For each $x \in \mathcal{S}$, $\mu(R(x)) > 0$, and, given $\mathcal{B} \subset \mathcal{S}$, there exists $n \in \mathbb{N}$ such that $\mathcal{B}$ is reachable in n steps from x.*

The conditions imposed in Assumption 4 guarantee that, for any state $x \in \mathbb{X}$: (i) the set of states reachable from x in one step has strictly positive volume; (ii) any subset of the safe set $\mathcal{S}$ is reachable in at most n steps from x. These conditions are closely related to the familiar notion of controllability of control theory. Under these conditions, we have the following proposition and its immediate corollary.

Proposition 1. *Under Assumptions 2, 3, 4, for given θ and any subset $\mathcal{B} \subset \mathcal{S}$ satisfying $\mu(\mathcal{B}) > 0$, the Markov chain induced by π_θ^C on $\mathcal{S}$ enters $\mathcal{B}$ with strictly positive probability.*

Corollary 1. *$\{x_n\}$ is μ-irreducible on $\mathcal{S}$.*

Now that we are assured that the objective function (3) is well-defined, we are justified in attempting to perform gradient ascent on it. In order to accomplish this, however, we need access to gradient estimates. This is the subject of the next section.

3.2 Policy Gradients

Despite the presence of C in π_θ^C, under mild assumptions on the underlying policy π_θ, we can apply the classic policy gradient theorem of Sutton et al. (2000) to (3) to obtain a gradient expression from which we can sample. Let $d_\theta^C(\cdot) := (1-\gamma)\sum_{k=0}^\infty \gamma^t P(x_k \in \cdot \mid \pi_\theta^C)$ denote the discounted state occupancy measure of the Markov chain induced by policy π_θ^C on $\mathcal{S}$. Furthermore, let $Q^{\pi_\theta^C}(x,u) = \mathbb{E}_{\pi_\theta^C}\left[\sum_{k=0}^\infty \gamma^k r(x_k,u_k) \mid x_0 = x, u_0 = u\right]$. We make the following assumption:

Assumption 5. *$\pi_\theta(u|x) > 0$ and $\pi_\theta(u|x)$ is differentiable in θ, for all $x \in \mathcal{X}, u \in \mathcal{U}$.*

Recall from (2) that $\pi_\theta^C(\cdot|x)$ is simply the probability density function $\pi_\theta(\cdot|x)$ truncated to the set $C(x)$. Note, since the value of $C(x)$ at a given x is independent of θ, we can take the derivative inside the integral sign in the latter expression to obtain $\nabla\pi_\theta(C(x)|x) = \int_{C(x)} \nabla\pi_\theta(u|x)du$, so $\pi_\theta^C(C(x)|x)$ is differentiable. Given these facts, combined with Assumption 5, the above expression for π_θ^C implies that, for any $x \in \mathcal{S}$, the policy $\pi_\theta^C(u|x)$ is differentiable with respect to θ, for any $u \in C(x)$. In short, π_θ^C satisfies its own version of Assumption 5, which we formalize in the following:

Lemma 1. *$\pi_\theta^C(u|x) > 0$ and $\pi_\theta^C(u|x)$ is differentiable in θ, for all $x \in \mathcal{S}$ and $u \in C(x)$.*

The policy gradient theorem (Konda, 2002) implies

$$\nabla J(\theta) = \frac{1}{1-\gamma} \mathbb{E}_{\pi_\theta^C} \left[Q^{\pi_\theta^C}(x,u) \nabla \log \pi_\theta^C(u|x) \right]. \quad (4)$$

In order to carry out gradient updates based on this expression, we first need to be able to estimate $\nabla_\theta \log \pi_\theta^C(u|x) = \nabla_\theta \pi_\theta^C(u|x)/\pi_\theta^C(u|x)$, for arbitrary u, x. We will discuss how to estimate $Q^{\pi_\theta^C}(x,u)$ in an unbiased manner in the following section. Since we already have access to $\pi_\theta^C(u|x)$, we can focus on estimating $\nabla_\theta \pi_\theta^C(u|x)$. Based on (2), the gradient of $\pi_\theta^C(u|x)$ with respect to θ is

$$\nabla \pi_\theta^C(u|x) = \nabla \left[\frac{\pi_\theta(u|x)}{\pi_\theta(C(x)|x)} \right] \quad (5)$$

$$= \frac{\nabla \pi_\theta(u|x)}{\pi_\theta(C(x)|x)} - \frac{\pi_\theta(u|x)}{[\pi_\theta(C(x)|x)]^2} \nabla \pi_\theta(C(x)|x) \quad (6)$$

$$= \frac{1}{\pi_\theta(C(x)|x)} \left[\nabla \pi_\theta(u|x) - \pi_\theta(u|x) \nabla \log \pi_\theta(C(x)|x) \right]. \quad (7)$$

To estimate $\pi_\theta(C(x)|x)$, we need to be able to estimate $\int_{C(x)} \pi_\theta(u|x)du$. Given access to $\pi_\theta(\cdot|x)$ and $C(x)$, we can use numerical integration or Monte Carlo techniques to approximate this integral. The standard Monte Carlo approach is to uniformly sample M elements $u_i \sim U(C(x))$ from $C(x)$, then estimate

$$\widehat{\pi_\theta}(C(x)|x) = \mu(C(x)) \frac{1}{M} \sum_{i=1}^{M} \pi_\theta(u_i|x), \quad (8)$$

where $\mu(C(x))$ is the volume of $C(x)$. This estimate is based on the fact that

$$\pi_\theta(C(x)|x) = \int_{C(x)} \pi_\theta(u|x)du \quad (9)$$

$$= \mu(C(x)) \int_{C(x)} \frac{\pi_\theta(u|x)}{\mu(C(x))} du \quad (10)$$

$$= \mu(C(x)) E_{u \sim U(C(x))}[\pi_\theta(u|x)] \quad (11)$$

$$= \mu(C(x)) \lim_{M \to \infty} \frac{1}{M} \sum_{i=1}^{M} \pi_\theta(u_i|x), \quad (12)$$

where the last equality holds by the law of large numbers. Since $C(x)$ is fixed given x, gradient estimates $\widehat{\nabla \log \pi_\theta}(C(x)|x)$ and ultimately $\widehat{\nabla \log \pi_\theta^C}(u|x)$ can also be obtained by estimating the integral $\int_{C(x)} \nabla \pi_\theta(u|x)du$. In the Monte Carlo situation, this can be obtained from (8) by differentiating each term with respect to θ.

3.3 Algorithm

In this section, we present a hard safety-constrained random-horizon policy gradient (Safe-RPG) algorithm. Our algorithm is based on the random-horizon policy gradient (RPG) scheme developed in Zhang et al. (2020), which uses a random rollout horizon and recent advances in non-convex optimization to obtain unbiased policy gradient estimates and ensure finite-time convergence to approximately locally optimal policies. As discussed in the following section, our convergence results ensure asymptotic convergence of Algorithm 2 to a stationary point of (3), but can likely be strengthened to prove finite-time convergence to approximately locally optimal policies. The main algorithm is presented in Algorithm 2, which depends on the action-value function estimation subroutine in Algorithm 1.

3.4 Convergence

In this section we show asymptotic convergence of Algorithm 2 to the set of stationary points of (3). The key challenge in this result revolves around the need to establish that the policies we consider satisfy important differentiability and continuity properties, which necessitates a careful analysis of the Lipschitz properties of the score functions of our truncated policies in the proof of Lemma 2. To proceed, we need the following assumption on the reward function r and underlying, untruncated policy class $\{\pi_\theta\}_{\theta \in \Theta}$.

Assumption 6. *The reward function r and parameterized policy class $\{\pi_\theta\}_{\theta \in \Theta}$ satisfy the following:*

1. *The absolute value of the reward r is uniformly bounded, i.e., there exists U_r such that $0 \leq \sup_{(x,u) \in \mathcal{X} \times \mathcal{U}} |r(x,u)| \leq U_r$.*
2. *For all $x \in \mathcal{X}, u \in \mathcal{U}$, $\nabla \log \pi_\theta(u|x)$ exists, and there exist $L_\Theta \geq 0$ and $B_\Theta \geq 0$ such that, for all $x \in \mathcal{X}, u \in \mathcal{U}$,*
    (a) $\|\nabla \log \pi_\theta(u|x) - \nabla \log \pi_{\theta'}(u|x)\| \leq L_\Theta \|\theta - \theta'\|$, for all $\theta, \theta' \in \Theta$
    (b) $\|\nabla \log \pi_\theta(u|x)\| \leq B_\Theta$, for all $\theta \in \Theta$.

Assumptions 5 and 6 were used to prove asymptotic convergence of the RPG algorithm with *untruncated* policies to stationary points in (Zhang et al., 2020, Theorem 4.4). For an analogous result to apply to the truncated policies we consider, it must be shown that the Lipschitz and differentiability conditions in part 2 of Assumption 6 hold for the constrained policies $\{\pi_\theta^C\}_{\theta \in \Theta}$. It turns out that, under the same conditions on the untruncated policy $\{\pi_\theta\}_{\theta \in \Theta}$, these properties are automatically satisfied for $\{\pi_\theta^C\}_{\theta \in \Theta}$.

Algorithm 1: `EstQ`: Unbiasedly Estimating Q

```
Data: $x, u, \theta$.
Result: Unbiased estimate of $Q^{\pi_\theta^C}(x, u)$.
1  Initialization: Sample $T \sim \text{Geom}(1-\gamma^{1/2})$ and
   initialize $\hat{Q} \leftarrow 0, x_0 \leftarrow x, u_0 \leftarrow u$.
2  for $t = 0, \dots, T-1$ do
3      $\hat{Q} \leftarrow \hat{Q} + \gamma^{t/2} r(x_t, u_t)$
4      $x_{t+1} \sim \mathcal{P}(\cdot|x_t, u_t)$
5      $u_{t+1} \sim \pi_{\theta_k}^C(\cdot|x_{t+1})$
6  end
7  $\hat{Q} \leftarrow \hat{Q} + \gamma^{T/2} r(x_T, u_T)$
8  return $\hat{Q}$
```

Algorithm 2: `Safe-RPG`: Hard safety-constrained Random-horizon Policy Gradient

```
Data: $x_0, \theta_0$, Monte Carlo sample size $M$.
Result: Locally optimal policy
1  Initialization: Set $k \leftarrow 0$.
2  repeat
3      Sample $T_{k+1} \sim \text{Geom}(1-\gamma)$, $u_0 \sim \pi_{\theta_k}^C(\cdot|x_0)$.
4      for $t = 0, \dots, T_{k+1}-1$ do
5          $x_{t+1} \sim \mathcal{P}(\cdot|x_t, u_t)$
6          $u_{t+1} \sim \pi_{\theta_k}^C(\cdot|x_{t+1})$
7      end
8      $\hat{Q}^{\pi_{\theta_k}^C}(x_{T_{k+1}}, u_{T_{k+1}}) = \texttt{EstQ}(x_{T_{k+1}}, u_{T_{k+1}}, \theta_k)$
9      Uniformly sample $\{u_l\}_{l=1,\dots,M}$ from $C(x_{T_{k+1}})$,
        then use them to compute
        $\widehat{\nabla \log \pi_{\theta_k}^C}(u_{T_{k+1}}|x_{T_{k+1}})$
10     $\theta_{k+1} \leftarrow \theta_k +$
        $\frac{\alpha_k}{1-\gamma} \hat{Q}^{\pi_{\theta_k}^C}(x_{T_{k+1}}, u_{T_{k+1}}) \widehat{\nabla \log \pi_{\theta_k}^C}(u_{T_{k+1}}|x_{T_{k+1}})$
11     $k \leftarrow k+1$
12 until convergence;
```

Lemma 2. *Under Assumptions 1, 5, and 6, $\nabla \log \pi_\theta^C(u|x)$ exists, for all $x \in \mathcal{X}, u \in \mathcal{U}$. Furthermore, there exist constants $L_\Theta^C \geq 0$ and $B_\Theta^C \geq 0$ such that, for all $x \in \mathcal{X}, u \in \mathcal{U}$,*
(i) $\left\|\nabla \log \pi_\theta^C(u|x) - \nabla \log \pi_{\theta'}^C(u|x)\right\| \leq L_\Theta^C \|\theta - \theta'\|$, for all $\theta, \theta' \in \Theta$, and
(ii) $\left\|\nabla \log \pi_\theta^C(u|x)\right\| \leq B_\Theta^C$, for all $\theta \in \Theta$.

With Lemma 2 in hand, we have the following result.

Theorem 1. *Let Assumptions 3, 4, 5, and 6 hold. Let $\{\theta_k\}_{k\in\mathbb{N}}$ be the sequence generated by Algorithm 2 with stepsize sequence $\{\alpha_k\}_{k\in\mathbb{N}}$ satisfying $\sum_{k=0}^\infty \alpha_k = \infty$ and $\sum_{k=0}^\infty \alpha_k^2 < \infty$. Then $\lim_k \theta_k \in \Theta^*$, where Θ^* is the set of stationary points of* (3).

Remark 1. By arguments analogous to those in the proof of (Zhang et al., 2019, Thm. 3), it can also be shown that, under the same assumptions as in Theorem 1 and appropriate stepsize selection, Algorithm 2 achieves ε-approximate first-order stationarity with a finite-time sample complexity of $\mathcal{O}(\varepsilon^{-2})$.

Given Lemma 2, the proof of the theorem follows directly from that of (Zhang et al., 2020, Theorem 4.4). With suitable modifications to Algorithm 2 incorporating periodically increasing stepsizes, these results can likely be strengthened to obtain finite-time convergence to an ε-locally optimal policy using the machinery developed in Zhang et al. (2020). We leave this to future work.

4 EXPERIMENTAL RESULTS

We now experimentally demonstrate the effectiveness of our sampling-based safe RL approach. Specifically, we evaluate the use of CBF-constrained Beta policies combined with the popular Proximal Policy Optimization (PPO) (Schulman et al., 2017) algorithm on safety-constrained inverted pendulum and quadcopter navigation environments. The use of Beta policies with variable action space constraints allows us to directly sample from a CBF-constrained action space at each timestep. In addition to providing a practical example of how truncated policies can be used to ensure safety, this method extends the work Chou et al. (2017) on the use of Beta policies for deep RL from constant to state-dependent action space constraints.

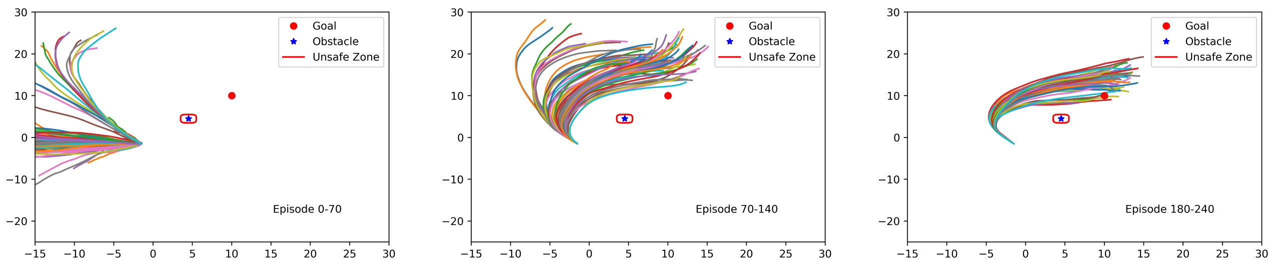
4.1 CBF-Constrained Beta Policies

When actions must be restricted to lie within fixed, predetermined bounds due to physical or numerical constraints, the common practice of simply clipping policies with infinite support (e.g., Gaussian policies) can cause bias and performance issues. Chou et al. (2017) propose and leverage finite-support Beta distribution-based policies to overcome these issues. We extend this approach to obtain policies that sample directly from the safe control actions prescribed by the CBF at a given state.

In order to describe these CBF-constrained Beta policies, let us first recall the probability density function (p.d.f.) of a one-dimensional Beta distribution:

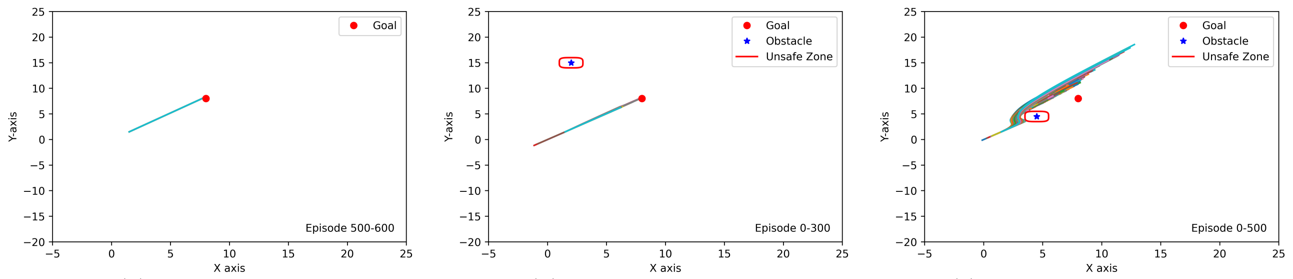
$$f(u; \alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} u^{\alpha-1}(1-u)^{\beta-1}, \tag{13}$$

where $u \in [0,1]$, $\alpha, \beta > 0$, and $\Gamma(z) = \int_0^\infty u^{z-1} e^{-u} du$ is the Gamma function defined for $z \in \mathbb{C}$ with $\text{Re}(z) > 0$. A Beta policy sampling from the fixed interval $[0,1]$ is given by $f(u; \alpha_\theta(x), \beta_\theta(x))$, where $\alpha_\theta, \beta_\theta : \mathcal{X} \to \mathbb{R}^+$ are parameterized functions (e.g., neural networks) mapping states to the parameters α, β of the Beta distribution. When the action space $\mathcal{U} \subset \mathbb{R}^n$



(a) Initial exploration. (b) Discovering goal direction. (c) Reaching the goal.

Figure 1: **Safety and convergence of CBF-Constrained Beta policies:** Agent was trained with PPO on the quadrotor navigation problem with an obstacle. Safety was maintained and goal was eventually reached.



(a) No obstacle. (b) Distant obstacle. (c) Interfering obstacle.

Figure 2: **Failure of benchmark safety-filtered Gaussian policies:** Agents were trained with PPO using three different obstacle configurations. When the obstacle is distant or non-existent, the method succeeds. When the obstacle is in the way, the resulting policy is suboptimal. In all cases, safety is maintained.

is of dimension $n \geq 2$ and the CBF constraint set (or an inner approximation of it), $C(x)$, can be expressed as a hyperrectangle with lower and upper bounds $a(x), b(x) \in \mathbb{R}^n$, respectively, we maintain independent Beta distributions, $f^i(\cdot; \alpha^i_\theta(x), \beta^i_\theta(x))$, over each dimension i of the unit box $[0,1]^n$, and samples from these distributions are shifted and rescaled to lie within the bounds given by $a(x), b(x)$. Specifically, our CBF-constrained Beta policies, denoted π_θ, sample $u \sim \pi_\theta(\cdot|x)$ from $C(x)$ by first sampling $\hat{u}^i \sim f^i(\cdot; \alpha^i_\theta(x), \beta^i_\theta(x))$, then performing the simple transformation $u = a(x) + \text{diag}(\hat{u}^1, \ldots, \hat{u}^n)(b(x) - a(x))$, where $\text{diag}(\hat{u}^1, \ldots, \hat{u}^n)$ denotes the diagonal matrix with elements $\hat{u}^1, \ldots, \hat{u}^n$ along the diagonal.

4.2 Implementation

We now describe the implementation details of our Beta policies. For a given state x, the parameter vectors $\alpha(x), \beta(x)$ are outputted by a two-layer, fully connected neural network. Control inputs at state x were obtained by first creating an independent PyTorch (Paszke et al., 2019) Beta distribution object with parameters $\alpha^i_\theta(x), \beta^i_\theta(x)$, for each dimension $i \in \{1, \ldots, n\}$ of the action space, then sampling $u = [u^1 \ldots u^n]^T$ from these distributions, and finally scaling and translating to lie within the current CBF set $C(x)$. Similarly, the Gaussian policies we used for comparison used distribution parameters outputted by a two-layer, fully connected neural network. Control inputs were then selected from the corresponding distribution by sampling, then following the standard practice (Chou et al., 2017) of clipping to a fixed set of permissible controls. The PPO implementation used in the experiments was adapted with minor modifications from Stable Baselines 3 (Raffin et al., 2021).

4.3 Case study 1 : Quadcopter Navigation

Experiment Setup. For this experiment, we consider the problem of learning to safely navigate a quadcopter around an obstacle to a goal location. In this section, we present an overview of the dynamical model that we use for this quadcopter, which was previously considered in Xu and Sreenath (2018), and describe our derivation of a hyperrectangular inner approximation of the safe control set, satisfying the CBF condition, that is amenable to sampling using our Beta policies. We finally briefly describe the reward function. See the supplementary material for a detailed exposition of the environment and sampling procedure.

We denote quadcopter and obstacle position by $r = (r_x, r_y, r_z)$ and $r_{obs} = (r_{o_x}, r_{o_y}, r_{o_z})$, respectively, and the quadcopter's relative position with respect to the obstacle as $\Delta r = r - r_{obs}$. The quadrotor dynamics are then given by

$$\dot{x} = Ax + Bu, x = \begin{bmatrix} r \\ \dot{r} \end{bmatrix}, A = \begin{bmatrix} \mathbf{0}_{3\times3} & I_3 \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix}, B = \begin{bmatrix} \mathbf{0}_{3\times3} \\ \mathbf{1}_{3\times3} \end{bmatrix},$$

with input u consisting the desired accelerations in each x, y, and z dimensions. For the obstacle avoidance problem, we characterize the safe set as $\mathcal{S} = \{r : h(r) \geq 0\}$, where

$$h(r) = (\Delta r_x/a)^4 + (\Delta r_y/b)^4 + (\Delta r_z/c)^4 - r_s, \quad (14)$$

$a, b, c > 0$ parameterize the obstacle's shape, which is assumed to be elliptical, and r_s represents the desired safety margin. Given the quadcopter dynamics and (14), the first time derivative $\dot{h}(r)$ does not explicitly contain the control input u. We therefore use the standard ECBF formulation Ames et al. (2019) to develop our safety condition using $\ddot{h}(r)$, which explicitly contains u. This ECBF condition is expressed as $\ddot{h} + K \cdot [h \quad \dot{h}]^T \geq 0$, where $K = [K_1 \quad K_2]^T, K_1 = 6, K_2 = 8$, are application-specific design parameters, and can be rewritten as $A_r u \leq b_r$, where A_r is a matrix and b_r is a vector, both depending on r. Then, the state-dependent safe control set is given by $C(r) = \{u \ : \ A_r u \leq b_r\}$ (see the supplementary for details). The dynamics (and consequently the ECBF conditions) are discretized with time step $dt = 0.1$.

We consider navigation in the x, y dimension as in Xu and Sreenath (2018), resulting in a two-dimensional action space. We take the actuator constraint to be defined by the hyperrectangle $H := \{u_{min}, u_{max}\}$, where $u_{min} := (u^x_{min}, u^y_{min}) \in \mathbb{R}^2$ and $u_{max} := (u^x_{max}, u^y_{max}) \in \mathbb{R}^2$ are the minimum and maximum input values. In order to sample from the safe control set $C(r)$ at a given r with our Beta policies, we need a hyperrectangular inner approximation. We obtain this inner approximation by formulating and solving a convex optimization problem yielding the highest volume hyperrectangle, $H_c(r)$, contained within $C(r)$.

Finally, we designed a reward providing an ℓ_2 penalty based on agent distance from the goal, as well as a sizeable bonus for reaching the goal and a significant penalty for approaching the edge of the map. See the supplementary for details.

Results. The experiments we conducted illustrate that safety-filter based approaches like those considered in Cheng et al. (2019) fail on simple cases of our safety-constrained quadcopter problem (see Figure 2), while our CBF-constrained Beta policy succeeds (Figure 1). For illustration purposes, Figures 1 and 2 present trajectories generated over the course of training. The corresponding learning curves are included in the supplementary material. As illustrated in Figure 2, the safety-filter approach is effective at ensuring safety and also learns to successfully reach the goal when the obstacle is nonexistent or distant. However, it ultimately fails to reach the goal when the obstacle lies directly between the start and goal positions. We hypothesize that this is due to the fact that the projection-based approach attempts to learn an optimal policy for the unconstrained navigation problem, while projection causes it to deviate from its learned policy to maintain safety. Furthermore, the resulting safety-filtered policy cannot recover from these projections without an additional control layer (such as a derivative or PID controller) due to the repeated perturbation from the projection procedure. Our method, on the other hand, learns to successfully solve the problem as shown in Figure 1 while maintaining safety throughout training, since we directly learn policies for the CBF-constrained problem.

4.4 Case study 2: Inverted pendulum

Experiment Setup. For the second set of experiments, we considered a safety-constrained inverted pendulum environment building on the baseline Gym implementation (Brockman et al., 2016). The goal in this environment is to swing an inverted pendulum upright while maintaining it within a fixed safe set. We tested PPO with the two different policies on this environment for two different safe sets: $\mathcal{S}_{0.5} = \{\theta \mid -0.5 \leq \theta \leq 0.5\}$ and $\mathcal{S}_{1.0} = \{\theta \mid -1.0 \leq \theta \leq 1.0\}$. Due to space limitations, we include the experiments with $\mathcal{S}_{0.5}$ in Figure 3 and the experiments with $\mathcal{S}_{1.0}$ with the supplementary material. As a baseline, we compare the proposed method to PPO with unconstrained Gaussian policies. This comparison highlights the effectiveness of the proposed method in guaranteeing safety as well as accelerating learning.

Results. Our experiments are summarized in Figure 3. There are two main points to be drawn from these results. First, the top panel shows that incorporating prior knowledge about properties such as safety can encourage learning and accelerate convergence by forcing the Beta policy agent to concentrate on higher-value subsets of the state space. The Gaussian agent, on the other hand, is unable to benefit from this prior knowledge and convergence suffers as it spends a greater portion of its time exploring lower-value regions of the state space. Second, the bottom panel illustrates that Beta policies are highly effective at maintaining safety throughout training, while Gaussian policies without safety constraints naturally fail to remain inside the safe set. This is expected, but illustrates the need to use constraint-aware policies such as Beta policies when prior knowledge is available.

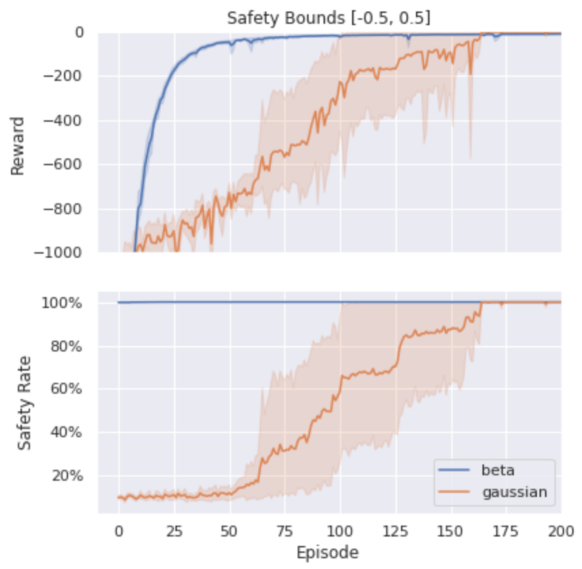


Figure 3: CBF-constrained Beta vs. unconstrained Gaussian on inverted pendulum environment with safe set $\mathcal{S}_{0.5} = \{\theta \mid -0.5 \leq \theta \leq 0.5\}$. "Safety Rate" denotes percentage of time spent in safe set. Curves present mean and 95% confidence intervals over 5 replications.

5 CONCLUSION

We have developed a sampling-based approach to learning policies ensuring hard constraint satisfaction in RL. Unlike existing, projection-based methods that ensure safety but lack convergence guarantees, our scheme provably does both. In addition to our theoretical contributions, we have also presented a practical solution method that leverages CBF-constrained Beta policies to ensure safety, and experimentally demonstrated its effectiveness on safe quadcopter navigation and inverted pendulum environments. Interesting directions for future work including extensions to the case where the constraint set must be estimated and application of our CBF-constrained Beta policies to real-world robotics problems.

Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful comments and Mostafa Mohamed Fa Abdelnaby of Purdue University for pointing out Remark 1. The work of W. A. Suttle was supported by a Distinguished Postdoctoral Fellowship with the U.S. Army Research Laboratory. V. K. Sharma was partially funded by a grant from the Purdue Engineering Initiative on Autonomous and Connected Systems. The work of J. Liu was supported in part by the National Science Foundation (NSF) under grant 2230101, by the Air Force Office of Scientific Research (AFOSR) under award number FA9550-23-1-0175, and by U.S. Air Force Task Order FA8650-23-F-2603. The work of K. C. Kosaraju and V. Gupta was partially supported by Army Research Office grants W911NF2310111, W911NF2310266, and W911NF-23-1-0316, AFOSR grant F.10052139.02.005, Office of Naval Research grants F.10052139.02.009 and F.10052139.02.012, and NSF grant 2300355.

References

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine learning*, 2004.

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

Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. Optimality and approximation with policy gradient methods in markov decision processes. In *Conference on Learning Theory*, pages 64–66. PMLR, 2020.

Ayush Agrawal and Koushil Sreenath. Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation. In *Robotics: Science and Systems*, volume 13, pages 1–10, 2017.

Eitan Altman. *Constrained Markov Decision Processes*. Routledge, 2021.

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.

Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European Control Conference*, pages 3420–3431, 2019.

Anil Aswani, Humberto Gonzalez, S Shankar Sastry, and Claire Tomlin. Provably safe and robust learning-based model predictive control. *Automatica*, 49(5):1216–1226, 2013.

Qinbo Bai, Amrit Singh Bedi, Mridul Agarwal, Alec Koppel, and Vaneet Aggarwal. Achieving zero constraint violation for constrained reinforcement learning via primal-dual approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 3682–3689, 2022.

Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Advances in Neural Information Processing Systems*, pages 908–918, 2017.

Shalabh Bhatnagar, Richard Sutton, Mohammad Ghavamzadeh, and Mark Lee. Natural actor-critic algorithms. *Automatica*, 45(11):2471–2482, 2009.

Vivek S Borkar. An actor-critic algorithm for constrained Markov decision processes. *Systems & Control Letters*, 54(3):207–213, 2005.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. `OpenAI Gym`. *arXiv preprint arXiv:1606.01540*, 2016.

Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:411–444, 2022.

Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W 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*, volume 33, pages 3387–3395, 2019.

Po-Wei Chou, Daniel Maturana, and Sebastian Scherer. Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution. In *International Conference on Machine Learning*, pages 834–843. PMLR, 2017.

Steven Diamond and Stephen Boyd. CVXPY: A python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research*, 17(1):2909–2913, 2016.

Maryam Fazel, Rong Ge, Sham Kakade, and Mehran Mesbahi. Global convergence of policy gradient methods for the linear quadratic regulator. In *International Conference on Machine Learning*, pages 1467–1476. PMLR, 2018.

Jaime F Fisac, Anayo K Akametalu, Melanie N Zeilinger, Shahab Kaynama, Jeremy Gillula, and Claire J Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7): 2737–2752, 2018.

Gerald B Folland. *Real Analysis: Modern Techniques and Their Applications*, volume 40. John Wiley & Sons, 1999.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

Cherie* Ho, Katherine* Shih, Jaskaran Grover, Changliu Liu, and Sebastian Scherer. "Provably safe" in the wild: testing control barrier functions on a vision-based quadrotor in an outdoor environment. In *RSS 2020 Workshop in Robust Autonomy*, 2020. URL `https://openreview.net/pdf?id=CrBJIgBr2BK`.

V. Konda. *Actor-Critic Algorithms*. PhD thesis, Massachusetts Institute of Technology, 2002.

Krishna Chaitanya Kosaraju, Seetharaman Sivaranjani, Wesley Suttle, Vijay Gupta, and Ji Liu. Reinforcement learning based distributed control of dissipative networked systems. *IEEE Transactions on Control of Network Systems*, 9(2):856–866, 2021.

Zhaojian Li, Uroš Kalabić, and Tianshu Chu. Safe reinforcement learning: Learning with supervision using a constraint-admissible set. In *2018 Annual American Control Conference*, pages 6390–6395, 2018.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Haitong Ma, Jianyu Chen, Shengbo Eben, Ziyu Lin, Yang Guan, Yangang Ren, and Sifa Zheng. Model-based constrained reinforcement learning using generalized control barrier function. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4552–4559, 2021.

Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, 31(5):664–674, 2012.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

Santiago Paternain, Luiz Chamon, Miguel Calvo-Fullana, and Alejandro Ribeiro. Constrained reinforcement learning has zero duality gap. *Advances in Neural Information Processing Systems*, 32, 2019.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *The Journal of Machine Learning Research*, 22(1):12348–12355, 2021.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

Jens Schreiter, Duy Nguyen-Tuong, Mona Eberts, Bastian Bischoff, Heiner Markert, and Marc Toussaint. Safe exploration for active learning with gaussian processes. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 133–149. Springer, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Vipul K Sharma, Wesley A Suttle, and Krishna C Kosaraju. `https://github.com/sharma1256/cbf-constrained_ppo`, 2024.

Yanan Sui, Alkis Gotovos, Joel Burdick, and Andreas Krause. Safe exploration for optimization with gaussian processes. In *International Conference on Machine Learning*, pages 997–1005. PMLR, 2015.

Wesley A Suttle, Amrit Bedi, Bhrij Patel, Brian M Sadler, Alec Koppel, and Dinesh Manocha. Beyond exponentially fast mixing in average-reward reinforcement learning via multi-level Monte Carlo actor-critic. In *International Conference on Machine Learning*, pages 33240–33267. PMLR, 2023.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction.* MIT Press, 2018.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1063, 2000.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature Methods*, 17(3):261–272, 2020.

Kim P Wabersich, Andrew J Taylor, Jason J Choi, Koushil Sreenath, Claire J Tomlin, Aaron D Ames, and Melanie N Zeilinger. Data-driven safety filters: Hamilton-Jacobi reachability, control barrier functions, and predictive methods for uncertain systems. *IEEE Control Systems Magazine*, 43(5):137–177, 2023.

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.

Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust Markov decision processes. *Mathematics of Operations Research*, 38(1):153–183, 2013.

Bin Xu and Koushil Sreenath. Safe teleoperation of dynamic uavs through control barrier functions. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7848–7855, 2018.

Kaiqing Zhang, Alec Koppel, Hao Zhu, and Tamer Başar. Convergence and iteration complexity of policy gradient method for infinite-horizon reinforcement learning. In *2019 58th Conference on Decision and Control*, pages 7415–7422, 2019.

Kaiqing Zhang, Alec Koppel, Hao Zhu, and Tamer Başar. Global convergence of policy gradient methods to (almost) locally optimal policies. *SIAM Journal on Control and Optimization*, 58(6):3586–3612, 2020.

Kaiqing Zhang, Bin Hu, and Tamer Başar. Policy optimization for $\mathcal{H}_2$ linear control with $\mathcal{H}_\infty$ robustness guarantee: Implicit regularization and global convergence. *SIAM Journal on Control and Optimization*, 59(6):4081–4109, 2021.

Checklist

1. For all models and algorithms presented, check if you include:
   - (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes: see §§2-3]
   - (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes: see Theorem 1]
   - (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes: see Sharma et al. (2024)]
2. For any theoretical claim, check if you include:
   - (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   - (b) Complete proofs of all theoretical results. [Yes: see appendix]
   - (c) Clear explanations of any assumptions. [Yes]
3. For all figures and tables that present empirical results, check if you include:
   - (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes: see Sharma et al. (2024)]

   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes: see §4 and appendix]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes: see figure captions]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes: see appendix]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes: see Brockman et al. (2016); Paszke et al. (2019); Raffin et al. (2021)]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes: see Sharma et al. (2024)]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Supplementary Material

A Proofs

Proof of Proposition 1. We construct a sequence of ε-balls, each reachable from the previous element of the sequence, that leads from x_0 to $\mathcal{B}$, then show that the head of the Markov chain lies inside this sequence with positive probability. Fix $\varepsilon > 0$ and let $\{y_0, y_1, \ldots, y_N\} \subset \mathcal{S}$ be such that $B_\varepsilon(y_{k+1}) \subset R(B_\varepsilon(y_k))$, for $k = 1, \ldots, N-1$, and $B_\varepsilon(y_N) \cap \mathcal{B} \neq \emptyset$ (see Figure 1 for an illustration). For a given θ, let $\{x_n\}$ be the Markov chain induced on $\mathcal{S}$ by π_θ^C such that $x_0 = y_0$. We show that the trajectory $(x_0, x_1, \ldots, x_N)$ is contained within the set $\{y_0\} \times B_\varepsilon(y_1) \times \ldots \times B_\varepsilon(y_N)$ with strictly positive probability, which will imply that $\{x_n\}$ enters $\mathcal{B}$ with strictly positive probability. For each $k = 1, \ldots, N$, consider the probability measure ν_k defined as

$$\nu_k(S) = P(x \in S \mid x_{k-1}) = \int_{\mathcal{T}_{x_{k-1}}^{-1}(S)} \pi_\theta^C(a|x_{k-1}) \; da,$$

for any μ-measurable subset S of $\mathcal{S}$. Note that ν_k is absolutely continuous with respect to μ, written $\nu_k \ll \mu$, since $\mu(S) > 0$ if and only if $\mathcal{T}_{x_{k-1}}^{-1}(S) > 0$, by Assumption 2. The Lebesgue-Radon-Nikodym Theorem implies that there exists a μ-integrable function $f_k : \mathcal{S} \to \mathbb{R}$, called the Radon-Nikodym derivative of ν_k, such that $\nu_k(S) = \int_S f_k(x)dx$ (see Folland (1999) for details). To make the link between f_k and ν_k perfectly clear, let us write

$$\begin{aligned} f_k(x) &= \int_{\mathcal{T}_{x_{k-1}}^{-1}(x)} \pi_\theta^C(a|x_{k-1}) \; da, \\ \nu_k(S) &= \int_S \int_{\mathcal{T}_{x_{k-1}}^{-1}(x)} \pi_\theta^C(a|x_{k-1}) \; da \; dx. \end{aligned}$$

By Assumptions 2 and 3, we also have $\mu \ll \nu_k$. Since both $\nu_k \ll \mu$ and $\mu \ll \nu_k$, the two measures are said to be *equivalent*, meaning that they agree on which sets have measure zero. Since μ and ν_k are equivalent, a standard result from real analysis allows us to take the Radon-Nikodym derivative f_k to be strictly positive μ-almost everywhere. As a first consequence, notice that

$$\begin{aligned} & P\Big((x_0, x_1, x_2) \in \{y_0\} \times B_\varepsilon(y_1) \times B_\varepsilon(y_2) \mid x_0 = y_0\Big) \\ &= P\Big((x_1, x_2) \in B_\varepsilon(y_1) \times B_\varepsilon(y_2) \mid x_0 = y_0\Big) \cdot P(x_0 = y_0) \\ &= P\Big((x_1, x_2) \in B_\varepsilon(y_1) \times B_\varepsilon(y_2) \mid x_0 = y_0\Big) \\ &= \int_{B_\varepsilon(y_1)} \int_{T_{x_1}^{-1}(B_\varepsilon(y_2))} \pi_\theta^C(a_1|x_1) f_1(x_1) \; da_1 \; dx_1 \qquad (15) \\ &= \int_{B_\varepsilon(y_1)} \int_{T_{x_1}^{-1}(B_\varepsilon(y_2)} \pi_\theta^C(a_1|x_1) \left[\int_{T_{x_0}^{-1}(x_1)} \pi_\theta^C(a_0|x_0) \; da_0 \right] \; da_1 \; dx_1. \end{aligned}$$

Given Assumption 3, equation (15) is strictly positive, since f_1 is strictly positive almost everywhere and the integrals are taken over sets of positive volume.

Building on equation (15), we have

$$P\Big((x_1, \ldots, x_{N-1}, x_N) \in \tag{16}$$

$$B_\varepsilon(y_1) \times \ldots \times B_\varepsilon(y_{N-1}) \times (B_\varepsilon(y_N) \cap \mathcal{B}) \mid x_0 = y_0\Big) \tag{17}$$

$$\begin{aligned} &= \int_{B_\varepsilon(y_1)} \int_{T_{x_1}^{-1}(B_\varepsilon(y_2))} \pi_\theta^C(a_1|x_1) \cdot \int_{B_\varepsilon(y_2)} \int_{T_{x_2}^{-1}(B_\varepsilon(y_3))} \pi_\theta^C(a_2|x_2) \cdot \ldots \\ &\quad \ldots \cdot \int_{B_\varepsilon(y_{N-1})} \int_{T_{x_{N-1}}^{-1}(B_\varepsilon(y_N)\cap\mathcal{B})} \pi_\theta^C(a_{N-1}|x_{N-1})\ f_{N-1}(x_{N-1})\ da_{N-1}\ dx_{N-1} \cdot \ldots \\ &\quad\quad \ldots \cdot f_2(x_2)\ da_2\ dx_2 \cdot f_1(x_1)\ da_1\ dx_1. \end{aligned}$$

Note in the innermost integral that $\mu(B_\varepsilon(y_N) \cap \mathcal{B}) > 0$, since both sets are open and their intersection is non-empty by hypothesis. Finally, given Assumption 3, we have that (17) is strictly positive, since all integrals are taken over sets of positive volume and f_i is strictly positive almost everywhere, for each $i \in \{1, \ldots N-1\}$. $\square$

Proof of Lemma 2. Recalling the definition of $\pi_\theta^C(u|x)$ in (2),

$$\begin{aligned} \nabla \log \pi_\theta^C(u|x) &= \nabla \log \pi_\theta(u|x) - \nabla \log \int_{C(x)} \pi_\theta(w|x) dw \\ &= \nabla \log \pi_\theta(u|x) - \frac{\int_{C(x)} \nabla \pi_\theta(w|x) dw}{\int_{C(x)} \pi_\theta(w|x) dw}. \end{aligned} \tag{18}$$

To see that, for all $x \in \mathcal{X}, u \in \mathcal{U}$, $\nabla \log \pi_\theta^C(u|x)$ exists, for all $\theta \in \Theta$, we simply need to verify that $(\int_{C(x)} \pi_\theta(w|x) dw)^{-1}$ is always finite. But this follows immediately from Assumption 3 and the fact that $\mu(C(x)) \geq m > 0$ by Assumption 1.

We next prove part 1) of the Lemma. The claim holds for the first term in (18) by part 2a) of Assumption 6, so we just need to show that it holds for the second term. To do this, we prove that, for a given $x \in \mathcal{X}$, this term is Lipschitz in θ, then argue that the largest minimal Lipschitz constant over all $x \in \mathcal{X}$ is finite. We know by part 2b) of Assumption 6 that, for all $x \in \mathcal{X}, u \in \mathcal{U}$, $\|\nabla \pi_\theta(u|x)\| \leq \nabla \log \pi_\theta(u|x) \leq B_\Theta$, for all $\theta \in \Theta$. This means that, for all $x \in \mathcal{X}$,

$$\begin{aligned} &\Big| \int_{C(x)} \pi_\theta(w|x) dw - \int_{C(x)} \pi_{\theta'}(w|x) dw \Big| \\ &\quad = \Big| \int_{C(x)} \left(\pi_\theta(w|x) - \pi_{\theta'}(w|x)\right) dw \Big| \\ &\quad \leq \int_{C(x)} |\pi_\theta(w|x) - \pi_{\theta'}(w|x)| dw \\ &\quad \leq \int_{C(x)} B_\Theta \left\|\theta - \theta'\right\| dw = B_\Theta \mu(C(x)) \left\|\theta - \theta'\right\| \\ &\quad \leq B_\Theta M \left\|\theta - \theta'\right\|, \end{aligned}$$

for all $\theta, \theta' \in \Theta$. So $\int_{C(x)} \pi_\theta(w|x) dw$ is Lipschitz in θ, for each $x \in \mathcal{X}$, and the largest Lipschitz constant over $\mathcal{X}$ is finite. In addition, $\int_{C(x)} \pi_\theta(w|x) dw$ is clearly uniformly bounded.

Notice that $\inf_{x \in \mathcal{X}} \mu(C(x)) \geq m > 0$, by Assumption 1. Thus, by Assumption 3, $\inf_{x \in \mathcal{X}} \int_{C(x)} \pi_\theta(w|x) dw > 0$, for all $\theta \in \Theta$. Since Θ is compact, this means $\inf_{\theta \in \Theta} \inf_{x \in \mathcal{X}} \int_{C(x)} \pi_\theta(w|x) dw > 0$. This implies that $(\int_{C(x)} \pi_\theta(w|x) dw)^{-1}$ is uniformly bounded. Since $\int_{C(x)} \pi_\theta(w|x) dw$ is Lipschitz, $(\int_{C(x)} \pi_\theta(w|x) dw)^{-1}$ is therefore Lipschitz and bounded in $\theta \in \Theta$, for all $x \in \mathcal{X}$. We also know that, for each $x \in \mathcal{X}$, $\int_{C(x)} \nabla \pi_\theta(w|x) dw$ is Lipschitz and bounded in $\theta \in \Theta$, by Assumption 6, part 2a). Fix $x \in \mathcal{X}$. Since the product of Lipschitz, bounded functions is Lipschitz and bounded, the function $\int_{C(x)} \nabla \pi_\theta(w|x) dw / \int_{C(x)} \pi_\theta(w|x) dw$ is Lipschitz and bounded in θ. Since this function is uniformly bounded over $x \in \mathcal{X}, \theta \in \Theta$, there therefore exists $L > 0$ such that, for all $x \in \mathcal{X}$,

$$\left\| \frac{\int_{C(x)} \nabla \pi_\theta(w|x) dw}{\int_{C(x)} \pi_\theta(w|x) dw} - \frac{\int_{C(x)} \nabla \pi_{\theta'}(w|x) dw}{\int_{C(x)} \pi_{\theta'}(w|x) dw} \right\| \leq L \left\|\theta - \theta'\right\|,$$

for all $\theta, \theta' \in \Theta$. Combined with part 2a) of Assumption 6, this implies that, for all $x \in \mathcal{X}, u \in \mathcal{U}$, $\left\|\nabla \log \pi_\theta^C(u|x) - \nabla \log \pi_{\theta'}^C(u|x)\right\| \leq (L_\Theta + L) \left\|\theta - \theta'\right\|$, for all $\theta, \theta' \in \Theta$. This completes the proof of part 1).

Part 2) follows from the fact that $\int_{C(x)} \nabla \pi_\theta(w|x) dw / \int_{C(x)} \pi_\theta(w|x) dw$ is uniformly bounded and that, for all $x \in \mathcal{X}, u \in \mathcal{U}$, $\|\nabla \log \pi_\theta(u|x)\| \leq B_\Theta$, for all $\theta \in \Theta$, by part 2) of Assumption 6.

B Background: Barrier Functions

In this section, we provide an overview of barrier functions for convenience. The theory of barrier functions revolve around controlled set invariance for dynamical systems. Safety can be represented through a set, say $\mathcal{S}$, defined as a level set of this barrier function, say h. Then, we write the condition on this barrier function to guarantee the forward invariance of this safety set under the given dynamics.

B.1 Control Barrier Functions (CBF)

Consider the following nonlinear system

$$\dot{r} = f(r, u), \tag{19}$$

where $r \in D \subset \mathbb{R}^n$ and $u \in U \subset \mathbb{R}^m$ denote the state and control input, and f is a locally Lipschitz function that models the state transition. The following definition and the theorem follows the development in Ames et al. (2019); Agrawal and Sreenath (2017).

Theorem 2. *Consider a function $h : \mathbb{R}^n \to \mathbb{R}$ that is continuously differentiable. Define a closed set $\mathcal{S}$ as the super-level set of this function as follows:*

$$\mathcal{S} \triangleq \{r \in \mathbb{R}^n \mid h(r) \geq 0\}. \tag{20}$$

The function h is a control barrier function, for (19) and with state s, if there exists an extended κ_∞ function α such that for all $r \in \mathcal{S}$, $t \in \mathbb{R}_+$,

$$\dot{h} \geq -\alpha(h). \tag{21}$$

Further, if we define the safe control set as

$$\mathcal{C}(r) \triangleq \left\{u \in \mathbb{R}^m | \dot{h}(r, u) \geq -\alpha(h(r))\right\}. \tag{22}$$

then any input $u \in \mathcal{C}(r)$ will render the set $\mathcal{S}$ forward invariant.

When designing safe controller with control values u sampled from this safe control set, we need the time-derivative of h i.e. $\dot{h}$ to explicitly contain u. However, the above forward variance condition is restricted to barrier function with relative degree $d_r = 1$. At this point, we also note that for our quadcopter experiment, our CBF has a relative degree $d_r = 2$, since only the second time-derivative $\ddot{h}$ explicitly contains the control input u. Therefore, for barrier functions with relative degree more than 1, which are often referred to as Exponential Control Barrier Functions (ECBF), we need a seperate discussion on forward invariance conditions.

B.2 Exponential Control Barrier Functions

We now discuss exponential CBFs for control affine nonlinear dynamical system. Consider the following control affine nonlinear dynamical system:

$$\dot{r} = f(r) + g(r)u, \tag{23}$$

with f and g locally lipshitz, $r \in D \subset \mathbb{R}^n$ and $u \in U \subset \mathbb{R}^m$. We suppose that the Lipschitz constant for f and g are L_f and L_g respectively, and the vector containing the first $d_r - 1$ time derivatives of $h(r)$ including $h(r)$ is given as: $\eta_b(r) = \begin{bmatrix} h(r) \\ \dot{h}(r) \\ \ddot{h}(r) \\ \vdots \\ h^{d_r-1}(r) \end{bmatrix}$. Further suppose that the matrices F, G, and C are defined as follows:

$$F = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}, G = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, \text{ and, } C = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Theorem 3. *Consider a function $h : \mathbb{R}^n \to \mathbb{R}$ that is continuously differentiable. Define a closed set $\mathcal{S}$ as the super-level set of this function as follows:*

$$\mathcal{S} \triangleq \{r \in \mathbb{R}^n \mid h(r) \geq 0\}. \tag{24}$$

Then the function h is an ECBF, with relative degree d_r, for system in (23), if there exist a row vector $K_\alpha \in \mathbb{R}^r$ such that

$$\sup_{u \in U} [L_f^{d_r} h(r) + L_g L_f^{d_r - 1} h(r) u] \geq -K_\alpha \eta_b(r) \tag{25}$$

$\forall r \in Int(\mathcal{S})$, implies $h(r(t)) \geq Ce^{(F - GK_\alpha)t} \eta_b(r) r(t_0) \geq 0$, whenever $h(r(t_0)) \geq 0$. Further, if we define the safe control set $C(r)$ as

$$\mathcal{C}(r) \triangleq \{u \in U | [L_f^{d_r} h(r) + L_g L_f^{d_r - 1} h(r) u] \geq -K_\alpha \eta_b(r)\}, \tag{26}$$

then any input $u \in \mathcal{C}(r)$ will render the set $\mathcal{S}$ forward invariant.

C Experiments: Additional Details

In this section, we provide additional details regarding the experiments presented in §4.

C.1 Inverted Pendulum Experiments

The safety-constrained inverted pendulum environment that we considered in §4.4 was obtained by modifying the standard implementation from Brockman et al. (2016) to include CBF-based safety constraints. In this section we describe the dynamical model and CBF used to obtain these constraints, then present implementation details and an additional experiment.

C.1.1 Dynamical Model

Consider the model of a simple inverted pendulum

$$\begin{bmatrix} \theta_{k+1} \\ \dot{\theta}_{k+1} \end{bmatrix} = \begin{bmatrix} \theta_k + \delta t \dot{\theta}_k + \delta t^2 \left(\frac{3g}{2l} \sin \theta_k + \frac{3}{ml^2} u_k \right) \\ \dot{\theta}_k + \delta t \left(\frac{3g}{2l} \sin \theta_k + \frac{3}{ml^2} u_k \right) \end{bmatrix}, \tag{27}$$

where θ_k, $\dot{\theta}_k$ denote the states (angle and angular velocity), u_k denote the input (torque), m and l denotes the mass and the length of the pendulum, respectively, g denotes the acceleration due to gravity and $\delta t > 0$ denotes the discretization time. Denote the safe operating region by

$$\mathcal{S} = \left\{ \theta \in \mathbb{R} | h(\theta) := \begin{bmatrix} \theta + 1 \\ 1 - \theta \end{bmatrix} \geq 0 \right\}. \tag{28}$$

C.1.2 Control Barrier Function

The following corollary is a direct consequence of Theorem 2, presented in §B.

Corollary 2. *Let*

$$U(\theta_k, \dot{\theta}_k) = \left\{ u_k \in \mathbb{R} | \left(\delta t \dot{\theta}_k + c(\theta_k, u_k) \right) \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \eta \begin{bmatrix} \theta_k + 1 \\ 1 - \theta_k \end{bmatrix} \geq 0 \right\} \tag{29}$$

where $c(\theta_k, u_k) := \delta t^2 \left(\frac{3g}{2l} \sin \theta_k + \frac{3}{ml^2} u_k \right)$, and $0 < \eta < 1$. Consider system (27) with $u_k \in U(\theta_k, \dot{\theta}_k)$. Let $(\theta_0, \dot{\theta}_0) \in \mathcal{S} \times \mathbb{R}$ and assume $U(\theta_0, \dot{\theta}_0)$ is non-empty. The set $\mathcal{S}$ is forward invariant.

The safe set (29) is used to provide the state-dependent constraints to the Beta policies learned in our experiments.

policy learning rate	0.0003
value learning rate	0.0003
entropy coefficient	0.0
clip range	0.2
weight decay	0.0
layer size	64
batch size	64
buffer size	300
number of epochs	10
rollout length	300
discount factor	0.99

(a) Gaussian hyperparameters.

policy learning rate	0.01
value learning rate	0.01
entropy coefficient	0.0
clip range	0.2
weight decay	0.0
layer size	64
batch size	64
buffer size	300
number of epochs	10
rollout length	300
discount factor	0.99

(b) Beta hyperparameters.

Figure 4: PPO hyperparameters for the inverted pendulum experiments.

C.1.3 Implementation Details

We next describe the implementation details of our experiments. As mentioned above, the environment was adapted from the implementation of (Brockman et al., 2016), with modifications to compute the CBF safe set (29). The reward function and other details are as in Brockman et al. (2016). The Beta and Gaussian policies used the corresponding distributions from the PyTorch library (Paszke et al., 2019). As described in §4.2, for a given state x, the parameters $\alpha(x), \beta(x)$ of the Beta distribution were outputted by a two-layer, fully connected neural network. Control inputs were obtained by sampling from this distribution, then translating and rescaling to lie within the current CBF set $C(x) = [a(x), b(x)]$. The Gaussian policy parameters were outputted by a two-layer, fully connected neural network. Control inputs were subsequently selected from the corresponding distribution by sampling, then, following standard practice (Chou et al., 2017), were clipped to a set of permissible controls, which was chosen to be $[-15.0, 15.0]$. The hyperparameters used are presented in Figure 4.

C.1.4 Additional Results

Figure 5 presents an experiment providing additional support to the discussion presented in §4.4.

C.2 Quadcopter Experiments

In this section, we provide additional details regarding the environment and experiments presented in §4.3.

C.2.1 Dynamical Model

We summarize the dynamical model of the quadcopter derived in Xu and Sreenath (2018). We consider the body frame, say $\mathbb{F}_b$, and world frame, say $\mathbb{F}_w$, and discuss the transformation between these two frames using the rotation matrix $\mathbb{R}_{wb}$ defined as

$$\mathbb{R}_{wb} := \begin{bmatrix} \cos\psi\cos\theta - \sin\phi\sin\psi\sin\theta & -\cos\phi\sin\psi & \cos\psi\sin\theta + \cos\theta\sin\phi\sin\psi \\ \cos\theta\sin\psi + \cos\psi\sin\phi\sin\theta & \cos\phi\cos\psi & \sin\psi\sin\theta - \cos\psi\cos\theta\sin\phi \\ -\cos\phi\sin\theta & \sin\phi & \cos\phi\cos\theta \end{bmatrix}, \tag{30}$$

where ϕ, θ, and ψ denote the Z-X-Y Euler angles corresponding to the roll, pitch, and yaw of the quadcopter. Suppose that the 3-dimensional position coordinates of the quadcopter along the x-,y-, and z-axis with respect to its body frame $\mathbb{F}_b$ of and the world frame of reference $\mathbb{F}_w$ be given by $x_b := (x_b, y_b, z_b)$ and $r := (r_x, r_y, r_z)$ respectively, then $r = \mathcal{R}_{wb}x_b$.

Then, the quadcopter dynamics is given by $\dot{x} = Ax + Bu$, where the control input u comprises of the desired acceleration of the quadcopter. The dynamics of this controller under small angle assumptions on the Euler angles, that is $\sin\hat{e} \approx \hat{e}, \cos\hat{e} \approx 1, \hat{e} \in \{\phi, \theta, \psi\}$) evolves as Mellinger et al. (2012):

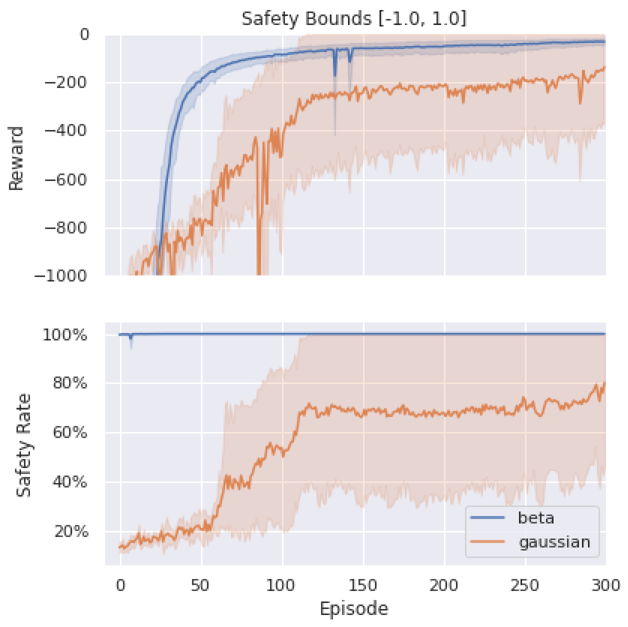


Figure 5: Comparison of safety-constrained Beta policy and unconstrained Gaussian policy on the inverted pendulum environment with constraint set $\mathcal{S}_{1.0} = \{\theta \mid -1.0 \leq \theta \leq 1.0\}$. The top figure presents learning curves, while the bottom figure presents the "safety rate", i.e., the percentage of time spent in $\mathcal{S}_{1.0}$ over the course of the episode. The curves represent means and 95% confidence intervals over five independent replications.

$$u = \begin{bmatrix} \ddot{r}_x^{des} \\ \ddot{r}_y^{des} \\ \ddot{r}_z^{des} \end{bmatrix} = \begin{bmatrix} g(\theta^{des}\cos\psi^{des} + \phi^{des}\sin\psi^{des}), \\ g(\theta^{des}\sin\psi^{des} - \phi^{des}\cos\psi^{des}) \\ \frac{\sum_{i=1}^{4} F_i^{des}}{m} - g \end{bmatrix}, \tag{31}$$

where m, g are respectively the mass of the quadcopter and gravitational constant, and $\ddot{r}_i^{des}, i \in \{x, y, z\}$ is the desired acceleration component of the quadcopter in the x-,y-, and z-direction respectively, computed using the desired specifications on the Euler angles ϕ^{des}, θ^{des}, and ψ^{des}, and $F_i^{des}, i \in \{1, 2, 3, 4\}$ is the desired thrust on the i-th rotor of the quadcopter. Lastly, the dynamical parameters for the quadcopter are setup as given in Ho et al. (2020).

C.2.2 Exponential Control Barrier Function

Recall, that the objective of our controller to enable the quadcopter to learn how to reach a target position r_{goal}, while avoiding an obstacle with position r_{obs}. For this obstacle avoidance, we now discuss our choice of CBF for the quadcopter experiment, as defined in (14), and reason why this is an exponential control barrier function (ECBF). We first derive expressions for $\dot{h}$ and $\ddot{h}$ using dynamical equations as follows:

$$\dot{h}(r) = 4((\Delta r_x/a)^3\,\dot{r}_x + (\Delta r_y/b)^3\,\dot{r}_y + (\Delta r_z/c)^3\,\dot{r}_z), \tag{32}$$

and

$$\ddot{h}(r) = 12((\Delta r_x/a)^2\,\dot{r}_x + (\Delta r_y/b)^2\,\dot{r}_y + (\Delta r_z/c)^2\,\dot{r}_z) + 4((\Delta r_x/a)^3\,\ddot{r}_x + (\Delta r_y/b)^3\,\ddot{r}_y + (\Delta r_z/c)^3\,\ddot{r}_z). \tag{33}$$

These equations can be re-written in vector form as follows:

$$\dot{h}(r) = \begin{bmatrix} 4(\Delta r_x^3/a^4) & 4(\Delta r_y^3/b^4) & 4(\Delta r_z^3/c^4) \end{bmatrix} \dot{r} \tag{34}$$

and

$$\ddot{h}(r) = \dot{r}^T \begin{bmatrix} 12({\Delta r_x}^2/a^4) & 0 & 0 \\ 0 & 12(\Delta r_y^2/b^4) & 0 \\ 0 & 0 & 12(\Delta r_z^2/c^4) \end{bmatrix} \dot{r} + \begin{bmatrix} 4({\Delta r_x}^3/a^4) & 4(\Delta r_y^3/b^4) & 4(\Delta r_z^3/c^4) \end{bmatrix} \ddot{r}. \tag{35}$$

Since $u = \ddot{r}$ from quadcopter dynamics, we can re-write the $\ddot{h}(r)$ as follows:

$$\ddot{h}(r) = \dot{r}^T D_r \dot{r} - A_r u, \tag{36}$$

where $D_r := \begin{bmatrix} 12({\Delta r_x}^2/a^4) & 0 & 0 \\ 0 & 12(\Delta r_y^2/b^4) & 0 \\ 0 & 0 & 12(\Delta r_z^2/c^4) \end{bmatrix}$, and $A_r := -\begin{bmatrix} 4({\Delta r_x}^3/a^4) & 4(\Delta r_y^3/b^4) & 4(\Delta r_z^3/c^4) \end{bmatrix}$.

Therefore, we note that $\ddot{h}(r)$ or the 2^{nd} time-derivative of $h(r)$ which explicitly depends on the control input u and therefore our choice of CBF $h(r)$ is an exponential CBF with a relative degree Ames et al. (2019) of 2.

Correspondingly, we use the following forward invariance condition for the set $\mathcal{S} = \{r : h(r) \geq 0\}$ as given in Xu and Sreenath (2018):

$$\ddot{h} + K \cdot [h \quad \dot{h}]^T \geq 0, \tag{37}$$

with $K = [K_1 \quad K_2]^T$.

The above equation can be re-arranged as follows:

$$-\ddot{h} \leq K_1 h + K_2 \dot{h},$$

and, using (34) and (36), we can re-write this equation as

$$A_r u \leq b_r, \tag{38}$$

where $b_r = \dot{r}^T D_r \dot{r} + K_1 h - K_2 A_r \dot{r}$. Thus, we can write the safe control set as $\mathcal{C}(r) = \{u \in \mathbb{R}^3 : A_r u \leq b_r\}$. For our quadcopter experiments, we consider navigation $x-y$ dimensions, therefore set the z-dimension position and velocity to be 0. Thus the control input only comprises the desired acceleration for x and y axes and therefore our action space becomes two-dimensional, and we only consider the x and y components in the above CBF calculations.

C.2.3 Maximal Inner Hyperrectangle Computation

We now describe the construction of the maximal inner hyperrectangle contained in the set $\mathcal{C}(r)$ under actuator constraints H. These are the sets that our Beta policies will sample from. We use the following optimization problem, with decision variables $u = (u^x, u^y)$, to get the maximal inner hyper-rectangle inside the safe set:

$$\mathcal{P}_A : \begin{array}{cl} \max\limits_u & \mathcal{A}(u) \\ \text{s.t.} & A_r u \leq b_r, \\ & u \in H, \end{array} \tag{39}$$

where $\mathcal{A}(u)$ is the area of a hyperrectangle inside $C(r) \cap H$ and the decision variables u^x, u^y are points on the line $A_r u \leq b_r$. One of the corner points of this hyperrectangle is formed by u^x, u^y and the rest of corner points lie on the boundary hyperrectangle formed by H. Suppose that (u_*^x, u_*^y) are solutions to $\mathcal{P}_A$, then the definition of Area $\mathcal{A}$ depends on how the line $A_r u \leq b_r$ intersects with H, and therefore, leads to the following four possibilities:

- $\mathcal{A} = (u^x - u^x_{min}) * (u^y - u^x_{min})$ and $H_c = \{u_{min}, (u^x_*, u^y_*)\}$
- $\mathcal{A} = (u^x - u^x_{min}) * (u^y_{max} - u^y)$ and $H_c = \{(u^x_{min}, u^y_*), (u^x_*, u^y_{max})\}$
- $\mathcal{A} = (u^x_{max} - u^x) * (u^x_{max} - u^y)$ and $H_c = \{(u^x_*, u^y_*), u_{max}\}$
- $\mathcal{A} = (u^x_{max} - u^x) * (u^y - u^y_{min})$ and $H_c = \{(u^x_*, u^y_{min}), (u^x_{max}, u^y_*)\}$.

policy learning rate	0.0004
value learning rate	0.0004
entropy coefficient	0.00000001
clip range	0.2
weight decay	0.0
layer size	256
batch size	256
buffer size	320
number of epochs	10
rollout length	320
discount factor	0.90

(a) Gaussian hyperparameters.

policy learning rate	0.0006
value learning rate	0.0006
entropy coefficient	0.0
clip range	0.2
weight decay	0.0
layer size	256
batch size	256
buffer size	180
number of epochs	10
rollout length	180
discount factor	0.90

(b) Beta hyperparameters.

Figure 6: PPO hyperparameters for the quadcopter experiments.

$\mathcal{P}_A$ is, in general, a non-convex program. However, through change-of-variables, we can transform this problem into a tractable problem through the following transformation. We perform a change of variables, with new variables denoted by $(\bar{u}^x, \bar{u}^y)$ defined by

- $\bar{u}^x = u^x - u^x_{min}$ and $\bar{u}^y = u^y - u^y_{min}$
- $\bar{u}^x = u^x - u^x_{min}$ and $\bar{u}^y = u^y_{max} - u^y$
- $\bar{u}^x = u^x_{max} - u^x$ and $\bar{u}^y = u^y_{max} - u^y$
- $\bar{u}^x = u^x_{max} - u^x$ and $\bar{u}^y = u^y - u^y_{min}$.

The corresponding objective function is given by $\bar{\mathcal{A}} = \bar{u}^x\bar{u}^y$. So long as the entries corresponding to A_r and b_r from (39) in the transformed problem are nonnegative, the resulting problem is a geometric program, which can be further transformed to a convex problem by standard methods and efficiently solved. In our quadcopter experiments, when $A_r, b_r \geq 0$, we solved the transformed geometric program using CVXPY (Diamond and Boyd, 2016), and used non-linear solvers from SCIPY (Virtanen et al., 2020) otherwise. We observed in our experiments that the transformation resulted in geometric programs in all but a handful of cases.

C.2.4 Reward

We now discuss reward shaping used in our quadcopter experiments.

Suppose that the $r_{min} := [r^x_{min} \quad r^y_{min}]^T$ and $r_{max} := [r^x_{max} \quad r^y_{max}]^T$ are environment boundaries, with x, y-axis boundary repectively defined by $[r^x_{min}, r^x_{max}]$ and $[r^y_{min}, r^y_{max}]$, that we employ for guiding exploration for both the quadcopter experiments. Then the reward used in our environment is defined by

$$R(r) = \begin{cases} 50 & \text{if } ||r - r_{goal}||_2 < \epsilon, \\ -||r - r_{goal}||_2 & \text{if } r_{max} > r > r_{min} \text{ and } ||r - r_{goal}||_2 \geq \epsilon, \\ -||r - r_{goal}||_2 - 400 & \text{if } r \geq r_{max} \text{ or } r \leq r_{min}, \end{cases}$$

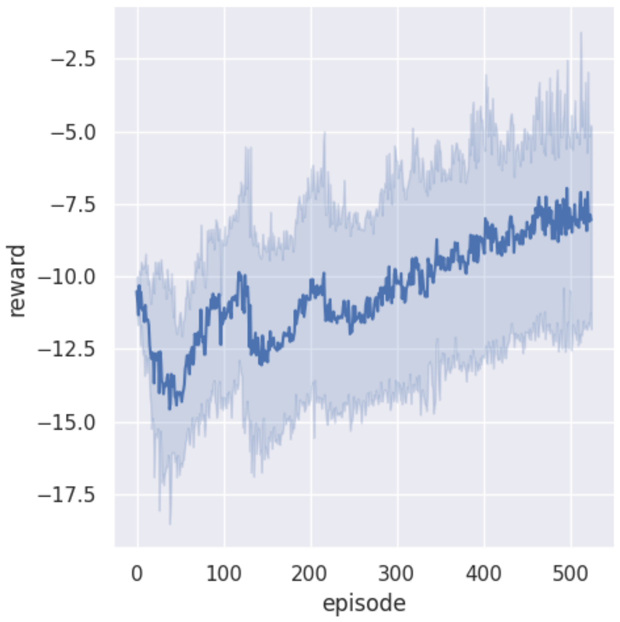
where $\epsilon = 0.25$ is the boundary around r_{goal} for which we give a constant positive reward of 50 to the agent, and the inequalities in the reward definition are element-wise. Moreover, when the agent is inside the boundary but outside the ϵ-neighborhood of the goal, then the reward is negative of the distance between the agent and the goal. Lastly, we penalize the agent if it goes outside the boundary defined by r_{min} and r_{max} to encourage exploration in the region around the goal.

C.2.5 Hyperparameters

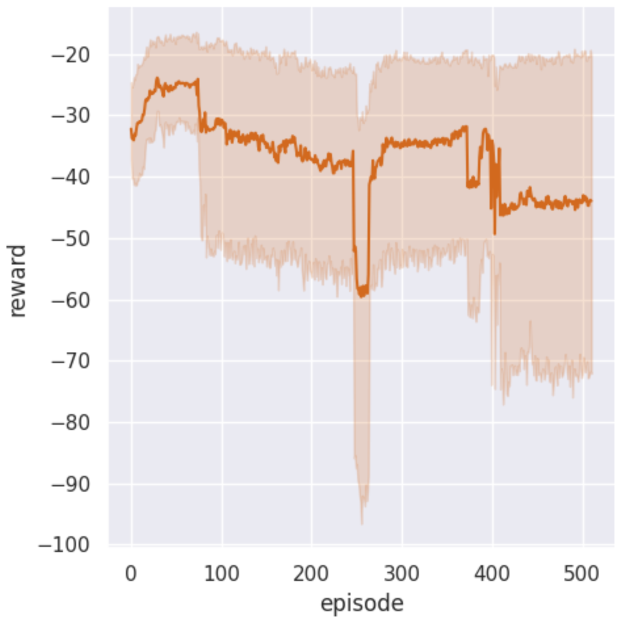
The hyperparameters used in the experiments are presented in Figure 6.

C.2.6 Learning Curves

Learning curves for the experiments illustrated in Figures 1 and 2c are presented in Figures 7a and 7b.



(a) Beta policy learning curve.



(b) Projected Gaussian learning curve.

Figure 7: Learning curves corresponding to experiments pictures in Figures 1 and 2c. Curves show means and 95% confidence intervals over 6 independent replications. Our CBF-constrained Beta policies clearly learn to improve reward and eventually find the goal, while the projection-based approach fails.

D Computing Resources

We ran our experiments on both a personal laptop and an HPC cluster. The laptop was configured with a 6-core i7-8750H, 2.20GHz CPU, an NVIDIA GeForce RTX 2070 GPU, and 32GB RAM . The HPC server node was configured with a 32-core Intel Xeon CPU , an 80GB Nvidia Tesla GPU, and 512 GB RAM .