Learning to Provably Satisfy High Relative Degree Constraints for Black-Box Systems*

Jean-Baptiste Bouvier, Kartik Nagpal and Negar Mehr

***Abstract*— In this paper, we develop a method for learning a control policy guaranteed to satisfy an affine state constraint of high relative degree in closed loop with a black-box system. Previous reinforcement learning (RL) approaches to satisfy safety constraints either require access to the system model, or assume control affine dynamics, or only discourage violations with reward shaping. Only recently have these issues been addressed with POLICEd RL, which guarantees constraint satisfaction for black-box systems. However, this previous work can only enforce constraints of relative degree 1. To address this gap, we build a novel RL algorithm explicitly designed to enforce an affine state constraint of high relative degree in closed loop with a black-box control system. Our key insight is to make the learned policy be affine around the unsafe set and to use this affine region to dissipate the inertia of the high relative degree constraint. We prove that such policies guarantee constraint satisfaction for deterministic systems while being agnostic to the choice of the RL training algorithm. Our results demonstrate the capacity of our approach to enforce hard constraints in the Gym inverted pendulum and on a space shuttle landing simulation.**

I. Introduction

The lack of safety guarantees in reinforcement learning (RL) has been impeding its wide deployment in real-world settings [1]. Safety in RL is traditionally captured by state constraints preventing the system from entering unsafe regions [2]. This issue has been investigated by numerous approaches and most commonly under the framework of constrained Markov decision processes (CMDPs) [3], [4], [5]. CMDPs only *encourage* policies to respect safety constraints by penalizing the expected violations [6]; however, they do not provide any satisfaction guarantees [7]. For safety-critical tasks, such as autonomous driving or human-robot interactions, safety guarantees are primordial and require the learned policy to maintain constraint respect.

A few RL attempts at learning provably safe policies have involved control barrier functions (CBFs) [8], backward reachable sets [9], and projection of control inputs onto safe sets [10], [11]. However, all these methods require precise knowledge of the system dynamics, which is usually not available in RL. To circumvent this issue and study black-box systems without an analytical model of the dynamics, the common approach has been to learn safety certificates [12], [13], [14]. Yet, by learning to approximate CBFs, the formal safety guarantees of these methods hinge upon the quality of their CBF approximation. More reliable safety guarantees have been established by recent work [15], whose POLICEd RL approach designs a repulsive buffer to enforce constraint satisfaction in closed-loop with a black-box system. However, work [15] along with most other safe RL works such as [10], [11], [12], [14] are limited to constraints of relative degree 1. In contrast, our approach enforces inviolable *constraints of high relative degree* in closed-loop with a learned control policy while exclusively using a *black-box model* of the system dynamics.

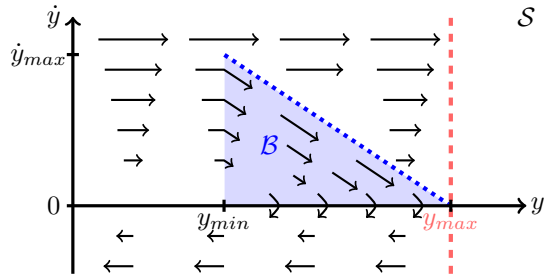

Fig. 1: Phase portrait of constrained output y illustrating our High Relative Degree POLICEd RL method on a system of relative degree 2. To prevent states from violating constraint $y \leq y_{max}$ (red dashed line), our policy guarantees that trajectories entering buffer region $\mathcal{B}$ (blue) cannot leave it through its upper bound (blue dotted line). Our policy makes $\ddot{y}$ sufficiently negative in buffer $\mathcal{B}$ to bring $\dot{y}$ to 0 in all trajectories entering $\mathcal{B}$. Once $\dot{y} < 0$, trajectories cannot approach the constraint. Due to the states' inertia, it is physically impossible to prevent all constraint violations. For instance, $y = y_{max}$, $\dot{y} >> 1$ will yield $y > y_{max}$ at the next timestep. Hence, we only aim at guaranteeing the safety of trajectories entering buffer $\mathcal{B}$.

The *relative degree of a constraint* describes how many times a constraint needs to be differentiated before a control input appears in its expression. The higher the relative degree, the more inertia the constraint has and the more challenging its satisfaction is [16]. Inspired by the recent extensions of CBFs to high relative degrees [17], [18], [16], we propose a backstepping inspired approach [19], which is compatible with systems non-affine in control and with black-box systems contrary to these CBF methods.

To learn our safe controller, we draw inspiration from the POLICEd RL method of [15] and transform the state space surrounding the affine state constraint into a buffer region that cannot be crossed. We overcome the major limitation of [15] by extending POLICEd RL to constraints of high relative degree. To dissipate the inertia of this high relative degree constraint, our key insight is to extend the buffer into the dimensions of the unactuated derivatives of the state constraint. While this task appears arduous, these derivatives are usually accessible since related to the states. In this buffer region, we train the policy to dissipate the state's inertia

*This work is supported by the National Science Foundation, under grants ECCS-2145134 CAREER Award, CNS-2218759, and CCF-2211542.

Jean-Baptiste Bouvier, Kartik Nagpal and Negar Mehr are with the Department of Mechanical Engineering, University of California Berkeley, Berkeley, CA 94709, USA {bouvier3, kartiknagpal, negar}@berkeley.edu

to progressively slow its progression towards the constraint, as illustrated in Fig. 1. This controller guarantees that trajectories entering the buffer do not violate the constraint. Since inertia cannot be dissipated instantly, some constraint violations are physically impossible to prevent. Inspired by [15], to easily verify the dissipative character of the controller in the buffer, we use the POLICE algorithm [20] to generate an affine policy over the buffer region.

In summary, our contributions in this work are as follows.

1) We introduce High Relative Degree POLICEd RL, a novel RL framework to guarantee satisfaction of an affine state constraint of high relative degree using a black-box model of the system in closed-loop with a trained policy.
2) We provide comprehensive proof, and we detail our method to evaluate our trained policy while directly guaranteeing constraint satisfaction.
3) We demonstrate the safety guarantees of our approach in a number of simulation studies involving an inverted pendulum and a space shuttle landing.

The remainder of this work is organized as follows. In Section II, we introduce our problem formulation along with our framework. In Section III, we establish the theoretical guarantees of our approach in enforcing the satisfaction of a high relative degree constraint. In Section IV, we illustrate our method on the Gym inverted pendulum and on a space shuttle landing scenario. Appendices A and B contain supporting lemmata and implementation details for our simulations.

Notation: We denote the positive integer interval from $a \in \mathbb{N}$ to $b \in \mathbb{N}$ inclusive by $[\![a, b]\!]$. We denote the component $i \in [\![1, n]\!]$ of a vector $x \in \mathbb{R}^n$ by x_i and the vector composed of components i to $j > i$ by $x_{i:j}$. The set of nonnegative real numbers is $\mathbb{R}^+$. We denote the k^{th} time derivative of a function y by $y^{(k)} = \frac{d^k y}{dt^k}$. If $x, y \in \mathbb{R}^n$, then $x \leq y$ denotes the element-wise inequalities $x_i \leq y_i$ for all $i \in [\![1, n]\!]$.

II. Framework

We consider a black-box deterministic system

$$\dot{x}(t) = f\big(x(t), u(t)\big), \quad u(t) \in \mathcal{U}, \quad x(0) \in \mathcal{X}, \tag{1}$$

with state space $\mathcal{X} \subseteq \mathbb{R}^n$ and admissible control set $\mathcal{U} \subseteq \mathbb{R}^m$. We consider dynamics (1) to be an implicit black-box, meaning that we can evaluate f but we do not have any explicit knowledge or analytical form of f. This is similar to the online RL setting where f is a simulator or a robot.

We assume that the system safety constraint is captured by a single affine inequality on output

$$y(t) := Cx(t) \leq y_{max}, \quad \text{for all } t \geq 0, \tag{2}$$

with $C \in \mathbb{R}^{1\times n}$ and $y_{max} \in \mathbb{R}$. We model our deterministic feedback policy $u(t) = \pi_\theta\big(x(t)\big) \in \mathcal{U}$ by a deep neural network parameterized by θ. Our objective is to train policy π_θ to respect constraint (2) and maximize the following expected discounted reward

$$\max_\theta \mathcal{G}(\pi_\theta) := \mathop{\mathbb{E}}_{x_0\sim\rho_0}\int_0^\infty \gamma^t R\big(x(t), \pi_\theta(x(t))\big)dt \text{ s.t. (2)}, \tag{3}$$

where $\gamma \in (0, 1]$ is a discount factor, R a reward function, and ρ_0 the distribution of initial states. The only stochasticity in our setting comes from the initial state sampling $x_0 \sim \rho_0$. We emphasize that constraint (2) is a *hard constraint* to be respected at all times. Contrary to previous work [15], we assume that constraint (2) has a *relative degree* at least 2.

Definition 1. *The* relative degree r *of output* y (2) *for dynamics* (1) *is the order of its input-output relationship, i.e.,* $r := \min\big\{p \in \mathbb{N} : \frac{\partial}{\partial u}\frac{\partial^p y}{\partial t^p}(t) \neq 0 \text{ for all } x \in \mathcal{X}\big\}$ [21].

In simpler words, the relative degree is the minimal number of times output y has to be differentiated until control input u appears. As argued in [22], relative degree r can be obtained by first-order principles without the knowledge of dynamics f. Hence, knowing r is compatible with our black-box model of f. Assuming $r \geq 2$, we have

$$\frac{\partial}{\partial u}\frac{\partial y(t)}{\partial t} = \frac{\partial \dot{y}(t)}{\partial u} = \frac{\partial}{\partial u} C\dot{x}(t) = C\frac{\partial f(x,u)}{\partial u} = 0, \tag{4}$$

for all $x \in \mathcal{X}$. Taking one further time derivative yields

$$\begin{aligned}\ddot{y}(t) = C\ddot{x}(t) &= C\frac{\partial f(x,u)}{\partial t} \\ &= C\frac{\partial f(x,u)}{\partial x}\frac{\partial x}{\partial t} + \underbrace{C\frac{\partial f(x,u)}{\partial u}}_{=0 \text{ from (4)}}\frac{\partial u}{\partial t} = C\mathcal{D}f(x,u),\end{aligned}$$

where $\mathcal{D}f(x,u) := \frac{\partial f(x,u)}{\partial x} f(x,u)$ differs from a Lie derivative since f depends not only on x but also on u since (1) is *not control affine*. Iterating this process yields

$$y^{(k)}(t) = C\mathcal{D}^{k-1}f(x,u)$$

for all $k \in [\![0, r-1]\!]$ with

$$\mathcal{D}^{k+1}f := \frac{\partial \mathcal{D}^k f}{\partial x} f \quad \text{and} \quad \mathcal{D}^0 f := f.$$

Having $r \geq 2$ means that u does not appear in the expression of $\dot{y}$. Thus, a change in control input will not immediately modify y. We follow [16] and refer to the unactuated derivatives of y $\big(y^{(k)}$ for $k \in [\![1, r-1]\!]\big)$ as generalized inertia, by analogy to inertia in kinematic systems. To enforce constraint (2), we need to dissipate this generalized inertia before reaching constraint line $y = y_{max}$. To easily assess this generalized inertia, we make the following assumption.

Assumption 1. *There exists an invertible map* T *between state* $x \in \mathbb{R}^n$ *and* $s \in \mathbb{R}^n$*, whose first* r *components are* $s_1 = y$*,* $s_2 = \dot{y}$*, ...,* $s_r = y^{(r-1)}$*, where* y *is output* (2)*. Transformation* $T(x) = s$ *gives rise to an equivalent state space* $\mathcal{S} := T(\mathcal{X})$*.*

Note that this is a rather mild assumption. Indeed, a transformation $s = T(x)$ always exists since $y = Cx$ and thus $s_{k+2} = y^{(k+1)} = C\frac{\partial^k \dot{x}}{\partial t} = C\frac{\partial^k f}{\partial t}(x,u)$. Assumption 1 is required for the invertibility of T and the fact that T can be determined without knowledge of black-box dynamics f. For typical control systems as studied in Section IV, T is a simple function satisfying Assumption 1. Following Assumption 1, we now have two equivalent state representations:

x denotes the original state of system (1), while s denotes the transformed state composed of the iterated derivatives of constrained output y.

We can now formally define our problem of interest.

Problem 1. *Given:*

1) *black-box control system (1);*
2) *state space $\mathcal{X} \subseteq \mathbb{R}^n$;*
3) *admissible input set $\mathcal{U} \subseteq \mathbb{R}^m$;*
4) *affine constraint (2) of relative degree $r \geq 2$;*
5) *neural network policy $\pi_\theta(x)$ parameterized by θ;*
6) *invertible transformation T of Assumption 1;*

Our goal is to solve $\theta^* = \arg\max_\theta \mathcal{G}(\pi_\theta)$ s.t. (1) and (2).

III. Constrained Reinforcement Learning

In this section, we devise a method to solve Problem 1 by designing a buffer preventing trajectories from breaching the constraint, as illustrated in Fig. 1. To enforce high relative degree constraint $y \leq y_{max}$, our safe controller must dissipate its generalized inertia before reaching constraint line $y = y_{max}$. We force this dissipation in a buffer region $\mathcal{B}$ as illustrated in Fig. 1. We will first build such a buffer, then show that trajectories entering $\mathcal{B}$ cannot breach the constraint.

A. Buffer Design

We formalize the concept of Fig. 1 and build buffer $\mathcal{B}$ as the state space region where our controller will dissipate generalized inertia $\dot{y}$,...,$y^{(r-1)}$ to prevent violation of constraint (2). To adapt $\mathcal{B}$ to this task, we design it in state space $\mathcal{S}$, because Assumption 1 states that the coordinates of $\mathcal{S}$ are the generalized inertia components $y^{(k)}$.

Assume we can choose $s \in \mathcal{B}$ and let us investigate what bounds the components of s should satisfy to remain in $\mathcal{B}$. Following Assumption 1, the first component of s is $s_1 = y$, and thus should satisfy $s_1 \leq y_{max}$ to respect constraint (2). We choose a lower bound for s_1 as $y_{min} < y_{max}$.

Following Assumption 1, the second component of s is $s_2 = \dot{y}$. To maintain $y \leq y_{max}$, we need $\dot{y} \leq 0$ when $y = y_{max}$. Requiring $\dot{y} \leq 0$ for all $s \in \mathcal{B}$ is the approach of [15] but restricts $\mathcal{B}$ to only include states already moving away from upper bound y_{max}, whereas we want to slow down and stop trajectories going towards y_{max}. Thus, we must allow states with $\dot{y} > 0$ in $\mathcal{B}$. Let $\dot{y}_{max} > 0$ be the maximal velocity in $\mathcal{B}$. Our controller will later require $\mathcal{B}$ to be a polytope. Thus, we naturally define the upper bound on $\dot{y} = s_2$ as

$$s_2^{max} := \beta(y_{max} - y) \quad \text{with} \quad \beta := \frac{\dot{y}_{max}}{y_{max} - y_{min}}, \tag{5}$$

so that $s_2^{max}(y) = \dot{y}_{max}$ when $y = y_{min}$ and $s_2^{max} = 0$ when $y = y_{max}$, as illustrated in Fig. 1. We choose a lower bound $s_2^{min} \leq 0$ so that $s_2^{min} \leq s_2^{max}(y)$ for all y. Note that

$$s_2 = \dot{y} \leq s_2^{max}(y) = \beta(y_{max} - y) \tag{6}$$

is a differential inequality on y that we designed to maintain $y \leq y_{max}$. To enforce (6) we need a control input, but only $y^{(r)}$ is actuated. Our key idea is then to make our controller enforce a differential inequality on $y^{(r)}$, whose iterated integrations will lead to (6).

Working backwards, we differentiate (6) into $\ddot{y} \leq -\beta\dot{y}$, i.e., $s_3 \leq -\beta s_2$. Thus, we choose bounds $s_3^{min} < s_3^{max}(s) := -\beta s_2$. Iterating this process until $y^{(r)}$ leads to lower and upper bounds $\underline{b}$ and $\overline{b}$ on the first r components of $s \in \mathcal{B}$. Then, $s_{1:r} \in \big[\underline{b}, \overline{b}(s)\big]$ element-wise with

$$\underline{b} := \big[y_{min}, \quad s_2^{min}, \quad s_3^{min}, \ \ldots, \quad s_r^{min} \ \big], \tag{7}$$

$$\overline{b}(s) := \big[y_{max}, \ \beta(y_{max} - s_1), \ -\beta s_2, \ \ldots, \ -\beta s_{r-1}\big]. \tag{8}$$

The remaining $n-r$ components of $s \in \mathcal{B}$ are not derivatives of y and thus are not involved in the constraint enforcement process. As mentioned previously, we will need $\mathcal{B}$ to be a polytope, hence we choose to bound the last $n-r$ components of $s \in \mathcal{B}$ by a polytope $\mathcal{P} \subset \mathbb{R}^{n-r}$ so that

$$\mathcal{B} := \big\{s \in \mathcal{S} : s_{1:r} \in \big[\underline{b}, \ \overline{b}(s)\big], \ s_{r+1:n} \in \mathcal{P}\big\}. \tag{9}$$

Note that the bounds we just derived only delimit region $\mathcal{B}$ in $\mathcal{S}$, but without adequate control input, trajectories will not respect these bounds. Similarly, the differential inequalities obtained above only reflect the desired dynamics that we want to enforce with our controller. On the other hand, bounds $s_{1:r} \geq \underline{b}$ and $s_{r+1:n} \in \mathcal{P}$ will not be specifically enforced by the controller, but should be designed to encompass all trajectories to be safeguarded from y_{max}.

By design, buffer $\mathcal{B}$ of (9) is then a compact convex polytope with a finite number N of vertices gathered in the set $\mathcal{V}(\mathcal{B}) := \{v^1, \ldots, v^N\}$[1].

B. Controller Design

Let us now design a controller to maintain trajectories in buffer $\mathcal{B}$. Inspired by [15], we model our control policy with a POLICEd neural network $\mu_\theta := \pi_\theta \circ T : \mathcal{S} \to \mathcal{U}$, with continuous piecewise affine activation functions such as ReLU [20]. This restriction is of little concern as ReLU is the most commonly used activation function. This POLICEd architecture allows us to make the outputs of μ_θ affine over a polytopic region of state space $\mathcal{S}$, which we chose to be $\mathcal{B}$. Then, there exist matrices $D_\theta \in \mathbb{R}^{m \times n}$ and $e_\theta \in \mathbb{R}^m$ such that

$$\mu_\theta(s) = D_\theta s + e_\theta \quad \text{for all } s \in \mathcal{B}. \tag{10}$$

Since buffer $\mathcal{B}$ and policy μ_θ are both in space $\mathcal{S}$ and not $\mathcal{X}$, we need to calculate the state dynamics in $\mathcal{S}$. State $s = T(x)$ following controller μ_θ satisfies

$$\dot{s} = \frac{\partial T}{\partial t}(x) = \frac{\partial T}{\partial x}\frac{\partial x}{\partial t} = \frac{\partial T}{\partial x}(x) f\Big(x, \ \mu_\theta\big(T(x)\big)\Big).$$

Then, by defining the map

$$\widetilde{f}(s; \mu_\theta) := \frac{\partial T}{\partial x}\big(T^{-1}(s)\big) f\big(T^{-1}(s), \ \mu_\theta(s)\big),$$

we can write $\dot{s}(t) = \widetilde{f}\big(s(t); \mu_\theta\big)$. We are mostly interested in the dynamics of the actuated derivative of output y:

$$y^{(r)}(t) = \frac{\partial y^{(r-1)}}{\partial t}(t) = \frac{\partial s_r}{\partial t}(t) = \widetilde{f}_r\big(s(t); \mu_\theta\big), \tag{11}$$

[1]The number of vertices of $\mathcal{B}$ is related to the Fibonacci sequence (see Lemma 5).

where s_r and $\widetilde{f}_r$ denote the r^{th} component of s and $\widetilde{f}$ respectively. If dynamics (11) were known and affine, their coupling with affine policy μ_θ on $\mathcal{B}$ would lead to a simple constraint enforcement process. However, dynamics (11) are a black-box and possibly nonlinear. We will thus use an affine approximation of (11) inside using the following definition.

Definition 2. *An* approximation measure ε *of dynamics* (11) *in buffer* (9) *is any* $\varepsilon \geq 0$ *for which there exists any matrices* $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, *and* $c \in \mathbb{R}^n$ *such that*

$$\left|\widetilde{f}_r\big(s;\mu_\theta\big) - C\big(As + B\mu_\theta(s) + c\big)\right| \leq \varepsilon, \tag{12}$$

for all $s \in \mathcal{B}$.

Intuitively, the value of ε quantifies how far from affine is function $\widetilde{f}_r$ over buffer $\mathcal{B}$. Having access to map T, controller μ_θ and to a black-box model of f, we can evaluate $\widetilde{f}_r$ and compute ε using linear least square approximation [23]. Since ε is estimated from data, it might not verify (12) for some $s \in \mathcal{B}$ absent from the dataset. To satisfy Definition 2, we need to *over*-approximate ε since any upper bound will verify (12). With such an upper bound, we will guarantee the satisfaction of constraint (2) with actual dynamics (1).

We now establish our central result demonstrating how to *guarantee* satisfaction of constraint (2) by black-box environment (1) armed *only* with an approximation measure ε and *without* knowing A, B, c, f, or $\widetilde{f}_r$.

Theorem 1. *Assume that for some approximation measure* ε, *dissipation condition*

$$\widetilde{f}_r(v;\mu_\theta) \leq -2\varepsilon - \beta v_r, \tag{13}$$

holds for all $v \in \mathcal{V}\big(\mathcal{B}\big)$, *where* v_r *is the* r^{th} *component of* v *and* β *comes from* (5). *If a trajectory* s *steered by* μ_θ *verifies*

$$s_{1:r}(t_0) < \overline{b}\big(s(t_0)\big) \tag{14}$$

for some $t_0 \geq 0$, *and satisfies*

$$s_{1:r}(t) \geq \underline{b} \quad \textit{and} \quad s_{r+1:n}(t) \in \mathcal{P} \tag{15}$$

for all $t \in [t_0, t_1)$, *then* $s_{1:r}(t) < \overline{b}\big(s(t)\big)$ *for all* $t \in [t_0, t_1)$.

In simpler words, Theorem 1 guarantees that trajectories entering buffer $\mathcal{B}$ below upper bound $\overline{b}$ cannot exit $\mathcal{B}$ through $\overline{b}$ as long as dissipation condition (13) is satisfied. Theorem 1 generates the bent arrows of the flow illustrated in Fig. 1 which prevent trajectories from violating constraint (2). The major strength of our approach is that dissipation condition (13) only needs to be enforced at the vertices of $\mathcal{B}$ and thus does not require knowledge of f or $\widetilde{f}_r$.

Proof of Theorem 1. The intuition behind this proof is to use the convexity of buffer $\mathcal{B}$ and affine approximation (12) to extend condition (13) to the entire $\mathcal{B}$. By combining this condition with the specific design of upper bound (8), we can derive bounds on the output derivatives y, ..., $y^{(r-1)}$ and show that they cannot cross upper bound $\overline{b}$.

We divide the proof into three lemmas. First, we show in Lemma 1 that condition (13) yields $\dot{s}_r \leq -\beta s_r$ for all s in $\mathcal{B}$. This condition combined in Lemma 2 with (15) yields the differential inequalities to be respected by y, $\dot{y}$,..., $y^{(r-1)}$ as long as trajectory s remains in $\mathcal{B}$. In Lemma 3 these differential equations are then paired with initial conditions (14) to obtain $s_{1:r}(t) < \overline{b}\big(s(t)\big)$ for all $t \in [t_0, t_1)$. □

C. Supporting Lemmata

We now extend dissipation condition (13) from the vertices of buffer $\mathcal{B}$ to the whole set $\mathcal{B}$.

Lemma 1. *If for some approximation measure* ε, *condition* (13) *holds for all* $v \in \mathcal{V}(\mathcal{B})$, *then controller* μ_θ *yields*

$$\dot{s}_r(t) \leq -\beta s_r(t), \quad \textit{i.e.,} \quad y^{(r)}(t) \leq -\beta y^{(r-1)}(t), \tag{16}$$

for all $s(t) \in \mathcal{B}$.

Proof. Since ε is an approximation measure, there exist A, B and c verifying (12) which we evaluate at $s = v \in \mathcal{V}\big(\mathcal{B}\big)$

$$\begin{aligned} &C\big(Av + B\mu_\theta(v) + c\big) \\ &\quad\leq \left|C\big(Av + B\mu_\theta(v) + c\big) - \widetilde{f}_r(v;\mu_\theta)\right| + \widetilde{f}_r(v;\mu_\theta) \\ &\quad\leq \varepsilon - 2\varepsilon - \beta v_r, \end{aligned} \tag{17}$$

where the first inequality is a triangular inequality, the second follows from (12) and (13). Using the convexity of polytope $\mathcal{B}$ of vertices $\mathcal{V}\big(\mathcal{B}\big) = \big\{v^1, \ldots, v^N\big\}$, for any $s \in \mathcal{B}$, there exist $\alpha^1, \ldots, \alpha^N \in \mathbb{R}^+$ such that $\sum_{k=1}^N \alpha^k = 1$ and $s = \sum_{k=1}^N \alpha^k v^k$. Controller (10) applied at $s \in \mathcal{B}$ yields

$$\begin{aligned} C\big(As+B\mu_\theta(s) + c\big) &= C\big(As + B(D_\theta s + e_\theta) + c\big) \\ &= C(A + BD_\theta)s + C(Be_\theta + c) \\ &= C(A + BD_\theta)\sum_{k=1}^N \alpha^k v^k + C(Be_\theta + c)\sum_{k=1}^N \alpha^k \\ &= \sum_{k=1}^N \alpha^k C\big((A + BD_\theta)v^k + Be_\theta + c\big) \\ &= \sum_{k=1}^N \alpha^k C\big(Av^k + B\mu_\theta(v^k) + c\big) \\ &\leq \sum_{k=1}^N \alpha^k \big(-\varepsilon - \beta v_r^k\big) = -\varepsilon\sum_{k=1}^N \alpha^k - \beta\sum_{k=1}^N \alpha^k v_r^k \\ &= -\varepsilon - \beta s_r, \end{aligned} \tag{18}$$

where the only inequality comes from (17) on each vertex v^k and the last equality stems from the linear decomposition of component r of state s between component r of vertices v^k. For any state $s \in \mathcal{B}$, (11) yields

$$\begin{aligned} y^{(r)} &= \widetilde{f}_r(s,\mu_\theta) \\ &\leq \left|\widetilde{f}_r(s,\mu_\theta) - C\big(As+B\mu_\theta(s)+c\big)\right| + C\big(As+B\mu_\theta(s) + c\big) \\ &\leq \varepsilon - \varepsilon - \beta s_r = -\beta y^{(r-1)}, \end{aligned}$$

where we first use the triangular inequality, then (12) and (18), and the last equality comes from the definition of state s in Assumption 1. □

Lemma 1 uses the convexity of $\mathcal{B}$ and affine approximation (12) to extend (13), valid *only* at the vertices of $\mathcal{B}$, into (16), valid all over $\mathcal{B}$. Without the POLICE algorithm [20],

μ_θ would not be affine over $\mathcal{B}$, and dissipation condition (13) would need to be enforced everywhere on the buffer at a prohibitive computational cost.

Lemma 2. *If (16) holds for all $s \in \mathcal{B}$, $s(t_0) \in \mathcal{B}$, and (15) holds for all $t \in [t_0, t_1)$ for some $t_1 > t_0 \geq 0$, then*

$$y(t) \leq \big(y(t_0) - y_{max}\big)e^{-\beta(t-t_0)} + y_{max} \tag{19}$$

and

$$y^{(k)}(t) \leq y^{(k)}(t_0)e^{-\beta(t-t_0)} \tag{20}$$

for all $k \in [\![1, r-1]\!]$ and all $t \in [t_0, t_1)$.

Proof. We apply the comparison lemma of [24] to differential inequality (16), which yields $y^{(r-1)}(t) \leq y^{(r-1)}(t_0)e^{-\beta(t-t_0)}$.

Initial condition $s(t_0) \in \mathcal{B}$ yields

$$s_r(t_0) = y^{(r-1)}(t_0) \leq -\beta s_{r-1}(t_0) = -\beta y^{(r-2)}(t_0).$$

Define function $g(t) := y^{(r-1)}(t) + \beta y^{(r-2)}(t)$. Then, (16) is equivalent to $\dot{g}(t) \leq 0$ and our initial condition is $g(t_0) \leq 0$. Therefore, $g(t) \leq 0$ for all $t \in [t_0, t_1)$, i.e., $y^{(r-1)}(t) \leq -\beta y^{(r-2)}(t)$. Using the comparison lemma of [24], we can solve this differential inequality and obtain $y^{(r-2)}(t) \leq y^{(r-2)}(t_0)e^{-\beta(t-t_0)}$.

We can iterate this process for $k \in \{r-3, \dots, 1\}$ and obtain $y^{(k+1)}(t) \leq -\beta y^{(k)}(t)$, which yields $y^{(k)}(t) \leq y^{(k)}(t_0)e^{-\beta(t-t_0)}$ for all $t \in [t_0, t_1)$.

For $k = 1$, we thus have $\ddot{y}(t) + \beta\dot{y}(t) \leq 0$. Initial condition $s(t_0) \in \mathcal{B}$ yields $\dot{y}(t_0) \leq \beta\big(y_{max} - y(t_0)\big)$, or equivalently $\dot{y}(t_0) + \beta y(t_0) \leq \beta y_{max}$. As previously, with $g(t) := \dot{y}(t) + \beta y(t)$, we have $\dot{g}(t) \leq 0$ and $g(t_0) \leq \beta y_{max}$. Thus, $g(t) \leq \beta y_{max}$, i.e., $\dot{y}(t) + \beta y(t) \leq \beta y_{max}$ for all $t \in [t_0, t_1)$. We solve this differential inequality using the comparison lemma of [24] and obtain (19) for all $t \in [t_0, t_1)$. □

Lemma 2 used (16) and upper bound $\bar{b}$ of (8) to obtain the differential equations verified by the output derivatives in $\mathcal{B}$. We will now use initial condition (14) to show that trajectories cannot leave $\mathcal{B}$ through its upper bound $\bar{b}$.

Lemma 3. *If (16) holds for all $s \in \mathcal{B}$, (19) and (20) hold for all $t \in [t_0, t_1)$, then (14) implies $s_{1:r}(t) \leq \bar{b}(s(t))$ for all $t \in [t_0, t_1)$.*

Proof. Condition $s(t_0) < \bar{b}(s(t_0))$ yields $y(t_0) < y_{max}$. This initial condition combined with (19) leads to $y(t) < y_{max}$ for all $t \in [t_0, t_1)$, i.e., $s_1(t) < \bar{b}(s(t))_1$.
Initial condition $s_2(t_0) < \bar{b}(s(t_0))_2$ yields $\dot{y}(t_0) < \beta(y_{max} - y(t_0))$. Starting from (20) for $k = 1$, we have

$$\dot{y}(t) \leq \dot{y}(t_0)e^{-\beta(t-t_0)} < \beta(y_{max} - y(t_0))e^{-\beta(t-t_0)} \tag{21}$$

for all $t \in [t_0, t_1)$. Reorganizing (19) leads to $-y(t_0)e^{-\beta(t-t_0)} \leq -y(t) + y_{max}(1 - e^{-\beta(t-t_0)})$, which can be combined with (21) into

$$\begin{aligned}\dot{y}(t) &< \beta\big(y_{max}e^{-\beta(t-t_0)} - y(t) + y_{max}(1 - e^{-\beta(t-t_0)})\big) \\ &< \beta\big(y_{max} - y(t)\big)e^{-\beta(t-t_0)},\end{aligned}$$

i.e., $s_2(t) < \bar{b}(s(t))_2$ for all $t \in [t_0, t_1)$.

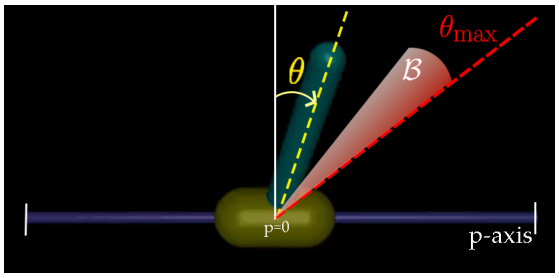

Fig. 2: The inverted pendulum Gym environment [25] annotated with cart position p, pendulum angle θ, and buffer $\mathcal{B}$.

Similarly for $k \in [\![2, r-1]\!]$, (20) combined with initial condition $y^{(k)}(t_0) < -\beta y^{(k-1)}(t_0)$ leads to

$$y^{(k)}(t) \leq y^{(k)}(t_0)e^{-\beta(t-t_0)} < -\beta y^{(k-1)}(t_0)e^{-\beta(t-t_0)} \tag{22}$$

for all $t \in [t_0, t_1)$. Reversing (20) at $k-1$ leads to $-y^{(k-1)}(t_0)e^{-\beta(t-t_0)} \leq -y^{(k-1)}(t)$, which can be combined with (22) into $y^{(k)}(t) < -\beta y^{(k-1)}(t)$, i.e., $s_{k+1}(t) < \bar{b}(s(t))_{k+1}$ for all $t \in [t_0, t_1)$. □

Remark 1. *Control set $\mathcal{U}$ of (1) might prevent the existence of an admissible stabilizing policy μ_θ. That is why we use RL to find policy μ_θ and we verify its safety with Theorem 1.*

Now that we have established our central result, we will illustrate its implementation on two numerical simulations.

IV. Numerical Simulations

A. Gym Inverted Pendulum

We consider the Inverted Pendulum Gym environment [25] with the MuJoCo dynamics engine [26] as illustrated in Fig. 2. The environment state x is composed, in that order, of the cart position p, the pole angle θ, and their derivatives $\dot{p}$ and $\dot{\theta}$. The objective within this environment is to maintain the pole close to the vertical, i.e., $|\theta| \leq 0.2$ rad. Let us focus on enforcing the upper constraint, $y := \theta \leq 0.2$ rad. This constraint has a relative degree $r = 2$ since the control input is the force exerted on the cart, which directly impacts $\ddot{\theta}$.

Following our choice of $y = \theta$, we define state $s = T(x)$

$$s := \begin{bmatrix} y \\ \dot{y} \\ s_3 \\ s_4 \end{bmatrix} = \begin{bmatrix} \theta \\ \dot{\theta} \\ p \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} p \\ \theta \\ \dot{p} \\ \dot{\theta} \end{bmatrix} = Tx.$$

Transformation T is linear and invertible, hence verifying Assumption 1. Additionally, we obtain T without violating the black-box assumption on dynamics f, since T only reorders the states.

Following Section III-A, we will now design buffer $\mathcal{B}$, whose architecture should help dissipate the inertia of trajectories arriving at $\theta = y_{min}$ with velocities $\dot{\theta} \leq \dot{y}_{max}$. We choose $y_{max} := 0.2$ rad, $y_{min} = 0.1$ rad, $s_2^{min} = 0$ rad/s

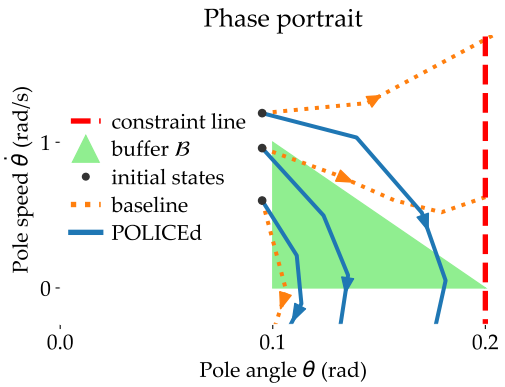

Fig. 3: Phase portrait of $(\theta, \dot{\theta})$ for the inverted pendulum. None of the POLICEd trajectories (blue) entering buffer $\mathcal{B}$ (green) cross constraint line $\theta = 0.2$ rad (dashed red), whereas some of the baseline trajectories do (dotted orange). Our approach guarantees that a pole arriving at $\theta = 0.1$ rad with a velocity $\dot{\theta} < 1$ rad/s will satisfy $\theta \leq 0.2$ rad. We do not guarantee the safety of POLICEd trajectories not entering the buffer.

and $\dot{y}_{max} = 1$ rad/s, and define buffer

$$\begin{aligned}\mathcal{B} := \big\{ s \in \mathcal{S} : s_1 = y = \theta \in [0.1,\, 0.2],\\ s_2 = \dot{y} = \dot{\theta} \in [0,\, 2 - 10\theta],\\ s_3 = p \in [-0.9,\, 0.9],\\ s_4 = \dot{p} \in [-1,\, 1] \big\}.\end{aligned}$$

This choice of $\mathcal{B}$ allows only $\dot{\theta} = 0$ rad/s when $\theta = 0.2$ rad, hence preventing θ in $\mathcal{B}$ from growing past 0.2 rad. Here polytope $\mathcal{P}$ of (9) is $\mathcal{P} := [-0.9, 0.9] \times [-1, 1]$.

Following Definition 2, we sample states in $\mathcal{B}$ and perform a linear regression on $\ddot{\theta}$ to obtain an approximation measure $\varepsilon = 0.53$. We model controller μ_θ with a deep neural network trained to stabilize the pole at $\theta = 0$ in a reinforcement learning fashion using proximal policy optimization (PPO) [27]. We train two such policies, one being a standard multi-layer perceptron (MLP) to form a baseline, and the other having POLICEd layers [20] enforcing affine condition (10). Both policies follow the same training and are encouraged to enforce dissipation condition (13) at the vertices of $\mathcal{B}$, which translates to $\ddot{\theta} \leq -2\varepsilon - 10\dot{\theta}$.

To illustrate Theorem 1, assume that our POLICEd controller μ_θ enforces dissipation condition (13) and let us consider a trajectory s entering buffer $\mathcal{B}$ at time t_0. If initial state condition (14) holds, i.e., if $\theta(t_0) < 0.2$ rad and $\dot{\theta}(t_0) < 2 - 10\theta(t_0)$, then as long as $\dot{\theta}(t) \geq 0.1$ rad, $\dot{\theta}(t) \geq 0$ rad/s, and $\big(p(t), \dot{p}(t)\big) \in \mathcal{P}$, we have $\theta(t) < 0.2$ rad and $\dot{\theta}(t) < 2 - 10\theta(t)$. These equations generate the phase portrait of Fig. 3, which successfully reproduces the desired behavior exhibited in Fig. 1.

Our POLICEd controller guarantees that all trajectories entering $\mathcal{B}$ cannot cross its upper bound $\bar{b}$ and hence cannot violate the constraint, whereas some baseline trajectories cross $\bar{b}$ and violate the constraint as shown in Fig. 3.

B. Space Shuttle Landing

We now study the highly-nonlinear dynamics of the space shuttle landing [28]. Original state $x \in \mathbb{R}^3$ is composed of the altitude h of the shuttle, its flight path angle γ, and its velocity v, as seen in Fig. 4. The dynamics of these state are

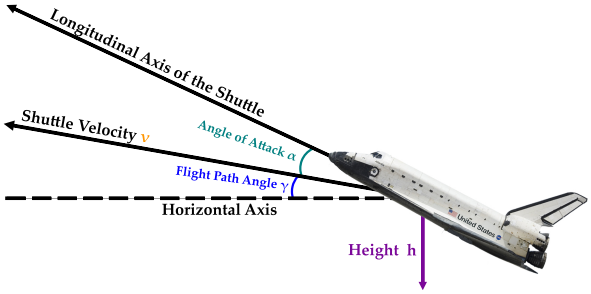

Fig. 4: Illustration of our Space Shuttle environment. The state $x \in \mathbb{R}^3$ is composed of the altitude or height h of the shuttle, its flight path angle γ, and its velocity v. The control action is the angle of attack α.

$$\dot{h}(t) = v(t)\sin\gamma(t) \tag{23a}$$

$$\dot{\gamma}(t) = \rho(t)v(t)C_L(t)\frac{S}{2m} - \frac{g\cos\gamma(t)}{v(t)} \tag{23b}$$

$$\dot{v}(t) = -\rho(t)v^2(t)C_D(t)\frac{S}{2m} - g\sin\gamma(t), \tag{23c}$$

where the air density satisfies $\rho(t) = \rho_0 e^{-h(t)/H}$ and the lift and drag coefficients take the form

$$\begin{aligned} C_L(t) &= C_{L_0}\sin^2\alpha(t)\cos\alpha(t)\\ C_D(t) &= C_{D_0} + KC_L^2(t). \end{aligned} \tag{24}$$

The other parameters are detailed in Table I. The control input is the angle-of-attack α of the shuttle, which makes these dynamics non-affine in control, and hence cannot be handled directly by any CBF method[2] [16], [17], [18].

In this scenario, the shuttle starts from a descent configuration with high vertical velocity $\dot{h}$, which must be drastically reduced to allow for a soft landing. More specifically, we consider initial states typical of a descent phase from a height $h_0 = 500$ ft, velocities $v_0 \in [300, 400]$ ft/s and flight path angles $\gamma_0 \in [-30°, -10°]$. The objective of our controller is to bring the shuttle to a low altitude $h \leq 50$ ft with vertical velocity $\dot{h} \leq 6$ ft/s sufficiently small to allow for a soft landing [28]. We choose an output constraint $y := -h \leq 0$, which has a relative degree 2 for control input α. We build a buffer with $y_{min} = -50$ ft, $y_{max} = 0$ ft, $s_2^{min} = 6$ ft/s and $\dot{y}_{max} = 100$ ft/s.

We introduce state $s := \big(h, \dot{h}, \gamma\big)$ and invertible transformation $T(x) = s$ is easily obtainable from pure geometric considerations in Fig. 4 as it only needs (23a). Thus, Assumption 1 is verified and determining T does not violate our black-box assumption on dynamics (23), (24).

We train two PPO policies [27] to minimize the vertical velocity at touchdown. One of these policies is a standard MLP used as a baseline and the other is our POLICEd version [15]. More implementation details are included in Appendix A. As seen in Fig. 5 our POLICEd policy successfully enforces the

[2]Adding an integrator $\dot{\alpha}(t) = u(t)$ renders dynamics (23) affine in control at the price of a higher relative degree [29].

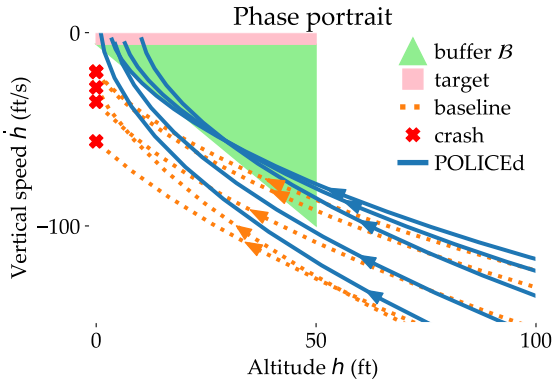

Fig. 5: Phase portrait of the space shuttle landing. POLICEd trajectories (blue) entering buffer $\mathcal{B}$ (green) all converge to a set of target conditions (pink) with small vertical velocity from which landing is feasible. However, the baseline trajectories (dotted orange) reach the ground $h = 0$ with high vertical velocities $\dot{h} \leq -6$ ft/s resulting in a crash of the shuttle (**x**).

dissipative buffer of Theorem 1 and ensures soft landing of the shuttle contrary to the baseline PPO policy.

V. Conclusion and Future Work

In this work, we established High Relative Degree POLICEd RL, a novel method to enforce a hard constraint of high relative degree on learned policies, while only using an implicit black-box model of the environment. We built a buffer region where the policy dissipates the generalized inertia of the high relative degree constraint to prevent trajectories from reaching the constraint line. We illustrated our theory on the MuJoCo inverted pendulum and on a space shuttle landing scenario.

Several avenues for future work seem especially interesting. Extending the POLICE algorithm of [20] to enforce multiple affine regions would allow a straightforward extension of this work to guarantee the satisfaction of multiple constraints of high relative degree. Another interesting pursuit would be to investigate how to guarantee constraint satisfaction during the training process of the policy.

Appendix

A. Space Shuttle Implementation Details

The baseline and POLICEd policies are both modeled by deep neural networks composed of 3 layers of 128 hidden units. Their reward function penalizes changes in the control input to encourage smooth variations of the angle of attack, and penalizes the final altitude and vertical velocity of the shuttle to promote soft landings. Its expression is

$$R(t) = -0.2\big|a(t) - a(t-dt)\big| - \mathbb{1}_{t=t_f}\big(|h(t_f)| + |\dot{h}(t_f)|\big),$$

where $\mathbb{1}_{t=t_f}$ is the final time t_f indicator function.

Remark 2. *Choosing α as input might seem unrealistic since α must be continuous. However, our control signal $u(t)$ is continuous by construction as a continuous function of the state $\mu_\theta(x(t))$. Adding an integrator $\dot{\alpha}(t) = u(t)$ as in [30] could make a reasonable input choice but at the price of increased complexity in calculating T^{-1} to recover α from s.*

TABLE I: Numerical values for the shuttle simulation from [28].

Parameter	Name	Value
S/m	surface area over mass	0.9118 ft^2/slug
C_{L_0}	zero-angle-of-attack lift coefficient	2.3
C_{D_0}	zero-lift drag coefficient	0.0975
K	lift-induced drag coef. parameter	0.1819
ρ_0	sea-level air density	0.0027 slugs/ft^3
g	Earth's gravitational acceleration	32, 174 ft/s^2
H	scale height	27890 ft

B. Supporting Lemmata

Recall that Theorem 1 guarantees the respect of upper bound $\overline{b}$ only while the trajectory remains in $\mathcal{B}$. We can thus strengthen Theorem 1 by deriving a lower bound $\underline{b}$ sufficiently low for upper bound $\overline{b}(s)$ never to cross $\underline{b}$ and cause a trajectory to prematurely exit buffer $\mathcal{B}$. To put it simply, we want $\underline{b} \leq \overline{b}(s)$ for all $s \in \mathcal{B}$.

Lemma 4. *Condition $\underline{b} \leq \overline{b}(s)$ for all $s \in \mathcal{B}$ is equivalent to $s_2^{min} \leq 0$, $s_{2k+1}^{min} \leq -\beta^{2k-1}\dot{y}_{max}$ and $s_{2k+2}^{min} \leq \beta^{2k} s_2^{min}$ for all $k \in [\![1, r/2]\!]$.*

Proof. Let $s \in \mathcal{B}$. Then, $s_2 \in \big[s_2^{min}, \beta(y_{max} - s_1)\big]$.

$$s_2^{min} \leq \min_s \overline{b}_2(s) = \min_s \beta(y_{max} - s_1) = 0,$$

since $s_1 \leq y_{max}$. Hence, $s_2^{min} \leq 0$. Similarly,

$$s_3^{min} \leq \min_s \overline{b}_3(s) = \min_s -\beta s_2 = -\beta \max_s s_2 = -\beta\dot{y}_{max}.$$

For $k \in [\![2, r-2]\!]$, we have

$$\begin{aligned} s_{k+2}^{min} &\leq \min_s \overline{b}_{k+2}(s) = \min_s -\beta s_{k+1} = -\beta \max_s s_{k+1} \\ &\leq -\beta \max_s \overline{b}_{k+1}(s) = -\beta \max_s -\beta s_k = \beta^2 \min_s s_k \\ &\leq \beta^2 s_k^{min}. \end{aligned}$$

Applying this inequality recursively leads to $s_{2k+1}^{min} \leq -\beta^{2k-1} s_3^{min} \leq -\beta^{2k-1}\dot{y}_{max}$ and $s_{2k+2}^{min} \leq \beta^{2k} s_2^{min}$ for all $k \in [\![1, r/2]\!]$. □

Armed with Lemma 4, we can now calculate the minimal number of vertices of buffer $\mathcal{B}$, which corresponds to the case where $\underline{b} = \min_s \overline{b}(s)$. We first need to introduce the Fibonacci sequence as $F_{n+2} = F_{n+1} + F_n$ for all $n \in \mathbb{N}$ with $F_0 = 0$ and $F_1 = 1$ [31]. We recall that r is the relative degree of output y of (2) with respect to dynamics (1). We denote the cardinality of a set $\mathcal{Z}$ as $\big|\mathcal{Z}\big|$.

Lemma 5. *If $\underline{b} = \min_s \overline{b}(s)$, then the number of vertices of $\mathcal{B}$ is $F_{r+2} \times \big|\mathcal{V}(\mathcal{P})\big|$.*

Proof. We introduce $\mathcal{B}^r := \big\{s_{1:r} \in [\underline{b},\ \overline{b}(s)] : s \in \mathcal{B}\big\}$. Then, buffer $\mathcal{B}$ of (9) can be written as the Cartesian product $\mathcal{B} = \mathcal{B}^r \times \mathcal{P}$. Since $\big|\mathcal{V}(\mathcal{P})\big|$ denotes the number of vertices of polytope $\mathcal{P}$, we only need to show that the number of vertices of $\mathcal{B}^r$ is equal to F_{r+2}.

Following Lemma 4, $\underline{b} = \min_s \overline{b}(s)$ yields, $s_{2k}^{min} = 0$ and $s_{2k+1}^{min} = -\beta^{2k-1}\dot{y}_{max}$ for all $k \in [\![1, r/2]\!]$. Since upper bound $\overline{b}_k(s)$ depends on the value of s_{k-1}, to enumerate the

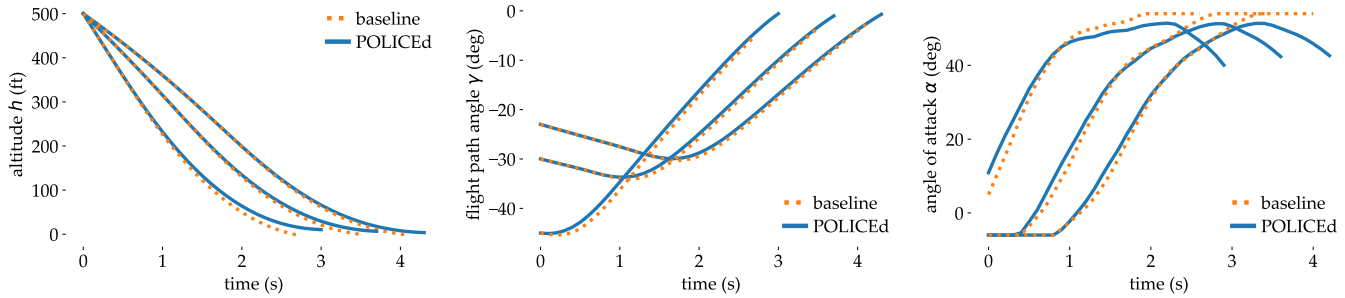

Fig. 6: Evolution of the altitude h, flight path angle γ and angle of attack α for three trajectories of the shuttle. All POLICEd trajectories reach level flight $\gamma = 0°$ at touchdown, which also appears as the flattening of the altitude plot at $h = 0$ ft, on the contrary to the baseline. The POLICEd controller also learns to reduce the angle of attack α for landing, while the baseline remain at saturation.

vertices of $\mathcal{B}^r$ we need to first list the possibilities for s_1, then for s_2, and so on. At dimension $k \in [\![2, r]\!]$, we have $s_k \in [s_k^{min}, -\beta s_{k-1}]$. If $s_{k-1} = s_{k-1}^{min}$, then $s_k \in [s_k^{min}, s_k^{max}]$. If $s_{k-1} = s_{k-1}^{max}$, then $s_k^{max} = -\beta s_{k-1} = s_k^{min}$ by assumption $\underline{b} = \min_s \bar{b}(s)$.

To enumerate the number of vertices of $\mathcal{B}^r$, we build a tree listing all the possibilities where each level correspond to a dimension as shown in Fig. 7.

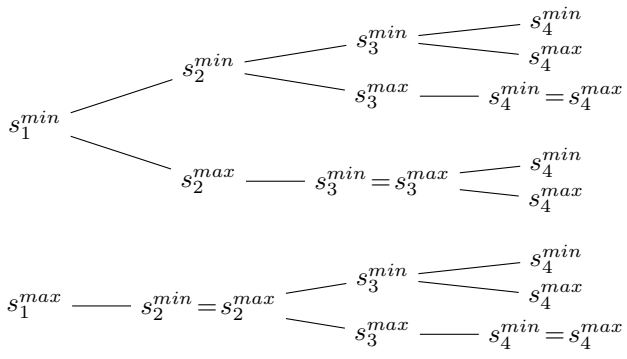

Fig. 7: Tree of all the vertex combinations for $\mathcal{B}_r$. A node s_k^{min} allows s_{k+1} to take value in its whole range $[s_{k+1}^{min}, s_{k+1}^{max}]$ and thus yields two nodes s_{k+1}^{min} and s_{k+1}^{max}. However, a node s_k^{max} causes $\bar{b}_k = \underline{b}_k$ and thus yields a single node $s_{k+1}^{min} = s_{k+1}^{max}$. Finally, a node $s_k^{min} = s_k^{max}$ forces $s_k = s_k^{min}$ and thus s_{k+1} can use its whole range, which yields two nodes s_{k+1}^{min} and s_{k+1}^{max}.

We count the nodes of the tree. At level $k \in [\![1, r]\!]$ we define the number of nodes where $s_k = s_k^{min}$ as $n^{min}(k)$, the number of nodes where $s_k = s_k^{max}$ as $n^{max}(k)$, the number of nodes where $s_k = s_k^{min} = s_k^{max}$ as $n^{nx}(k)$, and the total number of nodes as $n(k) := n^{min}(k) + n^{max}(k) + n^{nx}(k)$.

A node s_k^{min} allows s_{k+1} to take value in its whole range $[s_{k+1}^{min}, s_{k+1}^{max}]$ and thus yields two nodes s_{k+1}^{min} and s_{k+1}^{max}. However, a node s_k^{max} causes $\bar{b}_k = \underline{b}_k$ and thus yields a single node $s_{k+1}^{min} = s_{k+1}^{max}$. Finally, a node $s_k^{min} = s_k^{max}$ forces $s_k = s_k^{min}$ and thus s_{k+1} can use its whole range, which again yields two nodes s_{k+1}^{min} and s_{k+1}^{max}.

Thus, each node $s_{k+1}^{min} = s_{k+1}^{max}$ is created by a node s_k^{max}, so that $n^{nx}(k+1) = n^{max}(k)$. On the other hand, there is a s_{k+1}^{min} and s_{k+1}^{max} node for each s_k^{min} and each $s_k^{min} = s_k^{max}$ nodes, thus $n^{min}(k+1) = n^{max}(k+1) = n^{min}(k) + n^{nx}(k)$.

Then, for $k \in [\![3, r]\!]$, we have

$$\begin{aligned} n(k+1) &= n^{min}(k+1) + n^{max}(k+1) + n^{nx}(k+1) \\ &= 2n^{min}(k) + n^{max}(k) + 2n^{nx}(k) \\ &= n(k) + n^{min}(k) + n^{nx}(k) \\ &= n(k) + n^{min}(k-1) + n^{nx}(k-1) + n^{max}(k-1) \\ &= n(k) + n(k-1). \end{aligned}$$

Therefore, n verifies the recursion relation of the Fibonacci sequence [31]. Additionally, Fig. 7 shows that $n(1) = 2 = F_3$ and $n(2) = 3 = F_4$. Thus, $n(k) = F_{k+2}$. □

REFERENCES

[1] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis," *Machine Learning*, vol. 110, no. 9, pp. 2419 – 2468, 2021.

[2] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411 – 444, 2022.

[3] E. Altman, *Constrained Markov decision processes*. Routledge, 2021.

[4] W. Zhao, R. Chen, Y. Sun, T. Wei, and C. Liu, "State-wise constrained policy optimization," *arXiv preprint arXiv:2306.12594*, 2023.

[5] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *International Conference on Machine Learning*. PMLR, 2017, pp. 22 – 31.

[6] Z. Liu, Z. Guo, Y. Yao, Z. Cen, W. Yu, T. Zhang, and D. Zhao, "Constrained decision transformer for offline safe reinforcement learning," *arXiv preprint arXiv:2302.07351*, 2023.

[7] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, Y. Yang, and A. Knoll, "A review of safe reinforcement learning: Methods, theory and applications," *arXiv preprint arXiv:2205.10330*, 2022.

[8] W. Xiao, T.-H. Wang, R. Hasani, M. Chahine, A. Amini, X. Li, and D. Rus, "BarrierNet: Differentiable control barrier functions for learning of safe robot control," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 2289 – 2307, 2023.

[9] N. Rober, S. M. Katz, C. Sidrane, E. Yel, M. Everett, M. J. Kochenderfer, and J. P. How, "Backward reachability analysis of neural feedback loops: Techniques for linear and nonlinear systems," *IEEE Open Journal of Control Systems*, vol. 2, pp. 108–124, 2023.

[10] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, "Safe exploration in continuous action spaces," *arXiv preprint arXiv:1801.08757*, 2018.

[11] T.-H. Pham, G. De Magistris, and R. Tachibana, "Optlayer - practical constrained optimization for deep reinforcement learning in the real world," in *International Conference on Robotics and Automation*. IEEE, 2018, pp. 6236–6243.

[12] Z. Qin, D. Sun, and C. Fan, "Sablas: Learning safe control for black-box dynamical systems," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1928–1935, 2022.

[13] H. Ma, C. Liu, S. E. Li, S. Zheng, and J. Chen, "Joint synthesis of safety certificate and safe control policy using constrained reinforcement learning," in *Learning for Dynamics and Control Conference*. PMLR, 2022, pp. 97–109.

[14] Y. Yang, Y. Jiang, Y. Liu, J. Chen, and S. E. Li, "Model-free safe reinforcement learning through neural barrier certificate," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1295–1302, 2023.

[15] J.-B. Bouvier, K. Nagpal, and N. Mehr, "POLICEd RL: Learning closed-loop robot control policies with provable satisfaction of hard constraints," *ArXiv*, 2024.

[16] J. Breeden and D. Panagou, "High relative degree control barrier functions under input constraints," in *60th Conference on Decision and Control*. IEEE, 2021, pp. 6119–6124.

[17] Q. Nguyen and K. Sreenath, "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints," in *American Control Conference*. IEEE, 2016, pp. 322 – 328.

[18] W. Xiao and C. Belta, "Control barrier functions for systems with high relative degree," in *58th Conference on Decision and Control*. IEEE, 2019, pp. 474 – 479.

[19] P. V. Kokotovic, "The joy of feedback: nonlinear and adaptive," *IEEE Control Systems Magazine*, vol. 12, no. 3, pp. 7–17, 1992.

[20] R. Balestriero and Y. LeCun, "POLICE: Provably optimal linear constraint enforcement for deep neural networks," in *International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2023, pp. 1–5.

[21] S. Sastry, *Nonlinear systems: analysis, stability, and control*. Springer Science & Business Media, 1999.

[22] T. Westenbroek, D. Fridovich-Keil, E. Mazumdar, S. Arora, V. Prabhu, S. S. Sastry, and C. J. Tomlin, "Feedback linearization for uncertain systems via reinforcement learning," in *International Conference on Robotics and Automation*. IEEE, 2020, pp. 1364 – 1371.

[23] G. Golub, "Numerical methods for solving linear least squares problems," *Numerische Mathematik*, vol. 7, pp. 206–216, 1965.

[24] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, 2002.

[25] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv preprint arXiv:1606.01540*, 2016.

[26] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[28] N. Harl and S. N. Balakrishnan, "Reentry terminal guidance through sliding mode control," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 1, pp. 186 – 199, 2010.

[29] J. Yao, "Finite time suboptimal control design of nonlinear systems with θ–D technique and implementation to aerospace applications," Ph.D. dissertation, Missouri University of Science and Technology, 2020.

[30] A. Heydari and S. N. Balakrishnan, "Optimal online path planning for approach and landing guidance," in *AIAA Atmospheric Flight Mechanics Conference*, 2011.

[31] L. Sigler, *Fibonacci's Liber Abaci: a translation into modern English of Leonardo Pisano's book of calculation*. Springer Science & Business Media, 2003.