Neural Network-based Universal Formulas for Control

Pol Mestres[a,*], Jorge Cortés[b], Eduardo D. Sontag[c]

[a]*Department of Mechanical and Civil Engineering, California Institute of Technology, Pasadena, CA, USA*
[b]*Department of Mechanical and Aerospace Engineering, UC San Diego, La Jolla, CA, USA*
[c]*Departments of Electrical and Computer Engineering and Bioengineering, Northeastern University, Boston, MA, USA*

Abstract

We study the problem of designing a controller that satisfies an arbitrary number of affine inequalities at every point in the state space. This is motivated by the fact that a variety of key control objectives, such as stability, safety, and input saturation, are guaranteed by closed-loop systems whose controllers satisfy such inequalities. Many works in the literature design such controllers as the solution to a state-dependent quadratic program (QP) whose constraints are precisely the inequalities. When the input dimension and number of constraints are high, computing a solution of this QP in real time can become computationally burdensome. Additionally, the solution of such optimization problems is not smooth in general, which can degrade the performance of the system. This paper provides a novel method to design a smooth controller that satisfies an arbitrary number of affine constraints. The controller is given at every state as the minimizer of a strictly convex function. To avoid computing the minimizer of such function in real time, we introduce a method based on neural networks (NN) to approximate the controller. Remarkably, this NN can be used to solve the controller design problem for any task with less than a fixed input dimension and number of affine constraints, and is completely independent of the state dimension. This is why we refer to such NN approximation as a NN-based universal formula for control. Additionally, we show that the NN-based controller only needs to be trained with datapoints from a bounded set in the state space, which significantly simplifies the training process. Various simulations showcase the performance of the proposed solution, and also show that the NN-based controller can be used to warmstart an optimization scheme that refines the approximation of the true controller in real time, significantly reducing the computational cost compared to a generic initialization.

Keywords: Universal formula, control Lyapunov function, control barrier function, neural network

1. Introduction

Modern autonomous systems, ranging from self-driving vehicles to aerospace systems, are usually subject to a variety of operational requirements. These include, but are not limited to, trajectory tracking, disturbance rejection, stabilization, and guaranteeing state and input constraints. Although there exist technical tools to design controllers that satisfy each of these specifications, combining them into a single controller that meets all of them is often challenging. For example, control Lyapunov (respectively, barrier) functions define state-dependent input constraints that guarantee stability (respectively, safety) of the underlying control system. However, designing a smooth control law that satisfies all such constraints and can be computed efficiently in real time is significantly difficult. Addressing this problem is the main motivation for our work here.

1.1. Literature Review

Control Lyapunov Functions (CLFs) [1, 2] provide a powerful tool to achieve the stabilization of nonlinear control systems. For control-affine systems, CLFs prescribe an affine inequality in the system input at every state. By design, controllers that

*Corresponding author
Email addresses: mestres@caltech.edu (Pol Mestres), cortes@ucsd.edu (Jorge Cortés), e.sontag@northeastern.edu (Eduardo D. Sontag)

satisfy this inequality, such as the pointwise minimum norm controller [3] or the CLF universal formula [4], ensure stability of the closed-loop system. More recently, Control Barrier Functions (CBFs) [5, 6, 7] have been introduced to achieve safety requirements for nonlinear control systems. Similarly to CLFs, CBFs also prescribe affine inequalities in the input at every state for control-affine systems, and various safe control designs leveraging this inequality have been proposed [5, 8, 9]. Affine inequalities in the input also arise if control authority is limited (as is always the case in practice), or if stability needs to be guaranteed in fixed time [10]. Often, it is common to face scenarios where the controller must meet multiple objectives, which are encoded by multiple affine inequalities in the input. There are also a variety of works in the literature that design controllers in this setting. For example, the CLF-CBF QP in [6, 11] include both affine constraints in a quadratic program (QP), whose solution defines the desired controller. Alternatively, [12] unites a CLF and a CBF into a unique function, called control Lyapunov barrier function (CLBF), and uses the known CLF universal formula to derive a controller. Other control designs include [13], which use penalty methods to impose one of the inequalities as a *hard* constraint and the other one as a *soft* constraint, [14], which defines a formula based on the centroids of the two feasible sets defined by the two inequalities, or [15], where different CBF constraints are united into a single one by designing a smooth approximation of the intersection of the different safe sets.

However, the aforementioned approaches only lead to a controller that can be obtained in closed-form or in a computationally tractable manner for a low number of constraints [16]. To the best of our knowledge, there does not exist an approach in the literature for obtaining a controller with such properties for an arbitrary number of state-dependent affine constraints. This problem raises intertwined theoretical, practical, and computational challenges. Theoretically, the solution of the QP should be computed at every state, which is impossible in practice. This has led to the use sampled-data implementations of such controllers [17, 18, 19]. Even with a sampled-data implementation, from a computational standpoint, solving the QP in real time becomes burdensome and the mismatch between the computed solution and the actual one may degrade system performance.

The problem of designing such a controller is even more challenging if it is required to be smooth. The use of smooth controllers is motivated by theoretical reasons (e.g., existence and uniqueness of solutions and use in backstepping designs [20]) and practical considerations (e.g., avoidance of chattering behavior in digital platforms). Although the existence of an infinitely-times differentiable controller satisfying an arbitrary number of affine state-dependent constraints is guaranteed by [14, Proposition 3.1], a constructive method for finding such a controller is lacking. In fact, optimization-based designs such as the QP in [6] can fail to be locally Lipschitz (as exemplified by Robinson's counterexample [21]) or even continuous [22], and additional constraint qualifications are needed in order to ensure their regularity [21, 23].

Our work here is also related to a line of research that uses neural networks (NNs) to approximate the solution of model predictive control (MPC) problems [24, 25, 26, 27]. In the case of linear dynamics and affine input and state constraints, the solution of such MPC problem is known [28] to be piecewise affine, but computing it explicitly is difficult because the number of regions grows exponentially with the number of constraints. These works show that an approximate solution computed offline with a NN can be used to warmstart the MPC optimization problem online and considerably speed up its computation.

Statement of Contributions

We study the problem of designing a smooth controller that satisfies an arbitrary number of affine constraints at every point in the state space. In our first contribution we introduce a novel controller that generalizes the CLF universal formula and satisfies all the affine inequalities. We show that such controller is smooth, and can be computed as the minimizer of a strictly convex function.

Since a closed-form expression for the controller is not available for cases with more than one constraint, our second contribution consists of a numerical scheme to approximate the controller using NNs. Remarkably, we show that a single neural network (appropriately trained with parameters equal to the coefficients of the affine inequalities) is enough to obtain controllers for any task involving systems with the same input dimension and the same number of inequalities, regardless of the state dimension. Additionally, we show that such NN approximation can be obtained by only sam-

pling data points over a compact set of parameters, and that the NN-based approximate controller is smooth (by taking the activation functions of the NN to be smooth). Although such approximation is not guaranteed to be close to the true controller and to satisfy the inequalities at every point, the universal approximation theorem of NNs ensures that it becomes an arbitrarily good approximation of the true controller as the number of parameters of the NN increases. Additionally, using recent advances in training NNs with hard constraints, the NN can be trained so that the NN-based controller satisfies the affine constraints at every point by construction, although in that case it is not guaranteed to be smooth. Furthermore, in real-time control applications, the NN approximations can be used to warmstart an optimization scheme to compute the true controller with higher accuracy.

Finally, in our third contribution we use the NN approximations in two safe stabilization tasks, both directly and as a warmstart of an optimization scheme to compute the true controller value. We show that the use of the NN significantly reduces the execution time of the controller compared to other standard controllers in the literature such as the minimum-norm CLF-CBF QP controller [6], which need to be computed online by solving an optimization problem at every state.

Notation

We denote by $\mathbb{Z}_{>0}$ and $\mathbb{R}$ the set of positive integers and real numbers, respectively. For $N \in \mathbb{Z}_{>0}$, we write $[N] = \{1, \dots, N\}$. Given $l \in \mathbb{Z}_{>0}$ and $\mathcal{S} \subset \mathbb{R}^n$, the set of l-times continuously differentiable functions in $\mathcal{S}$ is denoted by $\mathcal{C}^l(\mathcal{S})$. Vectors are represented by boldface symbols whereas scalars are represented by non-boldface symbols. The zero vector in $\mathbb{R}^n$ is denoted by $\mathbf{0}_n$, and the identity matrix of dimension $n \times n$, by $\mathbf{I}_n$. Given $\mathbf{x} \in \mathbb{R}^n$, $\|\mathbf{x}\|$ denotes its Euclidean norm. A function $\beta : \mathbb{R} \to \mathbb{R}$ is of class $\mathcal{K}$ if $\beta(0) = 0$ and β is strictly increasing. If moreover, $\lim_{t\to\infty} \beta(t) = \infty$ and $\lim_{t\to-\infty} \beta(t) = -\infty$, then β is of extended class $\mathcal{K}_\infty$. A function $V : \mathbb{R}^n \to \mathbb{R}$ is positive definite if $V(\mathbf{0}_n) = \mathbf{0}_n$ and $V(\mathbf{x}) > 0$ for all $\mathbf{x} \neq \mathbf{0}_n$. The ball centered at $\mathbf{p} \in \mathbb{R}^n$ with radius $r > 0$ is denoted by $\mathcal{B}_r(\mathbf{p})$. Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be a locally Lipschitz vector field and consider the dynamical system $\dot{\mathbf{x}} = F(\mathbf{x})$. Local Lipschitzness of F ensures that, for every initial condition $\mathbf{x}_0 \in \mathbb{R}^n$, there exists $T > 0$ and a unique trajectory $x : [0, T] \to \mathbb{R}^n$ such that $x(0) = \mathbf{x}_0$ and $\dot{x}(t) = F(x(t))$. If all solutions exist for all $t \geq 0$, then we say the dynamical system is forward complete. In this case, we let $\Phi_t : \mathbb{R}^n \to \mathbb{R}^n$ denote the flow map, defined by $\Phi_t(\mathbf{x}) = x(t)$, where $x(t)$ is the unique solution of the dynamical system starting at $x(0) = \mathbf{x}$. Given a forward complete dynamical system, a set $\mathcal{K} \subset \mathbb{R}^n$ is (positively) forward invariant if $\mathbf{x} \in \mathcal{K}$ implies that $\Phi_t(\mathbf{x}) \in \mathcal{K}$ for all $t \geq 0$. A point $\mathbf{p}$ is Lyapunov stable if, for every open set U containing $\mathbf{p}$, there exists an open set $\bar{U}$ also containing $\mathbf{p}$ such that for all $\mathbf{x} \in \bar{U}$, $\Phi_t(\mathbf{x}) \in U$ for all $t \geq 0$. An equilibrium point $\mathbf{p}$ is asymptotically stable if it is Lyapunov stable and there exists an open set $\bar{U}$ containing $\mathbf{p}$ such that $\Phi_t(\mathbf{x}) \to \mathbf{p}$ as $t \to \infty$ for all $\mathbf{x} \in \bar{U}$.

2. Problem Statement

Consider a nonlinear control-affine system of the form

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}, \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are locally Lipschitz, with $\mathbf{x} \in \mathbb{R}^n$ the state and $\mathbf{u} \in \mathbb{R}^m$ the input. We assume that $f(\mathbf{0}_n) = \mathbf{0}_n$, so that the origin is an equilibrium of the unforced system. The following motivating examples provide instances on how state-dependent affine inequalities in the input arise when designing controllers aimed at meeting specific control objectives.

Example 2.1. (Control Lyapunov function for stabilization): *A continuously differentiable positive definite function $V : \mathbb{R}^n \to \mathbb{R}$ with compact level sets is a control Lyapunov function (CLF) if there exists a neighborhood $\mathcal{D}$ of the origin such that for all $\mathbf{x} \in \mathcal{D}$, there exists $\mathbf{u} \in \mathbb{R}^m$ such that*

$$\nabla V(\mathbf{x})^\top f(\mathbf{x}) + \nabla V(\mathbf{x})^\top g(\mathbf{x})\mathbf{u} + W(\mathbf{x}) \leq 0, \tag{2}$$

where $W : \mathbb{R}^n \to \mathbb{R}$ is a positive definite function. A locally Lipschitz controller $k : \mathbb{R}^n \to \mathbb{R}^m$ such that $\mathbf{u} = k(\mathbf{x})$ satisfies the inequality (2) at every $\mathbf{x} \in \mathcal{D}$ renders the origin of the closed-loop system $\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})k(\mathbf{x})$ asymptotically stable. •

Example 2.2. (Control barrier function for ensuring safety): *Let $\mathcal{C} \subset \mathbb{R}^n$ be a safe set of interest, and suppose that $\mathcal{C}$ is given by the 0-superlevel set of a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$, i.e., $\mathcal{C} = \{\mathbf{x} \in \mathbb{R}^n : h(\mathbf{x}) \geq 0\}$. This function is*

a control barrier function (CBF) of $\mathcal{C}$ if, for every $\mathbf{x} \in \mathcal{C}$, there exists $\mathbf{u} \in \mathbb{R}^m$ such that

$$\nabla h(\mathbf{x})^\top f(\mathbf{x}) + \nabla h(\mathbf{x})^\top g(\mathbf{x})\mathbf{u} + \alpha(h(\mathbf{x})) \geq 0, \quad (3)$$

where $\alpha : \mathbb{R} \to \mathbb{R}$ is an extended class $\mathcal{K}_\infty$ function. A locally Lipschitz controller $k : \mathbb{R}^n \to \mathbb{R}^m$ such that $\mathbf{u} = k(\mathbf{x})$ satisfies the inequality (3) at every $\mathbf{x} \in \mathcal{C}$ renders the set $\mathcal{C}$ forward invariant for the closed-loop system $\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})k(\mathbf{x})$. •

Examples 2.1 and 2.2 provide two illustrations of control-theoretic properties (stability and safety) that can be achieved by designing controllers that satisfy state-dependent affine inequalities in the input. In general, one can have an arbitrary number of such affine inequalities to encode, for instance, safety constraints related to collision avoidance with different obstacles in the environment.

Formally, let $N \in \mathbb{Z}_{>0}$, and for $i \in [N]$, let $a_i : \mathbb{R}^n \to \mathbb{R}$, $b_i : \mathbb{R}^n \to \mathbb{R}^m$ be $\mathcal{C}^l(\mathbb{R}^n)$ functions for some $l \in \mathbb{Z}_{\geq 0}$. Let $\mathcal{F}$ be the set of points where the inequalities $\{a_i(\mathbf{x}) + b_i(\mathbf{x})^\top \mathbf{u} < 0\}_{i\in[N]}$ are simultaneously strictly feasible, i.e.,

$$\mathcal{F} := \{\mathbf{x} \in \mathbb{R}^n : \exists \mathbf{u} \in \mathbb{R}^m \text{ s.t. } a_i(\mathbf{x}) + b_i(\mathbf{x})^\top \mathbf{u} < 0, \ \forall i \in [N]\}.$$

Since $\mathcal{F}$ is the projection in $\mathbb{R}^n$ of the open set $\{(\mathbf{x}, \mathbf{u}) \in \mathbb{R}^n \times \mathbb{R}^m : a_i(\mathbf{x}) + b_i(\mathbf{x})^\top \mathbf{u} < 0, \ \forall i \in [N]\}$, $\mathcal{F}$ is also open. In the case where $N = 2$ and the two inequalities correspond to the strict versions of the CLF and CBF inequalities (2) and (3), respectively, we have

$$\begin{aligned} a_1(\mathbf{x}) &= \nabla V(\mathbf{x})^\top f(\mathbf{x}) + W(\mathbf{x}), \\ b_1(\mathbf{x}) &= g(\mathbf{x})^\top \nabla V(\mathbf{x}), \\ a_2(\mathbf{x}) &= -\nabla h(\mathbf{x})^\top f(\mathbf{x}) - \alpha(h(\mathbf{x})), \\ b_2(\mathbf{x}) &= -g(\mathbf{x})^\top \nabla h(\mathbf{x}). \end{aligned}$$

In this case, the property that $\mathcal{F} \neq \emptyset$ is known as strict compatibility of the CLF and CBF pair, cf. [14, 29]. [14, Proposition 3.1] is an extension of Artstein's Theorem (cf. [30]) showing that if $\mathcal{F}$ is non-empty, there exists a $\mathcal{C}^\infty$ controller $k : \mathcal{F} \to \mathbb{R}^m$ such that $a_i(\mathbf{x}) + b_i(\mathbf{x})^\top k(\mathbf{x}) < 0$ for all $\mathbf{x} \in \mathcal{F}$ and all $i \in [N]$. The proof of this result, however, is not constructive, which is undesirable for the real-time control of the nonlinear system (1). Hence, in this paper we set out to solve the following problem.

Problem 1. *Design a smooth controller that satisfies the constraints $a_i(\mathbf{x}) + b_i(\mathbf{x})^\top \mathbf{u} < 0$ for all $i \in [N]$ and all $\mathbf{x} \in \mathcal{F}$.*

3. A Universal Formula for an Arbitrary Number of Affine Constraints

In this section we introduce a smooth universal formula for a controller satisfying an arbitrary number of affine inequalities. Let $A_1, \ldots, A_N \in \mathbb{R}$ and $\mathbf{B}_1, \ldots, \mathbf{B}_N \in \mathbb{R}^m$. Let $\mathbf{p} = (A_1, \ldots, A_N, \mathbf{B}_1, \ldots, \mathbf{B}_N)$ and define $\mathcal{K}_\mathbf{p} := \{\mathbf{k} \in \mathbb{R}^m : A_i + \mathbf{B}_i^\top \mathbf{k} < 0, \ \forall i \in [N]\}$. Note that $\mathcal{K}_\mathbf{p}$ is a convex polytope. Now, define the function $J_\mathbf{p} : \mathcal{K}_\mathbf{p} \to \mathbb{R}$ as

$$J_\mathbf{p}(\mathbf{k}) = -\sum_{i=1}^{N} \frac{\|\mathbf{B}_i\|^2 + \|\mathbf{k}\|^2}{2(A_i + \mathbf{B}_i^\top \mathbf{k})}. \quad (4)$$

Our first result shows that $J_\mathbf{p}$ is strictly convex.

Proposition 3.1. (Strict convexity): *Let $\mathbf{p} \in \mathbb{R}^{N+mN}$. The function $J_\mathbf{p}$ is strictly convex in the convex domain $\mathcal{K}_\mathbf{p}$.*

Proof. Let $J_{i,\mathbf{p}} : \mathcal{P} \to \mathbb{R}$ be defined as

$$J_{i,\mathbf{p}}(\mathbf{k}) = -\frac{\|\mathbf{B}_i\|^2 + \|\mathbf{k}\|^2}{2(A_i + \mathbf{B}_i^\top \mathbf{k})}.$$

Note that the Hessian of $J_{i,\mathbf{p}}$ is given by

$$\nabla^2 J_{i,\mathbf{p}}(\mathbf{k}) = \frac{-\Gamma}{(A_i + \mathbf{B}_i^\top \mathbf{k})^3},$$

where $\Gamma = (A_i + \mathbf{B}_i^\top \mathbf{k})^2 \mathbf{I}_n - (\mathbf{k}\mathbf{B}_i^\top + \mathbf{B}_i \mathbf{k}^\top)(A_i + \mathbf{B}_i^\top \mathbf{k}) + (\|\mathbf{B}_i\|^2 + \|\mathbf{k}\|^2)\mathbf{B}_i \mathbf{B}_i^\top$. Let us show that Γ is positive definite, i.e., $\mathbf{x}^\top \Gamma \mathbf{x} > 0$ whenever $\mathbf{x} \in \mathbb{R}^m \backslash \{\mathbf{0}_m\}$. This is true iff

$$\begin{aligned} &(A_i + \mathbf{B}_i^\top \mathbf{k})^2 \|\mathbf{x}\|^2 - 2(A_i + \mathbf{B}_i^\top \mathbf{k})(\mathbf{B}_i^\top \mathbf{k})(\mathbf{k}^\top \mathbf{x}) + \\ &(\|\mathbf{B}\|_i^2 + \|\mathbf{k}\|^2)(\mathbf{B}_i^\top \mathbf{x})^2 > 0, \end{aligned}$$

for all $\mathbf{x} \in \mathbb{R}^m \backslash \{\mathbf{0}_m\}$. Let $z = \mathbf{B}_i^\top \mathbf{x}$. We want to show that the scalar quadratic function in z given by $(A_i + \mathbf{B}_i^\top \mathbf{k})^2 \|\mathbf{x}\|^2 - 2(A_i + \mathbf{B}_i^\top \mathbf{k})(\mathbf{k}^\top \mathbf{x})z + (1 + \|\mathbf{k}\|^2)z^2$ is strictly positive for all $\mathbf{x} \in \mathbb{R}^m \backslash \{\mathbf{0}_m\}$. Note that the discriminant of the quadratic is

$$\Delta = 4(A_i + \mathbf{B}_i^\top \mathbf{k})^2 \Big((\mathbf{k}^\top \mathbf{x})^2 - \|\mathbf{k}\|^2 \|\mathbf{x}\|^2 - \|\mathbf{x}\|^2 \|\mathbf{B}_i\|^2 \Big)$$

By the Cauchy-Schwartz inequality, $(\mathbf{k}^\top \mathbf{x})^2 \leq \|\mathbf{k}\|^2 \|\mathbf{x}\|^2$. Therefore, $(\mathbf{k}^\top \mathbf{x})^2 - \|\mathbf{k}\|^2 \|\mathbf{x}\|^2 - \|\mathbf{k}\|^2 \|\mathbf{B}_i\|^2 < 0$ unless $\mathbf{B}_i = \mathbf{0}_m$ (in which case $(A_i + \mathbf{B}_i^\top \mathbf{k})^2 \|\mathbf{x}\|^2 - 2(A_i + \mathbf{B}_i^\top \mathbf{k})(\mathbf{k}^\top \mathbf{x})z + (1 + \|\mathbf{k}\|^2)z^2 > 0$ for $\mathbf{x} \neq \mathbf{0}_n$) or $\mathbf{x} = \mathbf{0}_m$. This implies that $\Delta < 0$ unless $\mathbf{x} = \mathbf{0}_m$ and therefore the quadratic $(A_i + \mathbf{B}_i^\top \mathbf{k})^2 \|\mathbf{x}\|^2 - 2(A_i + \mathbf{B}_i^\top \mathbf{k})(\mathbf{k}^\top \mathbf{x})z + (1 + \|\mathbf{k}\|^2)z^2$

is strictly positive for all $\mathbf{x} \in \mathbb{R}^m \backslash \{\mathbf{0}_m\}$. Hence, Γ is positive definite, which means that $\nabla^2 J_{i,\mathbf{p}}(\mathbf{k})$ is positive definite for all $\mathbf{k} \in \mathcal{K}$, and $J_{i,\mathbf{p}}$ is strictly convex in $\mathcal{K}$. Since $J_{\mathbf{p}}(\mathbf{k}) = \sum_{i=1}^{N} J_{i,\mathbf{p}}(\mathbf{k})$, $J_{\mathbf{p}}$ is the sum of strictly convex functions and is therefore also strictly convex in $\mathcal{K}_{\mathbf{p}}$. □

Since $J_{\mathbf{p}}$ is strictly convex in $\mathcal{K}_{\mathbf{p}}$ by Proposition 3.1, and $J_{\mathbf{p}}$ goes to infinity as it approaches the boundary of $\mathcal{K}_{\mathbf{p}}$, it follows that $J_{\mathbf{p}}$ has a unique minimizer in $\mathcal{K}_{\mathbf{p}}$. Let

$$\mathcal{P} = \{\mathbf{p} \in \mathbb{R}^{N+mN} : \mathcal{K}_{\mathbf{p}} \neq \emptyset\}.$$

Then, we let $k^* : \mathcal{P} \to \mathbb{R}^m$ be the function mapping every tuple $\mathbf{p} \in \mathcal{P}$ to the unique minimizer of $J_{\mathbf{p}}$ in $\mathcal{K}_{\mathbf{p}}$. The following result shows that the mapping k^* is smooth.

Lemma 3.2. (Smoothness): *The function k^* is $\mathcal{C}^\infty(\mathcal{P})$.*

Proof. Since $\mathbf{p} \in \mathcal{P}$, $\mathcal{K}_{\mathbf{p}} \neq \emptyset$. First note that since $\lim_{\|\mathbf{k}\| \to \infty} J_{\mathbf{p}}(\mathbf{k}) = \infty$, $k^*(\mathbf{p})$ is finite for all $\mathbf{p}$. Since $J_{\mathbf{p}}$ is strictly convex, continuous in $\mathcal{K}_{\mathbf{p}}$ and $k^*(\mathbf{p})$ is its unique minimizer, there exist neighborhoods $\mathcal{N}_1$, $\mathcal{N}_2$ of $k^*(\mathbf{p})$ and a constant $M > 0$ such that if $\mathbf{k} \in \mathcal{N}_1$ then $J_{\mathbf{p}}(\mathbf{k}) < M$, and if $\mathbf{k} \notin \mathcal{N}_2$ then $J_{\mathbf{p}}(\mathbf{k}) > 2M$. Now, since $J_{\mathbf{p}}$ is continuous with respect to $\mathbf{p}$, there exists a neighborhood $\mathcal{N}_{\mathbf{p}}$ of $\mathbf{p}$ such that for any $\bar{\mathbf{p}} \in \mathcal{N}_{\mathbf{p}}$, we have that if $\mathbf{k} \in \mathcal{N}_1$ then $J_{\bar{\mathbf{p}}}(\mathbf{k}) < \frac{3M}{2}$, and if $\mathbf{k} \notin \mathcal{N}_2$ then $J_{\mathbf{p}}(\mathbf{k}) > \frac{3M}{2}$. Hence, $k^*(\bar{\mathbf{p}}) \in \mathcal{N}_1$ for all $\bar{\mathbf{p}} \in \mathcal{N}_{\mathbf{p}}$. Now, define the function $J : \mathcal{N}_1 \times \mathcal{N}_1 \to \mathbb{R}$ as $J(\mathbf{p}, \mathbf{k}) = J_{\mathbf{p}}(\mathbf{k})$. Note that since k^* is the minimizer of $J_{\mathbf{p}}$, it satisfies the equation $\frac{\partial J}{\partial \mathbf{k}}(\bar{\mathbf{p}}, k^*(\bar{\mathbf{p}})) = \mathbf{0}_m$ for all $\bar{\mathbf{p}} \in \mathcal{N}_1$. Additionally, note that since $J_{\mathbf{p}}$ is strictly convex as shown in Proposition 3.1, $\frac{\partial^2 J}{\partial^2 \mathbf{k}}(\bar{\mathbf{p}}, k^*(\bar{\mathbf{p}}))$ is non-singular. Now, by the Implicit Function Theorem [31, Proposition 1B.5], since J is $\mathcal{C}^\infty$, k^* is also $\mathcal{C}^\infty$ at $\mathbf{p}$. This argument is valid for any $\mathbf{p} \in \mathcal{P}$. □

Therefore, k^* is $\mathcal{C}^\infty$ and satisfies the constraints $A_i + \mathbf{B}_i^\top k^*(\mathbf{p}) < 0$ for all $i \in [N]$ and $\mathbf{p} \in \mathcal{P}$. Define now the controller $u^* : \mathcal{F} \to \mathbb{R}^m$ by

$$u^*(\mathbf{x}) = k^*(a_1(\mathbf{x}), \ldots, a_N(\mathbf{x}), b_1(\mathbf{x}), \ldots, b_N(\mathbf{x})). \tag{5}$$

Note that $\mathbf{x} \in \mathcal{F}$ implies that $(a_1(\mathbf{x}), \ldots, a_N(\mathbf{x}), b_1(\mathbf{x}), \ldots, b_N(\mathbf{x})) \in \mathcal{P}$. Therefore, u^* is $\mathcal{C}^l(\mathcal{F})$ and satisfies the constraints $a_i(\mathbf{x}) + b_i(\mathbf{x})^\top u^*(\mathbf{x})$ for all $i \in [N]$ and $\mathbf{x} \in \mathcal{F}$, solving Problem 1.

Remark 3.3. (Connection with CLF universal formula): *A noteworthy property of the controller u^* is that when $N = 1$, $m = 1$, and the constraint corresponds to a CLF constraint, i.e., $a_1(\mathbf{x}) = \nabla V(\mathbf{x})^\top f(\mathbf{x}) + W(\mathbf{x})$, $b_1(\mathbf{x}) = \nabla V(\mathbf{x})^\top g(\mathbf{x})$ for some CLF V and positive definite function W, then k^* equals the well-known universal formula [4] for stabilization. Indeed, the minimizer of $J_{\mathbf{p}}$ can be found by solving the nonlinear equation $\nabla J_{\mathbf{p}}(k) = -\frac{k}{A_1 + B_1 k} + \frac{B_1^2 + k^2}{2(A_1 + B_1 k)^2} B_1 = 0$. By solving the resulting quadratic equation, one obtains the CLF universal formula:*

$$k^*(A_1, B_1) = \begin{cases} -\frac{A_1 + \sqrt{A_1^2 + B_1^4}}{B_1}, & \text{if } B_1 \neq 0, \\ 0, & \text{otherwise.} \end{cases}$$ •

Remark 3.4. (Dynamical controller): *Instead of computing the minimizer u^* at every state, one can choose to run the following dynamical controller:*

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}, \tag{6a}$$

$$\dot{\mathbf{u}} = -\tau \frac{\partial J}{\partial \mathbf{u}}(\mathbf{x}, \mathbf{u}), \tag{6b}$$

where $J(\mathbf{x}, \mathbf{u}) = J_{\mathbf{x}}(\mathbf{u})$ and $\tau > 0$ is a design parameter. Then, by using singular perturbation theory [32, Chapter 11], one can show that for sufficiently large τ, the evolution of the state variable $\mathbf{x}$ according to (6) can be made arbitrarily close to the evolution of the state variable for $\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})u^*(\mathbf{x})$. •

Remark 3.5. (Alternative universal formula): *Given a vector of positive weights $\mathbf{w} \in \mathbb{R}^N$ (i.e., with components $w_i > 0$ for all $i \in [N]$), one can instead consider a variation of the function $J_{\mathbf{p}}$ where each of the summands is assigned different weights:*

$$J_{\mathbf{p}}^{\mathbf{w}}(\mathbf{k}) = -\sum_{i=1}^{N} w_i \frac{\|\mathbf{B}_i\|^2 + \|\mathbf{k}\|^2}{2(A_i + \mathbf{B}_i^\top \mathbf{k})}.$$

By following an argument analogous to the one in the proof of Proposition 3.1, it follows that $J_{\mathbf{p}}^{\mathbf{w}}$ is strictly convex in $\mathcal{K}_{\mathbf{p}}$ and therefore it has a unique minimizer in $\mathcal{K}_{\mathbf{p}}$, which defines an alternative smooth function satisfying all the constraints. •

The following result shows that for safe stabilization problems, even though u^* is not defined at the origin, trajectories in its neighborhood converge to it.

Proposition 3.6. (Convergence to the origin): *Let $V : \mathbb{R}^n \to \mathbb{R}$ be a CLF, $h : \mathbb{R}^n \to \mathbb{R}$ be a CBF of a safe set $\mathcal{C} \subset \mathbb{R}^n$. Let $N = 2$ and define $a_1(\mathbf{x}) = \nabla V(\mathbf{x})^\top f(\mathbf{x}) + W(\mathbf{x})$, $b_1(\mathbf{x}) = g(\mathbf{x})^\top \nabla V(\mathbf{x})$, $a_2(\mathbf{x}) = -\nabla h(\mathbf{x})^\top f(\mathbf{x}) - \alpha h(\mathbf{x})$, $b_2(\mathbf{x}) = -g(\mathbf{x})^\top \nabla h(\mathbf{x})$. Suppose that there exists $\bar{k} > 0$ such that*

$$\mathcal{O}_{\bar{k}} = \{\mathbf{x} \in \mathbb{R}^n : 0 < V(\mathbf{x}) < \bar{k}\} \subset \mathcal{F}.$$

Let $\mathbf{x}_0 \in \mathcal{O}_{\bar{k}}$ with $V(\mathbf{x}_0) = k \leq \bar{k}$ and consider the maximal solution of $\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})u^*(\mathbf{x})$ in $\mathcal{F}$ with initial condition at $\mathbf{x}_0$. We let $\mathbf{x}(\cdot; \mathbf{x}_0)$ be such solution and $[0, T)$ its interval of existence (with either $T = \infty$ or $T < \infty$). Then, we have

$$\lim_{t \to T} \mathbf{x}(t; \mathbf{x}_0) = \mathbf{0}_n.$$

Proof. Suppose first that there exists a compact subset $C \subset \mathcal{F}$ such that $x(t) \in C$ for all $t \in [0, T)$. Then, by [1, Proposition C.3.6], it follows that $T = \infty$. Since $\mathbf{x}(\cdot; \mathbf{x}_0)$ is precompact, the omega limit set of $\mathbf{x}_0$, denoted as $\omega^+(\mathbf{x}_0)$ is non-empty. Since V is a LaSalle function in $\mathcal{O}_{\bar{k}}$, for all $\boldsymbol{\xi} \in \omega^+(\mathbf{x}_0)$ we have $\dot{V}(\boldsymbol{\xi}) = 0$. However, since $\xi \in \omega^+(\mathbf{x}_0) \in \mathcal{O}_{\bar{k}}$, we have $\dot{V}(\boldsymbol{\xi}) < 0$, reaching a contradiction. Therefore, there exists no such set C. Let $\epsilon > 0$ and consider the set $C_{\epsilon,k} = \{\mathbf{x} \in \mathbb{R}^n : \epsilon \leq V(\mathbf{x}) \leq k\}$. Since $C_{\epsilon,k}$ is a compact set contained in $\mathcal{F}$, there exists $t_1 \in (0, T)$ such that $\mathbf{x}(t_1; \mathbf{x}_0) \notin C_{\epsilon,k}$. Let $t_0 = \max_s\{s \leq t_1 : \mathbf{x}(s; \mathbf{x}_0) \in C_{\epsilon,k}\ \forall s \leq t\}$. First note that $V(\mathbf{x}(t_0; \mathbf{x}_0)) = k$ is impossible. Indeed, for such a t_0 we have $\dot{V}(\mathbf{x}(t_0; \mathbf{x}_0)) < 0$, so $V(\mathbf{x}(t; \mathbf{x}_0)) > V(\mathbf{x}(t_1; \mathbf{x}_0))$ for all $t \in [t_0 - \bar{\epsilon}, t_0]$ for some small enough $\bar{\epsilon} > 0$ (by continuity of $\dot{V}$), and therefore $V(\mathbf{x}(t; \mathbf{x}_0)) > k$ for $t \in [t_0 - \bar{\epsilon}, t_0]$, which contradicts the definition of t_0. Since $V(\mathbf{x}(t_0; \mathbf{x}_0)) = k$ is impossible, we necessarily have $V(\mathbf{x}(t_0; \mathbf{x}_0)) = \epsilon$. Since $\dot{V}(\mathbf{x}(t; \mathbf{x}_0)) < 0$ for all $t \in [0, T)$, we conclude that $V(\mathbf{x}(t; \mathbf{x}_0)) < \epsilon$ for all $t \in (t_0, T)$. Since ϵ is arbitrary, we necessarily have $\lim_{t \to T} \mathbf{x}(t; \mathbf{x}_0) = \mathbf{0}_n$. □

Moving beyond the case of one constraint discussed in Remark 3.3, the controller u^* is not available in closed form if multiple constraints are present, and therefore its implementation requires solving a minimization problem at every $\mathbf{x}$. Although $J_{\mathbf{p}}$ is strictly convex and therefore $u^*(\mathbf{x})$ can be found by using off-the-shelf convex solvers, this computation needs to be done at every state in continuous time, which is not possible in practice. Instead, one often implements a sample-and-hold version of the controller at a sufficiently high frequency. However, computing the minimizers at such high frequencies becomes computationally burdensome and puts into question the validity of the convergence guarantees. This motivates our ensuing discussion.

4. Neural Network Approximation of the Universal Formula

In this section, we present an approach to avoid the computational burden of solving a minimization problem at every state to compute the controller u^* defined in (5). Our approach is based on computing an approximation of the controller k^* using a NN. This NN needs to be trained with input-output pairs of the form $(\mathbf{p}, k^*(\mathbf{p}))$, with $\mathbf{p} \in \mathcal{P} \subset \mathbb{R}^{N+mN}$. However, this presents a challenge because the set $\mathcal{P}$ is potentially unbounded, which means that the set of possible inputs $\mathbf{p}$ to the NN is unbounded. The following result resolves this issue by showing that the values of $J_{\mathbf{p}}$ can be inferred by only looking at a bounded subset of $\mathbb{R}^{N+mN}$.

Lemma 4.1. (Scaling property of cost function): *Let $\tilde{\mathbf{p}} = (\tilde{A}_1, \ldots, \tilde{A}_N, \tilde{\mathbf{B}}_1, \ldots, \tilde{\mathbf{B}}_N) \in \mathcal{P}$, $\mathbf{q} = (\tilde{A}_1, \ldots, \tilde{A}_N, \tilde{\mathbf{B}}_1, \ldots, \tilde{\mathbf{B}}_N, r) \in \mathcal{P} \times [0, 1]$ and define the function $\tilde{J}_{\mathbf{q}} : \mathcal{K}_{\tilde{\mathbf{p}}} \to \mathbb{R}$ as*

$$\tilde{J}_{\mathbf{q}}(\mathbf{k}) = -\sum_{i=1}^{N} \frac{\|\tilde{\mathbf{B}}_i\|^2 + r\|\mathbf{k}\|^2}{2(\tilde{A}_i + \tilde{\mathbf{B}}_i^\top \mathbf{k})}.$$

Then, $\tilde{J}_{\mathbf{q}}$ is strictly convex in $\mathcal{K}_{\tilde{\mathbf{p}}}$. Furthermore, given $\mathbf{p} = (A_1, \ldots, A_N, \mathbf{B}_1, \ldots, \mathbf{B}_N)$, let

$$\mathbf{q}(\mathbf{p}) = \Big(\frac{\mathbf{p}}{M}, \frac{1}{M^2}\Big) = \Big(\frac{A_1}{M}, \ldots, \frac{A_N}{M}, \frac{\mathbf{B}_1}{M}, \ldots, \frac{\mathbf{B}_N}{M}, \frac{1}{M^2}\Big),$$

where $M = \max\{|A_1|, \ldots, |A_N|, |\|\mathbf{B}_1\|, \ldots, \|\mathbf{B}_N\|, 1\}$. Then $J_{\mathbf{p}}(\mathbf{k}) = \tilde{J}_{\mathbf{q}(\mathbf{p})}(\mathbf{k})$ for all $\mathbf{k} \in \mathcal{K}_{\mathbf{p}}$. As a consequence, if $\tilde{k}^* : \mathcal{P} \times [0, 1] \to \mathbb{R}^m$ denotes the function that maps each $\mathbf{q} \in \mathcal{P} \times [0, 1]$ to the minimizer of $\tilde{J}_{\mathbf{q}}$, we have $k^*(\mathbf{p}) = \tilde{k}^*(\mathbf{q}(\mathbf{p}))$ for all $\mathbf{p} \in \mathcal{K}_{\mathbf{p}}$.

Proof. The proof that $\tilde{J}_{\mathbf{q}}$ is strictly convex follows an argument analogous to that of Proposition 3.1. Given $\mathbf{p} = (A_1, \ldots, A_N, \mathbf{B}_1, \ldots, \mathbf{B}_N)$, note that

$$\begin{aligned}\tilde{J}_{\mathbf{q}(\mathbf{p})}(\mathbf{k}) &= -M \sum_{i=1}^{N} \frac{\frac{\|\mathbf{B}_i\|^2}{M^2} + \frac{\|\mathbf{k}\|^2}{M^2}}{2(A_i + \mathbf{B}_i^\top \mathbf{k})} \\ &= -\frac{1}{M} \sum_{i=1}^{N} \frac{\|\mathbf{B}_i\|^2 + \|\mathbf{k}\|^2}{2(A_i + \mathbf{B}_i^\top \mathbf{k})} = J_{\mathbf{p}}(\mathbf{k}).\end{aligned}$$

Therefore, the minimizers of $\tilde{J}_{\tilde{\mathbf{p}}}$ and $J_{\mathbf{p}}$ are the same. □

Lemma 4.1 shows that by approximating $\tilde{k}^*$ with values of $\mathbf{q}$ in $\mathcal{T} = \big([-1,1]^N \times \mathcal{B}_1(\mathbf{0}_m)^N\big) \cap \mathcal{P} \times [0,1]$, we can recover any value of k^*. Therefore, our approach consists in approximating the minimizer of $\tilde{J}_{\mathbf{q}}$ with a NN for $\mathbf{q}$ in the set $\mathcal{T}$. Since $\mathcal{P}$ is an open set, $\mathcal{T}$ is not compact. Given that the universal approximation theorem of neural networks [33] is only valid in compact sets, it does not apply to $\mathcal{T}$. However, for any compact subset of $\mathcal{T}$ (which we can take to be arbitrarily close to $\mathcal{T}$), the universal approximation theorem ensures that a NN with a sufficiently large number of parameters can approximate k^* arbitrarily well. Note also that by choosing the activation functions of the NN to be smooth, the resulting approximation of k^* is $\mathcal{C}^\infty(\mathcal{P})$, and therefore the approximation of the controller u^* is $\mathcal{C}^l(\mathcal{P})$.

Remark 4.2. (Applicability for multiple controllers): *Note that once we have obtained an approximation of k^*, we can use it in many different instantiations of the functions $a_1, \dots, a_N, b_1, \dots, b_N$, i.e., in a variety of different control problems. The NN only depends on the number of constraints N and the dimension of the input m, but remarkably, does not depend on n, the dimension of the state. In fact, given a NN trained with a given input dimension m and number of constraints N, all control problems with input dimension at most m and at most N constraints can be solved with the same NN (in case the input dimension is strictly less than m or the number of constraints is strictly less than N one can simply assign the coefficients of the additional inputs or constraints so that they are trivially satisfied). In particular, this implies the remarkable fact that all safe stabilization problems for systems with a given input dimension can be solved with the same NN approximation.* •

Remark 4.3. (Robustness): *If one of the inequalities corresponds to a CLF inequality* (2)*, and the corresponding CLF is an input-to-state stable (ISS) Lyapunov function for the system* (1) *(cf. [34, Section 3.3]), then if we let $\hat{k}$ be an approximated version of the controller k^* (like the one obtained by approximating it with a NN), and the bound $\big\|\hat{k}(\mathbf{x}) - k^*(\mathbf{x})\big\| \leq \sigma$ holds for all $\mathbf{x} \in \mathcal{X}$, with $\mathcal{X}$ some compact set, we have*

$$\begin{aligned}&\nabla V(\mathbf{x})^\top (f(\mathbf{x}) + g(\mathbf{x})\hat{k}(\mathbf{x}))\\ &= \nabla V(\mathbf{x})^\top (f(\mathbf{x}) + g(\mathbf{x})k^*(\mathbf{x})) + \gamma(\sigma),\end{aligned}$$

for some class $\mathcal{K}_\infty$ function γ. A similar result holds if the inequality is a CBF inequality (3) *and the CBF defining it is an input-to-state safe (ISSf) barrier function for the system* (1) *[35, Theorem 1]. Therefore, CLF and CBF inequalities are robust to approximation errors induced by the NN and therefore the NN-based controller, when applied to safety and stability tasks, makes the closed-loop system close to safe and stable, respectively.* •

Remark 4.4. (Hard-constrained NN): *We should also point out that the recently introduced HardNet method [36] shows that, by including a projection step at the output layer of the NN, the NN predictions can be guaranteed to satisfy a set of affine constraints. Although the universal approximation guarantees are retained, the projection operation in the last layer makes the controller obtained from it locally Lipschitz but not $\mathcal{C}^l(\mathcal{F})$, for $l \geq 1$.* •

5. Simulations

In this section we illustrate our approach in different simulation examples. We train a NN to approximate k^* as detailed in Section 4. We focus on the case of $m = 2$ and $N = 2$. We use a feedforward NN with 4 layers (with input dimension 7 and output dimension 2), each with 64 neurons, and we use the Sigmoid Linear Unit (SiLU) activation function $\phi : \mathbb{R} \to \mathbb{R}$ given by $\phi(s) = \frac{s}{1+\exp\{-s\}}$ [37], which is a smooth approximation of the ReLU activation function. The NN is trained with the Adam optimizer with learning rate 3×10^{-4} for 2000 epochs using a dataset with 25113 data points. Each data point $\mathbf{q}$ consists of a uniformly randomly sampled point in $([-1,1] \times \mathcal{B}_1(\mathbf{0}_2) \times [-1,1] \times \mathcal{B}_1(\mathbf{0}_2) \times [0,1]) \cap \mathcal{P} \subset \mathbb{R}^7$ (to do so, we first uniformly sample a point from $[-1,1] \times \mathcal{B}_1(\mathbf{0}_2) \times [-1,1] \times \mathcal{B}_1(\mathbf{0}_2) \times [0,1]$ and then check whether the corresponding 2 inequalities are feasible with the QP solver of the `cvxpy` library in Python [38]), along with the corresponding minimizer of the function $\tilde{J}_{\mathbf{q}}$. We compute such minimizer up to a tolerance of 10^{-6} by numerically integrating the gradient flow $\dot{\mathbf{k}} = -\nabla \tilde{J}_{\mathbf{q}}(\mathbf{k})$ using Python's `solve_ivp` function in the `SciPy` library [39]. We also train the same NN architecture with the HardNet method from [36], which includes a projection step at the output layer and guarantees that the NN predictions satisfy the two affine constraints (cf. Remark 4.4). All simulations were run in an Ubuntu PC with Intel Core i9-13900K 3 GHz 24-Core Processor.

Example 5.1. (Safe stabilization of single-integrator system): *Consider a single integrator on* $\mathbb{R}^2$, *i.e.,* $\dot{x} = u_x, \dot{y} = u_y$. *Suppose that our goal is to design a controller that stabilizes the system to the origin and stays in the safe set* $\mathcal{C} = \{(x, y) \in \mathbb{R}^2 : h_1(x, y) = x^2 + (y - 2.5)^2 - 1 \geq 0,\ h_2(x, y) = (x + 2)^2 + (y + 2)^2 - 1 \geq 0,\ h_3(x, y) = (x - 2)^2 + (y + 2)^2 - 1 \geq 0\}$. *Note that* $V(x, y) = \frac{1}{2}(x^2 + y^2)$ *is a CLF and* $h(x, y) := h_1(x, y)h_2(x, y)h_3(x, y)$ *is a CBF of* $\mathcal{C}$. *Take the positive definite function* W *in* (2) *to be* $W(x, y) = 0.1(x^2+y^2)$ *and the extended class* $\mathcal{K}_\infty$ *function* α *in* (3) *to be* $\alpha(s) = s$. *Now, using the inequalities* (2), (3), *we define* $a_1(x, y) = 0.1(x^2 + y^2)$, $b_1(x, y) = (x, y)$, $a_2(x, y) = -h(x, y)$, *and* $b_2(x, y) = -\nabla h(x, y)$. *By using [13, Lemma 5.2], we can show that the inequalities* $a_1(x, y) + b_1(x, y)^\top \mathbf{u} < 0$, $a_2(x, y) + b_2(x, y)^\top \mathbf{u} < 0$ *are simultaneously feasible at all points in* $\mathcal{C}$ *not satisfying* $\frac{\partial h}{\partial x}(x, y)y = \frac{\partial h}{\partial y}(x, y)x$. *More explicitely, the inequalities will be satisfied if there is a pair* (u_x, u_y) *so that the entries of*

$$\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} + \begin{pmatrix} x & y \\ -h_x & -h_y \end{pmatrix} \begin{pmatrix} u_x \\ u_y \end{pmatrix}$$

are both negative, which can achieved unless the determinant of the matrix is zero. Many points in the set where the determinant is zero are also in $\mathcal{F}$, *but this is sufficient to show that* $\mathcal{F}$ *consists of all points in* $\mathcal{C}$ *except for a set of measure zero. Figure 1 compares the trajectories obtained by directly applying the NN-based controller in closed-loop and using it to warmstart an optimization scheme to compute* u^* *online. We observe that the NN-based controller induces trajectories that are safe and converge to a small neighborhood around the origin, for the chosen initial conditions. Instead, the trajectories that use a controller obtained by numerically computing* u^* *online with the NN controller as a warmstart asymptotically converge to the origin (instead of a neighborhood of it) and are also safe. The same is true for the controllers obtained from the HardNet method, the CLF-CBF QP, or the computation of* u^* *online with the CLF-CBF QP as a warmstart.*

Table 1, top row reports the execution times of the various considered controllers. The NN-based controller is the fastest, but does not possess safety and stability guarantees. The HardNet-based controller is one order of magnitude slower but has safety and stability guarantees by construction, although no optimality guarantees. The CLF-CBF QP controller and the controllers obtained by computing u^* *online are two orders of magnitude slower than the NN-based controller, but are safe, stable, and optimal by construction up to numerical errors. It is also worth noting that warmstarting the numerical solver to optimize* $\tilde{J}_{\mathbf{q}}$ *with the NN prediction also improves its computational time compared to warmstarting it with the CLF-CBF QP.* •

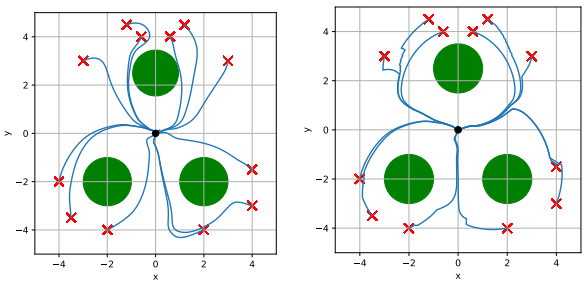

Figure 1: (left) Trajectories of the closed-loop system obtained from the neural network based controller for Example 5.1. (right) Trajectories of the closed-loop system obtained from numerically finding the controller u^* online, warmstarting the solver with the NN-based controller for Example 5.1. Initial conditions are denoted by red crosses, the origin is the black dot, and the green region denotes the unsafe set.

Example 5.2. (Safe stabilization of unicycle with drift): *Consider the following control system, modeling a unicycle with drift navigating on the plane:*

$$\dot{x} = v\cos(\theta), \quad \dot{y} = -y + v\sin(\theta), \quad \dot{\theta} = \omega,$$

with $\mathbf{u} = (v, \omega)$ *being the control input. Suppose that our goal is to design a controller that stabilizes the system to the origin and stays in the safe set* $\mathcal{C} = \{(x, y, \theta) \in \mathbb{R}^3 : h(x, y, \theta) = -y + (2x + 1)^2 + 1 \geq 0\}$ *(this example is taken from [14, Section VII]. Take the positive definite function* $W(x, y) = 0.1(x^2 + y^2 + \theta^2)$ *and* $\alpha(s) = s$. *Now, using the inequalities* (2), (3), *we can define* $b_1(x, y, \theta) = (x\cos(\theta) + y\sin(\theta), \theta)$ $a_1(x, y, \theta) = -y^2 + 0.1(x^2 + y^2 + \theta^2)$, $b_2(x, y, \theta) = (-4(2x + 1)\cos(\theta) + \sin(\theta), 0)$, $a_2(x, y, \theta) = -y - 2h(x, y)$. *As shown in [14, Section VII], the inequalities* $a_1(x, y, \theta)+b_1(x, y, \theta)^\top \mathbf{u} < 0$, $a_2(x, y, \theta)+b_2(x, y, \theta)^\top \mathbf{u} < 0$ *are simultaneously feasible in all of* $\mathcal{C}\backslash\{\mathbf{0}_2\}$, *so* $\mathcal{C}\backslash\{\mathbf{0}_2\} \subset \mathcal{F}$. *Since the dimension of the input is* 2 *and we have* 2 *constraints, we can use the same NN as in Example 5.1, even though the state dimension is different. Figure 2 compares the trajectories obtained by directly applying the NN-based controller in closed-loop and using it to warmstart an optimization scheme to compute* u^* *online. In this case, both controllers induce safe trajectories that asymptotically converge to the origin. Table 1,*

	NN	HardNet	CLF-CBF QP controller	u^* with CLF-CBF QP warmstart	u^* with NN warmstart
Time (ms) Ex. 5.1	0.053 ± 0.01	0.2 ± 0.08	2.2 ± 1.5	6.9 ± 2.0	4.7 ± 1.2
Time (ms) Ex. 5.2	0.05 ± 0.01	0.2 ± 0.04	2.2 ± 1.6	2.9 ± 1.9	1.3 ± 0.9

Table 1: Average execution times ($\pm$ standard deviation) in milliseconds over the trajectories in Figure 1 (top row) and Figure 2 (bottom row) for different controller implementations. The first column refers to the controller obtained directly as the prediction of the NN. The second column refers to the controller obtained as the prediction of the NN when trained with the HardNet method from [36]. The third column refers to the controller obtained by solving the CLF-CBF QP from [6] (using the `cvxpy` library in Python [38]). The fourth (resp. fifth) column refers to the controller obtained by finding the minimum of the function $\tilde{J}_{\mathbf{q}}$ using a numerical solver (the `solve_ivp` function in Python's `SciPy` library) and warmstarting the solver with the CLF-CBF QP (resp. the NN prediction).

bottom row reports the execution times of various controllers for this example as well. •

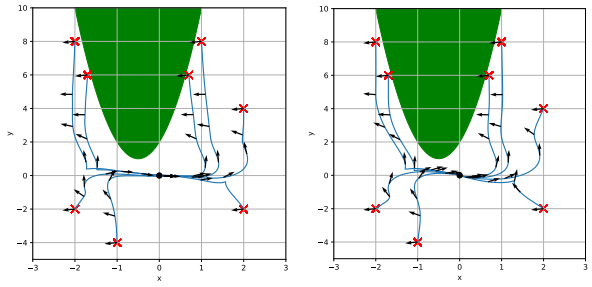

Figure 2: (left) Projection in the (x, y) plane of trajectories of the closed-loop system obtained from the neural network based controller for Example 5.2. (right) Projection in the (x, y) plane of trajectories of the closed-loop system obtained from numerically finding the controller u^* online and warmstarting the solver with the NN-based controller for Example 5.2. Initial conditions are denoted by red crosses (and all have an initial orientation $\theta_0 = \pi + 0.1$), the origin is the black dot, and the green region denotes the unsafe set. Black arrows indicate the orientation of the unicycle (i.e., the θ variable) at that point of the trajectory. Observe that the velocity v could be negative, so that at points near the right of the target, the vehicle is "backing up".

6. Conclusions, Limitations, and Future Work

We have studied the problem of designing a controller that satisfies an arbitrary number of affine inequalities at every point in the state space, which arise when enforcing safety, stability, and input constraints. We have provided a novel universal formula for controllers satisfying such affine inequalities. The control input is given at every state as the minimizer of a strictly convex function. To avoid the computation of such minimizer in real time, we have introduced a method based on NN to approximate it. Remarkably, this NN is universal in the sense that it can be used for any control task with input dimension and number of constraints less than some fixed value, and can be trained with data from just a small subset of the state space. We have shown the performance of the controller and its NN approximation in various simulation examples. A limitation of the presented approach is that it has not been tested in systems with high-dimensional states and inputs, or a high number of constraints. Additionally, the presented controller is only defined in the region where the constraints are simultaneously feasible, and is limited to affine constraints. Future work will focus on addressing these shortcomings.

References

[1] E. D. Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems*, volume 6 of *TAM*. Springer, 2 edition, 1998. ISBN 0387984895.

[2] E. D. Sontag. A Lyapunov-like characterization of asymptotic controllability. *SIAM Journal on Control and Optimization*, 21:462–471, 1983.

[3] R. A. Freeman and P. V. Kototovic. *Robust Nonlinear Control Design: State-space and Lyapunov Techniques*. Birkhauser Boston Inc., Cambridge, MA, USA, 1996.

[4] E. D. Sontag. A universal construction of Artstein's theorem on nonlinear stabilization. *Systems & Control Letters*, 13(2):117–123, 1989.

[5] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. Control barrier functions: theory and applications. In *European Control Conference*, pages 3420–3431, Naples, Italy, 2019.

[6] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2017.

[7] P. Wieland and F. Allgöwer. Constructive safety using control barrier functions. *IFAC Proceedings Volumes*, 40(12):462–467, 2007.

[8] M. Cohen, P. Ong, G. Bahati, and A. D. Ames. Characterizing smooth safety filters via the implicit function theorem. *IEEE Control Systems Letters*, 7:3890–3895, 2023.

[9] M. Li, Z. Sun, P. J. W. Koelewijn, and S. Weiland. A tunable universal formula for safety-critical control. *arXiv preprint arXiv:2403.06285*, 2024.

[10] K. Garg and D. Panagou. Robust control barrier and control Lyapunov functions with fixed-time convergence guarantees. In *American Control Conference*, pages 2292–2297, New Orleans, LA, July 2021.

[11] M. Li and Z. Sun. A graphical interpretation and universal formula for safe stabilization. In *American Control Conference*, pages 3012–3017, San Diego, California, 2023.

[12] M. Z. Romdlony and B. Jayawardhana. Stabilization with guaranteed safety using control Lyapunov-barrier function. *Automatica*, 66:39–47, 2016.

[13] P. Mestres and J. Cortés. Optimization-based safe stabilizing feedback with guaranteed region of attraction. *IEEE Control Systems Letters*, 7:367–372, 2023.

[14] P. Ong and J. Cortés. Universal formula for smooth safe stabilization. In *IEEE Conf. on Decision and Control*, pages 2373–2378, Nice, France, December 2019.

[15] T. G. Molnar and A. D. Ames. Composing control barrier functions for complex safety specifications. *IEEE Control Systems Letters*, 7:3615–3620, 2023.

[16] X. Tan and D. V. Dimarogonas. On the undesired equilibria induced by control barrier function based quadratic programs. *Automatica*, 159:111359, 2024.

[17] A. Singletary, Y. Chen, and A. D. Ames. Control barrier functions for sampled-data systems with input delays. pages 804–809, December 2020.

[18] A. J. Taylor, P. Ong, J. Cortés, and A. Ames. Safety-critical event triggered control via input-to-state safe barrier functions. *IEEE Control Systems Letters*, 5(3): 749–754, 2021.

[19] J. Breeden, K. Garg, and D. Panagou. Control barrier functions in sampled-data systems. *IEEE Control Systems Letters*, 6:367–372, 2022.

[20] A. J. Taylor, P. Ong, T. G. Molnar, and A. D. Ames. Safe backstepping with control barrier functions. In *IEEE Conf. on Decision and Control*, pages 5775–5782, Cancún, Mexico, 2022. doi: 10.1109/CDC51059.2022.9992763.

[21] B. J. Morris, M. J. Powell, and A. D. Ames. Continuity and smoothness properties of nonlinear optimization-based feedback controllers. In *IEEE Conf. on Decision and Control*, pages 151–158, Osaka, Japan, Dec 2015.

[22] M. Alyaseen, N. Atanasov, and J. Cortés. Continuity and boundedness of minimum-norm CBF-safe controllers. *IEEE Transactions on Automatic Control*, 70 (6):4148–4154, 2025.

[23] P. Mestres, A. Allibhoy, and J. Cortés. Regularity properties of optimization-based controllers. *European Journal of Control*, 81:101098, 2025.

[24] S. W. Chen, T. Wang, N. Atanasov, V. Kumar, and M. Morari. Large scale model predictive control with neural networks and primal active sets. *Automatica*, 135:109947, 2022.

[25] S. Chen, K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, and G. Pappas. Approximating explicit model predictive control using constrained neural networks. In *American Control Conference*, pages 1520–1527, Milwaukee, Wisconsin, USA, 2018.

[26] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer. Learning an approximate model predictive controller with guarantees. *IEEE Control Systems Letters*, 2(3): 543–548, 2018.

[27] X. Zhang, M. Bujarbaruah, and F. Borrelli. Safe and near-optimal policy learning for model predictive control using primal-dual neural networks. In *American Control Conference*, pages 354–359, Philadelphia, Pennsylvania, USA, 2019.

[28] F. Borrelli, A. Bemporad, and M. Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, Cambridge, UK, 2017.

[29] P. Mestres and J. Cortés. Converse theorems for certificates of safety and stability. *IEEE Transactions on Automatic Control*, 70(12), 2025. To appear. Available at `https://arxiv.org/abs/2406.14823`.

[30] Z. Artstein. Stabilization with relaxed controls. *Nonlinear Analysis*, 7(11):1163–1173, 1983.

[31] A. L. Dontchev and R. T. Rockafellar. *Implicit Functions and Solution Mappings: A View from Variational Analysis; 2nd ed.* Springer, New York, NY, 2014.

[32] H. Khalil. *Nonlinear Systems, 3rd ed.* Prentice Hall, Englewood Cliffs, NJ, 2002.

[33] G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing*, 7(2):279–301, 1989.

[34] E. D. Sontag. Input to state stability: Basic concepts and results. *Nonlinear and Optimal Control Theory*, 1932:163–220, 2008.

[35] S. Kolathaya and A. D. Ames. Input-to-state safety with control barrier functions. *IEEE Control Systems Letters*, 3(1):108–113, 2018.

[36] Y. Min and N. Azizan. Hard-constrained neural networks with universal approximation guarantees. *arXiv preprint arXiv:2410.10807*, 2025.

[37] S. Elfwing, E. Uchibe, and K. Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018.

[38] S. Diamond and S. Boyd. Cvxpy: A python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[39] P. Virtanen, R. Gommers, T. E. Oliphant, et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.