Certified Robust Invariant Polytope Training in Neural Controlled ODEs

Akash Harapanahalli AHARAPAN@GATECH.EDU
Samuel Coogan SAM.COOGAN@GATECH.EDU
School of Electrical and Computer Engineering, Georgia Institute of Technology

Editors: G. Sukhatme, L. Lindemann, S. Tu, A. Wierman, N. Atanasov

Abstract

We propose a framework for training neural network controllers with certified robust forward invariant polytopes. First, we parameterize a family of lifted control systems in a higher dimensional space, where the original neural controlled system evolves on an invariant subspace of each lifted system. We use interval analysis and neural network verifiers to further construct a family of lifted embedding systems, carefully capturing the knowledge of this invariant subspace. If the vector field of any lifted embedding system satisfies a sign constraint at a single point, then a certain convex polytope of the original system is robustly forward invariant. Treating the neural network controller and the lifted system parameters as variables, we propose an algorithm to train controllers with certified forward invariant polytopes in the closed-loop control system. Through two examples, we demonstrate how the simplicity of the sign constraint allows our approach to scale with system dimension to over 50 states, and outperform state-of-the-art Lyapunov-based sampling approaches in runtime.

Keywords: certification, robust training, dynamical systems, control theory, forward invariance

1. Introduction

Learning-enabled components are increasingly used in closed-loop control systems due to their ease of computation and ability to outperform optimization-based feedback control approaches (Chen et al., 2018). In safety-critical control systems, ensuring the reliability of these learning-enabled components is crucial for deployment. Recent work has focused on verifying and training robust neural networks, as summarized in Brix et al. (2023) and Meng et al. (2022). Neural networks in control loops introduce unique challenges such as the compounding of error via feedback, and verifying robustness in the closed-loop setting has also been the subject of recent work (Lopez et al., 2024). However, there are few methods for training safe neural network feedback controllers, despite advancements in certified robust training of isolated neural networks.

Certifying a robust forward invariant set is a well-studied and natural technique for ensuring infinite-time safe behavior of systems. These safe sets can represent a variety of different physical specifications, including operating regions, goal regions, or the complement of unsafe regions (*e.g.*, obstacles). There are many classical techniques for computationally certifying invariance specifications, including Lyapunov-based analysis using sum-of-squares programming (Papachristodoulou and Prajna, 2002; Topcu et al., 2008), barrier-based methods which introduce an online convex optimization problem to solve for each control input (Ames et al., 2019), and set-based approaches which require explicit characterizations of tangent cones (Blanchini, 1999). However, a direct application of these methods generally fails when facing high-dimensional and nonlinear neural network controllers in-the-loop.

Robustness of neural networks in isolation is a well studied field in the machine learning community—for a recent survey, see Meng et al. (2022). A key feature of many of these approaches is their implementation supporting automatic differentiation, to help promote robustness during the training procedure. As opposed to pure input-output robustness of neural networks, the challenge with closed-loop feedback is to capture the interactions of the network and the system. There is a growing community centered specifically around studying the safety of neural networks applied in feedback loops. Approaches based on computing reachable sets of the neural controlled system include POLAR (Huang et al., 2022), JuliaReach (Schilling et al., 2022), NNV (Tran et al., 2020), CORA (Kochdumper et al., 2023), and ReachMM (Jafarpour et al., 2024) for nonlinear systems, and ReachLP (Everett et al., 2021) and Reach-SDP (Hu et al., 2020) for linear systems. We refer to Lopez et al. (2024) for a comprehensive list of benchmarks and tools the community has been studying. However, to our knowledge, none of these approaches for control systems support autodifferentiation to help train neural network controllers with safety guarantees.

There are some papers that study forward invariance for neural networks in dynamical systems. The paper Saoud and Sanfelice (2021) verifies invariant interval sets for control-affine systems with independent inputs for each state variable, Jouret et al. (2023) finds invariant non-convex regions for linear systems with piecewise affine controllers, Yin et al. (2022) finds an ellipsoidal inner-approximation of a region of attraction for the system using Integral Quadratic Constraints (IQCs). In Dai et al. (2021), a Lyapunov-based approach is used to find robust invariant sets of control systems modeled by neural networks. For training robust neural ODEs, LyaNet (Rodriguez et al., 2022) is a Lyapunov-based approach to improve stability and Xiao et al. (2023) uses control barrier functions to filter parameters online to ensure set invariance. For neural ODEs and neural network controlled systems, Huang et al. (2023) sample along the boundary of a Lyapunov function to certify and train neural networks with robust invariance guarantees.

Contributions We first propose a novel technique to address the problem of certifying robust forward invariant polytopes. We introduce the lifted system, which is a lifting of the closed-loop system (2) into a n-dimensional subspace of $\mathbb{R}^m$ using a tall matrix H and a parameterized left inverse H^+. For any bounded and convex polytope, we construct a family of lifted embedding systems parameterized by H^+. A component-wise positivity check on the vector field at a single point, for any instance of H^+, obtains a sufficient condition for robust forward invariance for the original neural network controlled system (2). Compared to previous approaches, our method does not require sampling along the entire boundary of a desired robust invariant set, improving certification runtime and scalability with respect to state dimension. We then propose a novel method for training certified robust forward invariant polytopes by incorporating the positivity condition from the lifted embedding system into the optimization problem. A simple loss function induces the desired positivity in the lifted embedding space, which can subsequently be checked at each training iteration at little cost. Next, we add an unconstrained decision variable η by leveraging the parameterization of possible left inverses H^+, which reduces overconservatism by searching through the entire family of possible lifted dynamics. We implement the proposed algorithm in JAX, which allows us to just-in-time compile and vectorize the embedding system evaluations onto a GPU. The simplicity of our robust invariance condition allows our algorithm to demonstrate better training times and better scalability as compared to a previous sampling-based approach.

Notation Define the partial ordering $\leq$ on $\mathbb{R}^n$ such that $\underline{x} \leq \overline{x} \iff \underline{x}_i \leq \overline{x}_i$ for every $i = 1, \ldots, n$. Let $[\underline{x}, \overline{x}] := \{x \in \mathbb{R}^n : \underline{x} \leq x \leq \overline{x}\}$ denote a closed and bounded interval in $\mathbb{R}^n$, and let

$\mathbb{IR}^n$ denote the set of all such intervals. Define the upper triangle $\mathcal{T}^{2n}_{\geq 0} := \{[\frac{x}{\overline{x}}] \in \mathbb{R}^{2n} : \underline{x} \leq \overline{x}\}$, and note that $\mathbb{IR}^n \simeq \mathcal{T}^{2n}_{\geq 0}$. We denote this equivalence with $[[\frac{x}{\overline{x}}]] := [\underline{x}, \overline{x}]$. The partial order $\leq$ on $\mathbb{R}^n$ induces the southeast partial order $\leq_{\text{SE}}$ on $\mathbb{R}^{2n}$, where $\left[\begin{smallmatrix} x \\ \hat{x} \end{smallmatrix}\right] \leq_{\text{SE}} \left[\begin{smallmatrix} y \\ \hat{y} \end{smallmatrix}\right] \iff x \leq y$ and $\hat{y} \leq \hat{x}$. For $x^1 \in \mathbb{R}^{n_1}$, $x^2 \in \mathbb{R}^{n_2}$, $\ldots$, $x^m \in \mathbb{R}^{n_m}$, let $(x^1, x^2, \ldots, x^m) \in \mathbb{R}^{n_1+n_2+\cdots+n_m}$ denote their concatenation. For two vectors $x, y \in \mathbb{R}^n$ and $i \in \{1, \ldots, n\}$, let $x_{i:y} \in \mathbb{R}^n$ be the vector obtained by replacing the ith entry of x with that of y, *i.e.*, $(x_{i:y})_j = y_j$ if $i = j$ and otherwise $(x_{i:y})_j = x_j$. Let $\mathbf{I}_n \in \mathbb{R}^{n \times n}$ denote the $n \times n$ identity matrix. We represent a *convex polytope* as a nonempty set $\langle H, \underline{y}, \overline{y} \rangle := \{x \in \mathbb{R}^n : \underline{y} \leq Hx \leq \overline{y}\}$, for a full rank matrix $H \in \mathbb{R}^{m \times n}$. [1]

Preliminaries Consider the following nonlinear control system with disturbance,

$$\dot{x} = f(x, u, w), \tag{1}$$

where $x \in \mathbb{R}^n$ is the state of the system, $u \in \mathbb{R}^p$ is the control input, $w \in \mathcal{W} \subset \mathbb{R}^q$ is some disturbance in compact set $\mathcal{W}$, and $f : \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}^n$ is a locally Lipschitz vector field. Let $\pi : \mathbb{R}^n \to \mathbb{R}^p$ denote a neural network controller for the system, and define the closed-loop system

$$\dot{x} = f^\pi(x, w) := f(x, \pi(x), w). \tag{2}$$

For the system (2), let $[0, \infty) \ni t \mapsto \phi_{f^\pi}(t, x_0, \mathbf{w})$ denote its unique trajectory from initial condition x_0 at time 0 under piecewise continuous disturbance mapping $\mathbf{w} : [0, \infty) \to \mathcal{W}$. The set $\mathcal{S} \subseteq \mathbb{R}^n$ is *$\mathcal{W}$-robustly forward invariant* if $x_0 \in \mathcal{S}$ implies that $\phi_{f^\pi}(t, x_0, \mathbf{w}) \in \mathcal{S}$ for any $t \geq 0$ and any piecewise continuous $\mathbf{w} : [0, \infty) \to \mathcal{W}$. The goal of this work is to *train* the neural network π such that a given polytope $\langle H, \underline{y}, \overline{y} \rangle$ is a $\mathcal{W}$-robustly forward invariant set for the closed-loop system (2).

2. The Closed-Loop Embedding System

One of the biggest challenges in verifying neural network controlled dynamical systems is bounding the nonlinear behavior of the neural network controller while capturing its stabilizing closed-loop effects, which are important for invariance analysis. In this section, we recap the approach from Jafarpour et al. (2024), which builds inclusion functions and embedding systems that capture the first-order interactions of the system with the neural network.

2.1. Closed-Loop Inclusion Function

First, we discuss two computational tools we use to bound the closed-loop dynamics (2). The first tool is CROWN (Zhang et al., 2018), which obtains a *local affine bound*. Given an interval $[\underline{x}, \overline{x}]$, CROWN propogates linear bounds to obtain a tuple $(\underline{C}, \overline{C}, \underline{d}, \overline{d})$ satisfying

$$\underline{C}x + \underline{d} \leq \pi(x) \leq \overline{C}x + \overline{d},$$

valid for every $x \in [\underline{x}, \overline{x}]$. The second tool is a *mixed Jacobian-based inclusion* for the open-loop dynamics f from (1): given centering points and intervals $\mathring{x} \in [\underline{x}, \overline{x}]$, $\mathring{u} \in [\underline{u}, \overline{u}]$, $\mathring{w} \in [\underline{w}, \overline{w}]$, there are interval mappings $\mathsf{M}_x, \mathsf{M}_u, \mathsf{M}_w$ each with argument $(\underline{x}, \overline{x}, \underline{u}, \overline{u}, \underline{w}, \overline{w})$ such that

$$f(x, u, w) \in [\mathsf{M}_x](x - \mathring{x}) + [\mathsf{M}_u](u - \mathring{u}) + [\mathsf{M}_w](w - \mathring{w}) + f(\mathring{x}, \mathring{u}, \mathring{w}),$$

1. Under these assumptions, convex polytopes are bounded. We note that any compact H-rep convex polytope $\{x \in \mathbb{R}^n : Hx \leq y\}$ may be written as $\langle H, \underline{y}, y \rangle$ by taking, *e.g.*, sufficiently small $\underline{y}$.

for any $x \in [\underline{x}, \overline{x}]$, $u \in [\underline{u}, \overline{u}]$, $w \in [\underline{w}, \overline{w}]$. For instance, inclusion functions for the Jacobian matrices of the map with respect to (x, u, w) imply the inclusion by the mean value theorem—however, we use the *mixed Jacobian matrix* which is less conservative (Harapanahalli and Coogan, 2025). The toolbox `immrax` (Harapanahalli et al., 2024) automatically constructs these bounds using automatic differentiation and interval analysis. Given the open-loop system (1) and a neural network controller π, we next build a closed-loop mixed Jacobian-based *inclusion function* as

$$\begin{aligned}\mathsf{F}^\pi(\underline{x}, \overline{x}, \underline{w}, \overline{w}) = \begin{bmatrix} \underline{H}^+ - \underline{\mathsf{M}}_x & \underline{H}^- \\ \overline{H}^- - \overline{\mathsf{M}}_x & \overline{H}^+ \end{bmatrix} \begin{bmatrix} \underline{x} \\ \overline{x} \end{bmatrix} + \begin{bmatrix} -\underline{\mathsf{M}}_w^- & \underline{\mathsf{M}}_w^- \\ -\overline{\mathsf{M}}_w^+ & \overline{\mathsf{M}}_w^+ \end{bmatrix} \begin{bmatrix} \underline{w} \\ \overline{w} \end{bmatrix} + \begin{bmatrix} -\underline{\mathsf{M}}_u \underline{u} + \underline{\mathsf{M}}_u^+ \underline{d}_{[\underline{x},\overline{x}]} + \underline{\mathsf{M}}_u^- \overline{d}_{[\underline{x},\overline{x}]} + f(\underline{x}, \underline{u}, \underline{w}) \\ -\overline{\mathsf{M}}_u \underline{u} + \overline{\mathsf{M}}_u^+ \overline{d}_{[\underline{x},\overline{x}]} + \overline{\mathsf{M}}_u^- \underline{d}_{[\underline{x},\overline{x}]} + f(\underline{x}, \underline{u}, \underline{w}) \end{bmatrix}, \\ \underline{H} = \underline{\mathsf{M}}_x + \underline{\mathsf{M}}_u^+ \underline{C} + \underline{\mathsf{M}}_u^- \overline{C}, \quad \overline{H} = \overline{\mathsf{M}}_x + \overline{\mathsf{M}}_u^+ \overline{C} + \overline{\mathsf{M}}_u^- \underline{C}, \end{aligned} \tag{3}$$

where $(A^+)_{ij} = \max\{A_{ij}, 0\}$ and $A^- = A - A^+$ for any matrix A. As shown in (Jafarpour et al., 2024, Thm. 3), this inclusion function satisfies the bound $f^\pi(x, w) \in \mathsf{F}^\pi(\underline{x}, \overline{x}, \underline{w}, \overline{w})$ for every $x \in [\underline{x}, \overline{x}]$ and $w \in [\underline{w}, \overline{w}]$. In general, this approach works for any choice of $(\mathring{x}, \mathring{u}, \mathring{w})$ equal to a corner of the box $[\underline{x}, \overline{x}] \times [\underline{u}, \overline{u}] \times [\underline{w}, \overline{w}]$, with slight modifications to the expression (3). As shown in Jafarpour et al. (2024), the inclusion function (3) captures the first-order interactions between the dynamics and the neural network controller in the first term multiplying $\left[\begin{smallmatrix} \underline{x} \\ \overline{x} \end{smallmatrix}\right]$.

2.2. Hyperrectangle Invariance Using the Embedding System

The mixed Jacobian-based closed-loop inclusion function provides an efficient and scalable technique for bounding the output of the closed-loop vector field f^π in (2). The inclusion function (3) builds a *closed-loop embedding system*,

$$\begin{aligned} \dot{\underline{x}}_i &= \left(\underline{\mathsf{E}}(\underline{x}, \overline{x}, \underline{w}, \overline{w})\right)_i := \left(\underline{\mathsf{F}}^\pi(\underline{x}, \overline{x}_{i:\underline{x}}, \underline{w}, \overline{w})\right)_i, \\ \dot{\overline{x}}_i &= \left(\overline{\mathsf{E}}(\underline{x}, \overline{x}, \underline{w}, \overline{w})\right)_i := \left(\overline{\mathsf{F}}^\pi(\underline{x}_{i:\overline{x}}, \overline{x}, \underline{w}, \overline{w})\right)_i, \end{aligned} \tag{4}$$

where $\left[\begin{smallmatrix} \underline{x} \\ \overline{x} \end{smallmatrix}\right] \in \mathcal{T}_{\geq 0}^{2n}$, $\left[\begin{smallmatrix} \underline{w} \\ \overline{w} \end{smallmatrix}\right] \in \mathcal{T}_{\geq 0}^{2q}$, and $\mathsf{E} : \mathcal{T}_{\geq 0}^{2n} \times \mathcal{T}_{\geq 0}^{2q} \to \mathbb{R}^{2n}$. The embedding system can be thought of as evolving each face of the hyperrectangle separately, in a manner that contains the behavior of the original dynamics. The evaluation separately on each face ($[\underline{x}, \overline{x}_{i:\underline{x}}]$ and $[\underline{x}_{i:\overline{x}}, \overline{x}]$) of the hyperrectangle is a key feature of our approach. In our Python implementation, we use JAX to vectorize these faces for efficient evaluation on a GPU. The next Proposition is from (Harapanahalli et al., 2023, Prop. 1), and describes how robust forward invariance is simplified to a single evaluation of the vector field (4).

Proposition 1 (Invariant hyperrectangles) *Consider the closed-loop system (2), with the inclusion function F^π from (3), and the induced embedding system E from (4). If*

$$\mathsf{E}(\underline{x}_0, \overline{x}_0, \underline{w}, \overline{w}) \geq_{\mathrm{SE}} 0,$$

then the hyperrectangle $[\underline{x}_0, \overline{x}_0]$ is $[\underline{w}, \overline{w}]$-robustly forward invariant for the closed-loop system (2).

Recall that $\mathsf{E}(\underline{x}_0, \overline{x}_0, \underline{w}, \overline{w}) \geq_{\mathrm{SE}} 0$ if and only if $\underline{\mathsf{E}}(\underline{x}_0, \overline{x}_0, \underline{w}, \overline{w}) \geq 0$ and $\overline{\mathsf{E}}(\underline{x}_0, \overline{x}_0, \underline{w}, \overline{w}) \leq 0$. Proposition 1 characterizes a boundary condition on the hyperrectangle $[\underline{x}, \overline{x}]$ based on the Nagumo theorem (Blanchini, 1999, Thm. 3.1)—namely, these conditions ensure that the vector field points inside of the box along the entire boundary $\partial[\underline{x}, \overline{x}]$, thus, trajectories can never escape the box.

3. Polytope Invariance Using the Lifted Embedding System

The approach from the previous section is sufficient for forward invariant intervals of $\mathbb{R}^n$, *i.e.*, axis-aligned hyperrectangles. However, there are many systems that do not admit interval invariant sets under any controller. For instance, the phenomenon illustrated in the following example is inherent to many second-order control systems, such as mechanical systems for which the control is applied as a force, which is integrated twice to give the position state. In this section, we extend the theory to verify arbitrary compact polytopes.

Example 1 (Mechanical systems) *Consider the following mechanical system*

$$\dot{x}_1 = x_2,\ \dot{x}_2 = u,$$

where x_1 is the position, x_2 is the velocity, and u is the applied force. In these coordinates, it is impossible to design a controller $u := \pi(x)$ such that a hyperrectangle is forward invariant. To see this, consider the box $\mathcal{S}_1 = [-1, 1] \times [-1, 1]$. The point $[1, 1]^T \in \mathcal{S}_1$, and the vector field is $f([1, 1]^T) = [1, u]^T$. Regardless of the control, $\dot{x}_1 > 0$, and the system will leave $\mathcal{S}_1$. A similar argument holds for any other nonempty interval around the origin. Next, consider $u := \pi(x) = -2x_1 - 3x_2$, and the transformation $T = \left[\begin{smallmatrix} 1 & 1 \\ -1 & -2 \end{smallmatrix}\right]$. In transformed coordinates $y := T^{-1}x$,

$$\dot{y} = T^{-1}(A + BK)Ty = \left[\begin{smallmatrix} -1 & 0 \\ 0 & -2 \end{smallmatrix}\right] y.$$

Since the system is diagonal with negative eigenvalues, the vector field always points towards the origin, thus any hyperrectangle containing the origin is forward invariant for this transformed system. For example, the box $[-\frac{1}{2}, \frac{1}{2}] \times [-\frac{1}{2}, \frac{1}{2}]$ is forward invariant for the transformed system, implying that $\mathcal{S}_2 = \langle T^{-1}, \begin{bmatrix} -1/2 \\ -1/2 \end{bmatrix}, \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} \rangle$ is forward invariant for the original system. The sets $\mathcal{S}_1$ and $\mathcal{S}_2$ are visualized in Figure 1 in blue and green respectively.

3.1. Lifted System

The *lifted system* embeds the original system into a n-dimensional subspace of $\mathbb{R}^m$.

Definition 2 (Lifted system) *Consider the closed-loop system (2), and let $H \in \mathbb{R}^{m \times n}$ be a full rank matrix. Let $H^+ \in \mathbb{R}^{n \times m}$ be any matrix satisfying $H^+H = \mathbf{I}_n$. Then*

$$\dot{y} = g(y, w) := Hf^\pi(H^+y, w) = Hf(H^+y, \pi(H^+y), w), \tag{5}$$

*with state $y := Hx$, is the (H, H^+)-*lifted system *of (2).*

In the next Proposition, we parameterize the set of left inverses H^+, which will allow us to incorporate this matrix as an unconstrained decision variable in the training problem.

Proposition 3 (Parameterization of left inverses) *Let $H \in \mathbb{R}^{m \times n}$ be full rank. Let $N \in \mathbb{R}^{m \times (m-n)}$ be a basis spanning the left nullspace of H and let $H^\dagger = (H^TH)^{-1}H^T$ be the Moore-Penrose Pseudoinverse of H. Then the following characterizes the set of matrices satisfying $H^+H = \mathbf{I}_n$:*

$$\{H^+ \in \mathbb{R}^{n \times m} : H^+ = H^\dagger + \eta N^T,\ \eta \in \mathbb{R}^{n \times (m-n)}\}.$$

A key property of the lifted system is that the original system lies on an invariant n-dimensional subspace of the lifted state space $\mathbb{R}^m$. The invariance of $\mathcal{H}$ will be incorporated as extra knowledge for building a good embedding system for the lifted system (5) in the next section.

Proposition 4 (Invariant subspace) *Consider the closed-loop system (2), with the (H, H^+)-lifted system (5). For any $x_0 \in \mathbb{R}^n$ and piecewise continuous $\mathbf{w} : [0, \infty) \to \mathcal{W}$,*

$$H\phi_{f^\pi}(t, x_0, \mathbf{w}) = \phi_g(t, Hx_0, \mathbf{w}).$$

Moreover, the linear subspace $\mathcal{H} := \{Hx : x \in \mathbb{R}^n\}$ is forward invariant for the lifted system.

3.2. Lifted Embedding System

One approach is to simply embed the lifted system and obtain a valid embedding system for the lifted dynamics (5). However, this would discard the *a priori* knowledge that the original system lives on the invariant subspace from Proposition 4. A technique for incorporating this information was explored in Shen and Scott (2017), where a refinement operator was used in conjunction with model redundancies to improve interval reachable set estimates for dynamical systems. The following Definition allows us to incorporate the knowledge that the original system lies on the subspace $\mathcal{H}$, and continue with the efficient interval analysis framework previously developed.

Definition 5 (Interval refinement operator) *Let $\mathcal{H} \subset \mathbb{R}^m$ be a subset. $\mathcal{I}_\mathcal{H} : \mathcal{T}_{\geq 0}^{2m} \to \mathcal{T}_{\geq 0}^{2m}$ is an* interval refinement operator *on $\mathcal{H}$ if for every $[\underline{y}, \overline{y}] \in \mathbb{IR}^m$,*

$$\mathcal{H} \cap [\underline{y}, \overline{y}] \subseteq [\underline{\mathcal{I}}_\mathcal{H}(\underline{y}, \overline{y}), \overline{\mathcal{I}}_\mathcal{H}(\underline{y}, \overline{y})] \subseteq [\underline{y}, \overline{y}].$$

For the case where $\mathcal{H} = \{Hx : x \in \mathbb{R}^n\}$ is a subspace, we can use the fact that the left null space of H encodes $(m-n)$ constraints on $\mathbb{R}^m$ equivalently defining the subspace $\mathcal{H}$. Given a library of left null vectors $A \in \mathbb{R}^{N \times m}$, where $AH = 0$, the following defines a valid interval refinement operator,

$$\left[\mathcal{I}_\mathcal{H}(\underline{y}, \overline{y})\right]_j = [\underline{y}_j, \overline{y}_j] \bigcap_{A_{i,j} \neq 0} -\frac{1}{A_{i,j}} \sum_{k \neq i} A_{i,k}[\underline{z}_k, \overline{z}_k] \tag{6}$$

This is essentially equivalent to $\mathcal{I}_G$ from Shen and Scott (2017) which defines an interval refinement operator with the explicit knowledge that $Mz = b$ for some matrices M and b, and is further explored in Gould et al. (2025), where left null vectors are sampled in a structured manner to promote sparsity. We provide a proof that (6) provides a valid refinement operator on $\mathcal{H}$ in the appendices.

Definition 6 (Lifted embedding system) *Given a $\mathcal{H}$-refinement operator $\mathcal{I}_\mathcal{H}$ and an inclusion function G for the lifted closed-loop dynamics (5), define the (H, H^+)-lifted embedding system,*

$$\begin{aligned} \dot{\underline{y}}_i &= \left(\underline{\mathsf{G}}(\underline{\mathcal{I}}_\mathcal{H}(\underline{y}, \overline{y}_{i:\underline{y}}), \overline{\mathcal{I}}_\mathcal{H}(\underline{y}, \overline{y}_{i:\underline{y}}), \underline{w}, \overline{w})\right)_i, \\ \dot{\overline{y}}_i &= \left(\overline{\mathsf{G}}(\underline{\mathcal{I}}_\mathcal{H}(\underline{y}_{i:\overline{y}}, \overline{y}), \overline{\mathcal{I}}_\mathcal{H}(\underline{y}_{i:\overline{y}}, \overline{y}), \underline{w}, \overline{w})\right)_i. \end{aligned} \tag{7}$$

Since we know the original system lives on $\mathcal{H}$ by Proposition 4, each face of the $[\underline{y}, \overline{y}]$ hyperrectangle can be refined to a smaller interval containing the intersection of $\mathcal{H}$ and the face. Figure 1 demonstrates this procedure for the 6 faces of a 3 dimensional hyperrectangle intersecting a 2 dimensional subspace from Example 2 below.

The following Theorem describes how, once again, robust forward polytope forward invariance is simplified to a single evaluation of the vector field (7). The full proof is in the appendices.

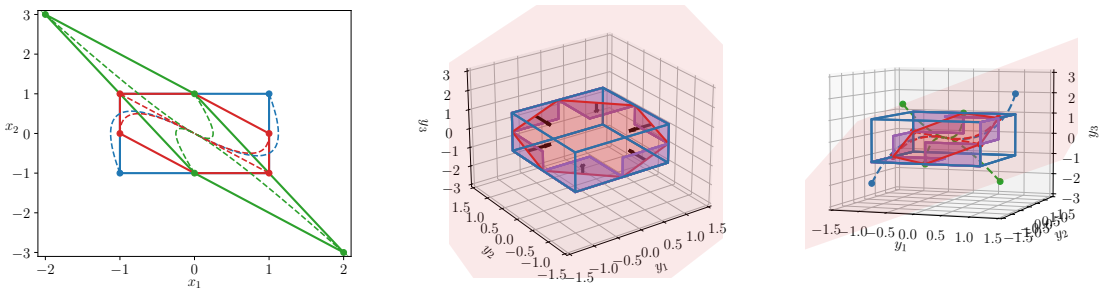

Figure 1: (**Left**) The mechanical system from Examples 1 and 2 is visualized with several polytopes in solid lines and solution trajectories in dotted lines. The blue set $\mathcal{S}_1$ is a hyperrectangle that cannot be forward invariant, the green invariant set $\mathcal{S}_2$ is obtained from the transformation associated with the eigenvalue decomposition, and the red invariant set $\mathcal{S}_3$ is obtained by lifting the system into 3-dimensions. (**Middle/Right**) $\mathcal{I}_{\mathcal{H}}$ is applied to every face of the box $[\underline{y}, \overline{y}] = [-1, 1]^3$ (blue) for the subspace $\mathcal{H}$ from Example 2 (red). The outputs (purple) are refined interval sets which still contain $\mathcal{H}$. The red outlined set ($\mathcal{H} \cap [\underline{y}, \overline{y}]$) corresponds to the polytope $\langle H, \underline{y}, \overline{y} \rangle$ from Example 2 and the red polytope from the left figure. The original system evolves on the subspace $\mathcal{H}$ by Proposition 4, and the positivity condition $\mathsf{E}_{H,H^+}(\underline{y}, \overline{y}, \underline{w}, \overline{w}) \geq_{\text{SE}} 0$ from Theorem 7 implies the vector component points in the direction indicated by the black arrows (**middle**) along each refined face. The corresponding trajectories from the left figure are shown in dotted lines (**right**).

Theorem 7 (Polytope invariant sets) *Consider the closed-loop system (2). Let $H \in \mathbb{R}^{m \times n}$, and $H^+ \in \mathbb{R}^{n \times m}$ satisfy $H^+ H = \mathbf{I}_n$. Let E_{H,H^+} denote the (H, H^+)-lifted embedding system (7). If*

$$\mathsf{E}_{H,H^+}(\underline{y}, \overline{y}, \underline{w}, \overline{w}) \geq_{\text{SE}} 0,$$

then the polytope $\langle H, \underline{y}, \overline{y} \rangle$ is $[\underline{w}, \overline{w}]$-robustly forward invariant for the original system (2).

Remark 8 (Comparison to the literature) *Theorem 7 generalizes (Harapanahalli et al., 2023, Thm. 2), which verified invariant polytopes when H is square. (taking $H^+ = H^{-1}$ recovers the result).*

Similar to the original embedding system, the lifted embedding system provides a scalable and trainable condition for checking the forward invariance of a polytope. In the next Example, we return to the mechanical system to demonstrate how this condition can be used to certify invariance.

Example 2 (Mechanical system, cont.) *Consider the mechanical control system from Example 1, with the same feedback controller. With the definitions $H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$, $\underline{y} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$, $\overline{y} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$, $\mathcal{I}_{\mathcal{H}}$ from (6), and $\mathsf{E}_{H,H^\dagger}$ as the $(H, H^\dagger)$-lifted embedding system (7),*

$$\mathsf{E}_{H,H^\dagger}(\underline{y}, \overline{y}) = [0, 1, \tfrac{4}{3}, 0, -1, -\tfrac{4}{3}]^T \geq_{\text{SE}} 0,$$

thus the polytope $\mathcal{S}_3 = \langle H, \underline{y}, \overline{y} \rangle$ is a forward invariant set for the original system. The polytope $\mathcal{S}_3$ is visualized in Figure 1 in green. Additionally, the box $[\underline{y}, \overline{y}]$, subspace $\mathcal{H} = \{Hx : x \in \mathbb{R}^2\}$, intersection $\mathcal{H} \cap [\underline{y}, \overline{y}]$, and outputs of $\mathcal{I}_{\mathcal{H}}(\underline{y}, \overline{y}_{i:\underline{y}})$ and $\mathcal{I}_{\mathcal{H}}(\underline{y}_{i:\overline{y}}, \overline{y})$ are all visualized in Figure 1. The black arrows show the direction $(+/-)$ of the embedding vector field along those components, showing how the polytope is certified to be forward invariant using Theorem 7.

4. Certified Polytope Invariance Training

Using the lifted embedding system, in this section we construct a loss for training controllers with certified forward invariant polytopes. First, we assume there already exists an original loss $\mathcal{L}^{\text{data}}$. Ideally, we would add the positivity condition from Theorem 7 as a hard constraint, however, the complexity of neural networks often necessitates the use of unconstrained optimization algorithms. Instead, we use regularizing loss term, with the hope that the algorithm will eventually tend towards network parameters that evaluate as $\geq_{\text{SE}} 0$. For a desired polytope $\mathcal{S} = \langle H, \underline{y}, \overline{y} \rangle$, we use the loss

$$\begin{aligned} \mathcal{L}(\pi, \eta) &= \mathcal{L}^{\text{data}}(\pi) + \lambda \mathcal{L}^{\mathcal{S}}(\pi, \eta), \\ \mathcal{L}^{\mathcal{S}}(\pi, \eta) &= \sum_{i=1}^{m} \text{ReLU}(\overline{\mathsf{E}}_{H, H_\eta^+}(\underline{y}, \overline{y}, \underline{w}, \overline{w})_i + \varepsilon) + \sum_{i=1}^{m} \text{ReLU}(-\underline{\mathsf{E}}_{H, H_\eta^+}(\underline{y}, \overline{y}, \underline{w}, \overline{w})_i + \varepsilon) \end{aligned} \tag{8}$$

to induce $[\underline{w}, \overline{w}]$-robust forward invariance, where E_{H, H_η^+} is the (H, H_η^+)-lifted embedding system of f^π, $\eta \in \mathbb{R}^{n \times (m-n)}$ is a decision variable choosing H^+ according to Proposition 3, and $\varepsilon > 0$ is a small numerical constant. The ReLU avoids incurring negative loss when the constraint is satisfied, instead switching to purely minimizing the error to the data, allowing the model to improve its efficacy. Then, if the descent brings the optimization to a point where $\mathsf{E}_{H, H_\eta^+} \not\geq_{\text{SE}} 0$, the loss (8) appears again. As a result, large values of λ work well in practice.

Remark 9 (Choice of η) *While the choice of η and H_η^+ may seem inconsequential, its inclusion as a parameter is empirically crucial in choosing a lifted system for invariance analysis. Analyzing (5), the choice of H^+ changes the dynamics of the lifted system off of the invariant subspace $\mathcal{H}$, which can drastically reduce the overconservatism of the lifted embedding system in practice.*

5. Experiments

[2] We use `jax_verify` and `immrax` (Harapanahalli et al., 2024) to compute the embedding system (4). JAX (Bradbury et al., 2018) vectorizes the evaluations on each face of the lifted hyperrectangle from Theorem 7 onto the GPU, Equinox (Kidger and Garcia, 2021) helps autodifferentiate (8) for gradient evaluations, and Optax (DeepMind et al., 2020) provides the gradient-based optimizer.

Segway Model Consider the nonlinear dynamics of a segway from Gurriet et al. (2018),

$$\begin{bmatrix} \dot{\phi} \\ \dot{v} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ \frac{\cos\phi(-1.8u+11.5v+9.8\sin\phi)-10.9u+68.4v-1.2\dot{\phi}^2\sin\phi}{\cos\phi-24.7} \\ \frac{(9.3u-58.8v)\cos\phi+38.6u-234.5v-\sin\phi(208.3+\dot{\phi}^2\cos\phi)}{\cos^2\phi-24.7} \end{bmatrix} \tag{9}$$

with state $x = [\phi \; v \; \dot{\phi}]^T \in \mathbb{R}^3$. To compare with the literature, we mimic the training procedure from Huang et al. (2023), where the network is trained to imitate the LQR gains from the linearization of the system around the origin, while certifying a robust forward invariant region around the

2. Experiments were performed on a computer running Kubuntu 22.04 with Intel Xeon Gold 6230, NVIDIA Quadro RTX 8000, and 64 GB of RAM. All code for the experiments is available at https://github.com/gtfactslab/Polytope-Training. For all of the experiment details, please see the appendices.

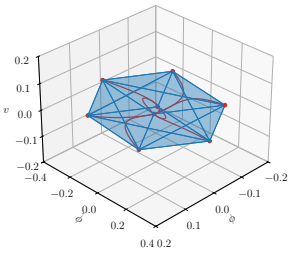

Figure 2: The certified robust invariant polytope in $\mathbb{R}^3$ for the segway is visualized in blue. Simulations of trajectories starting from its vertices are in red.

Method	Volume	Runtime (s) Setup (JIT)	Training	Total
Ours	0.00152	139.	**11.4**	**150.4**
FI-ODE	**0.158**	–	2758[3]	2758

Table 1: Comparison to robust invariance training literature

equilibrium. The robustness is with respect to a $\pm 2\%$ uncertainty in each of the system parameters, which we represent as a bounded multiplicative disturbance $(1 + w_k)$ for $w \in [-0.02, 0.02]^{11}$ applied independently to each system parameter from (9).

One approach to define a suitable polytope is to attempt to diagonalize the system, as demonstrated in Example 1 where a diagonalizing transformation yielded forward invariance since all eigenvalues were negative. This intuition extends to the nonlinear neural network controlled system: (i) let A_{cl} be the Jacobian matrix of the linearized system in closed-loop with the LQR gains, *i.e.*, $A_{\text{cl}} = \frac{\partial f}{\partial x}(0,0) + \frac{\partial f}{\partial u}(0,0)K$; (ii) let the Jordan decomposition of this matrix be $T^{-1}A_{cl}T = \Lambda$, where Λ is in Jordan form; (iii) choose the matrix $H = T^{-1}$, and fix offsets, *e.g.*, $[\underline{y}, \overline{y}] = [-2, 2]^3$.

We compare to FI-ODE (Huang et al., 2023), which to our knowledge, is the only other work that trains and certifies robust forward invariant sets in neural network controlled systems. Our method requires an initial setup time to just-in-time (JIT) compile the optimizer step. The positivity check in Theorem 7 verifies forward invariance in a fraction of the time compared to the sampling-based approaches by vectorizing over the faces of the hypercube in the lifted embedding space—allowing us to incorporate it directly into the objective. After compilation, it takes 11.4 seconds to train a robust neural network controller, using ADAM (Kingma and Ba, 2014) with a step size of 0.001. The robust forward invariant polytope and sample system trajectories are visualized in Figure 2.

The cost of our simple condition is possible overconservatism, demonstrated in our smaller volume compared with Huang et al. (2023) in Table 1. We suspect this is because FI-ODE incorporates the P matrix from their Lyapunov function into the initial training procedure, allowing them to shape the invariant set during training, while in this work we do not allow any shaping of the polytopes. However, FI-ODE trains the neural network first without any guarantees, then a post-training sampling-based robust verification step verifies that a sublevel set of the Lyapunov function is robustly forward invariant, which suffers when scaling to higher dimensions.

Platoon of Vehicles with Nonlinearities and Disturbances In this example, we investigate how our proposed approach scales with state dimension n. We consider a platoon of N vehicles, each with the following dynamics,

$$\dot{p}_j = v_j, \quad \dot{v}_j = \sigma(u_j)(1 + w_j), \tag{10}$$

where for each vehicle $j = 1, \ldots, N$, $p_j \in \mathbb{R}$ is its position, v_j is its velocity, $u_j \in \mathbb{R}$ is its control input, $w_j \in [-0.1, 0.1]$ is a bounded disturbance input, and $\sigma(u) = u_{\text{lim}} \tanh(u/u_{\text{lim}})$ is a softmax nonlinearity, $u_{\text{lim}} = 10$. Let $x_j = [p_j, v_j]^T \in \mathbb{R}^2$ for each j. Each vehicle is controlled by a shared

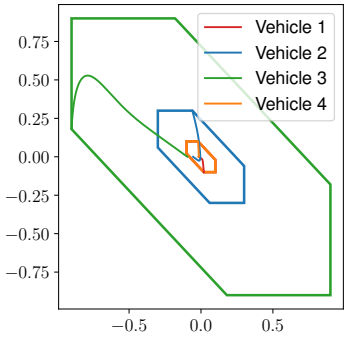

Figure 3: The invariant polytope and a sample system trajectory for the platoon with 4 vehicles is pictured. Vehicles 1 and 4 share the same invariant set in orange.

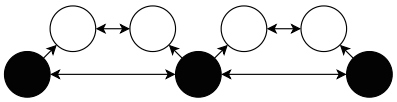

Figure 4: The connection topology of the platoon. Leaders are filled and arrows represent relative measurements.

N	# States Original	# States Lifted	Runtime (s) Setup	Runtime (s) Training (#iter)
4	8	12	35.8	6.90 (724)
10	20	30	64.4	57.7 (725)
16	32	48	128.	170. (807)
22	44	66	264.	408. (890)
28	56	84	478.	1040 (1267)

Table 2: Scalability with respect to number of vehicles

neural network control policy $\pi : \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$,

$$u_j = \begin{cases} \pi((x_j,\ x_{j-3} - x_j,\ x_j - x_{j+3})), & j = 3k, \\ \pi((0,\ x_{j-1} - x_j,\ x_j - x_{j+1})), & \text{otherwise}, \end{cases} \tag{11}$$

with $x_0 := 0$, $x_{N+1} := 0$. Every 3rd vehicle (leader) measures its true state, and the relative difference between the next two leaders. The rest of the platoon (followers) measures relative states between their nearest two neighbors. This communication scheme is pictured in Figure 4. The closed-loop system can be rewritten as $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \Pi(\mathbf{u}), \mathbf{w}) = \mathbf{f}^{\Pi}(\mathbf{x}, \mathbf{w})$, where $\mathbf{x} = (x_1, \ldots, x_N) \in \mathbb{R}^{2N}$, $\mathbf{u} = (u_1, \ldots, u_N) \in \mathbb{R}^N$, $\mathbf{w} = (w_1, \ldots, w_N) \in \mathbb{R}^N$, $\mathbf{f} : \mathbb{R}^{2N} \times \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}^{2N}$ is the dynamics (10), $\Pi : \mathbb{R}^N \to \mathbb{R}^N$ represents the policy (11).

We would like to train the shared feedback policy π such that the closed-loop platoon renders the polytope $\langle H, \underline{y}, \overline{y} \rangle$, for $H = \mathbf{I}_N \otimes \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{smallmatrix}\right]$, $\overline{y} = \underbrace{[1, 3, 9, 1, 3, 9, \ldots, 1, 3, 9, 1]^T}_{\in \mathbb{R}^N} \otimes [0.1, 0.1, 0.08]^T$, $\underline{y} = -\overline{y}$, robustly forward invariant. We use the loss (8) with no data loss ($\mathcal{L}^{\text{data}} = 0$), $\lambda = 1$, a $6 \times 32 \times 32 \times 32 \times 1$ fully connected ReLU network for π, and a range of different numbers of vehicles N. The set $\langle H, \underline{y}, \overline{y} \rangle$ is illustrated in Figure 2. In Table 2, we outline how the training time scales with the number of state dimensions of the system and the lifted system. Compared to sampling based approaches which suffer from the curse of dimensionality, our approach scales reasonably in both setup and training time.

6. Conclusions

In this paper, we proposed a framework for training certified robust forward invariant polytopes in neural network controlled dynamical systems, using a novel lifted embedding system where a single evaluation certifies forward invariance. Through two experiments, we demonstrated how our approach both improves on existing sampling-based approaches in runtime, and scales well with state dimension. In future work, we plan to address the overconservatism of our approach by incorporating the polytope itself into the optimization.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation under awards #2219755, #2333488, and #2440387, and by the Air Force Office of Scientific Research under Grant FA9550-23-1-0303.

References

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

F. Blanchini. Set invariance in control. *Automatica*, 35(11):1747–1767, 1999. doi: 10.1016/S0005-1098(99)00113-2.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Christopher Brix, Mark Niklas Müller, Stanley Bak, Taylor T Johnson, and Changliu Liu. First three years of the international verification of neural networks competition (vnn-comp). *International Journal on Software Tools for Technology Transfer*, 25(3):329–339, 2023.

Steven Chen, Kelsey Saulnier, Nikolay Atanasov, Daniel D. Lee, Vijay Kumar, George J. Pappas, and Manfred Morari. Approximating explicit model predictive control using constrained neural networks. In *2018 Annual American Control Conference (ACC)*, pages 1520–1527, 2018. doi: 10.23919/ACC.2018.8431275.

H. Dai, L. Landry, B.and Yang, M. Pavone, and R. Tedrake. Lyapunov-stable neural-network control. *arXiv preprint arXiv:2109.14152*, 2021.

DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020. URL http://github.com/google-deepmind.

M. Everett, G. Habibi, C. Sun, and J. How. Reachability analysis of neural feedback loops. *IEEE Access*, 9:163938–163953, 2021. doi: 10.1109/ACCESS.2021.3133370.

Brendan Gould, Akash Harapanahalli, and Samuel Coogan. Automatic and scalable safety verification using interval reachability with subspace sampling. *IEEE Control Systems Letters*, 9:1592–1597, 2025. doi: 10.1109/LCSYS.2025.3582206.

Thomas Gurriet, Andrew Singletary, Jacob Reher, Laurent Ciarletta, Eric Feron, and Aaron Ames. Towards a framework for realizable safety critical control through active set invariance. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 98–106, 2018. doi: 10.1109/ICCPS.2018.00018.

Akash Harapanahalli and Samuel Coogan. A linear differential inclusion for contraction analysis to known trajectories. *IEEE Transactions on Automatic Control*, pages 1–8, 2025. doi: 10.1109/TAC.2025.3626266.

Akash Harapanahalli, Saber Jafarpour, and Samuel Coogan. Forward invariance in neural network controlled systems. *IEEE Control Systems Letters*, 7:3962–3967, 2023. ISSN 2475-1456. doi: 10.1109/lcsys.2023.3341980. URL http://dx.doi.org/10.1109/LCSYS.2023.3341980.

Akash Harapanahalli, Saber Jafarpour, and Samuel Coogan. immrax: A parallelizable and differentiable toolbox for interval analysis and mixed monotone reachability in jax. *IFAC-PapersOnLine*, 58(11):75–80, 2024.

H. Hu, M. Fazlyab, M. Morari, and G. J. Pappas. Reach-SDP: Reachability analysis of closed-loop systems with neural network controllers via semidefinite programming. In *59th IEEE Conference on Decision and Control (CDC)*, pages 5929–5934, 2020. doi: 10.1109/CDC42340.2020.9304296.

C. Huang, J. Fan, X. Chen, W. Li, and Q. Zhu. POLAR: A polynomial arithmetic framework for verifying neural-network controlled systems. In *Automated Technology for Verification and Analysis*, pages 414–430. Springer International Publishing, 2022.

Yujia Huang, Ivan Dario Jimenez Rodriguez, Huan Zhang, Yuanyuan Shi, and Yisong Yue. Fi-ode: Certifiably robust forward invariance in neural odes, 2023.

Saber Jafarpour, Akash Harapanahalli, and Samuel Coogan. Efficient interaction-aware interval analysis of neural network feedback loops. *IEEE Transactions on Automatic Control*, pages 1–16, 2024. doi: 10.1109/TAC.2024.3420968.

Louis Jouret, Adnane Saoud, and Sorin Olaru. Safety verification of neural-network-based controllers: a set invariance approach. *IEEE Control Systems Letters*, 2023.

Patrick Kidger and Cristian Garcia. Equinox: neural networks in JAX via callable PyTrees and filtered transformations. *Differentiable Programming workshop at Neural Information Processing Systems 2021*, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Niklas Kochdumper, Christian Schilling, Matthias Althoff, and Stanley Bak. Open-and closed-loop neural network verification using polynomial zonotopes. In *NASA Formal Methods Symposium*, pages 16–36. Springer, 2023.

Diego Manzanas Lopez, Matthias Althoff, Luis Benet, Clemens Blab, Marcelo Forets, Yuhao Jia, Taylor T Johnson, Manuel Kranzl, Tobias Ladner, Lukas Linauer, et al. Arch-comp24 category report: Artificial intelligence and neural network control systems (ainncs) for continuous and hybrid systems plants. In *Proceedings of the 11th Int. Workshop on Applied*, volume 103, pages 64–121, 2024.

Mark Huasong Meng, Guangdong Bai, Sin Gee Teo, Zhe Hou, Yan Xiao, Yun Lin, and Jin Song Dong. Adversarial robustness of deep neural networks: A survey from a formal verification perspective. *IEEE Transactions on Dependable and Secure Computing*, 2022.

Antonis Papachristodoulou and Stephen Prajna. On the construction of lyapunov functions using the sum of squares decomposition. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 3, pages 3482–3487. IEEE, 2002.

Ivan Dario Jimenez Rodriguez, Aaron Ames, and Yisong Yue. Lyanet: A lyapunov framework for training neural odes. In *International Conference on Machine Learning*, pages 18687–18703. PMLR, 2022.

A. Saoud and R. G. Sanfelice. Computation of controlled invariants for nonlinear systems: Application to safe neural networks approximation and control. *IFAC-PapersOnLine*, 54(5):91–96, 2021. doi: 10.1016/j.ifacol.2021.08.480. Conference on Analysis and Design of Hybrid Systems (ADHS).

C. Schilling, M. Forets, and S. Guadalupe. Verification of neural-network control systems by integrating Taylor models and zonotopes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022. doi: 10.1609/aaai.v36i7.20790.

Kai Shen and Joseph K. Scott. Rapid and accurate reachability analysis for nonlinear dynamic systems by exploiting model redundancy. *Computers & Chemical Engineering*, 106:596–608, 2017. ISSN 0098-1354. doi: 10.1016/j.compchemeng.2017.08.001. ESCAPE-26.

U. Topcu, A. Packard, and P. Seiler. Local stability analysis using simulations and sum-of-squares programming. *Automatica*, 44(10):2669–2675, 2008. ISSN 0005-1098. doi: 10.1016/j.automatica.2008.03.010.

H.-D. Tran, X. Yang, D. Manzanas Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, and T. T. Johnson. NNV: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In *Computer Aided Verification*, pages 3–17. Springer International Publishing, 2020.

Wei Xiao, Tsun-Hsuan Wang, Ramin Hasani, Mathias Lechner, Yutong Ban, Chuang Gan, and Daniela Rus. On the forward invariance of neural odes. In *International conference on machine learning*, pages 38100–38124. PMLR, 2023.

H. Yin, P. Seiler, and M. Arcak. Stability analysis using quadratic constraints for systems with neural network controllers. *IEEE Transactions on Automatic Control*, 67(4):1980–1987, 2022. doi: 10.1109/TAC.2021.3069388.

H. Zhang, T-W. Weng, P-Y. Chen, C-J. Hsieh, and L. Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in Neural Information Processing Systems*, volume 31, page 4944–4953, 2018.

Appendix A. Algorithm for Certified Robust Forward Invariant Neural Network Training

The following Algorithm describes how to use the loss function (8) to promote forward invariance, and how to use the condition from Theorem 7 to easily check for certified forward invariance.

Algorithm 1: Certified polytope training using the lifted embedding system

Input: Desired polytope $\langle H, \underline{y}, \overline{y} \rangle = \{x \in \mathbb{R}^n : \underline{y} \leq Hx \leq \overline{y}\}$, $H \in \mathbb{R}^{m \times n}$ full rank, $\underline{y}, \overline{y} \in \mathbb{R}^m$, regularization constant $\lambda \geq 0$, disturbance set $[\underline{w}, \overline{w}]$, data loss $\mathcal{L}^{\text{data}}$

$N \leftarrow \text{null}(H^T)$

$\eta \leftarrow \mathbf{0}^{(m-n)\times m}$

repeat

$H_\eta^+ \leftarrow H^\dagger + \eta N^T$

$\mathsf{E} \leftarrow (H, H_\eta^+)$-Lifted Embedding System (7) for (2)

$\mathcal{L} \leftarrow \mathcal{L}^{\text{data}}(\pi) + \lambda \mathcal{L}^{\mathcal{S}}(\pi, \eta)$

$(\pi, \eta) \leftarrow \text{step\_optimizer}(\nabla_{(\pi,\eta)} \mathcal{L})$

until $\mathsf{E}(\underline{y}, \overline{y}, \underline{w}, \overline{w}) \succeq_{\text{SE}} 0$

Return: Neural network controller π with certified forward invariant polytope $\langle H, \underline{y}, \overline{y} \rangle$ for (2).

Appendix B. Proofs of Main Results

B.1. Proof of Proposition 3 - Parameterization of left inverses

Statement: Let $H \in \mathbb{R}^{m \times n}$ be full rank. Let $N \in \mathbb{R}^{m \times (m-n)}$ be a basis spanning the left nullspace of H and let $H^\dagger = (H^T H)^{-1} H^T$ be the Moore-Penrose Pseudoinverse of H. Then the set

$$\{H^+ \in \mathbb{R}^{n \times m} : H^+ = H^\dagger + \eta N^T,\ \eta \in \mathbb{R}^{n \times (m-n)}\} \tag{12}$$

characterizes the set of matrices satisfying $H^+ H = \mathbf{I}_n$.

Proof Let $A = \{H^+ \in \mathbb{R}^{n \times m} : H^+ = H^\dagger + \eta N^T,\ \eta \in \mathbb{R}^{n \times (m-n)}\}$, and let $B = \{H^+ \in \mathbb{R}^{n \times m} : H^+ H = \mathbf{I}_n\}$.

($\subseteq$) Consider any $\eta \in \mathbb{R}^{n \times (n-m)}$, and let $H_\eta^+ = H^\dagger + \eta N^T$. For any $x \in \mathbb{R}^n$,

$$H_\eta^+ Hx = ((H^T H)^{-1} H^T + \eta N^T) Hx = (H^T H)^{-1} H^T Hx + \eta N^T Hx = x + \eta \mathbf{0} x = x, \tag{13}$$

since $(H^T H)^{-1} H^T H = \mathbf{I}_n$, and since N is a basis for the left null-space of H. Thus, $A \subseteq B$.

($\supseteq$) Let H^+ be any matrix satisfying $H^+ H = \mathbf{I}_n$. Therefore, for any $x \in \mathbb{R}^n$,

$$H^+ Hx = x \iff H^\dagger Hx + (H^+ - H^\dagger) Hx = x \iff (H^+ - H^\dagger) Hx = 0, \tag{14}$$

which implies that each row $h_j \in \mathbb{R}^m$, $h_j := (H^+ - H^\dagger)_j$ is in the left nullspace of H. Since N is a basis for the left nullspace, for every $j = 1, \ldots, n$, there exists $\eta_j \in \mathbb{R}^{(m-n)}$ such that $N\eta_j = h_j$. Finally, with $\eta = [\eta_1 \cdots \eta_j \cdots \eta_n]^T$,

$$(H^+ - H^\dagger) = \eta N^T,$$

which implies that $H^+ = H^\dagger + \eta N^T$. Thus, $B \subseteq A$. ■

B.2. Proof of Proposition 4 - Invariant subspace

Statement: Consider the closed-loop system (2), with the (H, H^+)-lifted system (5). For any $x_0 \in \mathbb{R}^n$ and piecewise continuous $\mathbf{w} : [0, \infty) \to \mathcal{W}$,

$$H\phi_{f^\pi}(t, x_0, \mathbf{w}) = \phi_g(t, Hx_0, \mathbf{w}) \quad \text{and} \quad \phi_{f^\pi}(t, x_0, \mathbf{w}) = H^+\phi_g(t, Hx_0, \mathbf{w}).$$

Moreover, the linear subspace $\mathcal{H} := \{Hx : x \in \mathbb{R}^n\}$ is forward invariant for the lifted system.

Proof Let $t \mapsto x(t)$ be the trajectory of the closed-loop system (2) under piecewise continuous $t \mapsto \mathbf{w}(t)$, *i.e.*, $\phi_{f^\pi}(t, x_0, \mathbf{w})$. Let $t \mapsto y(t)$ be the trajectory of the (H, H^+)-lifted system (5) from initial condition Hx_0 under $\mathbf{w}$, *i.e.*, $\phi_g(t, Hx_0, \mathbf{w})$.

Let $\hat{y}(t) := Hx(t)$ for every $t \geq 0$, which in turn implies that $H^+\hat{y}(t) = x(t)$. Note that

$$\dot{\hat{y}}(t) = H\dot{x}(t) = Hf(x(t), w(t)) = Hf(H^+\hat{y}(t), w(t)), \tag{15}$$

which is the same dynamics as the (H, H^+)-lifted system (5). Additionally, note that $\hat{y}(0) = Hx(0) = Hx_0$. Thus, since $y(t)$ and $\hat{y}(t)$ have the same dynamics and the same initial condition, and f was assumed to be locally Lipschitz, the uniqueness of solutions to ODEs implies that $y(t) = \hat{y}(t)$. Thus, for every $t \geq 0$,

$$y(t) = Hx(t), \text{ which also implies that } H^+y(t) = x(t).$$

Moreover, we have shown that for any initial condition $x_0 \in \mathbb{R}^n$, $y_0 = Hx_0 \implies y(t) = Hx(t)$ for every $t \geq 0$, which implies that the linear subspace $\mathcal{H} = \{Hx : x \in \mathbb{R}^n\}$ is forward invariant for the lifted system (5). ■

B.3. Proof of Interval Refinement Operator - $\mathcal{I}_\mathcal{H}$ implementation

Need to show: Given $AH = 0$, $\mathcal{I}_\mathcal{H}$ from (6)

$$\left[\mathcal{I}_\mathcal{H}(\underline{y}, \overline{y})\right]_j = [\underline{y}_j, \overline{y}_j] \bigcap_{A_{i,j} \neq 0} -\frac{1}{A_{i,j}} \sum_{k \neq i} A_{i,k}[\underline{z}_k, \overline{z}_k]$$

satisfies

$$\mathcal{H} \cap [\underline{y}, \overline{y}] \subseteq [\underline{\mathcal{I}}_\mathcal{H}(\underline{y}, \overline{y}), \overline{\mathcal{I}}_\mathcal{H}(\underline{y}, \overline{y})] \subseteq [\underline{y}, \overline{y}].$$

Proof Let $H \in \mathbb{R}^{m \times n}$ be a full rank matrix, and let $A \in \mathbb{R}^{N \times m}$ satisfy $AH = 0$. Thus, $AHx = 0$ for any $x \in \mathbb{R}^n$, and for any $y \in \mathcal{H} = \{Hx : x \in \mathbb{R}^n\}$,

$$Ay = 0 \implies \sum_{k=1}^{m} A_{i,k}y_k = 0 \text{ for every } i = 1, \ldots, m - n,$$

as the equation $Ay = 0$ is the same as the system of equalities. Thus, for every $i = 1, \ldots, N$, and every $j = 1, \ldots, m$,

$$y_j = -\frac{1}{A_{i,k}} \sum_{k \neq j} A_{i,k}y_k.$$

With the additional information that $y \in [\underline{y}, \overline{y}]$, interval analysis on the RHS is still an overapproximation of $[\underline{y}_j, \overline{y}_j]$, but may provide a better overestimate. If not, the intersection with the original $[\underline{y}_j, \overline{y}_j]$ ensures the right inclusion. Thus, $\mathcal{H} \cap [\underline{y}, \overline{y}] \subseteq [\underline{\mathcal{I}}_\mathcal{H}(\underline{y}, \overline{y}), \overline{\mathcal{I}}_\mathcal{H}(\underline{y}, \overline{y})] \subseteq [\underline{y}, \overline{y}]$. ■

B.4. Proof of Theorem 7 - Polytope invariant sets

Statement: Consider the closed-loop system (2). Let $H \in \mathbb{R}^{n\times m}$, and $H^+ \in \mathbb{R}^{m\times n}$ satisfy $H^+H = \mathbf{I}_n$. Let E_{H,H^+} denote the H, H^+-lifted embedding system (7). If

$$\mathsf{E}_{H,H^+}(\underline{y}, \overline{y}, \underline{w}, \overline{w}) \geq_{\mathrm{SE}} 0,$$

then the polytope $\langle H, \underline{y}, \overline{y}\rangle$ is $[\underline{w}, \overline{w}]$-robustly forward invariant for the original system.

Proof By the equivalence from Proposition 4, it suffices to show that $\mathcal{H} \cap [\underline{y}, \overline{y}]$ is $[\underline{w}, \overline{w}]$-robustly forward invariant for the lifted system. Indeed, assume that $\mathcal{H} \cap [\underline{y}, \overline{y}]$ is a $\mathcal{W}$-robustly forward invariant set for the lifted system (5). Let $t \mapsto \mathbf{w}(t) \in \mathcal{W}$ be any piecewise continuous disturbance curve, and let $y_0 \in \mathcal{H} \cap [\underline{y}, \overline{y}]$. Since $y_0 \in \mathcal{H}$, $y_0 = Hx_0$ for some x_0. Since $\mathcal{H} \cap [\underline{y}, \overline{y}]$ is $\mathcal{W}$-robustly forward invariant, $\phi_g(t, Hx_0, \mathbf{w}) \in \mathcal{H} \cap [\underline{y}, \overline{y}]$ for every $t \geq 0$. By Proposition 4, this implies that $H\phi_f(t, x_0, \mathbf{w}) \in \mathcal{H} \cap [\underline{y}, \overline{y}]$ for every $t \geq 0$. But, since $Hx \in \mathcal{H}$ for any $x \in \mathbb{R}^n$, this is the same as

$$\underline{y} \leq H\phi_f(t, x_0, \mathbf{w}) \leq \overline{y} \iff \phi_f(t, x_0, \mathbf{w}) \in \langle H, \underline{y}, \overline{y}\rangle,$$

for every $t \geq 0$.

It remains to show that $\mathcal{H} \cap [\underline{y}, \overline{y}]$ is indeed $[\underline{w}, \overline{w}]$-robustly forward invariant. Let $[\underline{y}, \overline{y}]$ satisfy that $\mathsf{E}_{H,H^+}(\underline{y}, \overline{y}, \underline{w}, \overline{w}) \geq_{\mathrm{SE}} 0$. Therefore, for every $i = 1, \ldots, m$,

$$0 \leq \underline{\mathsf{G}}_i(\underline{\mathcal{I}}_{\mathcal{H}}(\underline{y}, \overline{y}_{i:\underline{y}}), \overline{\mathcal{I}}_{\mathcal{H}}(\underline{y}, \overline{y}_{i:\underline{y}}), \underline{w}, \overline{w}).$$

Since $\mathcal{I}_{\mathcal{H}}$ is a refinement operator,

$$\mathcal{H} \cap [\underline{y}, \overline{y}_{i:\underline{y}}] \subseteq [\underline{\mathcal{I}}_{\mathcal{H}}(\underline{y}, \overline{y}_{i:\underline{y}}), \overline{\mathcal{I}}_{\mathcal{H}}(\underline{y}, \overline{y}_{i:\underline{y}})],$$

therefore, it follows that

$$0 \leq \underline{\mathsf{G}}_i(\underline{\mathcal{I}}_{\mathcal{H}}(\underline{y}, \overline{y}_{i:\underline{y}}), \overline{\mathcal{I}}_{\mathcal{H}}(\underline{y}, \overline{y}_{i:\underline{y}}), \underline{w}, \overline{w}) \leq \inf_{y\in\mathcal{H}\cap[\underline{y}, \overline{y}_{i:\underline{y}}], w\in[\underline{w}, \overline{w}]} g_i(y, w).$$

This implies that $g_i(y, w) \geq 0$ for every $y \in \mathcal{H} \cap [\underline{y}, \overline{y}_{i:\underline{y}}]$, in other words, the intersection of the lower i-th face of the hyperrectangle $[\underline{y}, \overline{y}]$ and $\mathcal{H}$. Similarly, $g_i(y, w) \leq 0$ for every $y \in \mathcal{H} \cap [\underline{y}_{i:\overline{y}}, \overline{y}]$, in other words, the intersection of the upper i-th face of the hyperrectangle $[\underline{y}, \overline{y}]$ and $\mathcal{H}$. Thus, for every $y \in \mathcal{H} \cap \partial[\underline{y}, \overline{y}] = \partial(\mathcal{H} \cap [\underline{y}, \overline{y}])$, the vector field $g(y, w)$ points into the set $[\underline{y}, \overline{y}]$. But, by Proposition 4, we know that $\mathcal{H}$ is forward invariant, thus, for every $y \in \partial(\mathcal{H} \cap [\underline{y}, \overline{y}])$ the vector field $g(y, w)$ points into the set $\mathcal{H} \cap [\underline{y}, \overline{y}]$. Thus, by Nagumo's theorem Blanchini (1999, Theorem 3.1), the closed set $\mathcal{H} \cap [\underline{y}, \overline{y}]$ is $[\underline{w}, \overline{w}]$-robustly forward invariant. ■

Appendix C. Experiment Implementation Details

C.1. Segway Model

Disturbance Set Partitioning To handle the multiplicative disturbance $(1 + w_k)$ for each system parameter from (9), with $w \in [-0.02, 0.02]^{11}$, we partition the disturbance set into $2^{11} = 2048$ different regions, *i.e.*,

$$\mathbf{W} = \{\mathcal{W}^j = W_1 \times \cdots \times W_{11} : W_i \in \{[-0.02, 0], [0, 0.02]\}\}.$$

One can create a valid embedding system for the whole disturbance set $\mathcal{W} = [-0.02, 0.02]^{11}$ by considering the worst case for each of the disturbance partitions on each output of the embedding system's vector field,

$$\underline{\dot{y}}_i = \left(\underline{\mathsf{G}}(\underline{\mathcal{I}}_{\mathcal{H}}(\underline{y}, \overline{y}_{i:\underline{y}}), \overline{\mathcal{I}}_{\mathcal{H}}(\underline{y}, \overline{y}_{i:\underline{y}}), \underline{w}, \overline{w})\right)_i = \min_{[\underline{w}^j, \overline{w}^j] \in \mathbf{W}} \left(\underline{\mathsf{G}}(\underline{\mathcal{I}}_{\mathcal{H}}(\underline{y}, \overline{y}_{i:\underline{y}}), \overline{\mathcal{I}}_{\mathcal{H}}(\underline{y}, \overline{y}_{i:\underline{y}}), \underline{w}^j, \overline{w}^j)\right)_i,$$

$$\dot{\overline{y}}_i = \left(\overline{\mathsf{G}}(\underline{\mathcal{I}}_{\mathcal{H}}(\underline{y}_{i:\overline{y}}, \overline{y}), \overline{\mathcal{I}}_{\mathcal{H}}(\underline{y}_{i:\overline{y}}, \overline{y}), \underline{w}, \overline{w})\right)_i = \max_{[\underline{w}^j, \overline{w}^j] \in \mathbf{W}} \left(\overline{\mathsf{G}}(\underline{\mathcal{I}}_{\mathcal{H}}(\underline{y}_{i:\overline{y}}, \overline{y}), \overline{\mathcal{I}}_{\mathcal{H}}(\underline{y}_{i:\overline{y}}, \overline{y}), \underline{w}^j, \overline{w}^j)\right)_i.$$

These embedding system and $\min/\max$ evaluations are vectorized using JAX for efficient evaluation on the GPU.

Training Setup

- Network: We train a 2 hidden layer network with 32 neurons each, $3 \times 32 \times 32 \times 1$ with ReLU activations.
- Data Loss: At each step we uniformly sample 1000 points $\{x_k\}$ from the set $[-\frac{\pi}{2}, \frac{\pi}{2}] \times [-5, 5] \times [-2\pi, 2\pi]$. We build

$$\mathcal{L}^{\text{data}}(\pi) = \frac{1}{N} \sum_{k=1}^{N} \|\pi(x_k) - K x_k\|_2^2.$$

- Polytope Loss: We use the loss $\mathcal{L}^{\mathcal{S}}$ (8) with $\lambda = 1000$ and $\varepsilon = 0.1$.
- Optimizer: Algorithm 1 terminates in 595 steps of ADAM with step size 0.001.

C.2. Platoon of Vehicles with Nonlinearities and Disturbances

Training Setup

- Network: We train the shared policy π as a 3 hidden layer network with 32 neurons each, $6 \times 32 \times 32 \times 32 \times 1$ with ReLU activations.
- Data Loss: We use no data loss for this example, *i.e.*, $\mathcal{L}^{\text{data}} = 0$.
- Polytope Loss: We use the loss (8) with $\lambda = 1$ and $\varepsilon = 0.02$.
- Optimizer: Algorithm 1 terminates in $\{724, 725, 807, 890, 1267\}$ steps of ADAM with step size 0.001 for platoons with $\{4, 10, 16, 22, 28\}$ vehicles.