

Certified Robust Invariant Polytope Training in Neural Controlled ODEs

Akash Harapanahalli

*School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA*

AHARAPAN@GATECH.EDU

Samuel Coogan

*School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA*

SAM.COOGAN@GATECH.EDU

Abstract

We consider a nonlinear control system modeled as an ordinary differential equation subject to disturbance, with a state feedback controller parameterized as a feedforward neural network. We propose a framework for training controllers with certified robust forward invariant polytopes, where any trajectory initialized inside the polytope remains within the polytope, regardless of the disturbance. 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., 2023). However, there are few methods for training safe neural network feedback controllers, despite advancements in certified robust training of isolated neural networks.

1.1 Problem Formulation

Consider a continuous-time closed-loop dynamical control system with disturbance given by the ordinary differential equation (ODE)

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

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 $f : \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^q \rightarrow \mathbb{R}^n$ is a locally Lipschitz vector field.

Problem 1 *For the system (1), design a neural network full state feedback control policy $\pi : \mathbb{R}^n \rightarrow \mathbb{R}^p$ such that the closed loop system*

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

renders a given safe set \mathcal{S} robustly forward invariant, i.e., trajectories starting in \mathcal{S} remain in \mathcal{S} for all time.

In control theory, 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). 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.

1.2 Contributions

The contributions of this work are two-fold:

- (i) **Certification:** 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.
- (ii) **Training:** We then propose a novel algorithm 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 GPU¹. 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.

1.3 Literature Review

Control Theoretic Computational Approaches to Invariance There are many classical techniques in the controls literature for computationally certifying invariance specifications, including Lyapunov-based analysis using sum-of-squares programming (Pachristodoulou 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.

Neural Network Verification Robustness of neural networks is a well studied field in the machine learning community. One well-studied approach for certifying the performance of neural networks is to provide an adversarial guarantee, where an overapproximation of the set of possible outputs of the network guarantees the performance of the network under a given set of possible inputs. These approaches include interval bound propagation across each layer given an interval input (Gowal et al., 2018), convex relaxations of each neuron which can be propagated to bound the output between two convex function bounds (Xu et al., 2020), and mixed integer linear programming for exact output characterization with piecewise affine activation functions (Tjeng et al., 2019). 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.

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., 2023, 2024) for nonlinear systems, and ReachLP (Everett et al., 2021) and ReachSDP (Hu et al., 2020) for linear systems. We refer to Lopez et al. (2023) for a comprehensive list of benchmarks and tools the community has been studying. However, to our knowledge, none of these approaches for control systems have been used to help train neural network controllers with safety guarantees.

Certifying Forward Invariance in Neural (Controlled) ODEs An alternative approach to safety analysis is to certify a forward invariant set for the neural controlled dynamics. 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 ap-

1. All code for the experiments is available at <https://github.com/gtfactslab/Polytope-Training>.

proach 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.

2 Mathematical Preliminaries

For the dynamical 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 $\mathbf{w} : [0, \infty) \rightarrow \mathcal{W}$.²

Definition 2 (Robust forward invariance) *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 $\mathbf{w} : [0, \infty) \rightarrow \mathcal{W}$.*

Define the partial ordering \leq on \mathbb{R}^n such that $\underline{x} \leq \bar{x} \iff \underline{x}_i \leq \bar{x}_i$ for every $i = 1, \dots, n$. Let $[\underline{x}, \bar{x}] := \{x \in \mathbb{R}^n : \underline{x} \leq x \leq \bar{x}\}$ denote a closed and bounded interval in \mathbb{R}^n , and let $\mathbb{I}\mathbb{R}^n$ denote the set of all such intervals. Define the upper triangle $\mathcal{T}_{\geq 0}^{2n} := \{[\frac{x}{\bar{x}}] \in \mathbb{R}^{2n} : \underline{x} \leq \bar{x}\}$, and note that $\mathbb{I}\mathbb{R}^n \simeq \mathcal{T}_{\geq 0}^{2n}$. We denote this equivalence with $[[\frac{x}{\bar{x}}]] := [\underline{x}, \bar{x}]$.

Definition 3 (Inclusion function) *Given a function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, the function $\mathbf{F} = \begin{bmatrix} \underline{\mathbf{F}} \\ \bar{\mathbf{F}} \end{bmatrix} : \mathcal{T}_{\geq 0}^{2n} \rightarrow \mathcal{T}_{\geq 0}^{2m}$ is called an inclusion function for f if for every $x \in [\underline{x}, \bar{x}]$,*

$$\mathbf{F}(\underline{x}, \bar{x}) \leq f(x) \leq \bar{\mathbf{F}}(\underline{x}, \bar{x}).$$

Notationally, we use the upper triangular interpretation $\mathcal{T}_{\geq 0}^{2n}$ for convenience when constructing the embedding system in Sections 3 and 4. Most references equivalently define inclusion functions as mappings on $\mathbb{I}\mathbb{R}^n$. For $[\underline{a}, \bar{a}], [\underline{b}, \bar{b}] \in \mathbb{I}\mathbb{R}$ and $[\underline{A}, \bar{A}] \in \mathbb{I}\mathbb{R}^{m \times p}$, $[\underline{B}, \bar{B}] \in \mathbb{I}\mathbb{R}^{p \times n}$, define the operations

- $[\underline{a}, \bar{a}] + [\underline{b}, \bar{b}] = [\underline{a} + \underline{b}, \bar{a} + \bar{b}]$ (also on $\mathbb{I}\mathbb{R}^n$ element-wise);
- $[\underline{a}, \bar{a}] \cdot [\underline{b}, \bar{b}] = [\min\{\underline{ab}, \underline{a}\bar{b}, \bar{a}\underline{b}, \bar{a}\bar{b}\}, \max\{\underline{ab}, \underline{a}\bar{b}, \bar{a}\underline{b}, \bar{a}\bar{b}\}]$;
- $([\underline{A}, \bar{A}][\underline{B}, \bar{B}])_{i,j} = \sum_{k=1}^p [\underline{A}_{i,k}, \bar{A}_{i,k}] \cdot [\underline{B}_{k,j}, \bar{B}_{k,j}]$.

In this work, we use a two-sided representation of a (bounded) convex polytope, for convenience in our embedding system formulation.

Definition 4 (Convex polytope) *We represent a convex polytope as a nonempty set $\langle H, \underline{y}, \bar{y} \rangle := \{x \in \mathbb{R}^n : \underline{y} \leq Hx \leq \bar{y}\}$, for a full rank matrix $H \in \mathbb{R}^{m \times n}$. Under these assumptions, convex polytopes are bounded, and in general any bounded traditional H -rep convex polytope $\{x \in \mathbb{R}^n : Hx \leq y\}$ may be written in this form by taking sufficiently small \underline{y} .*

The partial order \leq on \mathbb{R}^n induces the southeast partial order \leq_{SE} on \mathbb{R}^{2n} , where $[\frac{x}{\bar{x}}] \leq_{\text{SE}} [\frac{y}{\bar{y}}] \iff x \leq y$ and $\hat{y} \leq \hat{x}$. For $x^1 \in \mathbb{R}^{n_1}$, $x^2 \in \mathbb{R}^{n_2}$, \dots , $x^m \in \mathbb{R}^{n_m}$, let $(x^1, x^2, \dots, x^m) \in \mathbb{R}^{n_1+n_2+\dots+n_m}$ denote their concatenation. For two vectors $x, y \in \mathbb{R}^n$ and $i \in \{1, \dots, n\}$, let $x_{i:y} \in \mathbb{R}^n$ be the vector obtained by replacing the i th entry of x with that of y , i.e., $(x_{i:y})_j = y_j$ if $i = j$ and otherwise $(x_{i:y})_j = x_j$. For $n \in \mathbb{N}$, let $\mathbf{I}_n \in \mathbb{R}^{n \times n}$ denote the identity matrix.

2. We assume local Lipschitz f and piecewise continuous \mathbf{w} to guarantee existence and uniqueness of this trajectory.

3 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.

3.1 Closed-Loop Inclusion Function

We assume bounds of the following forms for the neural network and the open-loop dynamics, which we obtain in practice using existing computational tools.

Assumption 5 (Local affine bound of neural network) *Given a neural network $\pi : \mathbb{R}^n \rightarrow \mathbb{R}^p$, there exists an algorithm generating the tuple $(\underline{C}_{[\underline{x}, \bar{x}]}, \overline{C}_{[\underline{x}, \bar{x}]}, \underline{d}_{[\underline{x}, \bar{x}]}, \overline{d}_{[\underline{x}, \bar{x}]})$ where for every $x \in [\underline{x}, \bar{x}]$,*

$$\underline{C}_{[\underline{x}, \bar{x}]}x + \underline{d}_{[\underline{x}, \bar{x}]} \leq \pi(x) \leq \overline{C}_{[\underline{x}, \bar{x}]}x + \overline{d}_{[\underline{x}, \bar{x}]}.$$

In the input-output neural network verification literature, there are many algorithms that can obtain bounds of this form, including CROWN (Zhang et al., 2018), FastLin (Weng et al., 2018), and even Lipschitz constant-based approaches such as LipSDP (Fazlyab et al., 2019). In this paper, we focus on CROWN, a lightweight linear bound propagator, and its implementation in `jax_verify`.

Assumption 6 (Mixed Jacobian-based bound of open-loop dynamics) *For the open-loop dynamics f in (1), given centering points and intervals $\hat{x} \in [\underline{x}, \bar{x}] \in \mathbb{IR}^n$, $\hat{u} \in [\underline{u}, \bar{u}] \in \mathbb{IR}^p$, $\hat{w} \in [\underline{w}, \bar{w}] \in \mathbb{IR}^q$, there exists interval mappings $\mathbf{M}_x, \mathbf{M}_u, \mathbf{M}_w$ such that*

$$f(x, u, w) \in [\mathbf{M}_x(\underline{x}, \bar{x}, \underline{u}, \bar{u}, \underline{w}, \bar{w})](x - \hat{x}) + [\mathbf{M}_u(\underline{x}, \bar{x}, \underline{u}, \bar{u}, \underline{w}, \bar{w})](u - \hat{u}) \\ + [\mathbf{M}_w(\underline{x}, \bar{x}, \underline{u}, \bar{u}, \underline{w}, \bar{w})](w - \hat{w}),$$

for any $x \in [\underline{x}, \bar{x}]$, $u \in [\underline{u}, \bar{u}]$, $w \in [\underline{w}, \bar{w}]$.

This is not a restrictive assumption in general, and the toolbox `immrax` (Harapanahalli et al., 2024) constructs $\mathbf{M}_x, \mathbf{M}_u, \mathbf{M}_w$ from f using automatic differentiation and interval arithmetic techniques in JAX. There are different techniques for obtaining these mappings. For example, using an inclusion function for the Jacobian matrices, the right hand side is a direct result of the mean value theorem. In practice, one can obtain better bounds by considering a componentwise usage of the mean value theorem, and for a more detailed discussion we refer to Jaulin et al. (2001, Section 2.4.4) and Harapanahalli et al. (2024, Proposition 7).

Using bounds obtained from these two tools, given the open-loop system (1) and a neural network controller π , we next build a closed-loop mixed Jacobian-based inclusion function as

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

where $(A^+)_{ij} = \max\{A_{ij}, 0\}$ and $A^- = A - A^+$ for any matrix A , $(\underline{C}_{[\underline{x}, \bar{x}]}, \bar{C}_{[\underline{x}, \bar{x}]}, \underline{d}_{[\underline{x}, \bar{x}]}, \bar{d}_{[\underline{x}, \bar{x}]})$ is from Assumption 5, $[\frac{u}{\bar{w}}] := \left[\frac{C^+}{\bar{C}^-} \frac{C^-}{\bar{C}^+} \right] [\frac{x}{\bar{x}}] + \left[\frac{d}{\bar{d}} \right]$, and $\mathbf{M}_s := \mathbf{M}_s(\underline{x}, \bar{x}, \underline{u}, \bar{u}, \underline{w}, \bar{w})$ for $\mathbf{s} \in \{x, u, w\}$ is as defined in Assumption 6 for $(\dot{x}, \dot{u}, \dot{w}) = (\underline{x}, \underline{u}, \underline{w})$. In general, this approach works for any choice of $(\dot{x}, \dot{u}, \dot{w})$ equal to a corner of the box $[\underline{x}, \bar{x}] \times [\underline{u}, \bar{u}] \times [\underline{w}, \bar{w}]$, with slight modifications to the expression (3). Jafarpour et al. (2024, Theorem 3) proves that (3) is indeed a valid inclusion function for the closed-loop vector field f^π (2). 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 $[\frac{x}{\bar{x}}]$.

3.2 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). This inclusion function builds an embedding system, whose evaluation at a single point yields a sufficient condition for robust forward invariant hyperrectangles.

Definition 7 (Embedding system) *Consider the neural network controlled system (2). The closed-loop inclusion function (3) induces the closed-loop embedding system*

$$\begin{aligned} \dot{x}_i &= (\underline{\mathbf{E}}(\underline{x}, \bar{x}, \underline{w}, \bar{w}))_i := (\underline{\mathbf{F}}^\pi(\underline{x}, \bar{x}_{i:\underline{x}}, \underline{w}, \bar{w}))_i, \\ \dot{\bar{x}}_i &= (\bar{\mathbf{E}}(\underline{x}, \bar{x}, \underline{w}, \bar{w}))_i := (\bar{\mathbf{F}}^\pi(\underline{x}_{i:\bar{x}}, \bar{x}, \underline{w}, \bar{w}))_i, \end{aligned} \quad (4)$$

where $[\frac{x}{\bar{x}}] \in \mathcal{T}_{\geq 0}^{2n}$, $[\frac{w}{\bar{w}}] \in \mathcal{T}_{\geq 0}^{2q}$, and $\mathbf{E} : \mathcal{T}_{\geq 0}^{2n} \times \mathcal{T}_{\geq 0}^{2q} \rightarrow \mathbb{R}^{2n}$.

Consider any point $[\frac{x_0}{\bar{x}_0}] \in \mathcal{T}_{\geq 0}^{2n}$, and the trajectory $t \mapsto \left[\frac{x(t)}{\bar{x}(t)} \right]$ of (4) from this initial condition. The equivalence $\mathcal{T}_{\geq 0}^{2n} \simeq \mathbb{I}\mathbb{R}^n$ associates each point $\left[\frac{x(t)}{\bar{x}(t)} \right]$ along the trajectory of the embedding system with an interval $[\underline{x}(t), \bar{x}(t)] \subset \mathbb{R}^n$ in the state space. The trajectory of the original system starting from any initial condition in $[\underline{x}_0, \bar{x}_0]$ and with any disturbance mapping is guaranteed to be contained in these intervals for every $t \geq 0$, i.e., $\phi_{f^\pi}(t, x_0, \mathbf{w}) \in [\underline{x}(t), \bar{x}(t)]$ for every $x_0 \in [\underline{x}_0, \bar{x}_0]$ and any disturbance mapping $\mathbf{w} : [0, \infty) \rightarrow [\underline{w}, \bar{w}]$ (Jafarpour et al., 2023, Proposition 5). Thus, 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.

This fact that the embedding system evaluates the inclusion function separately on each face of the hyperrectangle is a key feature of our approach—in (4), the faces of the hyperrectangle $[\underline{x}, \bar{x}]$ are represented by $[\underline{x}, \bar{x}_{i:\underline{x}}]$ and $[\underline{x}_{i:\bar{x}}, \bar{x}]$ for each $i \in \{1, \dots, n\}$. As a result, both the neural network verification step from Assumption 5 and the mixed Jacobian-based bounding step from Assumption 6 are computed separately for each face. 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, Proposition 1).

Proposition 8 (Invariant hyperrectangles) *Consider the closed-loop system (2), with the inclusion function \mathbf{F}^π from (3), and the induced embedding system \mathbf{E} from (4). If*

$$\mathbf{E}(\underline{x}_0, \bar{x}_0, \underline{w}, \bar{w}) \geq_{\text{SE}} 0,$$

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

Recall that $E(x_0, \bar{x}_0, \underline{w}, \bar{w}) \geq_{SE} 0$ if and only if $\underline{E}(x_0, \bar{x}_0, \underline{w}, \bar{w}) \geq 0$ and $\bar{E}(x_0, \bar{x}_0, \underline{w}, \bar{w}) \leq 0$. Proposition 8 characterizes a boundary condition on the hyperrectangle $[\underline{x}, \bar{x}]$ based on the Nagumo theorem (Blanchini, 1999, Theorem 3.1). In particular, since the embedding system bounds each face of the hyperrectangle separately, its positivity with respect to the southeast order is equivalent to the vector field pointing inside of the box along the entire boundary $\partial[\underline{x}, \bar{x}]$, thus, trajectories can never escape the box. The condition from Proposition 8 is simple and suitable for repeated evaluation during a training procedure and for the design of an objective function in Section 5. In particular, it does not require sampling along the boundary of the set like previous approaches (Huang et al., 2023).

4 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. The following Example illustrates this principle in greater detail.

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

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

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 system's vector field at this point is $f([1 \ 1]^T) = [1 \ u]^T$. Regardless of the control applied, $\dot{x}_1 > 0$, and thus 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 $T = \begin{bmatrix} 1 & 1 \\ -1 & -2 \end{bmatrix}$. In transformed coordinates $y := T^{-1}x$,

$$\dot{y} = T^{-1}(A + BK)Ty = \begin{bmatrix} -1 & 0 \\ 0 & -2 \end{bmatrix} y.$$

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.

The phenomenon illustrated in Example 1 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. To allow for richer set geometries while retaining computational tractability, we extend the theory from the previous sections by introducing a parameterized family of lifted systems in a higher dimensional space where hyperrectangles in the larger space directly correspond to polytopes for the original system.

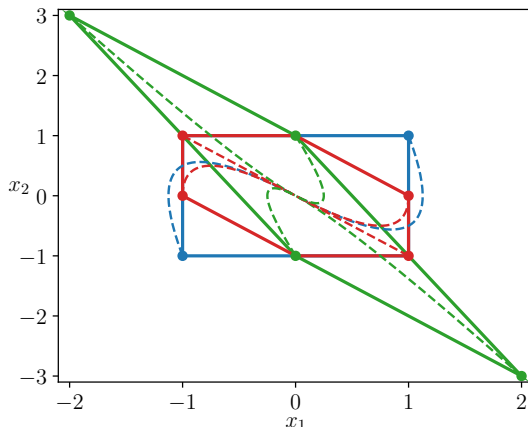


Figure 1: The mechanical system from Examples 1 and 2 is visualized with several important polytopes in solid lines and solution trajectories in dotted lines. The blue set \mathcal{S}_1 is an example of a hyperrectangle that cannot be forward invariant, the green invariant set \mathcal{S}_2 is obtained 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.

4.1 Lifted System

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

Definition 9 (Lifted system) Consider the closed-loop system (2), and let $H \in \mathbb{R}^{m \times n}$ be a matrix with full rank. Let $H^+ \in \mathbb{R}^{n \times m}$ be any matrix satisfying $H^+H = \mathbf{I}_n$. With $y := Hx$,

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

is the (H, H^+) -lifted system of (2).

In the following 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 10 (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 $H^\dagger = (H^T H)^{-1} H^T$ be the Moore-Penrose Pseudoinverse of H . Then the set

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

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

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 , which we show in the next Proposition.

Proposition 11 (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 $\mathbf{w} : [0, \infty) \rightarrow \mathcal{W}$,

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

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

Algorithm 1 Subspace $\mathcal{I}_{\mathcal{H}}$ implementation

Input: Interval $[y, \bar{y}] \in \mathbb{IR}^m$, Subspace $\mathcal{H} = \text{col}(H)$, $H \in \mathbb{R}^{m \times n}$ full rank
 $N \leftarrow \text{null}(H^T) \in \mathbb{R}^{m \times (m-n)}$
 $A \leftarrow N^T \in \mathbb{R}^{(m-n) \times m}$
 $[z, \bar{z}] \leftarrow [y, \bar{y}]$
for $i = 1, \dots, m - n$ **do**
 for $j = 1, \dots, m$ **do**
 $[z_i, \bar{z}_i] \leftarrow \left(-\frac{1}{A_{i,j}} \sum_{k \neq i} A_{i,k} [z_k, \bar{z}_k] \right) \cap [z_i, \bar{z}_i]$
 end for
end for
Output: $[z, \bar{z}]$ with $\mathcal{H} \cap [y, \bar{y}] \subseteq [z, \bar{z}] \subseteq [y, \bar{y}]$.

The invariance of \mathcal{H} is crucial for building a good embedding system for the lifted system (5), which we demonstrate in the next section.

4.2 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 11. 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 12 (Interval refinement operator) Let $\mathcal{H} \subset \mathbb{R}^m$ be a subset. $\mathcal{I}_{\mathcal{H}} : \mathcal{T}_{\geq 0}^{2m} \rightarrow \mathcal{T}_{\geq 0}^{2m}$ is an interval refinement operator on \mathcal{H} if for every $[y, \bar{y}] \in \mathbb{IR}^m$,

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

Algorithm 1 provides an implementation of $\mathcal{I}_{\mathcal{H}}$ for the case where $\mathcal{H} = \{Hx : x \in \mathbb{R}^n\}$ is a subspace, using the fact that the left null space of H encodes $(m - n)$ constraints on \mathbb{R}^m which yield the same subspace \mathcal{H} . Once the matrix $A = N^T$ is obtained, where the columns of N are a basis for the left null space of H , the algorithm 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 . We provide a full proof that Algorithm 1 is indeed a valid refinement operator on \mathcal{H} in Appendix A.3.

Given a \mathcal{H} -refinement operator $\mathcal{I}_{\mathcal{H}}$ and an inclusion function \mathbb{G} for the lifted closed-loop dynamics (5), a (H, H^+) -lifted embedding system can be constructed by applying $\mathcal{I}_{\mathcal{H}}$ on each face of the lifted hyperrectangle before evaluating the inclusion function,

$$\begin{aligned} \dot{y}_i &= (\mathbb{G}(\mathcal{I}_{\mathcal{H}}(y, \bar{y}_{i:y}), \bar{\mathcal{I}}_{\mathcal{H}}(y, \bar{y}_{i:y}), \underline{w}, \bar{w}))_i, \\ \dot{\bar{y}}_i &= (\bar{\mathbb{G}}(\mathcal{I}_{\mathcal{H}}(\underline{y}_{i:\bar{y}}), \bar{\mathcal{I}}_{\mathcal{H}}(\underline{y}_{i:\bar{y}}, \bar{y}), \underline{w}, \bar{w}))_i. \end{aligned} \tag{6}$$

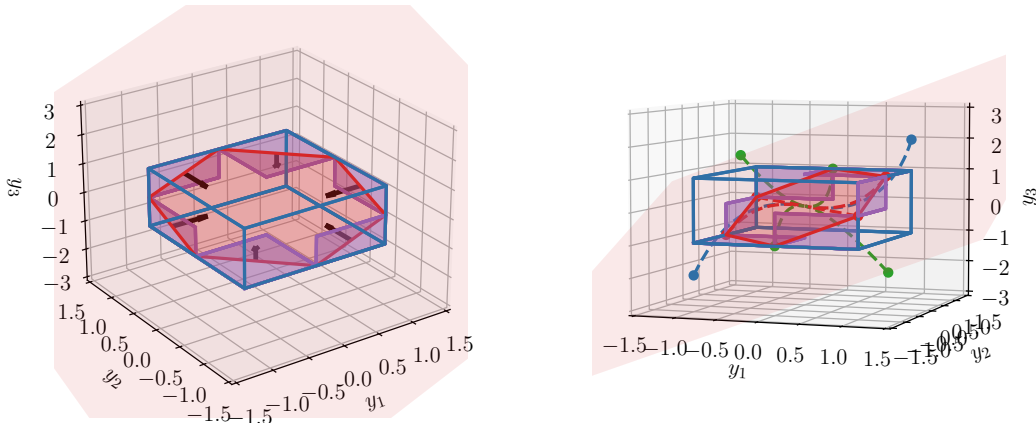


Figure 2: $\mathcal{I}_{\mathcal{H}}$ is applied to every face of the box $[\underline{y}, \bar{y}] = [-1, 1]^3$ (blue) for the subspace \mathcal{H} from Example 2 (red), *i.e.*, normal to $N = [1 \ 1 \ -1]^T \in \mathbb{R}^3$. The outputs (purple) are refined interval sets which still contain \mathcal{H} . The red outlined set, which is the intersection $\mathcal{H} \cap [\underline{y}, \bar{y}]$, corresponds directly to the polytope $\langle H, \underline{y}, \bar{y} \rangle$ from Example 2 and the red polytope from Figure 1. The original system evolves on the subspace \mathcal{H} by Proposition 11. The southeast positivity condition $\mathbf{E}_{H, H^+}(\underline{y}, \bar{y}, \underline{w}, \bar{w}) \geq_{\text{SE}} 0$ from Theorem 13 bounds the i -th component of the vector field to point in the direction indicated by the black arrows (**left**) along each refined face. These two conditions together guarantee that as the system approaches any edge of the set $\mathcal{H} \cap [\underline{y}, \bar{y}]$, the vector field must push the system back into the set $\mathcal{H} \cap [\underline{y}, \bar{y}]$. The sample trajectories from Figure 1 are shown in dotted lines (**right**) for the lifted system.

Due to the refinement operator $\mathcal{I}_{\mathcal{H}}$, each face of the $[\underline{y}, \bar{y}]$ hyperrectangle is refined to a smaller interval using the knowledge that the original dynamics remain on the subspace \mathcal{H} . Figure 2 demonstrates this procedure for the 6 faces of a 3 dimensional hyperrectangle intersecting a 2 dimensional subspace from Example 2 below.

Theorem 13 (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 \mathbf{E}_{H, H^+} denote the (H, H^+) -lifted embedding system (6). If*

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

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

Remark 14 (Comparison to the literature) *Theorem 13 generalizes Harapanahalli et al. (2023, Theorem 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 simple, evaluable, and trainable condition for checking the forward invariance of a polytope. In the next Example, we return to the double integrator structure to demonstrate how this condition can be used to certify invariant polytopes.

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

$$\mathbf{E}_{H,H^\dagger}(\underline{y}, \bar{y}) = \left(0, 1, \frac{4}{3}, 0, -1, -\frac{4}{3} \right) \geq_{\text{SE}} 0,$$

which implies that the polytope $\mathcal{S}_3 = \langle H, \underline{y}, \bar{y} \rangle$ is a forward invariant set for the original system. The forward invariant polytope \mathcal{S}_3 is visualized in Figure 1 in green. Additionally, the box $[\underline{y}, \bar{y}]$, subspace $\mathcal{H} = \{Hx : x \in \mathbb{R}^2\}$, intersection $\mathcal{H} \cap [\underline{y}, \bar{y}]$, and outputs of $\mathcal{I}_{\mathcal{H}}(\underline{y}, \bar{y}_{i:\underline{y}})$ and $\mathcal{I}_{\mathcal{H}}(\bar{y}_{i:\bar{y}}, \bar{y})$ are all visualized in Figure 2. The black arrows show the direction (+/-) of the embedding system's vector field along those components, which demonstrates how the polytope is certified to be forward invariant using Theorem 13.

5 Certified Polytope Invariance Training

Using the lifted embedding system, we propose a framework for training certified forward invariant polytopes in neural network controlled ODEs. First, we assume there already exists an original loss $\mathcal{L}^{\text{data}}$. Ideally, we would add the positivity condition from Theorem 13 as a hard constraint, however, the complexity of neural networks often necessitates the use of unconstrained optimization algorithms. Instead, we relax the constraint in favor of regularizing loss terms, with the hope that the algorithm will eventually tend towards network parameters that evaluate as $\geq_{\text{SE}} 0$. For a desired polytope $\mathcal{S} = \langle H, \underline{y}, \bar{y} \rangle$, we use the loss

$$\mathcal{L}^{\mathcal{S}}(\pi, \eta) = \sum_{i=1}^m \text{ReLU}(\bar{\mathbf{E}}_{H,H_\eta^+}(\underline{y}, \bar{y}, \underline{w}, \bar{w})_i + \varepsilon) + \sum_{i=1}^m \text{ReLU}(-\underline{\mathbf{E}}_{H,H_\eta^+}(\underline{y}, \bar{y}, \underline{w}, \bar{w})_i + \varepsilon) \quad (7)$$

to induce $[\underline{w}, \bar{w}]$ -robust forward invariance, where \mathbf{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 10, 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 $\mathbf{E}_{H,H_\eta^+} \not\geq_{\text{SE}} 0$, the loss (7) appears again. As a result, large values of λ work well in practice.

Remark 15 (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.

Algorithm 2 implements these losses into a training procedure, and we incorporate the condition from Theorem 13 directly into the training loop.

Algorithm 2 Certified polytope training using the lifted embedding system

Input: Desired invariant polytope $\langle H, \underline{y}, \bar{y} \rangle = \{x \in \mathbb{R}^n : \underline{y} \leq Hx \leq \bar{y}\}$, $H \in \mathbb{R}^{m \times n}$ full rank, $\underline{y}, \bar{y} \in \mathbb{R}^m$, regularization constant $\lambda \geq 0$, disturbance set $[\underline{w}, \bar{w}]$, data loss $\mathcal{L}^{\text{data}}$
 $N \leftarrow \text{null}(H^T)$
 $\eta \leftarrow \mathbf{0}^{(m-n) \times m}$
repeat
 $H_\eta^+ \leftarrow H^+ + \eta N^T$
 $E \leftarrow (H, H_\eta^+)$ -Lifted Embedding System (6) for (2)
 $\mathcal{L} \leftarrow \mathcal{L}^{\text{data}}(\pi) + \lambda \mathcal{L}^S(\pi, \eta)$
 $(\pi, \eta) \leftarrow \text{step_optimizer}(\nabla_{(\pi, \eta)} \mathcal{L})$
until $E(\underline{y}, \bar{y}, \underline{w}, \bar{w}) \geq_{\text{SE}} 0$
Return: Neural network controller π with certified forward invariant polytope $\langle H, \underline{y}, \bar{y} \rangle$ for (2).

6 Experiments

³ We use `jax_verify` (JAX implementation of verification techniques like CROWN (Zhang et al., 2018)) to compute the affine bounds required in Assumption 5, and `immrax` (Harapanahalli et al., 2024) (JAX implementation of interval analysis) to compute the interval matrices from Assumption 6. JAX (Bradbury et al., 2018) vectorizes the evaluations on each face of the lifted hyperrectangle from Theorem 13 onto the GPU, and Equinox (Kidger and Garcia, 2021) helps autodifferentiate for gradient evaluations in Algorithm 2.

6.1 Segway Model

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

$$\begin{bmatrix} \dot{\phi} \\ \dot{v} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ \frac{\cos \phi(-1.8u+11.5v+9.8 \sin \phi)-10.9u+68.4v-1.2\dot{\phi}^2 \sin \phi}{\cos \phi-24.7} \\ \frac{(9.3u-58.8v) \cos \phi+38.6u-234.5v-\sin \phi(208.3+\dot{\phi}^2 \cos \phi)}{\cos^2 \phi-24.7} \end{bmatrix} \quad (8)$$

with state $x = [\phi \ v \ \dot{\phi}]^T \in \mathbb{R}^3$. To compare with the literature, we use a similar training procedure to that from Huang et al. (2023), where the network is trained to mimic the LQR gains from the linearization of the system around the origin, while certifying a robust forward invariant region around the equilibrium. The robustness is with respect to a $\pm 2\%$ uncertainty in each of the system parameters—this is represented as a multiplicative disturbance $(1 + w_k)$ for $w \in [-0.02, 0.02]$ ¹¹ applied independently to each system parameter from (8).

One approach to define a suitable polytope is to attempt to diagonalize the system, as demonstrated in Example 1 where a diagonalizing transformation easily yielded forward invariance since all eigenvalues were negative. This intuition extends to the nonlinear neural network controlled case: (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

3. All experiments were performed on a computer running Kubuntu 22.04 with Intel Xeon Gold 6230, NVIDIA Quadro RTX 8000, and 64 GB of RAM.

4. The time reported here includes both the training step (139 s) and the verification step (2619 s).

Table 1: Comparison to robust invariance training literature

| Method | Volume | Setup (JIT) | Runtime (s) | |
|-------------|--------------|-------------|-------------------|--------------|
| | | | Verified Training | Total |
| Ours | 0.00152 | 139. | 11.4 | 150.4 |
| FI-ODE | 0.158 | – | 2758 ⁴ | 2758 |

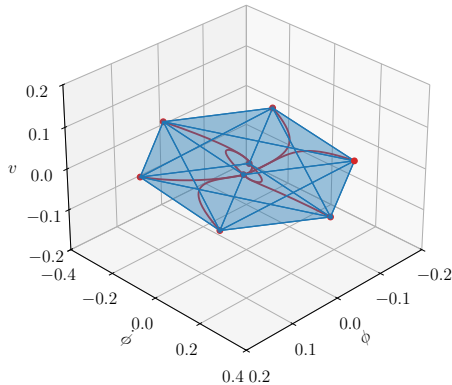


Figure 3: The certified robust forward invariant polytope in \mathbb{R}^3 for the segway model is visualized in blue, with simulations of real trajectories starting from each vertex of the polytope in red.

of this matrix be $T^{-1}A_{cl}T = \Lambda$, where Λ is in Jordan form; (iii) choose the matrix $H = T^{-1}$, and consider some offsets, *e.g.*, $[y, \bar{y}] = [-2, 2]^3$.

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 13 allows us to verify 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, the process to train a robust neural network is quick—it takes 11.4 seconds to train a certified forward invariant set for the segway, 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 3.

In our approach, a simple positivity check certifies robust forward invariance at each iteration of the training optimizer. However, the cost of this simple condition is possible overconservatism, demonstrated in our small volume compared with (Huang et al., 2023). In contrast, FI-ODE trains the neural network first without any guarantees by incorporating the Lyapunov function into the training process. Then, a post-training sampling-based robust verification step verifies that a sublevel set of the Lyapunov function is robustly forward invariant. Another difference is that FI-ODE incorporates the positive definite P matrix from their Lyapunov function into the initial training procedure, which allows them to shape the invariant set during training. In contrast, in this work we do not allow

Table 2: Scalability with respect to number of vehicles

| N | # States | | Runtime (s) | |
|-----|----------|--------|-------------|------------------|
| | Original | Lifted | Setup | 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) |

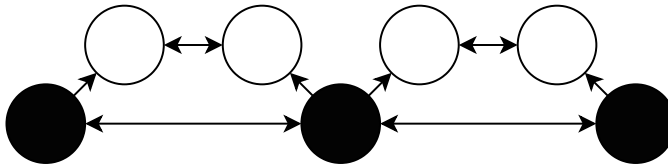


Figure 4: The connection graph for the vehicle platooning example. The leaders (filled units) measure their absolute positions, and arrows represent relative measurements.

any modification or shaping of the polytopic sets in the optimization, which is a possible direction for future work that may decrease conservatism.

6.2 Platoon of Vehicles with Nonlinearities and Disturbances

In this example, we examine how our proposed approach scales with state dimension n . We consider a platoon of N vehicles, each with the following dynamics,

$$\dot{p}_j = v_j, \quad \dot{v}_j = \sigma(u_j)(1 + w_j), \quad (9)$$

where for each vehicle $j = 1, \dots, N$, $p_j \in \mathbb{R}$ 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 $x_j = (p_j, v_j) \in \mathbb{R}^2$ for each j . Each vehicle is controlled by a shared neural network control policy $\pi : \mathbb{R}^6 \rightarrow \mathbb{R}$,

$$u_j = \begin{cases} \pi((x_j, x_{j-3} - x_j, x_j - x_{j+3})), & j = 3k, k \in \mathbb{N} \\ \pi((0, x_{j-1} - x_j, x_j - x_{j+1})), & \text{otherwise,} \end{cases} \quad (10)$$

with $x_0 := 0, x_{N+1} := 0$. Every 3rd vehicle (leader) measures its true displacement from the origin, as well as the distances between the next two leaders. The rest of the platoon (followers) measures relative displacements between their nearest two neighbors. This communication scheme is pictured in Figure 4. We represent the closed-loop system as,

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \Pi(\mathbf{u}), \mathbf{w}) = \mathbf{f}^\Pi(\mathbf{x}, \mathbf{w}), \quad (11)$$

where $\mathbf{x} = (x_1, \dots, x_N) \in \mathbb{R}^{2N}$, $\mathbf{u} = (u_1, \dots, u_N) \in \mathbb{R}^N$, $\mathbf{w} = (w_1, \dots, w_N) \in \mathbb{R}^N$, $\mathbf{f} : \mathbb{R}^{2N} \times \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}^{2N}$ is the dynamics (9), $\Pi : \mathbb{R}^N \rightarrow \mathbb{R}^N$ represents the policy (10).

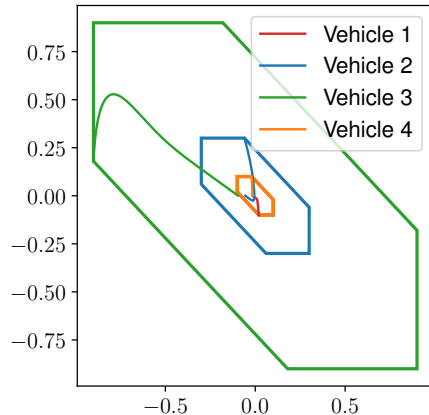


Figure 5: The invariant polytope and a sample system trajectory for the platoon with 4 vehicles is pictured. Vehicles 1 and 4 share the same invariant set in orange.

We would like to train the shared feedback policy π such that the closed-loop platoon (11) renders the polytope $\langle H, \underline{y}, \bar{y} \rangle$ robustly forward invariant, for $H = \mathbf{I}_N \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$, $\bar{y} = \underbrace{(1, 3, 9, 1, 3, 9, \dots, 1, 3, 9, 1)}_{\in \mathbb{R}^N} \otimes (0.1, 0.1, 0.08)$, $\underline{y} = -\bar{y}$.⁵ We use Algorithm 2 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 $\langle H, \underline{y}, \bar{y} \rangle$ is illustrated in Figure 5. In Table 2, we outline: (i) the number of vehicles N , state dimension of the original system, and state dimension of the lifted system; (ii) the time to just-in-time (JIT) compile the iteration loop; (iii) the time to train the neural network controller with the certified robust forward invariant polytope $\langle H, \underline{y}, \bar{y} \rangle$, and how many iterations of ADAM with step size 0.001 were used. Compared to sampling based approaches, where the complexity grows exponentially, our approach scales reasonably in both setup and training time.

7 Conclusions

In this paper, we developed a framework for training certified robust forward invariant polytopes in neural network controlled dynamical systems. We introduced the novel lifted embedding system, which provides a sufficient condition for robust forward invariance through a single pointwise evaluation of the positivity of a vector field in a higher dimensional space. All computations were efficiently dispatched to a GPU using JAX, and the proposed positivity condition was implemented into a training algorithm using automatic differentiation. Any policy returned by the training procedure is automatically certified with the invariant set, which is desirable compared to previous approaches requiring a sampling-based post-training verification step. Through numerical experiments, we demonstrated how our approach

5. \otimes is the Kronecker product.

both improves upon these sampling-based approaches in runtime, and scales well to high dimensional systems.

Appendix A. Proofs of Main Results

A.1 Proof of Proposition 10 - 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

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

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

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

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 $x \in \mathbb{R}^n$,

$$H^+ H x = x \iff H^\dagger H x + (H^+ - H^\dagger) H x = x \iff (H^+ - H^\dagger) H x = 0, \quad (14)$$

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 $j = 1, \dots, n$, there exists $\eta_j \in \mathbb{R}^{(m-n)}$ such that $N \eta_j = h_j$. Finally, with $\eta = [\eta_1 \cdots \eta_n]^T$,

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

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

A.2 Proof of Proposition 11 - 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 $\mathbf{w} : [0, \infty) \rightarrow \mathcal{W}$,

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

Moreover, the linear subspace $\mathcal{H} := \{H x : x \in \mathbb{R}^n\}$ 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 $H x_0$ under \mathbf{w} , *i.e.*, $\phi_g(t, H x_0, \mathbf{w})$.

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

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

which is the same dynamics as the (H, H^+) -lifted system (5). Additionally, note that $\hat{y}(0) = H x(0) = H x_0$. 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). \blacksquare

A.3 Proof of Algorithm 1 - $\mathcal{I}_{\mathcal{H}}$ implementation

Need to show: $\mathcal{I}_{\mathcal{H}}$ from Algorithm 1 satisfies

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

Proof Let $H \in \mathbb{R}^{m \times n}$ be a full rank matrix, and let N be a basis for its left nullspace. Thus, $N^T H = 0$, which implies that $N^T Hx = 0$ for any $x \in \mathbb{R}^n$. Let $A = N^T \in \mathbb{R}^{(m-n) \times m}$. Thus, for any $y \in \mathcal{H} = \{Hx : x \in \mathbb{R}^n\}$,

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

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

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

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

A.4 Proof of Theorem 13 - 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 \mathbf{E}_{H,H^+} denote the H, H^+ -lifted embedding system (6). If

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

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

Proof By the equivalence from Proposition 11, it suffices to show that $\mathcal{H} \cap [\underline{y}, \bar{y}]$ is $[\underline{w}, \bar{w}]$ -robustly forward invariant for the lifted system. Indeed, assume that $\mathcal{H} \cap [\underline{y}, \bar{y}]$ 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 $y_0 \in \mathcal{H} \cap [\underline{y}, \bar{y}]$. Since $y_0 \in \mathcal{H}$, $y_0 = Hx_0$ for some x_0 . Since $\mathcal{H} \cap [\underline{y}, \bar{y}]$ is \mathcal{W} -robustly forward invariant, $\phi_g(t, Hx_0, \mathbf{w}) \in \mathcal{H} \cap [\underline{y}, \bar{y}]$ for

every $t \geq 0$. By Proposition 11, this implies that $H\phi_f(t, x_0, \mathbf{w}) \in \mathcal{H} \cap [\underline{y}, \bar{y}]$ for every $t \geq 0$. But, since $Hx \in \mathcal{H}$ for any $x \in \mathbb{R}^n$, this is the same as

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

for every $t \geq 0$.

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

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

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

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

therefore, it follows that

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

This implies that $g_i(y, w) \geq 0$ for every $y \in \mathcal{H} \cap [\underline{y}, \bar{y}_{i:\underline{y}}]$, in other words, the intersection of the lower i -th face of the hyperrectangle $[\underline{y}, \bar{y}]$ and \mathcal{H} . Similarly, $g_i(y, w) \leq 0$ for every $y \in \mathcal{H} \cap [\underline{y}_{i:\bar{y}}, \bar{y}]$, in other words, the intersection of the upper i -th face of the hyperrectangle $[\underline{y}, \bar{y}]$ and \mathcal{H} . Thus, for every $y \in \mathcal{H} \cap \partial[\underline{y}, \bar{y}] = \partial(\mathcal{H} \cap [\underline{y}, \bar{y}])$, the vector field $g(y, w)$ points into the set $[\underline{y}, \bar{y}]$. But, by Proposition 11, we know that \mathcal{H} is forward invariant, thus, for every $y \in \partial(\mathcal{H} \cap [\underline{y}, \bar{y}])$ the vector field $g(y, w)$ points into the set $\mathcal{H} \cap [\underline{y}, \bar{y}]$. Thus, by Nagumo's theorem Blanchini (1999, Theorem 3.1), the closed set $\mathcal{H} \cap [\underline{y}, \bar{y}]$ is $[\underline{w}, \bar{w}]$ -robustly forward invariant. \blacksquare

Appendix B. Experiment Implementation Details

B.1 Segway Model

Disturbance Set Partitioning To handle the multiplicative disturbance $(1 + w_k)$ for each system parameter from (8), with $w \in [-0.02, 0.02]^{11}$, we partition the disturbance set into $2^{11} = 2048$ different regions, *i.e.*,

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

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,

$$\begin{aligned} \dot{\underline{y}}_i &= (\mathbf{G}(\mathcal{I}_{\mathcal{H}}(\underline{y}, \bar{y}_{i:\underline{y}}), \bar{\mathcal{I}}_{\mathcal{H}}(\underline{y}, \bar{y}_{i:\underline{y}}), \underline{w}, \bar{w}))_i = \min_{[w^j, \bar{w}^j] \in \mathbf{W}} (\mathbf{G}(\mathcal{I}_{\mathcal{H}}(\underline{y}, \bar{y}_{i:\underline{y}}), \bar{\mathcal{I}}_{\mathcal{H}}(\underline{y}, \bar{y}_{i:\underline{y}}), \underline{w}^j, \bar{w}^j))_i, \\ \dot{\bar{y}}_i &= (\bar{\mathbf{G}}(\mathcal{I}_{\mathcal{H}}(\underline{y}_{i:\bar{y}}, \bar{y}), \bar{\mathcal{I}}_{\mathcal{H}}(\underline{y}_{i:\bar{y}}, \bar{y}), \underline{w}, \bar{w}))_i = \max_{[w^j, \bar{w}^j] \in \mathbf{W}} (\bar{\mathbf{G}}(\mathcal{I}_{\mathcal{H}}(\underline{y}_{i:\bar{y}}, \bar{y}), \bar{\mathcal{I}}_{\mathcal{H}}(\underline{y}_{i:\bar{y}}, \bar{y}), \underline{w}^j, \bar{w}^j))_i. \end{aligned}$$

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

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

- Polytope Loss: We use the loss \mathcal{L}^S (7) with $\lambda = 1000$ and $\varepsilon = 0.1$.
- Optimizer: Algorithm 2 terminates in 595 steps of ADAM with step size 0.001.

B.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 (7) with $\lambda = 1$ and $\varepsilon = 0.02$.
- Optimizer: Algorithm 2 terminates in $\{724, 725, 807, 890, 1267\}$ steps of ADAM with step size 0.001 for platoons with $\{4, 10, 16, 22, 28\}$ vehicles.

References

- A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. 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.
- J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL <http://github.com/google/jax>.
- C. Brix, M. N. Müller, S. Bak, T. T. Johnson, and C. 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.
- S. Chen, K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, G. J. Pappas, and M. 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.
- 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.
- M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelovic, T. Mann, and P. Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.
- T. Gurriet, A. Singletary, J. Reher, L. Ciarletta, E. Feron, and A. 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.
- A. Harapanahalli, S. Jafarpour, and S. 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>.
- A. Harapanahalli, S. Jafarpour, and S. Coogan. `immrax`: A parallelizable and differentiable toolbox for interval analysis and mixed monotone reachability in jax, 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.
- Y. Huang, I. D. J. Rodriguez, H. Zhang, Y. Shi, and Y. Yue. Fi-ode: Certifiably robust forward invariance in neural odes, 2023.
- S. Jafarpour, A. Harapanahalli, and S. Coogan. Interval reachability of nonlinear dynamical systems with neural network controllers. In *Learning for Dynamics and Control Conference*. PMLR, 2023.
- S. Jafarpour, A. Harapanahalli, and S. 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.
- L. Jaulin, M. Kieffer, O. Didrit, and É. Walter. *Applied Interval Analysis*. Springer London, 2001.

- L. Jouret, A. Saoud, and S. Oлару. Safety verification of neural-network-based controllers: a set invariance approach. *IEEE Control Systems Letters*, 2023.
- P. Kidger and C. Garcia. Equinox: neural networks in JAX via callable PyTrees and filtered transformations. *Differentiable Programming workshop at Neural Information Processing Systems 2021*, 2021.
- D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- N. Kochdumper, C. Schilling, M. Althoff, and S. Bak. Open-and closed-loop neural network verification using polynomial zonotopes. In *NASA Formal Methods Symposium*, pages 16–36. Springer, 2023.
- D. M. Lopez, M. Althoff, M. Forets, T. T. Johnson, T. Ladner, and C. Schilling. Arch-comp23 category report: Artificial intelligence and neural network control systems (ainncs) for continuous and hybrid systems plants. In G. Frehse and M. Althoff, editors, *Proceedings of 10th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH23)*, volume 96 of *EPiC Series in Computing*, pages 89–125. EasyChair, 2023. doi: 10.29007/x38n. URL <https://easychair.org/publications/paper/Vfq4b>.
- M. H. Meng, G. Bai, S. G. Teo, Z. Hou, Y. Xiao, Y. Lin, and J. S. Dong. Adversarial robustness of deep neural networks: A survey from a formal verification perspective. *IEEE Transactions on Dependable and Secure Computing*, 2022.
- A. Papachristodoulou and S. 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.
- I. D. J. Rodriguez, A. Ames, and Y. 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.
- K. Shen and J. 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.
- V. Tjeng, K. Y. Xiao, and R. Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations*, 2019.

- 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. Manzananas 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.
- T.-W. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, D. Boning, I. S. Dhillon, and L. Daniel. Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning (ICML)*, *arXiv preprint arXiv:1804.09699*, jul 2018.
- W. Xiao, T.-H. Wang, R. Hasani, M. Lechner, Y. Ban, C. Gan, and D. Rus. On the forward invariance of neural odes. In *International conference on machine learning*, pages 38100–38124. PMLR, 2023.
- K. Xu, Z. Shi, H. Zhang, Y. Wang, K.-W. Chang, M. Huang, B. Kailkhura, X. Lin, and C.-J. Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems*, 33:1129–1141, 2020.
- 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.