Contraction-Guided Adaptive Partitioning for Reachability Analysis of Neural Network Controlled Systems

Akash Harapanahalli, Saber Jafarpour, Samuel Coogan

Georgia Institute of Technology
School of Electrical and Computer Engineering
https://akashhara.com

CDC 2023

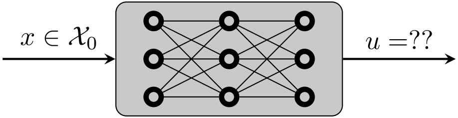
Acknowledgements

Mentor: Saber Jafarpour
`https://sites.engineering.ucsb.edu/~saber.jafarpour`

Advisor: Samuel Coogan
`https://coogan.ece.gatech.edu/`

Motivation

- Learning-enabled components are being increasingly used in control systems due to their ease of computation and ability to outperform traditional optimization-based approaches.
- However, neural networks are vulnerable to input perturbations.
- These uncertainties can compound in closed-loop applications.
- Ensuring safe operation in safety-critical applications is paramount.



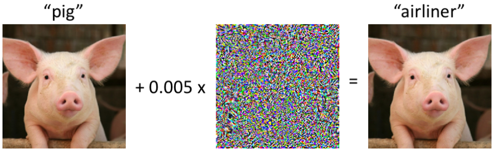
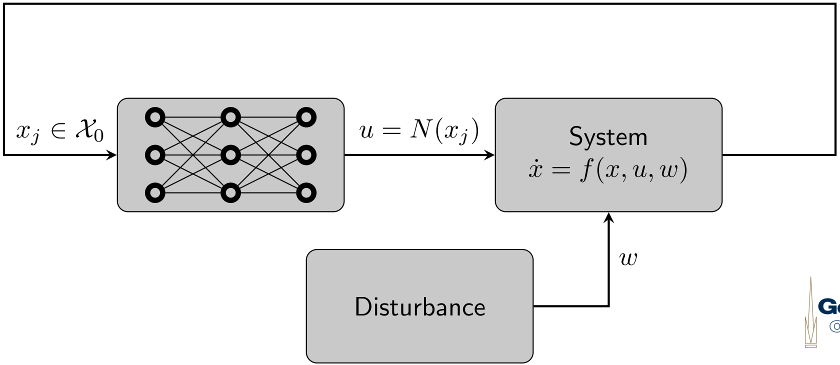


Image credit: MIT CSAIL

Problem Statement

- One way to ensure safety under uncertainty is to compute a *reachable set*, the set of all possible states a system can reach.
- **Problem:** Efficiently compute an accurate reachable set for a nonlinear system with a piecewise constant state-feedback neural network controller
- **Method:** Develop a compositional interval-based reachability framework with a contraction-guided adaptive partitioning algorithm



Outline

Part I

Interval Reachability of Learning-Enabled Systems

Interval Analysis: Inclusion Functions

Definition (Inclusion Functions)

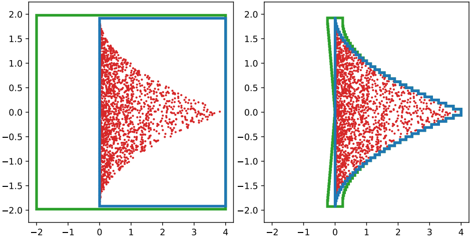
Given a map f, the map $\mathsf{F} = \begin{bmatrix} \underline{\mathsf{F}} \\ \overline{\mathsf{F}} \end{bmatrix}$ is an *inclusion function* for f if for every $\underline{x} \leq x \leq \overline{x}$,

$$\underline{\mathsf{F}}(\underline{x}, \overline{x}) \leq f(x) \leq \overline{\mathsf{F}}(\underline{x}, \overline{x}).$$

- *Minimal Inclusion Function*

$$\begin{bmatrix} \underline{\mathsf{F}}(\underline{x},\overline{x}) \\ \overline{\mathsf{F}}(\underline{x},\overline{x}) \end{bmatrix} = \begin{bmatrix} \inf_{x\in[\underline{x},\overline{x}]} f(x) \\ \sup_{x\in[\underline{x},\overline{x}]} f(x) \end{bmatrix}$$

- Inclusion functions provide a *sound* and *scalable* approach for bounding a mapping's output.



Interval Analysis: Natural Inclusion Functions in `npinterval`

Definition (Natural Inclusion Function)

Given $f = f_1 \circ f_2 \circ \cdots \circ f_N$, with inclusion functions $\mathsf{F}_1, \mathsf{F}_2, \ldots, \mathsf{F}_N$, the map $\mathsf{F}_1 \circ \mathsf{F}_2 \circ \cdots \circ \mathsf{F}_N$ is a *natural inclusion function* for F.

- Simple (non-unique) approach to building inclusion functions:
  1. Define minimal inclusion functions for elementary operations and standard functions
  2. Chain operations and functions together to build inclusion functions for general f
- `npinterval` [1] implements this in `numpy` as a new `interval` data-type. Standard `ufuncs` are implemented in compiled C.
- **Right:** Compare $f(x_1, x_2) = [(x_1 + x_2)^2, 4\sin((x_1 - x_2)/4)]^T$, $f(x_1, x_2) = [x_2^2 + 2x_1x_2 + x_2^2, 4\sin(x_1/4)\cos(x_2/4) - 4\cos(x_1/4)\sin(x_2/4)]^T$

[1] A. Harapanahalli, S. Jafarpour, S. Coogan, *WFVML* at *ICML*

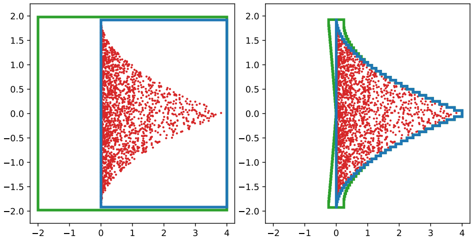
Interval Analysis: Natural Inclusion Functions in `npinterval`

Definition (Natural Inclusion Function)

Given $f = f_1 \circ f_2 \circ \cdots \circ f_N$, with inclusion functions $\mathsf{F}_1, \mathsf{F}_2, \ldots, \mathsf{F}_N$, the map $\mathsf{F}_1 \circ \mathsf{F}_2 \circ \cdots \circ \mathsf{F}_N$ is a *natural inclusion function* for F.

- Simple (non-unique) approach to building inclusion functions:
  1. Define minimal inclusion functions for elementary operations and standard functions
  2. Chain operations and functions together to build inclusion functions for general f

- `npinterval` [1] implements this in `numpy` as a new `interval` data-type. Standard `ufuncs` are implemented in compiled C.
- **Right:** Compare $f(x_1, x_2) = [(x_1 + x_2)^2, 4\sin((x_1 - x_2)/4)]^T$, $f(x_1, x_2) = [x_2^2 + 2x_1x_2 + x_2^2, 4\sin(x_1/4)\cos(x_2/4) - 4\cos(x_1/4)\sin(x_2/4)]^T$



[1] A. Harapanahalli, S. Jafarpour, S. Coogan, *WFVML* at *ICML*

Neural Network Verification: First-Order Inclusion Functions

Assumption (Local Affine Bounds of Neural Network)

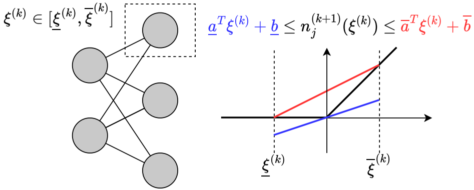
Given the neural network controller N, there exists an algorithm providing $(\underline{C}_{[\underline{y},\overline{y}]}, \overline{C}_{[\underline{y},\overline{y}]}, \underline{d}_{[\underline{y},\overline{y}]}, \overline{d}_{[\underline{y},\overline{y}]})$ valid for the localization $[\underline{y},\overline{y}]$, such that for any $x \in [\underline{x},\overline{x}] \subset [\underline{y},\overline{y}]$,

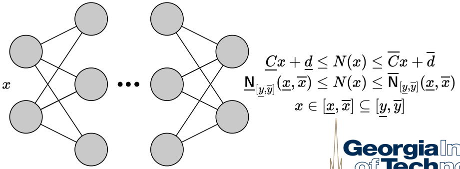
$$\underline{C}_{[\underline{y},\overline{y}]}x + \underline{d} \leq N(x) \leq \overline{C}_{[\underline{y},\overline{y}]}x + \overline{d},$$

which implies that N is a $[\underline{y},\overline{y}]$-localized inclusion function for N, where

$$\begin{bmatrix} \underline{\mathsf{N}} \\ \overline{\mathsf{N}} \end{bmatrix} = \begin{bmatrix} \underline{C}^{+}_{[\underline{y},\overline{y}]}\underline{x} + \underline{C}^{-}_{[\underline{y},\overline{y}]}\overline{x} + \underline{d}_{[\underline{y},\overline{y}]} \\ \overline{C}^{-}_{[\underline{y},\overline{y}]}\underline{x} + \overline{C}^{+}_{[\underline{y},\overline{y}]}\overline{x} + \overline{d}_{[\underline{y},\overline{y}]} \end{bmatrix}.$$

- **Notation:** $(C^{+})_{i,j} := \max(C_{i,j}, 0)$, $C^{-} = C - C^{+}$
- Many neural network verifiers can return bounds of this form, in particular CROWN [2]





Georgia Institute of Technology

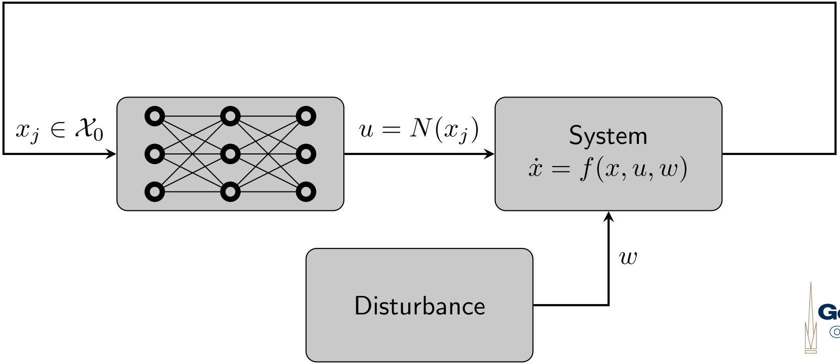
[2] H. Zhang, T-W. Weng, P-Y. Chen, C-J. Hsieh, and L. Daniel, *NeurIPS*

Closed-Loop System Inclusion Function

For the nonlinear system $\dot{x} = f(x, u, w)$ controlled by neural network $u := N(x_j)$, with inclusion functions F for f and N for N, if $[\underline{y}, \overline{y}] \supseteq [\underline{x}(t_j), \overline{x}(t_j)]$,

$$\mathsf{F}\left(\underline{x}, \overline{x}, \underline{\mathsf{N}}_{[\underline{y},\overline{y}]}(\underline{x}(t_j), \overline{x}(t_j)), \overline{\mathsf{N}}_{[\underline{y},\overline{y}]}(\underline{x}(t_j), \overline{x}(t_j)), \underline{w}, \overline{w}\right)$$

is a valid inclusion function for the closed-loop dynamics.



Georgia Institute of Technology

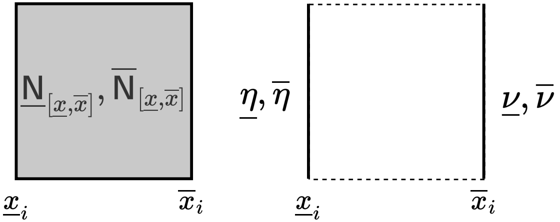
"Hybrid" Embedding System

- Embedding of the uncertain neural network controlled system into a $2n$-dimensional deterministic dynamical embedding system, evolving with state $\left[\begin{smallmatrix}\underline{x}\\ \overline{x}\end{smallmatrix}\right]$
- The neural network verification step is computationally expensive
- To facilitate runtime reachability, we use the "hybrid" mode. For $i = 1, \dots, n$,

$$
\begin{aligned}
\underline{\eta}_j &= \underline{\mathsf{N}}_{[\underline{x}(t_j),\overline{x}(t_j)]}\left(\underline{x}(t_j), \overline{x}(t_j)_{i:\underline{x}(t_j)}\right)\\
\overline{\eta}_j &= \overline{\mathsf{N}}_{[\underline{x}(t_j),\overline{x}(t_j)]}\left(\underline{x}(t_j), \overline{x}(t_j)_{i:\underline{x}(t_j)}\right)\\
\underline{\nu}_j &= \underline{\mathsf{N}}_{[\underline{x}(t_j),\overline{x}(t_j)]}\left(\underline{x}(t_j)_{i:\overline{x}(t_j)}, \overline{x}(t_j)\right)\\
\overline{\nu}_j &= \overline{\mathsf{N}}_{[\underline{x}(t_j),\overline{x}(t_j)]}\left(\underline{x}(t_j)_{i:\overline{x}(t_j)}, \overline{x}(t_j)\right)
\end{aligned}
$$

$$
\begin{aligned}
\dot{\underline{x}}_i &= \left(\underline{\mathsf{E}}(\underline{x},\overline{x},\underline{w},\overline{w})\right)_i := \left(\underline{\mathsf{F}}\left(\underline{x}, \overline{x}_{i:\underline{x}}, \underline{\eta}_j, \overline{\eta}_j, \underline{w}, \overline{w}\right)\right)_i\\
\dot{\overline{x}}_i &= \left(\overline{\mathsf{E}}(\underline{x},\overline{x},\underline{w},\overline{w})\right)_i := \left(\overline{\mathsf{F}}\left(\underline{x}_{i:\overline{x}}, \overline{x}, \underline{\nu}_j, \overline{\nu}_j, \underline{w}, \overline{w}\right)\right)_i
\end{aligned}
$$

Notation: $(x_{i:y})_j = \begin{cases} x_j & i \neq j \\ y_j & i = j \end{cases}$



Reachability Using the Embedding System

- The $2n$-dimensional deterministic dynamical embedding system provides computationally efficient bounds on the uncertain dynamics of the neural network controlled system.

Proposition (interval reachability via embedding system)

Given $\dot{x} = f(x, u, w)$ with $u := N(x_j)$, and inclusion function F for f,

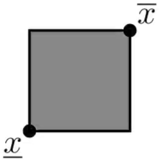
$$\begin{aligned}\mathcal{R}_f(t, t_0, [\underline{x}_0, \overline{x}_0], [\underline{w}, \overline{w}]) &= \{\text{trajectories of } f \text{ from } x_0 \in [\underline{x}_0, \overline{x}_0] \text{ with } w(t) \in [\underline{w}, \overline{w}]\} \\ &\subseteq [\underline{x}(t), \overline{x}(t)],\end{aligned}$$

for every $t \geq t_0$, provided $t \mapsto \begin{bmatrix} \underline{x}(t) \\ \overline{x}(t) \end{bmatrix}$ is the trajectory of the embedding system

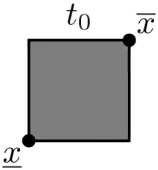
$$\begin{aligned}\dot{\underline{x}}_i &= \left(\underline{\mathsf{E}}(\underline{x}, \overline{x}, \underline{w}, \overline{w})\right)_i := \left(\underline{\mathsf{F}}\left(\underline{x}, \overline{x}_{i:\underline{x}}, \underline{\eta}_j, \overline{\eta}_j, \underline{w}, \overline{w}\right)\right)_i \\ \dot{\overline{x}}_i &= \left(\overline{\mathsf{E}}(\underline{x}, \overline{x}, \underline{w}, \overline{w})\right)_i := \left(\overline{\mathsf{F}}\left(\underline{x}_{i:\overline{x}}, \overline{x}, \underline{\nu}_j, \overline{\nu}_j, \underline{w}, \overline{w}\right)\right)_i\end{aligned}.$$

Embedding System Visualization

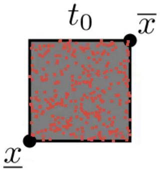
Embedding System Visualization



Embedding System Visualization

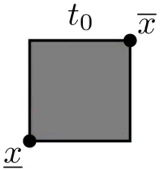


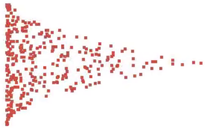
Embedding System Visualization



$$\dot{x} = f(x, N(x(t_0)), w)$$

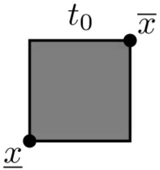
Embedding System Visualization





$$\dot{x} = f(x, N(x(t_0)), w)$$

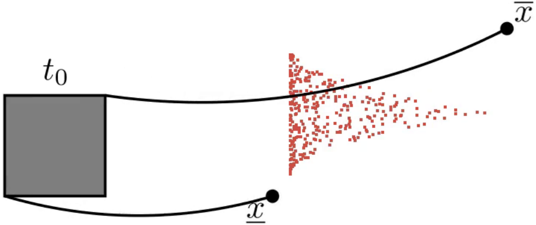
Embedding System Visualization



$$\dot{\underline{x}} = \underline{\mathsf{E}}(\underline{x},\overline{x},\underline{w},\overline{w}) = \underline{\mathsf{F}}(\underline{x}, \overline{x}_{i:\underline{x}}, \underline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x},\overline{x}_{i:\underline{x}}), \overline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x},\overline{x}_{i:\underline{x}}), \underline{w},\overline{w})$$

$$\dot{\overline{x}} = \overline{\mathsf{E}}(\underline{x},\overline{x},\underline{w},\overline{w}) = \overline{\mathsf{F}}(\underline{x}_{i:\overline{x}}, \overline{x}, \underline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x}_{i:\overline{x}},\overline{x}), \overline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x}_{i:\overline{x}},\overline{x}), \underline{w},\overline{w})$$

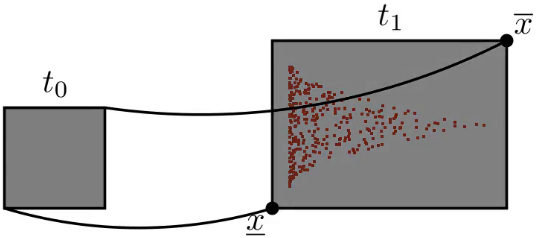
Embedding System Visualization



$$\dot{\underline{x}} = \underline{\mathsf{E}}(\underline{x}, \overline{x}, \underline{w}, \overline{w}) = \underline{\mathsf{F}}(\underline{x}, \overline{x}_{i:\underline{x}}, \underline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x}, \overline{x}_{i:\underline{x}}), \overline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x}, \overline{x}_{i:\underline{x}}), \underline{w}, \overline{w})$$

$$\dot{\overline{x}} = \overline{\mathsf{E}}(\underline{x}, \overline{x}, \underline{w}, \overline{w}) = \overline{\mathsf{F}}(\underline{x}_{i:\overline{x}}, \overline{x}, \underline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x}_{i:\overline{x}}, \overline{x}), \overline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x}_{i:\overline{x}}, \overline{x}), \underline{w}, \overline{w})$$

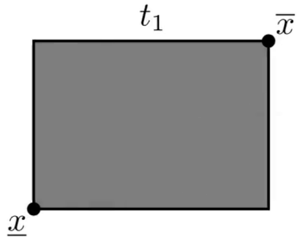
Embedding System Visualization



$$\dot{\underline{x}} = \underline{\mathsf{E}}(\underline{x},\overline{x},\underline{w},\overline{w}) = \underline{\mathsf{F}}(\underline{x},\overline{x}_{i:\underline{x}},\underline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x},\overline{x}_{i:\underline{x}}),\overline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x},\overline{x}_{i:\underline{x}}),\underline{w},\overline{w})$$

$$\dot{\overline{x}} = \overline{\mathsf{E}}(\underline{x},\overline{x},\underline{w},\overline{w}) = \overline{\mathsf{F}}(\underline{x}_{i:\overline{x}},\overline{x},\underline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x}_{i:\overline{x}},\overline{x}),\overline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x}_{i:\overline{x}},\overline{x}),\underline{w},\overline{w})$$

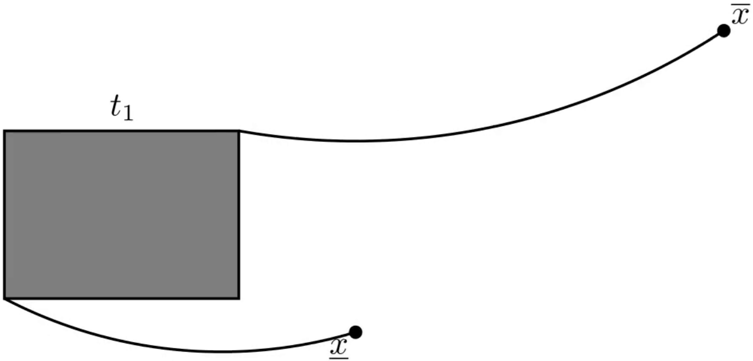
Embedding System Visualization



$$\dot{\underline{x}} = \underline{\mathsf{E}}(\underline{x}, \overline{x}, \underline{w}, \overline{w}) = \underline{\mathsf{F}}(\underline{x}, \overline{x}_{i:\underline{x}}, \underline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x}, \overline{x}_{i:\underline{x}}), \overline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x}, \overline{x}_{i:\underline{x}}), \underline{w}, \overline{w})$$

$$\dot{\overline{x}} = \overline{\mathsf{E}}(\underline{x}, \overline{x}, \underline{w}, \overline{w}) = \overline{\mathsf{F}}(\underline{x}_{i:\overline{x}}, \overline{x}, \underline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x}_{i:\overline{x}}, \overline{x}), \overline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x}_{i:\overline{x}}, \overline{x}), \underline{w}, \overline{w})$$

Embedding System Visualization

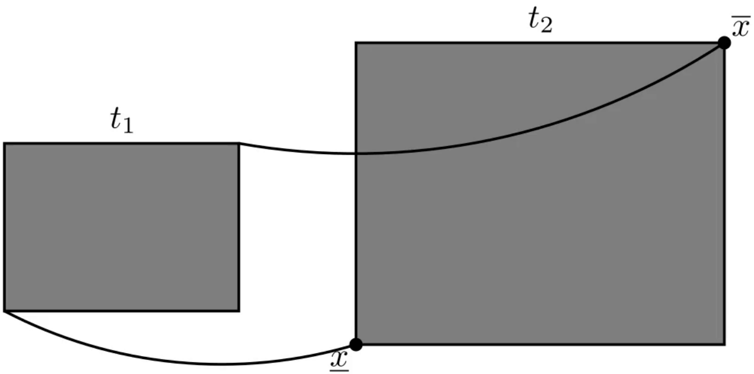


$$\dot{\underline{x}} = \underline{\mathsf{E}}(\underline{x}, \overline{x}, \underline{w}, \overline{w}) = \underline{\mathsf{F}}(\underline{x}, \overline{x}_{i:\underline{x}}, \underline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x}, \overline{x}_{i:\underline{x}}), \overline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x}, \overline{x}_{i:\underline{x}}), \underline{w}, \overline{w})$$

$$\dot{\overline{x}} = \overline{\mathsf{E}}(\underline{x}, \overline{x}, \underline{w}, \overline{w}) = \overline{\mathsf{F}}(\underline{x}_{i:\overline{x}}, \overline{x}, \underline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x}_{i:\overline{x}}, \overline{x}), \overline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x}_{i:\overline{x}}, \overline{x}), \underline{w}, \overline{w})$$

ia Institute
hnology

Embedding System Visualization



$$\dot{\underline{x}} = \underline{\mathsf{E}}(\underline{x}, \overline{x}, \underline{w}, \overline{w}) = \underline{\mathsf{F}}(\underline{x}, \overline{x}_{i:\underline{x}}, \underline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x}, \overline{x}_{i:\underline{x}}), \overline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x}, \overline{x}_{i:\underline{x}}), \underline{w}, \overline{w})$$

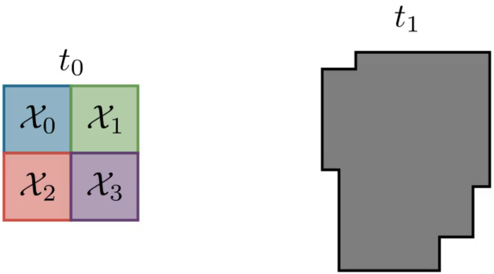
$$\dot{\overline{x}} = \overline{\mathsf{E}}(\underline{x}, \overline{x}, \underline{w}, \overline{w}) = \overline{\mathsf{F}}(\underline{x}_{i:\overline{x}}, \overline{x}, \underline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x}_{i:\overline{x}}, \overline{x}), \overline{\mathsf{N}}_{[\underline{x},\overline{x}]}(\underline{x}_{i:\overline{x}}, \overline{x}), \underline{w}, \overline{w})$$

Part II

Contraction-Guided Adaptive Partitioning

Basic Theory of Partitioning

- While interval methods are computationally efficient, they are notoriously overconservative
- Estimates tend to grow exponentially as the size of intervals grow
- Smaller initial sets have smaller overconservatism



Understanding Conservatism Using Contraction Theory

Theorem (informal, accuracy guarantees)

$$\left\| \begin{bmatrix} \underline{x}(t) \\ \overline{x}(t) \end{bmatrix} - \begin{bmatrix} x(t) \\ x(t) \end{bmatrix} \right\|_\infty \leq e^{c_x t} \cdot \boxed{\text{(1) size of initial set } [\underline{x}_0, \overline{x}_0]} + \frac{\ell_u (e^{c_x t} - 1)}{c_x} \cdot \boxed{\text{(2) neural network verification approximation error}} + \frac{\ell_w (e^{c_x t} - 1)}{c_x} \cdot \boxed{\text{(3) size of disturbance } [\underline{w}, \overline{w}]},$$

where c_x is the maximum rate of expansion of the closed-loop system, ℓ_u is the ℓ_∞-Lipschitz bound of the control input u on the open-loop system, and ℓ_w is the ℓ_∞-Lipschitz bound of the disturbance input w; all localized to a curve $[\underline{y}(t), \overline{y}(t)] \supseteq [\underline{x}(t), \overline{x}(t)]$.

1. We introduce *Contraction-Guided Adaptive Partitioning*, with three main features: *Separation*, *Spatial Awareness*, and *Temporal Awareness*.

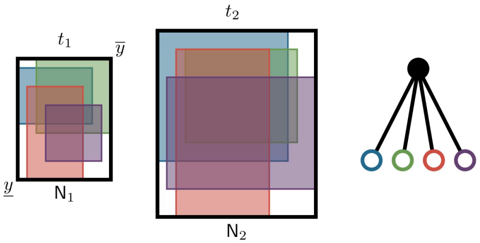
Feature 1: Separation

$$\left\| \begin{bmatrix} \underline{x}(t) \\ \overline{x}(t) \end{bmatrix} - \begin{bmatrix} x(t) \\ x(t) \end{bmatrix} \right\|_\infty \leq e^{c_x t} \cdot \boxed{\text{(1) size of initial set } [\underline{x}_0, \overline{x}_0]}$$
$$+ \frac{\ell_u (e^{c_x t} - 1)}{c_x} \cdot \boxed{\text{(2) neural network verification approximation error}}$$
$$+ \frac{\ell_w (e^{c_x t} - 1)}{c_x} \cdot \boxed{\text{(3) size of disturbance } [\underline{w}, \overline{w}]}$$

Local constants: $c_x =$ closed-loop rate of expansion. $\ell_u = \mathrm{Lip}_\infty^f(u)$, $\ell_w = \mathrm{Lip}_\infty^f(w)$ (Lipschitz)

Build the inclusion function $\mathsf{N}_{[\underline{y},\overline{y}]}$ for a set containing multiple partitions, and reuse.

Improves $\boxed{\text{(1) size of initial set } [\underline{x}_0, \overline{x}_0]}$ without spending extra computations on $\boxed{\text{(2) neural network verification approximation error}}$

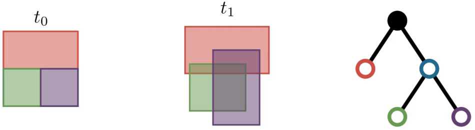


Feature 2: Spatial Awareness

$$\left\| \begin{bmatrix} \underline{x}(t) \\ \overline{x}(t) \end{bmatrix} - \begin{bmatrix} x(t) \\ x(t) \end{bmatrix} \right\|_\infty \leq e^{c_x t} \cdot \boxed{\text{(1) size of initial set } [\underline{x}_0, \overline{x}_0]}$$
$$+ \frac{\ell_u (e^{c_x t} - 1)}{c_x} \cdot \boxed{\text{(2) neural network verification approximation error}}$$
$$+ \frac{\ell_w (e^{c_x t} - 1)}{c_x} \cdot \boxed{\text{(3) size of disturbance } [\underline{w}, \overline{w}]}$$

Local constants: c_x = closed-loop rate of expansion. $\ell_u = \mathrm{Lip}^f_\infty(u)$, $\ell_w = \mathrm{Lip}^f_\infty(w)$ (Lipschitz)

Partition the regions of the state space that are expanding the fastest. Improves c_x, ℓ_u, ℓ_w where they are *spatially* the worst.

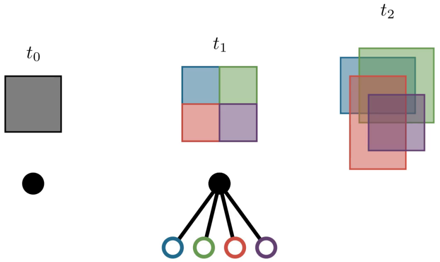


Feature 3: Temporal Awareness

$$\left\| \begin{bmatrix} \underline{x}(t) \\ \overline{x}(t) \end{bmatrix} - \begin{bmatrix} x(t) \\ x(t) \end{bmatrix} \right\|_\infty \leq e^{c_x t} \cdot \boxed{\text{(1) size of initial set } [\underline{x}_0, \overline{x}_0]}$$
$$+ \frac{\ell_u (e^{c_x t} - 1)}{c_x} \cdot \boxed{\text{(2) neural network verification approximation error}}$$
$$+ \frac{\ell_w (e^{c_x t} - 1)}{c_x} \cdot \boxed{\text{(3) size of disturbance } [\underline{w}, \overline{w}]}$$

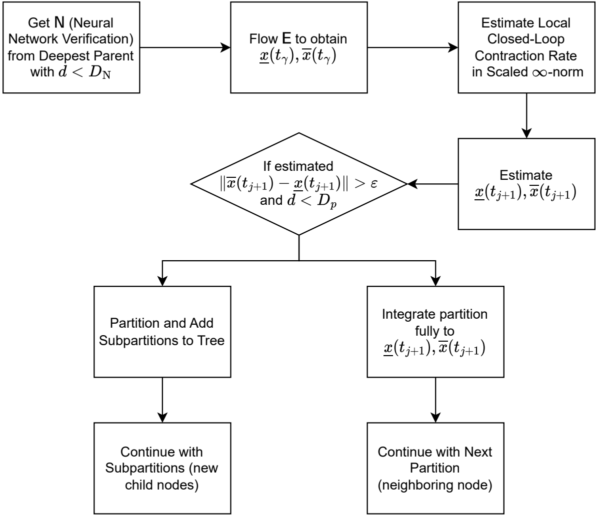
Local constants: c_x = closed-loop rate of expansion. $\ell_u = \mathrm{Lip}^f_\infty(u)$, $\ell_w = \mathrm{Lip}^f_\infty(w)$ (Lipschitz)

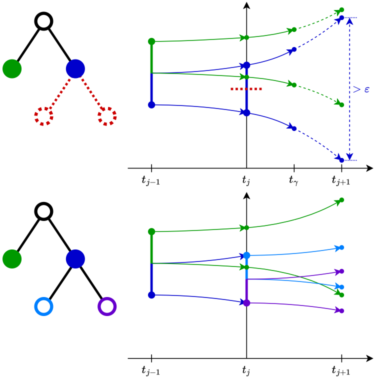
Saves computations by applying partitions along trajectories just before estimates begin to explode. Improves c_x, ℓ_u, ℓ_w where they are *temporally* the worst.



Algorithm

Scaled ∞-norm: for $\varepsilon \in \mathbb{R}^n$, $\|x\|_{\infty,\varepsilon} = \|\operatorname{diag}(\varepsilon)^{-1}x\|_\infty$.





Vehicle Model

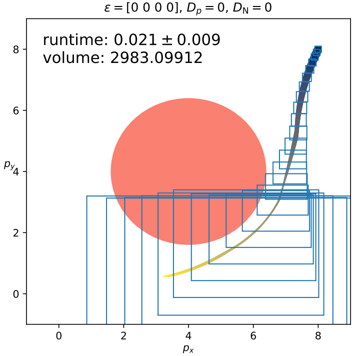
Kinematic bicycle controlled by a $4 \times 100 \times 100 \times 2$ ReLU network trained using imitation learning on an MPC stabilizing to the origin while avoiding an obstacle centered at $(4, 4)$.

$$\dot{p_x} = v\cos(\phi + \beta(u_2)) \qquad \dot{\phi} = \frac{v}{\ell_r}\sin(\beta(u_2))$$

$$\dot{p_y} = v\sin(\phi + \beta(u_2)) \qquad \dot{v} = u_1$$

ε	D_p, D_{N}	Runtime (s)	Volume
non-adaptive	$(2, 1)$	1.851 ± 0.010	1.988
$[0.2,\ 0.2,\ \infty,\ \infty]$	$(2, 1)$	1.583 ± 0.010	1.689
$[0.25,\ 0.25,\ \infty,\ \infty]$	$(2, 1)$	1.243 ± 0.008	1.846
non-adaptive	$(2, 2)$	4.274 ± 0.023	0.803
$[0.2,\ 0.2,\ \infty,\ \infty]$	$(2, 2)$	3.332 ± 0.012	0.787
$[0.25,\ 0.25,\ \infty,\ \infty]$	$(2, 2)$	2.636 ± 0.008	0.986

Table: The performance of ReachMM-CG on the vehicle model.



Vehicle Model

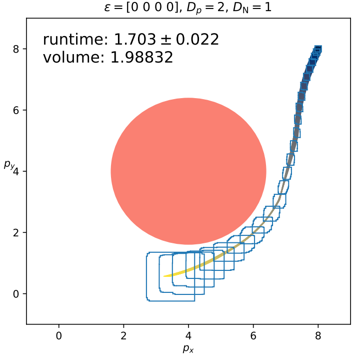
Kinematic bicycle controlled by a $4 \times 100 \times 100 \times 2$ ReLU network trained using imitation learning on an MPC stabilizing to the origin while avoiding an obstacle centered at $(4, 4)$.

$$\dot{p_x} = v\cos(\phi + \beta(u_2)) \qquad \dot{\phi} = \frac{v}{\ell_r}\sin(\beta(u_2))$$

$$\dot{p_y} = v\sin(\phi + \beta(u_2)) \qquad \dot{v} = u_1$$

ε	D_p, D_N	Runtime (s)	Volume
non-adaptive	$(2, 1)$	1.851 ± 0.010	1.988
$[0.2,\ 0.2,\ \infty,\ \infty]$	$(2, 1)$	1.583 ± 0.010	1.689
$[0.25,\ 0.25,\ \infty,\ \infty]$	$(2, 1)$	1.243 ± 0.008	1.846
non-adaptive	$(2, 2)$	4.274 ± 0.023	0.803
$[0.2,\ 0.2,\ \infty,\ \infty]$	$(2, 2)$	3.332 ± 0.012	0.787
$[0.25,\ 0.25,\ \infty,\ \infty]$	$(2, 2)$	2.636 ± 0.008	0.986

Table: The performance of ReachMM-CG on the vehicle model.



Vehicle Model

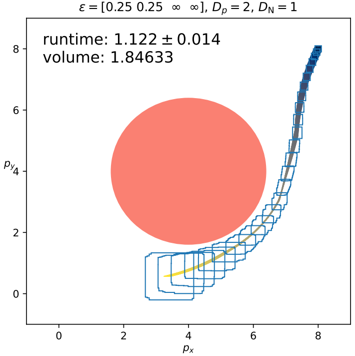
Kinematic bicycle controlled by a $4 \times 100 \times 100 \times 2$ ReLU network trained using imitation learning on an MPC stabilizing to the origin while avoiding an obstacle centered at $(4, 4)$.

$$\dot{p_x} = v\cos(\phi + \beta(u_2)) \qquad \dot{\phi} = \frac{v}{\ell_r}\sin(\beta(u_2))$$

$$\dot{p_y} = v\sin(\phi + \beta(u_2)) \qquad \dot{v} = u_1$$

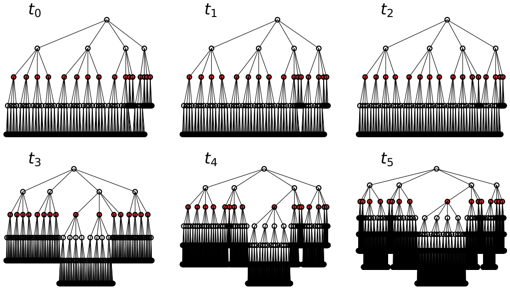
ε	D_p, D_N	Runtime (s)	Volume
non-adaptive	$(2, 1)$	1.851 ± 0.010	1.988
$[0.2,\ 0.2,\ \infty,\ \infty]$	$(2, 1)$	1.583 ± 0.010	1.689
$[0.25,\ 0.25,\ \infty,\ \infty]$	$(2, 1)$	1.243 ± 0.008	1.846
non-adaptive	$(2, 2)$	4.274 ± 0.023	0.803
$[0.2,\ 0.2,\ \infty,\ \infty]$	$(2, 2)$	3.332 ± 0.012	0.787
$[0.25,\ 0.25,\ \infty,\ \infty]$	$(2, 2)$	2.636 ± 0.008	0.986

Table: The performance of ReachMM-CG on the vehicle model.

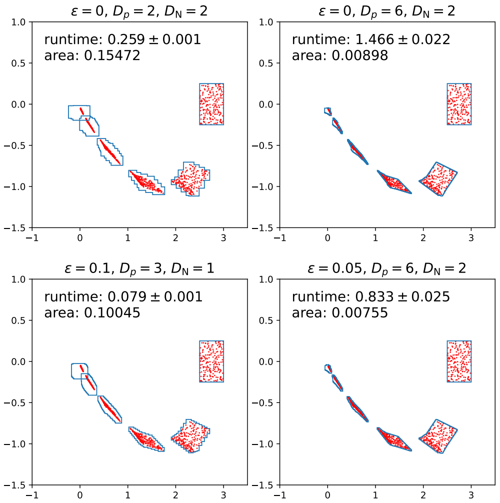


Discrete-Time Double Integrator

Benchmark from the literature. Discrete time double integrator with $2 \times 10 \times 5 \times 1$ ReLU neural network controller.

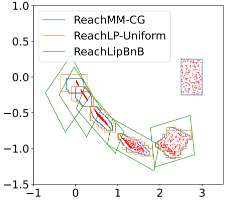


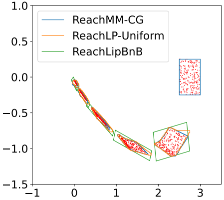
The partition tree for a run on the double integrator for $\varepsilon = 0.1$, $D_p = 10$, $D_N = 2$.



Discrete-Time Double Integrator - Comparison to the Literature

Method	Setup	Runtime	Area
ReachMM-CG (our method)	$(0.1,\ 3,\ 1)$	$\mathbf{0.079}$	$\mathbf{1.0 \cdot 10^{-1}}$
	$(0.05,\ 6,\ 2)$	$\mathbf{0.833}$	$\mathbf{7.5 \cdot 10^{-3}}$
ReachMM [3]	$(2,\ 2)$	0.259	$1.5 \cdot 10^{-1}$
	$(6,\ 2)$	1.466	$9.0 \cdot 10^{-3}$
ReachLP-Unif [4]	4	0.212	$1.5 \cdot 10^{-1}$
	16	3.149	$1.0 \cdot 10^{-2}$
ReachLP-GSG [4]	55	0.913	$5.3 \cdot 10^{-1}$
	205	2.164	$8.8 \cdot 10^{-2}$
ReachLipBnB [5]	0.1	0.956	$5.4 \cdot 10^{-1}$
	0.001	3.681	$1.2 \cdot 10^{-2}$





[3] S. Jafarpour, A. Harapanahalli, and S. Coogan, *L4DC*
[4] M. Everett, G. Habibi, C. Sun, and J. How, *IEEE Access*
[5] T. Entesari, S. Sharifi, and M. Fazlyab, *ICRA*

Part III

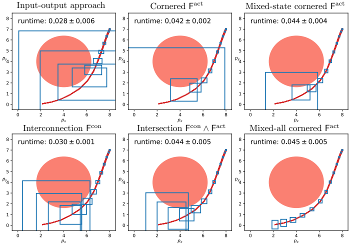
Conclusions

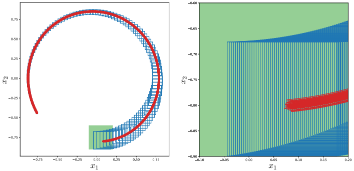
Conclusions

- Ensuring safety of learning enabled components in control systems is vital for safety-critical applications.
- While interval reachable set computation provides a fast and scalable approach, estimates can be overconservative.
- The efficiency of interval reachability approaches are expoited through partitioning, and borrowing ideas from contraction theory, the adaptive partitioning algorithm can help balance the accuracy/runtime tradeoff through separation, and spatial/temporal awareness.

In the Pipeline...

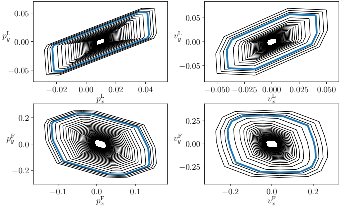
Interaction-Aware Jacobian-based Inclusion Functions





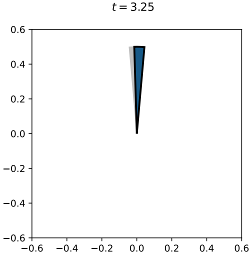
in review, arXiv:2307.14938

Forward Invariance in Neural Network Controlled Systems



accepted to IEEE L-CSS, arXiv:2309.09043

Differentiable and Parallel Implementation in JAX



# Part.	ReachMM (euler)	immrax (euler) CPU	immrax (euler) GPU	immrax (tsit5) CPU	immrax (tsit5) GPU
$1^4 = 1$	.0476	.00246	.00328	.0124	.0153
$2^4 = 16$	.690	.0271	.00357	.156	.0199
$3^4 = 81$	3.44	.123	.00652	.716	.0303
$4^4 = 256$	11.0	.273	.0163	1.63	.0510
$5^4 = 625$	27.1	.826	.0230	4.93	.104
$6^4 = 1296$	55.8	2.01	.0447	12.1	.200

awaiting submission

References I

A. Harapanahalli, S. Jafarpour, and S. Coogan.
A toolbox for fast interval arithmetic in `numpy` with an application to formal verification of neural network controlled system.
In *2nd ICML Workshop on Formal Verification of Machine Learning*, 2023.

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 4944â4953, 2018.

Saber Jafarpour, Akash Harapanahalli, and Samuel Coogan.
Interval reachability of nonlinear dynamical systems with neural network controllers.
In *Learning for Dynamics and Control Conference*, pages 12–25. PMLR, 2023.

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

References II

T. Entesari, S. Sharifi, and M. Fazlyab.
ReachLipBnB: A branch-and-bound method for reachability analysis of neural autonomous systems using lipschitz bounds.
In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1003–1010, 2023.