Safety of Autonomous Systems with Learning-enabled Feedbacks

Saber Jafarpour

University of Colorado Boulder

Georgia Tech

September 11, 2024

Alexander Davydov
UCSB

Matthew Abate
Georgia Tech

Pedro Cisneros-Velarde
UIUC

Akash Harapanahalli
Georgia Tech

Anton Proskurnikov
Politecnico di Torino

Francesco Bullo
UCSB

Samuel Coogan
Georgia Tech

Power grids

Delivery drones

Autonomous Vehicles

- large penetration of distributed renewable units in power grids
- urban air mobility support operations including transfer of passengers and cargo
- the increase in number of self-driving learning-enabled vehicles

Modern Autonomous Systems

Power grids

Delivery drones

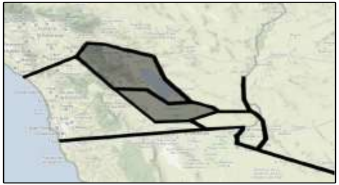
Autonomous Vehicles

- large penetration of distributed renewable units in power grids
- urban air mobility support operations including transfer of passengers and cargo
- the increase in number of self-driving learning-enabled vehicles

Autonomous systems in our societies are becoming **large-scale**, **interconnected** and **complex**.

A critical task

Desired performance while ensuring their **safety** and **reliability**.



2011 US Southwest blackout

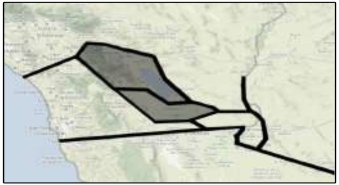
Postal Drone hit the building

Self-driving car accident

Modern Autonomous Systems
Safety and Reliability guarantees

A critical task

Desired performance while ensuring their **safety** and **reliability**.



2011 US Southwest blackout

Postal Drone hit the building

Self-driving car accident

My Research

Provide **guarantees** for safety and reliability of autonomous systems

Tools: Systems and Control (contraction theory, monotone system theory)

In this talk: Autonomous Systems with learning-based components

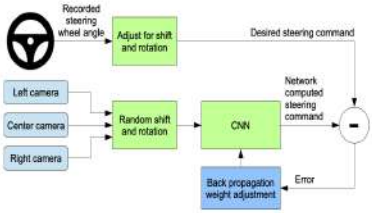
In this talk: Autonomous Systems with learning-based components

- Learning-based **controllers** or **motion planners** in safety-critical applications

In this talk: Autonomous Systems with learning-based components

- Learning-based **controllers** or **motion planners** in safety-critical applications
- Main issues with traditional controllers: computationally burdensome, executed by an expert, complicated representation.

Self driving vehicles:

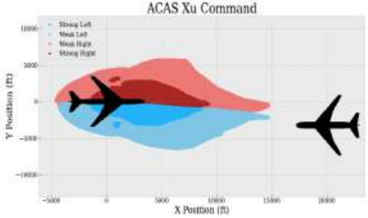


M. Bojarski, et al., NeurIPS, 2016.

Robotic motion planning:

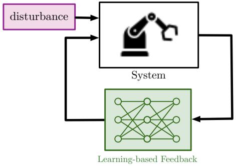
M. Everett, et. al., IROS, 2018.

Collision avoidance:

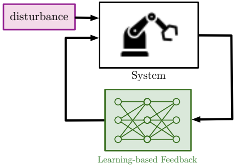


K. Julian, et. al., DASC, 2016.

Goal: ensure *safety and reliability* of the closed-loop system



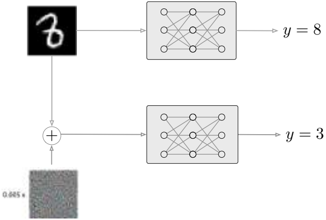
[1] C. Szegedy et. al. Intriguing properties of neural networks. In ICLR, 2014



Goal: ensure *safety and reliability* of the closed-loop system

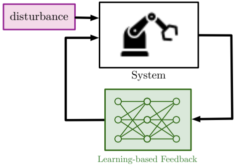
Issues with learning algorithms:

- large # of parameters with nonlinearity
- sensitive wrt to input perturbations[1]
- no safety guarantee in their training



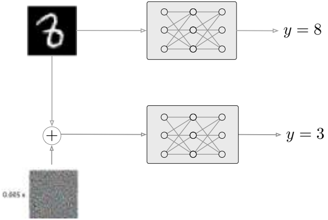
[1] C. Szegedy et. al. Intriguing properties of neural networks. In ICLR, 2014

Goal: ensure *safety and reliability* of the closed-loop system



Issues with learning algorithms:

- large # of parameters with nonlinearity
- sensitive wrt to input perturbations[1]
- no safety guarantee in their training

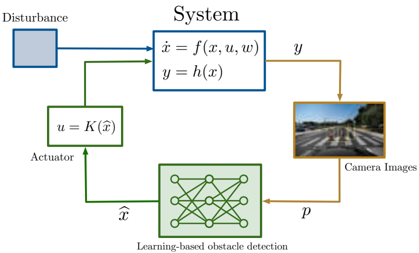


Analysis: how safe is the closed-loop system? (Verification)

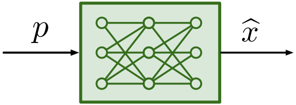
Design: how to design the learning component to ensure safety? (Training)

[1] C. Szegedy et. al. Intriguing properties of neural networks. In ICLR, 2014

Perception-based Obstacle Avoidance

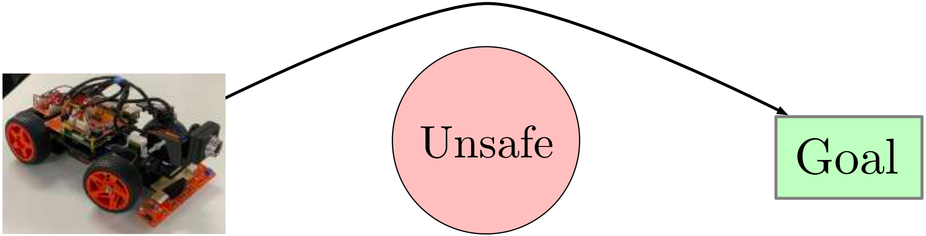


$$\dot{x} = f(x, u, w)$$
$$y = h(x)$$

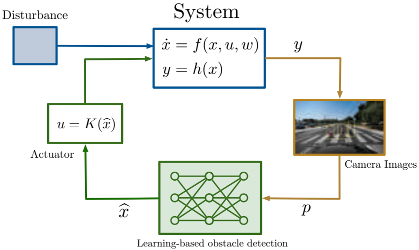


Learning-based obstacle detection

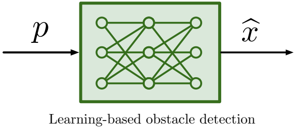
trained offline using images



Perception-based Obstacle Avoidance



$$\dot{x} = f(x, u, w)$$
$$y = h(x)$$

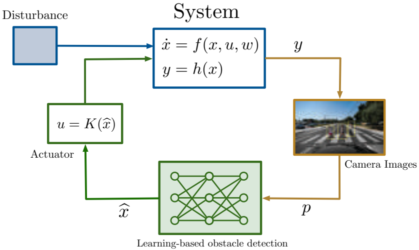


trained offline using images

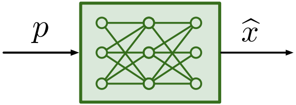
No guarantee to avoid the obstacle:

- out of distribution images
- changes in the environment

Perception-based Obstacle Avoidance

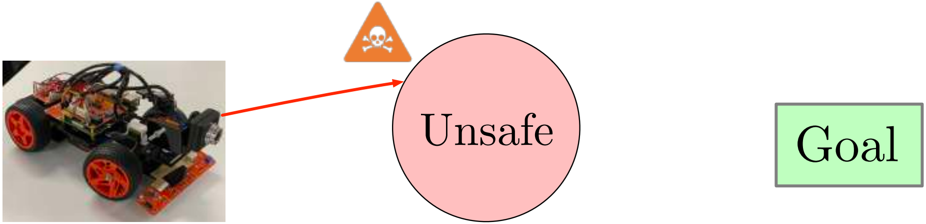


$$\dot{x} = f(x, u, w)$$
$$y = h(x)$$



Learning-based obstacle detection

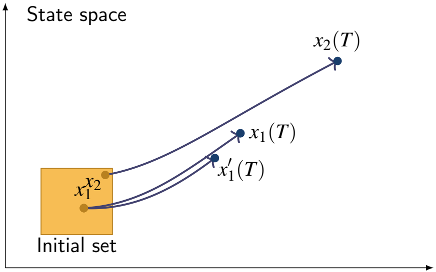
trained offline using images

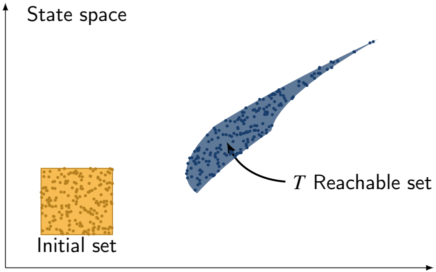


Outline of this talk

- **Reachability Analysis**
- Contraction and Monotone Theory
- Analysis of Learning-enabled Feedbacks

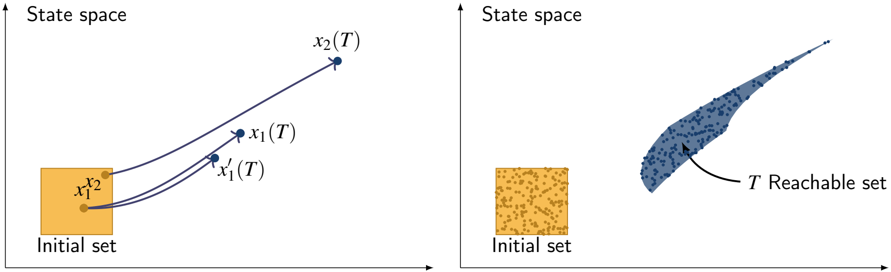
System : $\dot{x} = f(x, w)$ **State** : $x \in \mathbb{R}^n$ **Uncertainty** : $w \in \mathcal{W} \subseteq \mathbb{R}^m$





What are the possible states of the system at time T?

System : $\dot{x} = f(x, w)$ **State** : $x \in \mathbb{R}^n$ **Uncertainty** : $w \in \mathcal{W} \subseteq \mathbb{R}^m$



What are the possible states of the system at time T?

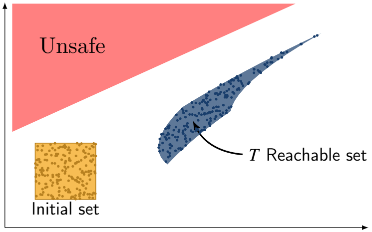
- T**-reachable sets** characterize evolution of the system

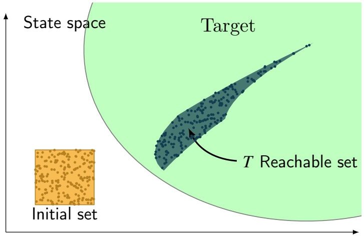
$$\mathcal{R}_f(T, \mathcal{X}_0, \mathcal{W}) = \{x_w(T) \mid x_w(\cdot) \text{ is a traj for some } w(\cdot) \in \mathcal{W} \text{ with } x_0 \in \mathcal{X}_0\}$$

A large number of **safety specifications** can be represented using T-reachable sets

A large number of **safety specifications** can be represented using T-reachable sets

- Example: Reach-avoid problem



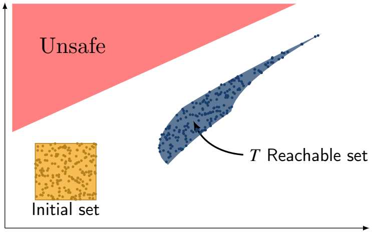


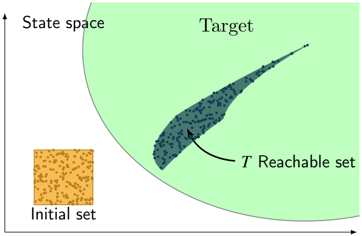
$$\mathcal{R}_f(T, \mathcal{X}_0, \mathcal{W}) \cap \text{ Unsafe set } = \emptyset$$

$$\mathcal{R}_f(T_{\text{final}}, \mathcal{X}_0, \mathcal{W}) \subseteq \text{Target set}$$

A large number of **safety specifications** can be represented using T-reachable sets

- Example: Reach-avoid problem



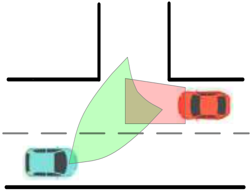


$$\mathcal{R}_f(T, \mathcal{X}_0, \mathcal{W}) \cap \text{ Unsafe set } = \emptyset$$

$$\mathcal{R}_f(T_{\text{final}}, \mathcal{X}_0, \mathcal{W}) \subseteq \text{Target set}$$

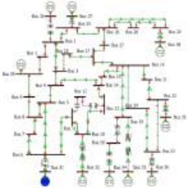
Combining different instantiation of Reach-avoid problem $\Longrightarrow$
diverse range of specifications
(complex planning using logics, invariance, stability)

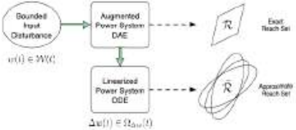
Autonomous Driving:



Althoff, 2014

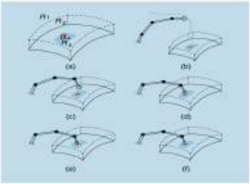
Power grids:



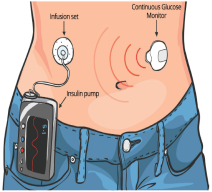


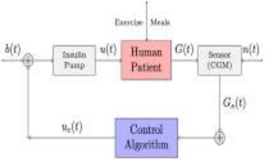
Chen and Domınguez-Garcıa, 2016

Robot-assisted Surgery:



Drug Delivery:





Chen, Dutta, and Sankaranarayanan, 2017

Computing the T-reachable sets are computationally challenging

Computing the T-reachable sets are computationally challenging

Solution: over-approximations and under-approximation of reachable sets

Computing the T-reachable sets are computationally challenging

Solution: over-approximations and under-approximation of reachable sets

- for safety verification $\implies$ over-approximations

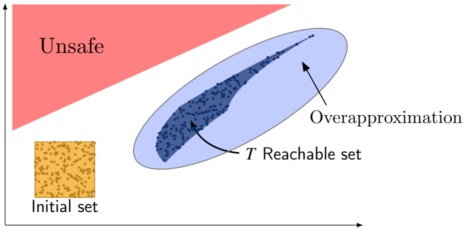
Over-approximation: $\mathcal{R}_f(T, \mathcal{X}_0, \mathcal{W}) \subseteq \overline{\mathcal{R}}_f(T, \mathcal{X}_0, \mathcal{W})$

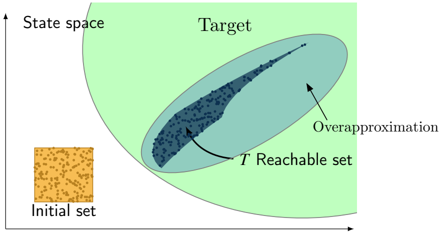
Computing the T-reachable sets are computationally challenging

Solution: over-approximations and under-approximation of reachable sets

- for safety verification $\implies$ over-approximations

Over-approximation: $\mathcal{R}_f(T, \mathcal{X}_0, \mathcal{W}) \subseteq \overline{\mathcal{R}}_f(T, \mathcal{X}_0, \mathcal{W})$





$\overline{\mathcal{R}}_f(T, \mathcal{X}_0, \mathcal{W}) \cap \text{Unsafe set} = \emptyset$

$\overline{\mathcal{R}}_f(T_{\text{final}}, \mathcal{X}_0, \mathcal{W}) \subseteq \text{Target set}$

In many autonomous systems safety cannot be **completely ensured** at the design level[2]. (stochastic environment, human-in-the-loop)

[2] Institute for Defense Analysis, The Status of Test, Evaluation, Verification, and Validation of Autonomous Systems, 2018

In many autonomous systems safety cannot be **completely ensured** at the design level[2]. (stochastic environment, human-in-the-loop)

Many autonomous systems contains **data-driven** components (neural network controllers, learning-based strategies)

[2] Institute for Defense Analysis, The Status of Test, Evaluation, Verification, and Validation of Autonomous Systems, 2018

In many autonomous systems safety cannot be **completely ensured** at the design level[2]. (stochastic environment, human-in-the-loop)

Many autonomous systems contains **data-driven** components (neural network controllers, learning-based strategies)

Many autonomous systems are **large-scale** with interconnected components (power grids, transportation networks)

[2] Institute for Defense Analysis, The Status of Test, Evaluation, Verification, and Validation of Autonomous Systems, 2018

In many autonomous systems safety cannot be **completely ensured** at the design level[2]. (stochastic environment, human-in-the-loop)

Many autonomous systems contains **data-driven** components (neural network controllers, learning-based strategies)

Many autonomous systems are **large-scale** with interconnected components (power grids, transportation networks)

Develop **reachability algorithms** that are

- computationally efficient
- adaptable to data-driven algorithms
- scalable to the size of the system

[2] Institute for Defense Analysis, The Status of Test, Evaluation, Verification, and Validation of Autonomous Systems, 2018

Reachability of dynamical system is an old problem: ~ 1980

Reachability of dynamical system is an old problem: $\sim$ 1980

Different approaches for approximating reachable sets

- Linear, and piecewise linear systems (Ellipsoidal methods) (Kurzhanski and Varaiya, 2000)
- Optimization-based approaches (Hamilton-Jacobi, Level-set method) (Bansal et al., 2017, Mitchell et al., 2002, Herbert et al., 2021)
- Matrix measure-based (Fan et al., 2018, Maidens and Arcak, 2015)

Reachability of dynamical system is an old problem: ~ 1980

Different approaches for approximating reachable sets

- Linear, and piecewise linear systems (Ellipsoidal methods) (Kurzhanski and Varaiya, 2000)
- Optimization-based approaches (Hamilton-Jacobi, Level-set method) (Bansal et al., 2017, Mitchell et al., 2002, Herbert et al., 2021)
- Matrix measure-based (Fan et al., 2018, Maidens and Arcak, 2015)

> Most of these classical and general approaches are computationally heavy and are not readily adaptable to data-driven algorithms.

Reachability of dynamical system is an old problem: ~ 1980

Different approaches for approximating reachable sets

- Linear, and piecewise linear systems (Ellipsoidal methods) (Kurzhanski and Varaiya, 2000)
- Optimization-based approaches (Hamilton-Jacobi, Level-set method) (Bansal et al., 2017, Mitchell et al., 2002, Herbert et al., 2021)
- Matrix measure-based (Fan et al., 2018, Maidens and Arcak, 2015)

Most of these classical and general approaches are computationally heavy and are not readily adaptable to data-driven algorithms.

In this talk: use control theoretic tools to develop computationally efficient approaches for reachability

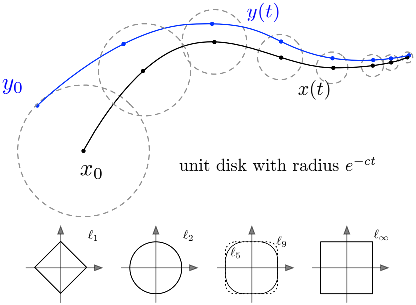
Outline of this talk

- Reachability Analysis
- **Contraction and Monotone Theory**
- Analysis of Learning-enabled Feedbacks

$\dot{x} = f(x, w)$ is contracting wrt $\|\cdot\|$ with rate c if
the dist between every two traj is decreasing/increasing with exp rate c wrt $\|\cdot\|$

Applications

- convergence to reference trajectories
- efficient equilibrium point computation
- input-output robustness
- entrainment to periodic orbits



In this talk: contraction theory for reachability analysis

How to characterize contractivity using vector fields?

Matrix measure

Given a matrix $A \in \mathbb{R}^{n\times n}$ and a norm $\|\cdot\|$:

$$\mu_{\|\cdot\|}(A) := \lim_{h\to 0^+} \frac{\|I_n + hA\| - 1}{h}$$

Given $\eta \in \mathbb{R}^n_{\geq 0}$

$$\mu_{2,\eta}(A) = \tfrac{1}{2}\lambda_{\max}(\mathrm{diag}(\eta)A + A^\top \mathrm{diag}(\eta))$$

$$\mu_{1,\eta}(A) = \max_j \left(a_{jj} + \sum\nolimits_{i\neq j} |a_{ij}|\tfrac{\eta_j}{\eta_i}\right)$$

$$\mu_{\infty,\eta}(A) = \max_i \left(a_{ii} + \sum\nolimits_{j\neq i} |a_{ij}|\tfrac{\eta_j}{\eta_i}\right)$$

- directional derivative of matrix norm $\|\cdot\|$ in direction of A at point I_n,
- **In the literature**: one-sided Lipschitz constant, logarithmic norm

Classical result

$\dot{x} = f(x,w)$ is contracting wrt $\|\cdot\|$ with rate c iff

$$\mu_{\|\cdot\|}\left(\tfrac{\partial f}{\partial x}(x,w)\right) \leq c, \qquad \text{for all } x, w$$

Approach #1: Contraction-based Reachability

A global bound

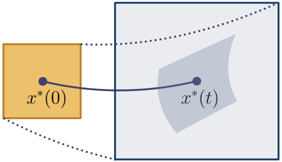
Assume $\mu_{\|\cdot\|}\left(\frac{\partial f}{\partial x}(x,w)\right) \leq c$ and $\left\|\frac{\partial f}{\partial w}(x,w)\right\| \leq \ell$

Theorem

If $\mathcal{X}_0 = B_{\|\cdot\|}(r_1, x_0^*)$ and $\mathcal{W} = B_{\|\cdot\|}(r_2, w^*)$, then

$$\mathcal{R}_f(t,\mathcal{X}_0) \subseteq B_{\|\cdot\|}(e^{ct} r_1 + \tfrac{\ell}{c}(e^{ct}-1) r_2, x^*(t))$$

where $x^*(\cdot)$ is the solution of $\dot{x} = f(x,w^*)$ with $x(0) = x_0^*$.



$$D^+\|x(t) - x^*(t)\| \leq c\|x(t) - x^*(t)\| + \ell\|w(t) - w^*\|$$

- generalized version of Grönwall's lemma
- overly conservative since c and ℓ are defined globally

A dynamical system $\dot{x} = f(x, w)$ is monotone[3] if

$$x_u(0) \leq y_w(0) \quad \text{and} \quad u \leq w \quad \Longrightarrow \quad x_u(t) \leq y_w(t) \quad \text{for all time}$$

where $\leq$ is the component-wise partial order.

[3] Angeli and Sontag, "Monotone control systems", IEEE TAC, 2003

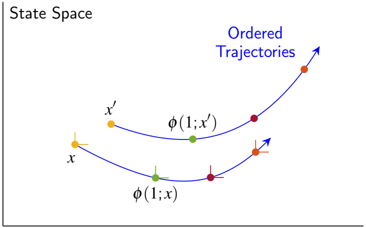
A dynamical system $\dot{x} = f(x, w)$ is monotone[3] if

$$x_u(0) \leq y_w(0) \quad \text{and} \quad u \leq w \quad \Longrightarrow \quad x_u(t) \leq y_w(t) \quad \text{for all time}$$

where $\leq$ is the component-wise partial order.

Monotonicity test

1. $\frac{\partial f}{\partial x}(x, w)$ is Metzler (off-diag ≥ 0)
2. $\frac{\partial f}{\partial w}(x, w) \geq 0$



[3] Angeli and Sontag, "Monotone control systems", IEEE TAC, 2003

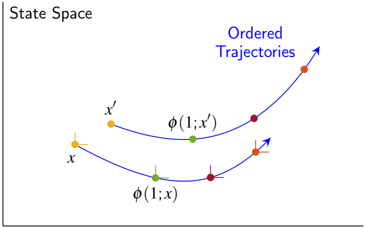
A dynamical system $\dot{x} = f(x, w)$ is monotone[3] if

$$x_u(0) \leq y_w(0) \quad \text{and} \quad u \leq w \quad \Longrightarrow \quad x_u(t) \leq y_w(t) \quad \text{for all time}$$

where $\leq$ is the component-wise partial order.

Monotonicity test

1. $\frac{\partial f}{\partial x}(x, w)$ is Metzler (off-diag ≥ 0)
2. $\frac{\partial f}{\partial w}(x, w) \geq 0$



In this talk: monotone system theory for reachability analysis

[3] Angeli and Sontag, "Monotone control systems", IEEE TAC, 2003

Theorem (classical result)

For a monotone system with $\mathcal{W} = [\underline{w}, \overline{w}]$

$$\mathcal{R}_f(t, [\underline{x}_0, \overline{x}_0]) \subseteq [x_{\underline{w}}(t), x_{\overline{w}}(t)]$$

where $x_{\underline{w}}(\cdot)$ (resp. $x_{\overline{w}}(\cdot)$) is the trajectory with disturbance $\underline{w}$ (resp. $\overline{w}$) starting at $\underline{x}_0$ (resp. $\overline{x}_0$)

Theorem (classical result)

For a monotone system with $\mathcal{W} = [\underline{w}, \overline{w}]$

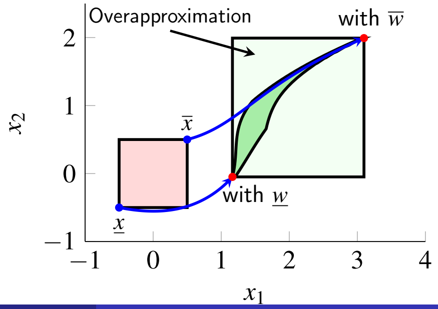
$$\mathcal{R}_f(t, [\underline{x}_0, \overline{x}_0]) \subseteq [x_{\underline{w}}(t), x_{\overline{w}}(t)]$$

where $x_{\underline{w}}(\cdot)$ (resp. $x_{\overline{w}}(\cdot)$) is the trajectory with disturbance $\underline{w}$ (resp. $\overline{w}$) starting at $\underline{x}_0$ (resp. $\overline{x}_0$)

Example:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2^3 - x_1 + w \\ x_1 \end{bmatrix}$$

$$\mathcal{W} = [2.2\ ,\ 2.3] \quad \mathcal{X}_0 = \left[\begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}, \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}\right]$$



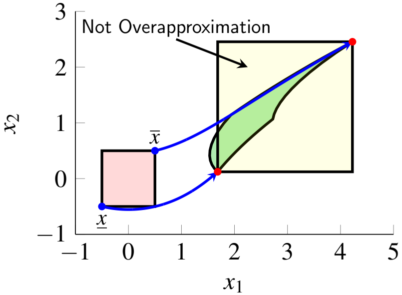
- For non-monotone dynamical systems the extreme trajectories do not provide any over-approximation of reachable sets

- For non-monotone dynamical systems the extreme trajectories do not provide any over-approximation of reachable sets

Example:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2^3 - x_2 + w \\ x_1 \end{bmatrix}$$

$$\mathcal{W} = [2.2\ ,\ 2.3] \quad \mathcal{X}_0 = \left[\begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}, \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}\right]$$



Approach #2: Mixed Monotone Theory

Embedding into a higher dimensional system

- **Key idea:** embed the dynamical system on $\mathbb{R}^n$ into a dynamical system on $\mathbb{R}^{2n}$
- Assume $\mathcal{W} = [\underline{w}, \overline{w}]$ and $\mathcal{X}_0 = [\underline{x}_0, \overline{x}_0]$

Original system

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

Embedding system

$$\dot{\underline{x}} = \underline{d}(\underline{x}, \overline{x}, \underline{w}, \overline{w}),$$
$$\dot{\overline{x}} = \overline{d}(\underline{x}, \overline{x}, \underline{w}, \overline{w})$$

$\underline{d}, \overline{d}$ are **decomposition functions** s.t.

1. $f(x, w) = \underline{d}(x, x, w, w)$ for every x, w
2. cooperative: $(\underline{x}, \underline{w}) \mapsto \underline{d}(\underline{x}, \overline{x}, \underline{w}, \overline{w})$
3. competitive: $(\overline{x}, \overline{w}) \mapsto \underline{d}(\underline{x}, \overline{x}, \underline{w}, \overline{w})$
4. the same properties for $\overline{d}$

Approach #2: Mixed Monotone Theory

Embedding into a higher dimensional system

- **Key idea:** embed the dynamical system on $\mathbb{R}^n$ into a dynamical system on $\mathbb{R}^{2n}$
- Assume $\mathcal{W} = [\underline{w}, \overline{w}]$ and $\mathcal{X}_0 = [\underline{x}_0, \overline{x}_0]$

Original system

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

Embedding system

$$\dot{\underline{x}} = \underline{d}(\underline{x}, \overline{x}, \underline{w}, \overline{w}),$$
$$\dot{\overline{x}} = \overline{d}(\underline{x}, \overline{x}, \underline{w}, \overline{w})$$

$\underline{d}, \overline{d}$ are **decomposition functions** s.t.

1. $f(x, w) = \underline{d}(x, x, w, w)$ for every x, w
2. cooperative: $(\underline{x}, \underline{w}) \mapsto \underline{d}(\underline{x}, \overline{x}, \underline{w}, \overline{w})$
3. competitive: $(\overline{x}, \overline{w}) \mapsto \underline{d}(\underline{x}, \overline{x}, \underline{w}, \overline{w})$
4. the same properties for $\overline{d}$

The embedding system is a monotone dynamical system on $\mathbb{R}^{2n}$ with respect to the **southeast** partial order $\leq_{\mathrm{SE}}$:

$$\begin{bmatrix} x \\ \widehat{x} \end{bmatrix} \leq_{\mathrm{SE}} \begin{bmatrix} y \\ \widehat{y} \end{bmatrix} \iff x \leq y \quad \text{and} \quad \widehat{y} \leq \widehat{x}$$

- f locally Lipschitz $\implies$ a decomposition function exists

The best (tightest) decomposition function is given by

$$\underline{d}_i(\underline{x}, \overline{x}, \underline{w}, \overline{w}) = \min_{\substack{z \in [\underline{x}, \overline{x}], z_i = \underline{x}_i \\ u \in [\underline{w}, \overline{w}]}} f_i(z, u), \qquad \overline{d}_i(\underline{x}, \overline{x}, \underline{w}, \overline{w}) = \max_{\substack{z \in [\underline{x}, \overline{x}], z_i = \overline{x}_i \\ u \in [\underline{w}, \overline{w}]}} f_i(z, u)$$

- f locally Lipschitz $\implies$ a decomposition function exists

The best (tightest) decomposition function is given by

$$\underline{d}_i(\underline{x}, \overline{x}, \underline{w}, \overline{w}) = \min_{\substack{z \in [\underline{x}, \overline{x}], z_i = \underline{x}_i \\ u \in [\underline{w}, \overline{w}]}} f_i(z, u), \qquad \overline{d}_i(\underline{x}, \overline{x}, \underline{w}, \overline{w}) = \max_{\substack{z \in [\underline{x}, \overline{x}], z_i = \overline{x}_i \\ u \in [\underline{w}, \overline{w}]}} f_i(z, u)$$

A short (and incomplete) history:

J-L. Gouze and L. P. Hadeler. Monotone flows and order intervals. Nonlinear World, 1994

G. Enciso, H. Smith, and E. Sontag. Nonmonotone systems decomposable into monotone systems with negative feedback . Journal of Differential Equations, 2006.

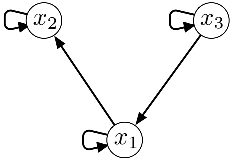
H. Smith. Global stability for mixed monotone systems. Journal of Difference Equations and Applications, 2008

- **Metzler/non-Metzler** decomposition: $A = \lceil A \rceil^{\mathrm{Mzl}} + \lfloor A \rfloor^{\mathrm{Mzl}}$
- Example: $A = \begin{bmatrix} 2 & 0 & -1 \\ 1 & -3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \implies \lceil A \rceil^{\mathrm{Mzl}} = \begin{bmatrix} 2 & 0 & 0 \\ 1 & -3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \lfloor A \rfloor^{\mathrm{Mzl}} = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

Linear systems

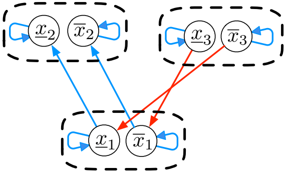
Original system

$$\dot{x} = Ax + Bw$$



Embedding system

$$\dot{\underline{x}} = \lceil A \rceil^{\mathrm{Mzl}} \underline{x} + \lfloor A \rfloor^{\mathrm{Mzl}} \overline{x} + B^{+} \underline{w} + B^{-} \overline{w}$$
$$\dot{\overline{x}} = \lceil A \rceil^{\mathrm{Mzl}} \overline{x} + \lfloor A \rfloor^{\mathrm{Mzl}} \underline{x} + B^{+} \overline{w} + B^{-} \underline{w}$$



Approach #2: Reachability using Embedding Systems

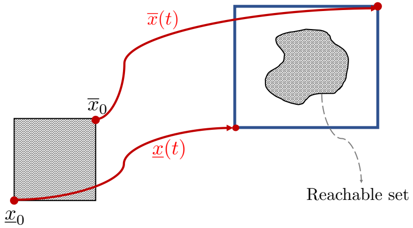
Hyper-rectangular over-approximations

Theorem[4]

Assume $\mathcal{W} = [\underline{w}, \overline{w}]$ and $\mathcal{X}_0 = [\underline{x}_0, \overline{x}_0]$ and

$$\begin{aligned} \dot{\underline{x}} &= \underline{d}(\underline{x}, \overline{x}, \underline{w}, \overline{w}), & \underline{x}(0) &= \underline{x}_0 \\ \dot{\overline{x}} &= \overline{d}(\overline{x}, \underline{x}, \overline{w}, \underline{w}), & \overline{x}(0) &= \overline{x}_0 \end{aligned}$$

Then $\mathcal{R}_f(t, \mathcal{X}_0) \subseteq [\underline{x}(t), \overline{x}(t)]$



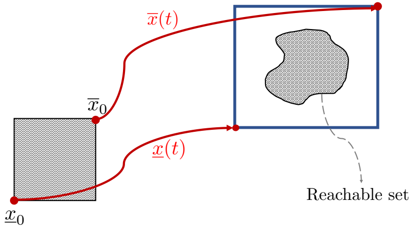
[4] Coogan and Arcak, "Efficient finite abstraction of mixed monotone systems", HSCC, 2015.

Theorem[4]

Assume $\mathcal{W} = [\underline{w}, \overline{w}]$ and $\mathcal{X}_0 = [\underline{x}_0, \overline{x}_0]$ and

$$\dot{\underline{x}} = \underline{d}(\underline{x}, \overline{x}, \underline{w}, \overline{w}), \qquad \underline{x}(0) = \underline{x}_0$$
$$\dot{\overline{x}} = \overline{d}(\overline{x}, \underline{x}, \overline{w}, \underline{w}), \qquad \overline{x}(0) = \overline{x}_0$$

Then $\mathcal{R}_f(t, \mathcal{X}_0) \subseteq [\underline{x}(t), \overline{x}(t)]$



(Scalable) a single trajectory of embedding system provides **lower bound** ($\underline{x}$) and **upper bound** ($\overline{x}$) for the trajectories of the original system.

[4] Coogan and Arcak, "Efficient finite abstraction of mixed monotone systems", HSCC, 2015.

Original System:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2^3 - x_2 + w \\ x_1 \end{bmatrix}$$

$$\mathcal{W} = [2.2\ ,\ 2.3] \quad \mathcal{X}_0 = \left[\begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}, \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}\right]$$

blue = cooperative, red = competitive

Decomposition function

$$\underline{d}(\underline{x}, \overline{x}, \underline{w}, \overline{w}) = \begin{bmatrix} \underline{x}_2^3 + \underline{w} \\ \underline{x}_1 \end{bmatrix} + \begin{bmatrix} -\overline{x}_2 \\ 0 \end{bmatrix}$$

$$\overline{d}(\underline{x}, \overline{x}, \underline{w}, \overline{w}) = \begin{bmatrix} \overline{x}_2^3 + \overline{w} \\ \overline{x}_1 \end{bmatrix} + \begin{bmatrix} -\underline{x}_2 \\ 0 \end{bmatrix}$$

Original System:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2^3 - x_2 + w \\ x_1 \end{bmatrix}$$

$$\mathcal{W} = [2.2\ ,\ 2.3] \quad \mathcal{X}_0 = \left[\begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}, \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}\right]$$

blue = cooperative, red = competitive

Embedding System:

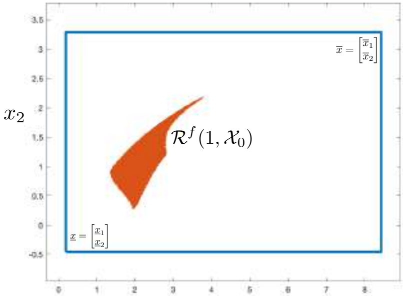
$$\frac{d}{dt}\begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \\ \overline{x}_1 \\ \overline{x}_2 \end{bmatrix} = \begin{bmatrix} \underline{x}_2^3 - \overline{x}_2 + \underline{w} \\ \underline{x}_1 \\ \overline{x}_2^3 - \underline{x}_2 + \overline{w} \\ \overline{x}_1 \end{bmatrix} \qquad \begin{bmatrix} \underline{w} \\ \overline{w} \end{bmatrix} = \begin{bmatrix} 2.2 \\ 2.3 \end{bmatrix}$$

$$\begin{bmatrix} \underline{x}_1(0) \\ \underline{x}_2(0) \end{bmatrix} = \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix} \qquad \begin{bmatrix} \overline{x}_1(0) \\ \overline{x}_2(0) \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

Decomposition function

$$\underline{d}(\underline{x}, \overline{x}, \underline{w}, \overline{w}) = \begin{bmatrix} \underline{x}_2^3 + \underline{w} \\ \underline{x}_1 \end{bmatrix} + \begin{bmatrix} -\overline{x}_2 \\ 0 \end{bmatrix}$$

$$\overline{d}(\underline{x}, \overline{x}, \underline{w}, \overline{w}) = \begin{bmatrix} \overline{x}_2^3 + \overline{w} \\ \overline{x}_1 \end{bmatrix} + \begin{bmatrix} -\underline{x}_2 \\ 0 \end{bmatrix}$$



Which approach is better?

Comparison between contraction and monotone reachability

Question: How to compare contraction and monotone reachability?

- In general these two approaches are not comparable

Contraction Reachability: norm-ball $\mapsto$ norm-ball
Monotone Reachability: hyper-rectangles $\mapsto$ hyper-rectangles

[5] Jafarpour and Coogan, "Monotoncity and contraction on polyhedral cones", under review, 2023

Question: How to compare contraction and monotone reachability?

- when contraction is wrt diagonally weighted ℓ_∞-norm: they are comparable

Theorem[5]

Let $\frac{d}{dt}\begin{bmatrix}\underline{x}\\ \overline{x}\end{bmatrix} = \begin{bmatrix}\underline{d}(\underline{x},\overline{x},\underline{w},\overline{w})\\ \overline{d}(\underline{x},\overline{x},\underline{w},\overline{w})\end{bmatrix} := e(\underline{x},\overline{x},\underline{w},\overline{w})$ be the embedding function with the tight decomposition functions for $\dot{x} = f(x,w)$. For any $\eta \in \mathbb{R}^n_{\geq 0}$

$$\mu_{\infty,\eta}\left(\frac{\partial f}{\partial x}(x,w)\right) \leq c \quad \iff \quad \mu_{\infty,\eta\otimes I_2}\left(\frac{\partial e}{\partial\left[\begin{smallmatrix}\underline{x}\\ \overline{x}\end{smallmatrix}\right]}(\underline{x},\overline{x},\underline{w},\overline{w})\right) \leq c$$

1. Monotone reachability is at least as accurate as contraction reachability
2. Monotone hyper-rectangles shrink/expand with rate of contraction of original system

[5] Jafarpour and Coogan, "Monotoncity and contraction on polyhedral cones", under review, 2023

- Reachability Analysis
- Contraction and Monotone Theory
- **Systems with Learning-enabled Feedbacks**

Given the open-loop nonlinear system with a neural network controller

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

study reachability of the closed-loop system

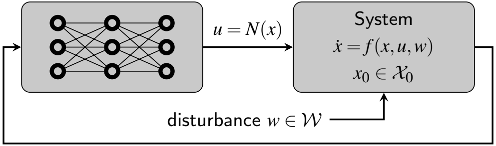
$$\dot{x} = f(x, N(x), w) := f^c(x, w)$$

Given the open-loop nonlinear system with a neural network controller

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

study reachability of the closed-loop system

$$\dot{x} = f(x, N(x), w) := f^c(x, w)$$



$u = N(x)$ is k-layer feed-forward neural net

$$\xi^{(i)}(x) = \phi^{(i)}(W^{(i-1)}\xi^{(i-1)}(x) + b^{(i-1)})$$
$$x = \xi^{(0)}, \quad u = W^{(k)}\xi^{(k)}(x) + b^{(k)} := N(x),$$

Given the open-loop nonlinear system with a neural network controller

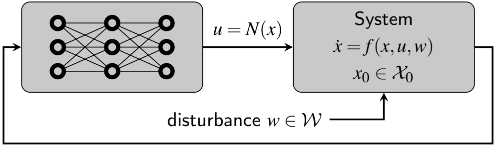
$$\dot{x} = f(x, u, w),$$
$$u = N(x),$$

study reachability of the closed-loop system

$$\dot{x} = f(x, N(x), w) := f^c(x, w)$$

$u = N(x)$ is k-layer feed-forward neural net

$$\xi^{(i)}(x) = \phi^{(i)}(W^{(i-1)}\xi^{(i-1)}(x) + b^{(i-1)})$$
$$x = \xi^{(0)}, \quad u = W^{(k)}\xi^{(k)}(x) + b^{(k)} := N(x),$$



Challenge: directly performing reachability on f^c is complicated

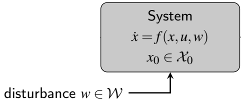
$N(x)$ is high dimensional and has a large # of parameters

Reachability of open-loop system treating u as a parameter

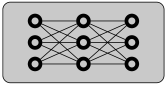
Reachability of open-loop system treating u as a parameter

Neural network verification algorithm for bounds on u

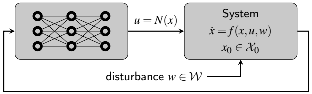
Reachability of open-loop system treating u as a parameter



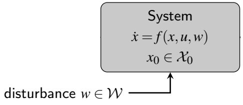
Neural network verification algorithm for bounds on u



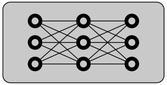
Reachability of open-loop system + Bounds from neural network verification algorithms



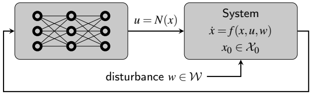
Reachability of open-loop system treating u as a parameter



Neural network verification algorithm for bounds on u



Reachability of open-loop system + Bounds from neural network verification algorithms



If not carefully implemented, it can lead to overly-conservative results.

- Everett and Habibi and Sun, and How, Reachability analysis of neural feedback loops, IEEE Access, 2021
- Hu and Fazlyab and Morari and Pappas,Reach-SDP: Reach- ability analysis of closed-loop systems with neural network controllers via semidefinite programming, CDC, 2020.
- Huang and Fan and Chen and Li and Zh, POLAR: A polynomial arithmetic framework for verifying neural-network controlled systems, ATVA, 2022.
- Schilling and Forets, and Guadalup, Verification of neural- network control systems by integrating Taylor models and zonotope, AAAI, 2022

- Everett and Habibi and Sun, and How, Reachability analysis of neural feedback loops, IEEE Access, 2021
- Hu and Fazlyab and Morari and Pappas,Reach-SDP: Reach- ability analysis of closed-loop systems with neural network controllers via semidefinite programming, CDC, 2020.
- Huang and Fan and Chen and Li and Zh, POLAR: A polynomial arithmetic framework for verifying neural-network controlled systems, ATVA, 2022.
- Schilling and Forets, and Guadalup, Verification of neural- network control systems by integrating Taylor models and zonotope, AAAI, 2022

The existing approaches in the literature are either

- only applicable to a specific class of systems and learning algorithms
- computationally burdensome

- Everett and Habibi and Sun, and How, Reachability analysis of neural feedback loops, IEEE Access, 2021
- Hu and Fazlyab and Morari and Pappas,Reach-SDP: Reach- ability analysis of closed-loop systems with neural network controllers via semidefinite programming, CDC, 2020.
- Huang and Fan and Chen and Li and Zh, POLAR: A polynomial arithmetic framework for verifying neural-network controlled systems, ATVA, 2022.
- Schilling and Forets, and Guadalup, Verification of neural- network control systems by integrating Taylor models and zonotope, AAAI, 2022

The existing approaches in the literature are either

- only applicable to a specific class of systems and learning algorithms
- computationally burdensome

In this talk: computationally efficient reachability using monotone theory
a system theoretic perspective toward composition

Jacobian-based: $\dot{x} = f(x,u)$ such that $\frac{\partial f}{\partial x} \in [\underline{J}_{[\underline{x},\overline{x}]}, \overline{J}_{[\underline{x},\overline{x}]}]$ and $\frac{\partial f}{\partial u} \in [\underline{J}_{[\underline{u},\overline{u}]}, \overline{J}_{[\underline{u},\overline{u}]}]$, then

$$\begin{bmatrix} \underline{d}(\underline{x},\overline{x},\underline{u},\overline{u}) \\ \overline{d}(\underline{x},\overline{x},\underline{u},\overline{u}) \end{bmatrix} = \begin{bmatrix} -[\underline{J}_{[\underline{x},\overline{x}]}]^- & [\underline{J}_{[\underline{x},\overline{x}]}]^- \\ -[\overline{J}_{[\underline{x},\overline{x}]}]^+ & [\overline{J}_{[\underline{x},\overline{x}]}]^+ \end{bmatrix} \begin{bmatrix} \underline{x} \\ \overline{x} \end{bmatrix} + \begin{bmatrix} -[\underline{J}_{[\underline{u},\overline{u}]}]^- & [\underline{J}_{[\underline{u},\overline{u}]}]^- \\ -[\overline{J}_{[\underline{u},\overline{u}]}]^+ & [\overline{J}_{[\underline{u},\overline{u}]}]^+ \end{bmatrix} \begin{bmatrix} \underline{u} \\ \overline{u} \end{bmatrix} + \begin{bmatrix} f(\underline{x},\underline{u}) \\ f(\underline{x},\underline{u}) \end{bmatrix}$$

is a decomposition function for $\dot{x} = f(x,u)$.

[6] Harapanahalli, Jafarpour, Coogan. "A Toolbox for Fast Interval Arithmetic in numpy with an Application to Formal Verification of Neural Network Controlled Systems", 2nd WFVML, ICML, 2023

Jacobian-based: $\dot{x} = f(x,u)$ such that $\frac{\partial f}{\partial x} \in [\underline{J}_{[\underline{x},\overline{x}]}, \overline{J}_{[\underline{x},\overline{x}]}]$ and $\frac{\partial f}{\partial u} \in [\underline{J}_{[\underline{u},\overline{u}]}, \overline{J}_{[\underline{u},\overline{u}]}]$, then

$$\begin{bmatrix} \underline{d}(\underline{x},\overline{x},\underline{u},\overline{u}) \\ \overline{d}(\underline{x},\overline{x},\underline{u},\overline{u}) \end{bmatrix} = \begin{bmatrix} -[\underline{J}_{[\underline{x},\overline{x}]}]^{-} & [\underline{J}_{[\underline{x},\overline{x}]}]^{-} \\ -[\overline{J}_{[\underline{x},\overline{x}]}]^{+} & [\overline{J}_{[\underline{x},\overline{x}]}]^{+} \end{bmatrix} \begin{bmatrix} \underline{x} \\ \overline{x} \end{bmatrix} + \begin{bmatrix} -[\underline{J}_{[\underline{u},\overline{u}]}]^{-} & [\underline{J}_{[\underline{u},\overline{u}]}]^{-} \\ -[\overline{J}_{[\underline{u},\overline{u}]}]^{+} & [\overline{J}_{[\underline{u},\overline{u}]}]^{+} \end{bmatrix} \begin{bmatrix} \underline{u} \\ \overline{u} \end{bmatrix} + \begin{bmatrix} f(\underline{x},\underline{u}) \\ f(\underline{x},\underline{u}) \end{bmatrix}$$

is a decomposition function for $\dot{x} = f(x,u)$.

- Interval arithmetic allows computing Jacobian bounds efficiently.

[6] Harapanahalli, Jafarpour, Coogan. "A Toolbox for Fast Interval Arithmetic in numpy with an Application to Formal Verification of Neural Network Controlled Systems", 2nd WFVML, ICML, 2023

Reachability of Open-loop System

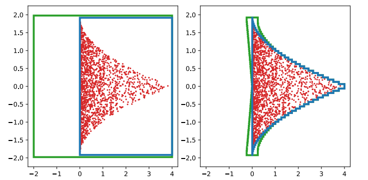
A Constructive Approach

Jacobian-based: $\dot{x} = f(x,u)$ such that $\frac{\partial f}{\partial x} \in [\underline{J}_{[\underline{x},\overline{x}]}, \overline{J}_{[\underline{x},\overline{x}]}]$ and $\frac{\partial f}{\partial u} \in [\underline{J}_{[\underline{u},\overline{u}]}, \overline{J}_{[\underline{u},\overline{u}]}]$, then

$$\begin{bmatrix} \underline{d}(\underline{x},\overline{x},\underline{u},\overline{u}) \\ \overline{d}(\underline{x},\overline{x},\underline{u},\overline{u}) \end{bmatrix} = \begin{bmatrix} -[\underline{J}_{[\underline{x},\overline{x}]}]^{-} & [\overline{J}_{[\underline{x},\overline{x}]}]^{-} \\ -[\underline{J}_{[\underline{x},\overline{x}]}]^{+} & [\overline{J}_{[\underline{x},\overline{x}]}]^{+} \end{bmatrix} \begin{bmatrix} \underline{x} \\ \overline{x} \end{bmatrix} + \begin{bmatrix} -[\underline{J}_{[\underline{u},\overline{u}]}]^{-} & [\overline{J}_{[\underline{u},\overline{u}]}]^{-} \\ -[\underline{J}_{[\underline{u},\overline{u}]}]^{+} & [\overline{J}_{[\underline{u},\overline{u}]}]^{+} \end{bmatrix} \begin{bmatrix} \underline{u} \\ \overline{u} \end{bmatrix} + \begin{bmatrix} f(\underline{x},\underline{u}) \\ f(\underline{x},\underline{u}) \end{bmatrix}$$

is a decomposition function for $\dot{x} = f(x,u)$.

- Interval arithmetic allows computing Jacobian bounds efficiently.
- `npinterval`[6]: Toolbox that implements intervals as native data-type in `numpy`.



$g(x_1,x_2) = [(x_1+x_2)^2, 4\sin((x_1-x_2)/4)]^T$

vs.

$g(x_1,x_2) =$
$[x_2^2 + 2x_1x_2 + x_2^2, 4\sin(x_1/4)\cos(x_2/4) - 4\cos(x_1/4)\sin(x_2/4)]^T$

[6] Harapanahalli, Jafarpour, Coogan. "A Toolbox for Fast Interval Arithmetic in numpy with an Application to Formal Verification of Neural Network Controlled Systems", 2nd WFVML, ICML, 2023

Input-output bounds: Given a neural network controller $u = N(x)$

$$\underline{u}_{[\underline{x},\overline{x}]} \leq N(x) \leq \overline{u}_{[\underline{x},\overline{x}]}, \quad \text{for all } x \in [\underline{x}, \overline{x}]$$

[7]Zhang, Weng, Chen, Hsieh, Daniel. "Efficient neural network robustness certification with general activation functions." NeurIPS, 2018.

Input-output bounds: Given a neural network controller $u = N(x)$

$$\underline{u}_{[\underline{x},\overline{x}]} \leq N(x) \leq \overline{u}_{[\underline{x},\overline{x}]}, \qquad \text{for all } x \in [\underline{x}, \overline{x}]$$

Neural network verification algorithms can produce these bounds (CROWN, LipSDP, IBP, etc)

[7] Zhang, Weng, Chen, Hsieh, Daniel. "Efficient neural network robustness certification with general activation functions." NeurIPS, 2018.

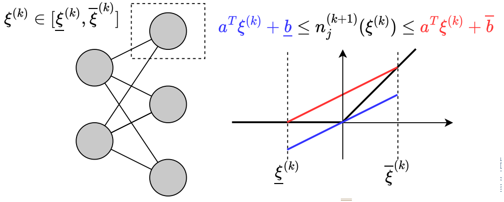
Input-output bounds: Given a neural network controller $u = N(x)$

$$\underline{u}_{[\underline{x},\overline{x}]} \leq N(x) \leq \overline{u}_{[\underline{x},\overline{x}]}, \qquad \text{for all } x \in [\underline{x}, \overline{x}]$$

Neural network verification algorithms can produce these bounds (CROWN, LipSDP, IBP, etc)

CROWN[7]

- Bounding the value of each neurons
- Linear upper and lower bounds on the activation function



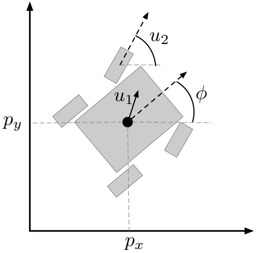
[7] Zhang, Weng, Chen, Hsieh, Daniel. "Efficient neural network robustness certification with general activation functions." NeurIPS, 2018.

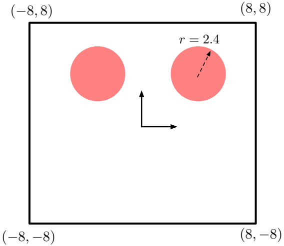
Dynamics of bicycle

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

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

$$\beta(u_2) = \arctan\left(\frac{l_r}{l_f + l_r}\tan(u_2)\right)$$



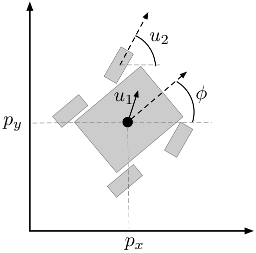


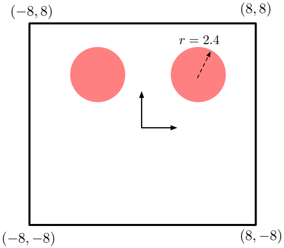
Dynamics of bicycle

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

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

$$\beta(u_2) = \arctan\left(\frac{l_r}{l_f + l_r}\tan(u_2)\right)$$





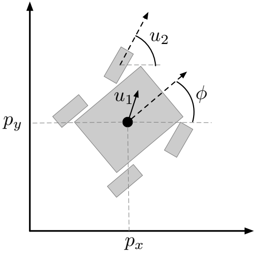
Goal: steer the bicycle to the origin avoiding the obstacles

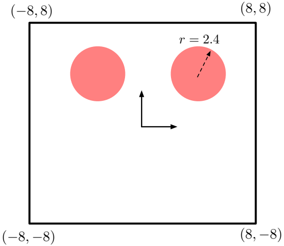
Dynamics of bicycle

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

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

$$\beta(u_2) = \arctan\left(\frac{l_r}{l_f + l_r}\tan(u_2)\right)$$





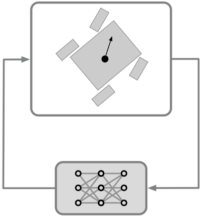
Goal: steer the bicycle to the origin avoiding the obstacles

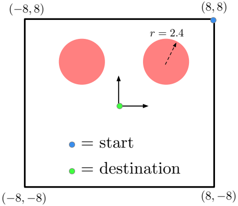
- train a feedforward neural network $4 \mapsto 100 \mapsto 100 \mapsto 2$ with ReLU activations using data from model predictive control

- start from $(8,8)$ toward $(0,0)$
- $\mathcal{X}_0 = [\underline{x}_0, \overline{x}_0]$ with

$$\underline{x}_0 = \begin{pmatrix} 7.95 & 7.95 & -\frac{\pi}{3} - 0.01 & 1.99 \end{pmatrix}^\top$$
$$\overline{x}_0 = \begin{pmatrix} 8.05 & 8.05 & -\frac{\pi}{3} + 0.01 & 2.01 \end{pmatrix}^\top$$

- CROWN for verification of neural network





Embedding system:

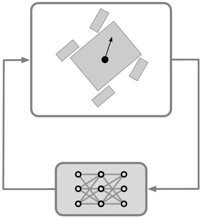
$$\dot{\underline{x}} = \underline{d}(\underline{x}, \overline{x}, \underline{u}, \overline{u}, \underline{w}, \overline{w})$$
$$\dot{\overline{x}} = \overline{d}(\underline{x}, \overline{x}, \underline{u}, \overline{u}, \underline{w}, \overline{w})$$

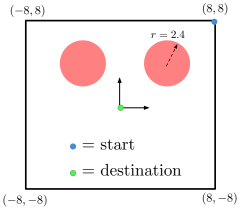
$\underline{u} \le N(x) \le \overline{u}$, for every $x \in [\underline{x}, \overline{x}]$.

- start from $(8, 8)$ toward $(0, 0)$
- $\mathcal{X}_0 = [\underline{x}_0, \overline{x}_0]$ with

$$\underline{x}_0 = \begin{pmatrix} 7.95 & 7.95 & -\frac{\pi}{3} - 0.01 & 1.99 \end{pmatrix}^\top$$
$$\overline{x}_0 = \begin{pmatrix} 8.05 & 8.05 & -\frac{\pi}{3} + 0.01 & 2.01 \end{pmatrix}^\top$$

- CROWN for verification of neural network

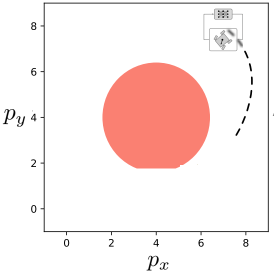


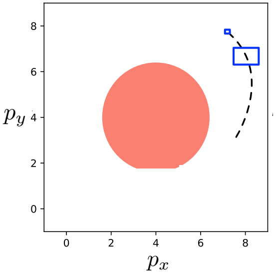


Euler integration with step h:

$$\underline{x}_1 = \underline{x}_0 + h\underline{d}(\underline{x}_0, \overline{x}_0, \underline{u}_0, \overline{u}_0, \underline{w}, \overline{w})$$
$$\overline{x}_1 = \overline{x}_0 + h\overline{d}(\underline{x}_0, \overline{x}_0, \underline{u}_0, \overline{u}_0, \underline{w}, \overline{w})$$

$\underline{u}_0 \leq N(x) \leq \overline{u}_0$, for every $x \in [\underline{x}_0, \overline{x}_0]$.

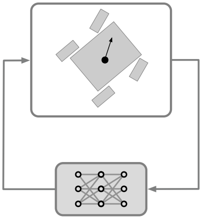


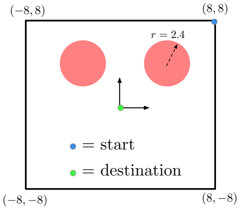


- start from $(8, 8)$ toward $(0, 0)$
- $\mathcal{X}_0 = [\underline{x}_0, \overline{x}_0]$ with

$$\underline{x}_0 = \begin{pmatrix} 7.95 & 7.95 & -\frac{\pi}{3} - 0.01 & 1.99 \end{pmatrix}^\top$$
$$\overline{x}_0 = \begin{pmatrix} 8.05 & 8.05 & -\frac{\pi}{3} + 0.01 & 2.01 \end{pmatrix}^\top$$

- CROWN for verification of neural network

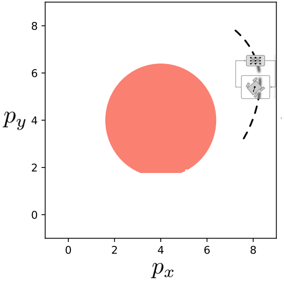


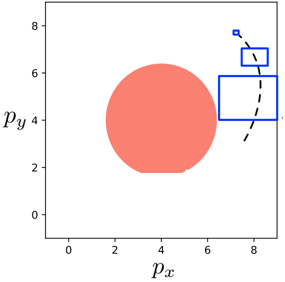


Euler integration with step h:

$$\underline{x}_2 = \underline{x}_1 + h\underline{d}(\underline{x}_1, \overline{x}_1, \underline{u}_1, \overline{u}_1, \underline{w}, \overline{w})$$
$$\overline{x}_2 = \overline{x}_1 + h\overline{d}(\underline{x}_1, \overline{x}_1, \underline{u}_1, \overline{u}_1, \underline{w}, \overline{w})$$

$\underline{u}_1 \leq N(x) \leq \overline{u}_1$, for every $x \in [\underline{x}_1, \overline{x}_1]$.

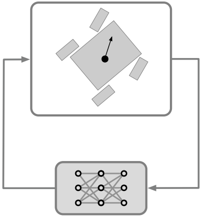


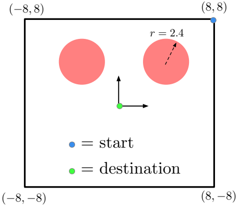


- start from $(8, 8)$ toward $(0, 0)$
- $\mathcal{X}_0 = [\underline{x}_0, \overline{x}_0]$ with

$$\underline{x}_0 = \begin{pmatrix} 7.95 & 7.95 & -\frac{\pi}{3} - 0.01 & 1.99 \end{pmatrix}^\top$$
$$\overline{x}_0 = \begin{pmatrix} 8.05 & 8.05 & -\frac{\pi}{3} + 0.01 & 2.01 \end{pmatrix}^\top$$

- CROWN for verification of neural network

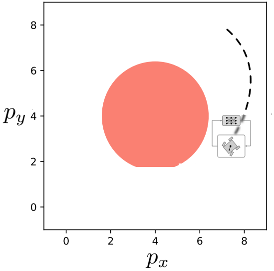


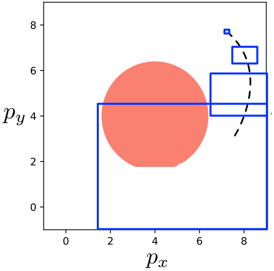


Euler integration with step h:

$$\underline{x}_3 = \underline{x}_2 + h\underline{d}(\underline{x}_2, \overline{x}_2, \underline{u}_2, \overline{u}_2, \underline{w}, \overline{w})$$
$$\overline{x}_3 = \overline{x}_2 + h\overline{d}(\underline{x}_2, \overline{x}_2, \underline{u}_2, \overline{u}_2, \underline{w}, \overline{w})$$

$\underline{u}_2 \leq N(x) \leq \overline{u}_2$, for every $x \in [\underline{x}_2, \overline{x}_2]$.





Neural network controller as **disturbances** (worst-case scenario)
This approach does not capture the **stabilizing** effect of the neural network.

Neural network controller as **disturbances** (worst-case scenario)
This approach does not capture the **stabilizing** effect of the neural network.

An illustrative example

$\dot{x} = x + u + w$ with controller $u = -Kx$, for some unknown $1 < K \leq 3$.

Neural network controller as **disturbances** (worst-case scenario)
This approach does not capture the **stabilizing** effect of the neural network.

An illustrative example

$\dot{x} = x + u + w$ with controller $u = -Kx$, for some unknown $1 < K \leq 3$.

Decomposition #1

First find the bounds $\underline{u} \leq Kx \leq \overline{u}$, then

$$\dot{\underline{x}} = \underline{x} + \underline{u} + \underline{w}$$
$$\dot{\overline{x}} = \overline{x} + \overline{u} + \overline{w}$$

System is unstable with contraction rate 1.

Decomposition #2

First replace $u = Kx$ in the system, then

$$\dot{\underline{x}} = (1 - K)\underline{x} + \underline{w}$$
$$\dot{\overline{x}} = (1 - K)\overline{x} + \overline{w}$$

System is stable with contraction rate $1 - K$.

Neural network controller as **disturbances** (worst-case scenario)
This approach does not capture the **stabilizing** effect of the neural network.

An illustrative example

$\dot{x} = x + u + w$ with controller $u = -Kx$, for some unknown $1 < K \leq 3$.

Decomposition #1

First find the bounds $\underline{u} \leq Kx \leq \overline{u}$, then

$$\dot{\underline{x}} = \underline{x} + \underline{u} + \underline{w}$$
$$\dot{\overline{x}} = \overline{x} + \overline{u} + \overline{w}$$

System is unstable with contraction rate 1.

Decomposition #2

First replace $u = Kx$ in the system, then

$$\dot{\underline{x}} = (1 - K)\underline{x} + \underline{w}$$
$$\dot{\overline{x}} = (1 - K)\overline{x} + \overline{w}$$

System is stable with contraction rate $1 - K$.

Key observation: capture stabilizing effect of neural networks in the original system

Neural network controller as **disturbances** (worst-case scenario)
This approach does not capture the **stabilizing** effect of the neural network.

An illustrative example

$\dot{x} = x + u + w$ with controller $u = -Kx$, for some unknown $1 < K \leq 3$.

Decomposition #1

First find the bounds $\underline{u} \leq Kx \leq \overline{u}$, then

$$\dot{\underline{x}} = \underline{x} + \underline{u} + \underline{w}$$
$$\dot{\overline{x}} = \overline{x} + \overline{u} + \overline{w}$$

System is unstable with contraction rate 1.

Decomposition #2

First replace $u = Kx$ in the system, then

$$\dot{\underline{x}} = (1 - K)\underline{x} + \underline{w}$$
$$\dot{\overline{x}} = (1 - K)\overline{x} + \overline{w}$$

System is stable with contraction rate $1 - K$.

Key observation: capture stabilizing effect of neural networks in the original system

Recall: monotone hyper-rectangles shrink/expand with contraction rate of the original system

We need to know the **functional** dependencies of neural network bounds

[8] Zhang, Weng, Chen, Hsieh, Daniel. "Efficient neural network robustness certification with general activation functions." NeurIPS, 2018.

We need to know the **functional** dependencies of neural network bounds

Functional bounds: Given a neural network controller $u = N(x)$

$$\underline{N}_{[\underline{x},\overline{x}]}(x) \leq N(x) \leq \overline{N}_{[\underline{x},\overline{x}]}(x), \qquad \text{for all } x \in [\underline{x}, \overline{x}]$$

[8] Zhang, Weng, Chen, Hsieh, Daniel. "Efficient neural network robustness certification with general activation functions." NeurIPS, 2018.

We need to know the **functional** dependencies of neural network bounds

Functional bounds: Given a neural network controller $u = N(x)$

$$\underline{N}_{[\underline{x},\overline{x}]}(x) \leq N(x) \leq \overline{N}_{[\underline{x},\overline{x}]}(x), \qquad \text{for all } x \in [\underline{x}, \overline{x}]$$

- Example: CROWN[8] can provide functional bounds.

CROWN functional bounds:

$$\underline{N}_{[\underline{x},\overline{x}]}(x) = \underline{A}_{[\underline{x},\overline{x}]} x + \underline{b}_{[\underline{x},\overline{x}]},$$
$$\overline{N}_{[\underline{x},\overline{x}]}(x) = \overline{A}_{[\underline{x},\overline{x}]} x + \overline{b}_{[\underline{x},\overline{x}]}$$

CROWN input-output bounds:

$$\underline{u}_{[\underline{x},\overline{x}]} = \underline{A}^{+}_{[\underline{x},\overline{x}]} \overline{x} + \overline{A}^{-}_{[\underline{x},\overline{x}]} \underline{x} + \underline{b}_{[\underline{x},\overline{x}]},$$
$$\overline{u}_{[\underline{x},\overline{x}]} = \overline{A}^{+}_{[\underline{x},\overline{x}]} \overline{x} + \underline{A}^{-}_{[\underline{x},\overline{x}]} \underline{x} + \overline{b}_{[\underline{x},\overline{x}]}$$

[8] Zhang, Weng, Chen, Hsieh, Daniel. "Efficient neural network robustness certification with general activation functions." NeurIPS, 2018.

Interaction Approach

A pictorial explanation

Original system:

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

closed-loop system

Embedding system:

$$\begin{bmatrix} \dot{\underline{x}} \\ \dot{\overline{x}} \end{bmatrix} = \begin{bmatrix} [\underline{H}]^{+} - \underline{J}_{[\underline{x},\overline{x}]} & [\underline{H}]^{-} \\ [\overline{H}]^{+} - \overline{J}_{[\underline{x},\overline{x}]} & [\overline{H}]^{-} \end{bmatrix} \begin{bmatrix} \underline{x} \\ \overline{x} \end{bmatrix} + \begin{bmatrix} -[\underline{J}_{[\underline{w},\overline{w}]}]^{-} & [\underline{J}_{[\underline{w},\overline{w}]}]^{+} \\ -[\overline{J}_{[\underline{w},\overline{w}]}]^{-} & [\overline{J}_{[\underline{w},\overline{w}]}]^{+} \end{bmatrix} \begin{bmatrix} \underline{w} \\ \overline{w} \end{bmatrix} + Q$$

closed-loop embedding system

Original system:

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

closed-loop system

Embedding system:

$$\begin{bmatrix} \dot{\underline{x}} \\ \dot{\overline{x}} \end{bmatrix} = \begin{bmatrix} [\underline{H}]^{+} - \underline{J}_{[\underline{x},\overline{x}]} & [\underline{H}]^{-} \\ [\overline{H}]^{+} - \overline{J}_{[\underline{x},\overline{x}]} & [\overline{H}]^{-} \end{bmatrix} \begin{bmatrix} \underline{x} \\ \overline{x} \end{bmatrix} + \begin{bmatrix} -[\underline{J}_{[\underline{w},\overline{w}]}]^{-} & [\underline{J}_{[\underline{w},\overline{w}]}]^{+} \\ -[\overline{J}_{[\underline{w},\overline{w}]}]^{-} & [\overline{J}_{[\underline{w},\overline{w}]}]^{+} \end{bmatrix} \begin{bmatrix} \underline{w} \\ \overline{w} \end{bmatrix} + Q$$

closed-loop embedding system

How does the interaction approach work?

- Closed-loop decomposition function = Jacobian based for $f(x, N(x), w)$.
- Neural Network affine functional bounds
  $\underline{N}_{[\underline{x},\overline{x}]} = \underline{A}_{[\underline{x},\overline{x}]} x + \underline{b}_{[\underline{x},\overline{x}]}$,
  $\overline{N}_{[\underline{x},\overline{x}]} = \overline{A}_{[\underline{x},\overline{x}]} x + \overline{b}_{[\underline{x},\overline{x}]}$
  are used to compute the interactions.

Theorem[9]

Let $\frac{\partial f}{\partial x} \in [\underline{J}_{[\underline{x},\overline{x}]}, \overline{J}_{[\underline{x},\overline{x}]}]$, $\frac{\partial f}{\partial u} \in [\underline{J}_{[\underline{u},\overline{u}]}, \overline{J}_{[\underline{u},\overline{u}]}]$, and $\frac{\partial f}{\partial w} \in [\underline{J}_{[\underline{w},\overline{w}]}, \overline{J}_{[\underline{w},\overline{w}]}]$. Then

$$\begin{bmatrix} \underline{d}_i^c(\underline{x},\overline{x},\underline{w},\overline{w}) \\ \overline{d}_i^c(\underline{x},\overline{x},\underline{w},\overline{w}) \end{bmatrix} = \begin{bmatrix} [\underline{H}]^+ - \underline{J}_{[\underline{x},\overline{x}]} & [\underline{H}]^- \\ [\overline{H}]^+ - \overline{J}_{[\underline{x},\overline{x}]} & [\overline{H}]^- \end{bmatrix} \begin{bmatrix} \underline{x} \\ \overline{x} \end{bmatrix} + \begin{bmatrix} -[\underline{J}_{[\underline{w},\overline{w}]}]^- & [\underline{J}_{[\underline{w},\overline{w}]}]^+ \\ -[\overline{J}_{[\underline{w},\overline{w}]}]^- & [\overline{J}_{[\underline{w},\overline{w}]}]^+ \end{bmatrix} \begin{bmatrix} \underline{w} \\ \overline{w} \end{bmatrix} + Q$$

where

$$\underline{H} = \underline{J}_{[\underline{x},\overline{x}]} + [\underline{J}_{[\underline{u},\overline{u}]}]^+ \underline{A}_{[\underline{x},\overline{x}]} + [\underline{J}_{[\underline{u},\overline{u}]}]^- \overline{A}_{[\underline{x},\overline{x}]}$$
$$\overline{H} = \overline{J}_{[\underline{x},\overline{x}]} + [\underline{J}_{[\underline{u},\overline{u}]}]^+ \overline{A}_{[\underline{x},\overline{x}]} + [\underline{J}_{[\underline{u},\overline{u}]}]^- \underline{A}_{[\underline{x},\overline{x}]}$$

is a decomposition function for the closed-loop system.

[9] Jafarpour, Harapanahalli, Coogan. "Efficient Interaction-aware Interval Reachability of Neural Network Feedback Loops", under review, 2021

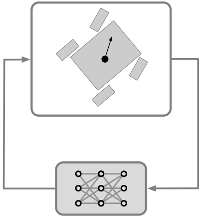
Case Study: Bicycle Model

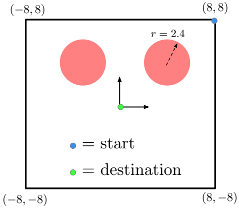
Numerical Experiments

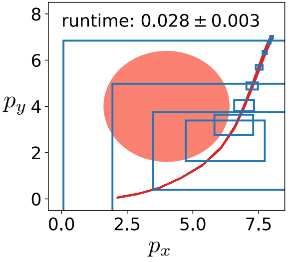
- start from $(8,7)$ toward $(0,0)$
- $\mathcal{X}_0 = [\underline{x}_0, \overline{x}_0]$ with

$$\underline{x}_0 = \begin{pmatrix} 7.95 & 6.95 & -\frac{2\pi}{3} - 0.01 & 1.99 \end{pmatrix}^\top$$
$$\overline{x}_0 = \begin{pmatrix} 8.05 & 7.05 & -\frac{2\pi}{3} + 0.01 & 2.01 \end{pmatrix}^\top$$

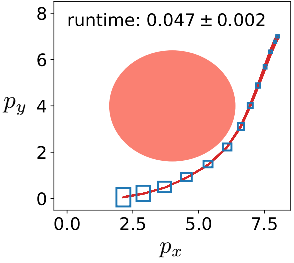
- CROWN for verification of neural network







Naive Composition



Interaction Approach

Dynamics of the jth vehicle

$$\dot{p}_x^j = v_x^j, \qquad \dot{v}_x^j = \tanh(u_x^j) + w_x^j,$$
$$\dot{p}_y^j = v_y^j, \qquad \dot{v}_y^j = \tanh(u_y^j) + w_y^j,$$

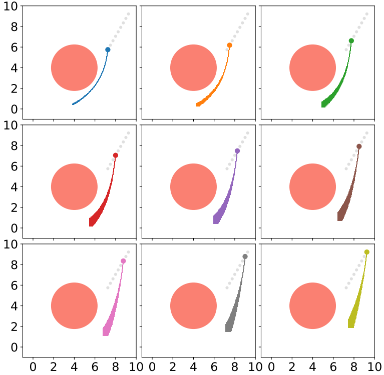
where $w_x^j, w_y^j \sim \mathcal{U}([-0.001, 0.001])$. First vehicle uses a neural network controller

$4 \times 100 \times 100 \times 2$, with ReLU activations

and other vehicles use PD controller

$$u_d^j = k_p\left(p_d^{j-1} - p_d^j - r\frac{v_d^{j-1}}{\|v^{j-1}\|_2}\right) + k_v(v_d^{j-1} - v_d^j),$$

where $d \in \{x, y\}$.



N (units)	# of states	Our Approach (s)	POLAR (s)	JuliaReach (s)
1	4	0.635	9.352	0.224
4	16	1.369	14.182	12.579
9	36	3.144	43.428	59.929
20	80	9.737	316.337	–
50	200	46.426	4256.435	–

Table: Run-time comparison with existing approaches

Conclusions

and follow-up work

- Reachability as a framework for safety certification
- Contraction and monotone theory as computationally efficient methods for reachability
- Reachability of neural network controlled systems
- Contraction theory can capture the interaction between system and neural network controller

Follow-up work: Forward invariance (safety guarantees for infinite time)

Harapanahalli, Jafarpour, and Coogan. Forward Invariance in Neural Network Controlled Systems. IEEE Control Systems Letters, Dec 2023

In monotone theory: uncertainty $w \in \mathcal{W} = [\underline{w}, \overline{w}]$ are treated as **worst-case** using $\underline{w}$ and $\overline{w}$

In monotone theory: uncertainty $w \in \mathcal{W} = [\underline{w}, \overline{w}]$ are treated as **worst-case** using $\underline{w}$ and $\overline{w}$

- In some applications, we can obtain some **statistical** knowledge of uncertainty v.
- In some applications we can **learn** statistics of the uncertainty.

In monotone theory: uncertainty $w \in \mathcal{W} = [\underline{w}, \overline{w}]$ are treated as **worst-case** using $\underline{w}$ and $\overline{w}$

- In some applications, we can obtain some **statistical** knowledge of uncertainty v.
- In some applications we can **learn** statistics of the uncertainty.
- Use data to approximate a probability distribution for the uncertainty $v \sim \mathcal{D}$

Stochastic dynamical system:

$$dx = f(x, w)dt + dv \text{ where } v \sim \mathcal{D}$$

In monotone theory: uncertainty $w \in \mathcal{W} = [\underline{w}, \overline{w}]$ are treated as **worst-case** using $\underline{w}$ and $\overline{w}$

- In some applications, we can obtain some **statistical** knowledge of uncertainty v.
- In some applications we can **learn** statistics of the uncertainty.
- Use data to approximate a probability distribution for the uncertainty $v \sim \mathcal{D}$

Stochastic dynamical system:

$$dx = f(x, w)dt + dv \text{ where } v \sim \mathcal{D}$$

- **Question:** how to incorporate this stochastic uncertainty in neural network algorithms?
- **Question:** how to incorporate this stochastic uncertainty in closed-loop reachability?

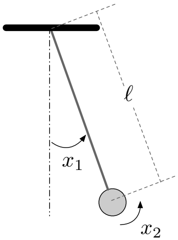
Future Research Directions

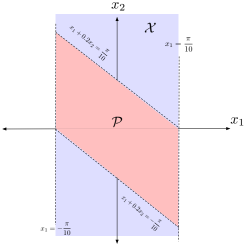
Beyond hyper-rectangular estimates

Monotone theory: hyper-rectangular over-approximation

Monotone theory: hyper-rectangular over-approximation

- In mechanical systems, hyper-rectangular over-approximations are too conservative
- Example: no hyper-rectangular invariant sets for a simple inverted pendulum





A dynamical system $\dot{x} = f(x, w)$ is monotone (with respect to cones K, C) if

$$x_u(0) \preceq_K y_w(0) \quad \text{and} \quad u \preceq_C w \quad \Longrightarrow \quad x_u(t) \preceq_K y_w(t) \quad \text{for all time}$$

where $\preceq_K$ ($\preceq_C$) is the partial order with induced by the cone K (cone C).

A dynamical system $\dot{x} = f(x, w)$ is monotone (with respect to cones K, C) if

$$x_u(0) \preceq_K y_w(0) \quad \text{and} \quad u \preceq_C w \qquad \Longrightarrow \qquad x_u(t) \preceq_K y_w(t) \quad \text{for all time}$$

where $\preceq_K$ ($\preceq_C$) is the partial order with induced by the cone K (cone C).

A **polyhedral cone** has the form

$$K = \underbrace{\{y \in \mathbb{R}^n \mid H_K y \geq \mathbb{0}_p\}}_{\text{halfspace rep}} = \underbrace{\{V_K y \mid y \geq \mathbb{0}_p\}}_{\text{vertex rep}}$$

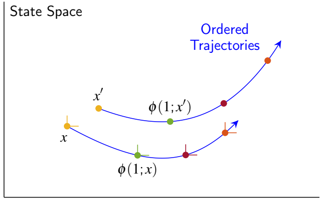
A dynamical system $\dot{x} = f(x, w)$ is monotone (with respect to cones K, C) if

$$x_u(0) \preceq_K y_w(0) \quad \text{and} \quad u \preceq_C w \quad \Longrightarrow \quad x_u(t) \preceq_K y_w(t) \quad \text{for all time}$$

where $\preceq_K$ ($\preceq_C$) is the partial order with induced by the cone K (cone C).

A **polyhedral cone** has the form

$$K = \underbrace{\{y \in \mathbb{R}^n \mid H_K y \geq \mathbb{0}_p\}}_{\text{halfspace rep}} = \underbrace{\{V_K y \mid y \geq \mathbb{0}_p\}}_{\text{vertex rep}}$$



Monotonicity test

1. $H_K(\frac{\partial f}{\partial x}(x, w) + \alpha(x, w) I_n) V_K \geq \mathbb{0}$ for some $\alpha(x, w)$
2. $H_K \frac{\partial f}{\partial w}(x, w) V_C \geq \mathbb{0}$

- **Question:** how to extend to mixed monotone systems?
- **Question:** how to search for the cone with tightest reachable set approximation?
- **Question:** how to incorporate the knowledge of trajectories of the system from data in this approach?

In this talk: verification of neural networks using state-of-the-art algorithms

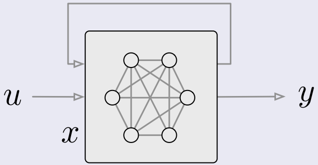
In this talk: verification of neural networks using state-of-the-art algorithms

How to design robust **standalone** neural networks? Input-output **Lipschitz constant**

In this talk: verification of neural networks using state-of-the-art algorithms

How to design robust **standalone** neural networks? Input-output **Lipschitz constant**

Implicit/Recurrent



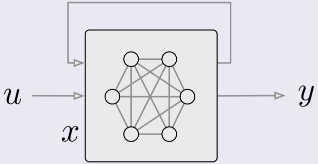
Fixed-point/dynamics

$$x = \Phi(Ax + Bu + b)$$
$$\dot{x} = -x + \Phi(Ax + Bu + b)$$

In this talk: verification of neural networks using state-of-the-art algorithms

How to design robust **standalone** neural networks? Input-output **Lipschitz constant**

Implicit/Recurrent



Fixed-point/dynamics

$$x = \Phi(Ax + Bu + b)$$
$$\dot{x} = -x + \Phi(Ax + Bu + b)$$

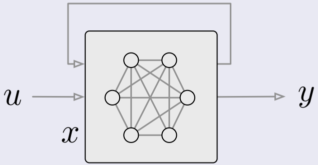
If $\mu_\infty(A) := \max_i(a_{ii} + \sum_{j\neq i} |a_{ij}|) < 1$ then

1. the dynamics is contracting with respect to $\|\cdot\|_\infty$
2. ℓ_∞-norm Lipschitz constant $= \frac{\|C\|_\infty \|B\|_\infty}{1-\mu_\infty(A)} + \|D\|_\infty$

In this talk: verification of neural networks using state-of-the-art algorithms

How to design robust **standalone** neural networks? Input-output **Lipschitz constant**

Implicit/Recurrent



Fixed-point/dynamics

$$x = \Phi(Ax + Bu + b)$$
$$\dot{x} = -x + \Phi(Ax + Bu + b)$$

If $\mu_\infty(A) := \max_i(a_{ii} + \sum_{j\neq i} |a_{ij}|) < 1$ then

1. the dynamics is contracting with respect to $\|\cdot\|_\infty$
2. ℓ_∞-norm Lipschitz constant $= \frac{\|C\|_\infty \|B\|_\infty}{1-\mu_\infty(A)} + \|D\|_\infty$

Closed-form expression for Lipschitz constant to train robust neural networks

- **Question:** measures of robustness for neural networks **in-the-loop**?
- **Question:** impose safety guarantees in training of learning algorithms? ex: forward invariance?