Safety Assurance in Neural Network-Controlled Systems:

A Mixed Monotone Approach

Saber Jafarpour

University of Colorado **Boulder**

Workshop on Formal Verification of Control Systems with Neural Network Components, ACC 2025

Acknowledgment

Akash Harapanahalli
Georgia Tech

Samuel Coogan
Georgia Tech

SJ and A. Harapanahalli and S. Coogan. Interval Reachability of Nonlinear Dynamical Systems with Neural Network Controllers. L4DC, 2023.

SJ and A. Harapanahalli and S. Coogan. Efficient interaction-aware interval analysis of neural network feedback loops. IEEE Transactions on Automatic Control, 2024

A. Harapanahalli and SJ and S. Coogan. `immrax`: A Parallelizable and Differentiable Toolbox for Interval Analysis and Mixed Monotone Reachability in JAX. ADHS 2024

Increase in deployment of learning components in autonomous systems

Increase in deployment of learning components in autonomous systems

- availability of data and computation tools
- performance and efficiency

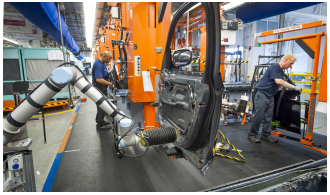
Increase in deployment of learning components in autonomous systems

- availability of data and computation tools
- performance and efficiency

Success stories and potential applications

Self-driving vehicles

Manufacturing



Fulfillment center

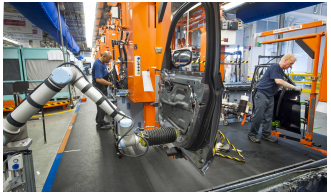
Increase in deployment of learning components in autonomous systems

- availability of data and computation tools
- performance and efficiency

Success stories and potential applications

Self-driving vehicles

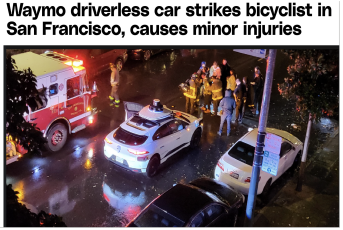
Manufacturing



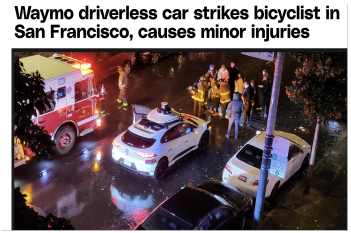
Fulfillment center

Learning-enabled systems are often deployed in **safety-critical** environments

But can we ensure their safety?

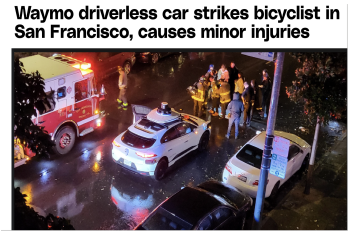


But can we ensure their safety?



What is special about Learning-based components?

But can we ensure their safety?



- highly sensitive to input perturbations

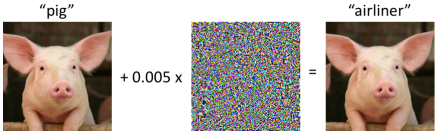
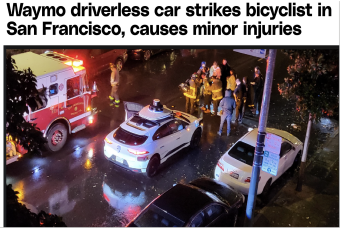
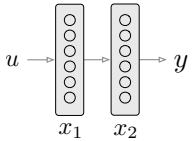


Image credit: MIT CSAIL

But can we ensure their safety?



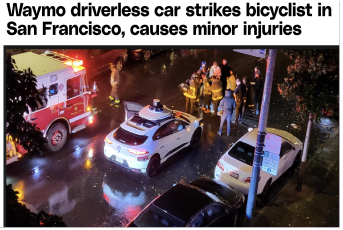
- highly sensitive to input perturbations
- large # of parameters with nonlinearity



$$478 \times 100 \times 100 \times 10$$

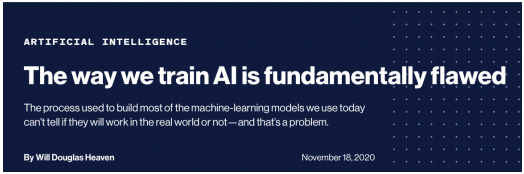
$$\#\text{ of parameters} \sim 90000$$
$$\#\text{ of activation patterns} \sim 10^{60}$$

But can we ensure their safety?

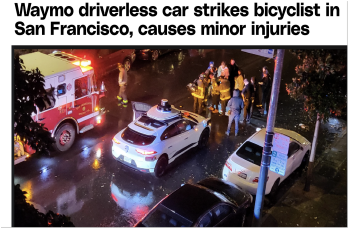


MIT Technology Review

- highly sensitive to input perturbations
- large # of parameters with nonlinearity
- limited guarantee in their design



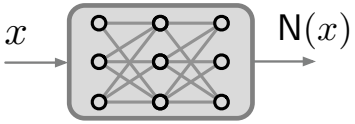
But can we ensure their safety?



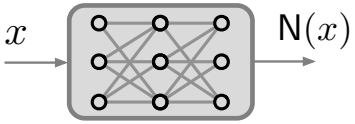
- highly sensitive to input perturbations
- large # of parameters with nonlinearity
- limited guarantee in their design

Scalable and **computationally efficient** methods for safety assurance

ML focus on safety and robustness of **stand-alone** learning algorithms



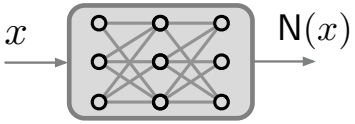
ML focus on safety and robustness of **stand-alone** learning algorithms



Different approaches:

- analysis (Goodfellow et al., 2015, Zhang et al., 2019, Fazlyab et al., 2023)
- design (Papernot et al., 2016, Carlini and Wagner, 2017, Madry et al., 2018)

ML focus on safety and robustness of **stand-alone** learning algorithms

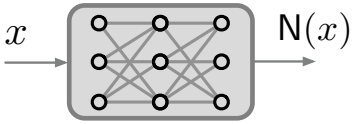


Different approaches:

- analysis (Goodfellow et al., 2015, Zhang et al., 2019, Fazlyab et al., 2023)
- design (Papernot et al., 2016, Carlini and Wagner, 2017, Madry et al., 2018)

In autonomous systems, learning algorithms are **a component of the system**
(controller, motion planner, obstacle detection)

ML focus on safety and robustness of **stand-alone** learning algorithms



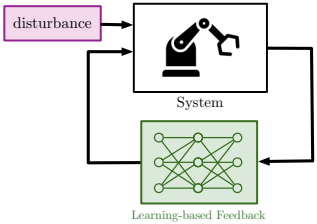
Different approaches:

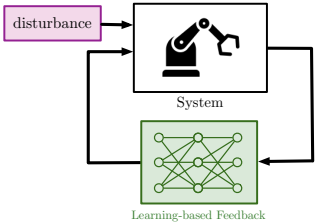
- analysis (Goodfellow et al., 2015, Zhang et al., 2019, Fazlyab et al., 2023)
- design (Papernot et al., 2016, Carlini and Wagner, 2017, Madry et al., 2018)

In autonomous systems, learning algorithms are **a component of the system** (controller, motion planner, obstacle detection)

New challenges arises when learning algorithms are used **in-the-loop**

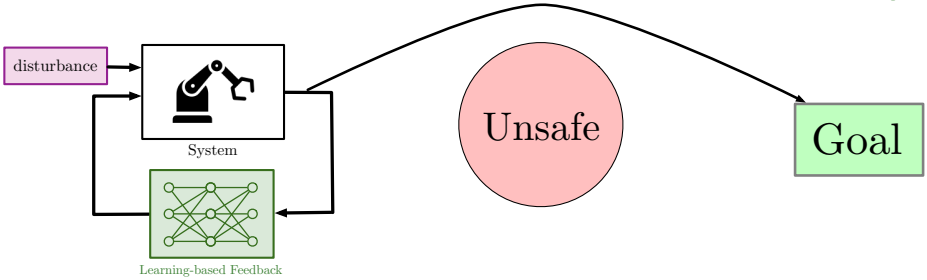
- **In this talk:** Learning-based component is a controller

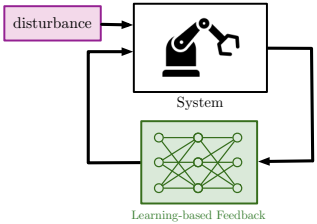




- **In this talk:** Learning-based component is a controller

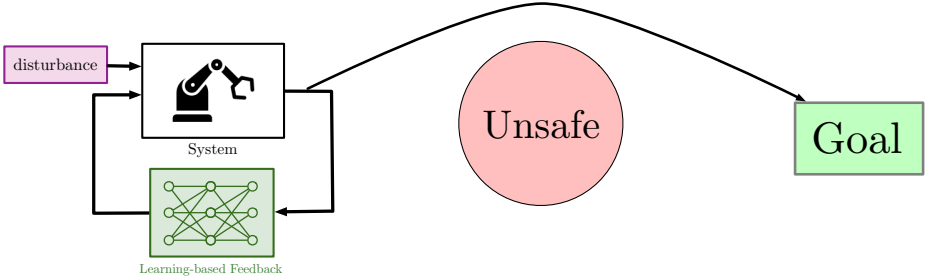
Goal: Ensure safety of the closed-loop system





- **In this talk:** Learning-based component is a controller

Goal: Ensure safety of the closed-loop system

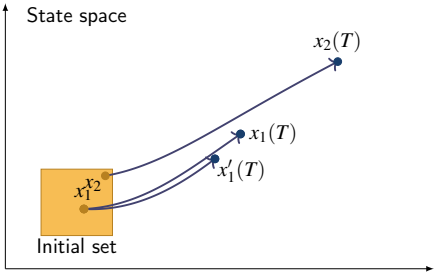


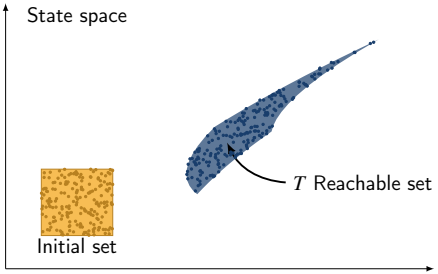
Safety of learning-based controlled systems using **reachability analysis**

Outline of this talk

- **Safety via Reachability Analysis**
- Mixed Monotone Reachability
- Neural Network-Controlled Systems

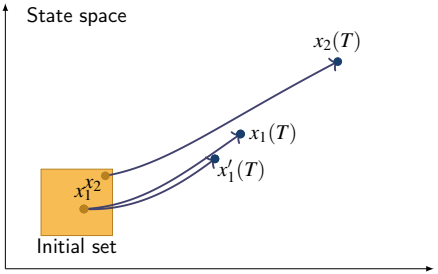
System : $\dot{x} = f(x, w)$ **State** : $x \in \mathbb{R}^n$ **Uncertainty** : $w \in \mathcal{W} \subseteq \mathbb{R}^m$

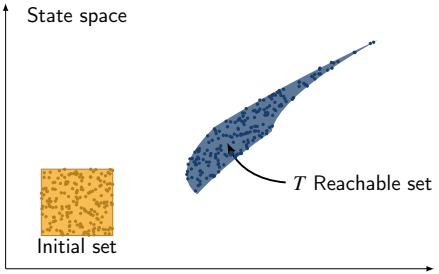




What are the possible states of the system at time T?

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?

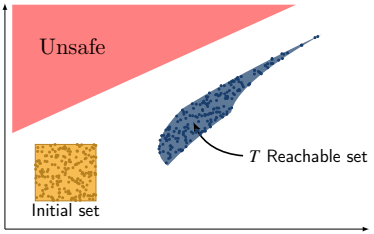
- T-**reachable sets** characterize evolution of the system

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

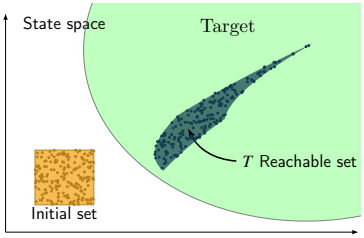
A large number of **safety specifications** can be represented using reachable sets

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

- Example: Reach-avoid problem



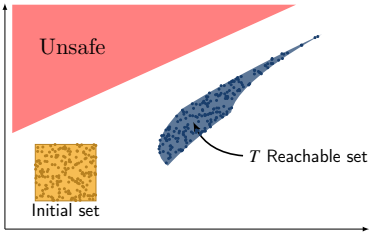
$\mathcal{R}_f(T, \mathcal{X}_0, \mathcal{W}) \cap$ Unsafe set $= \emptyset$

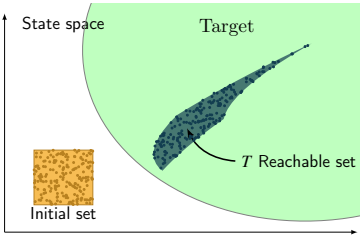


$\mathcal{R}_f(T_{\text{final}}, \mathcal{X}_0, \mathcal{W}) \subseteq$ Target set

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

- Example: Reach-avoid problem





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

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

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

Computing **exact** reachable sets are computationally challenging

Computing **exact** reachable sets are computationally challenging

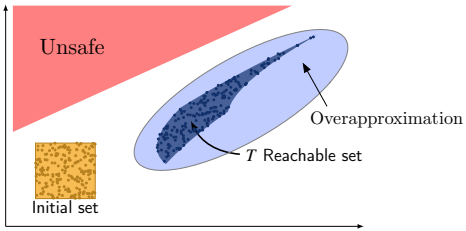
Solution: over-approximations of reachable sets

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

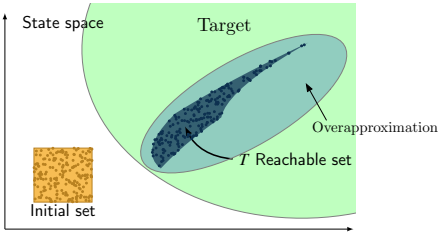
Computing **exact** reachable sets are computationally challenging

Solution: over-approximations of reachable sets

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



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

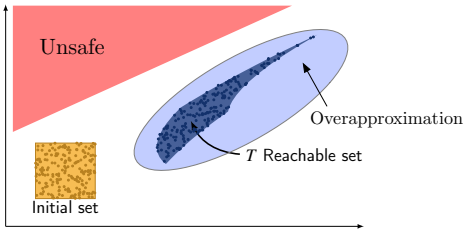


$\overline{\mathcal{R}}_f(T_{\text{final}}, \mathcal{X}_0, \mathcal{W}) \subseteq$ Target set

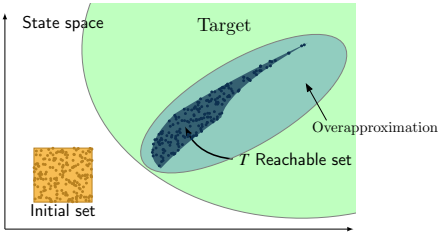
Computing **exact** reachable sets are computationally challenging

Solution: over-approximations of reachable sets

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



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



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

In this talk: computationally efficient methods for over-approximating reachable sets

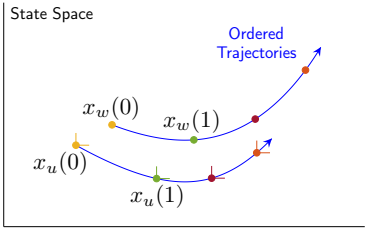
Outline of this talk

- Safety via Reachability Analysis
- **Mixed Monotone Reachability**
- Neural Network-Controlled Systems

A dynamical system $\dot{x} = f(x, w)$ is monotone 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.



[2] D. Angeli and E. Sontag, "Monotone control systems", TAC 2003

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

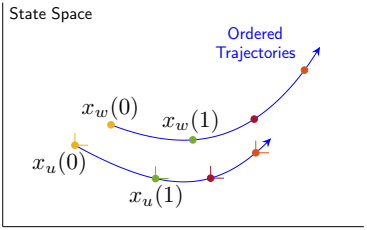
$$x_u(0) \leq y_w(0) \quad \text{and} \quad u \leq w \quad \Longrightarrow \quad x_u(t) \leq y_w(t) \quad \text{for all time}$$

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

Kamke–Müller condition[1]

A dynamical system $\dot{x} = f(x, w)$ is monotone iff

1. $\frac{\partial f}{\partial x}(x, w)$ is Metzler (off-diag ≥ 0) for all x, w
2. $\frac{\partial f}{\partial w}(x, w) \geq \mathbb{0}_{n \times m}$ for all x, w



[2]D. Angeli and E. Sontag, "Monotone control systems", TAC 2003

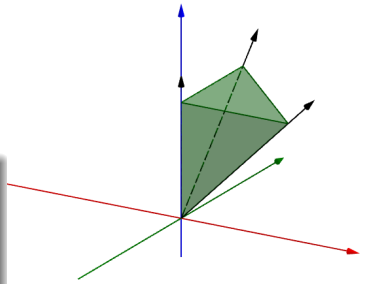
A dynamical system $\dot{x} = f(x, w)$ is monotone if

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

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

A **polyhedral cone** has the form

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



Kamke–Müller condition[2]

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

[2] **SJ** and S. Coogan, "Monotonicity and Contraction on Polyhedral Cones", TAC 2024.

Theorem (classical)[3]

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

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

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

[3] MW Hirsch, H Smith. "Monotone dynamical systems", 2006 [Book]

Theorem (classical)[3]

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

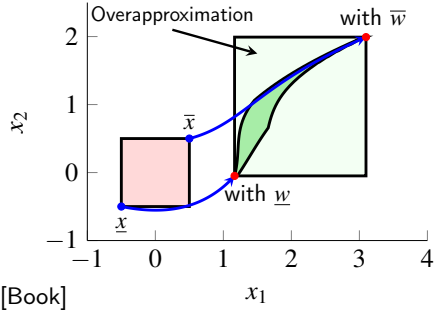
$$\mathcal{R}_f(t, [\underline{x}_0, \overline{x}_0], [\underline{w}, \overline{w}]) \subseteq [x_{\underline{w}}(t), x_{\overline{w}}(t)]$$

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

Example:

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

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



[3] MW Hirsch, H Smith. "Monotone dynamical systems", 2006 [Book]

Theorem (classical)[3]

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

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

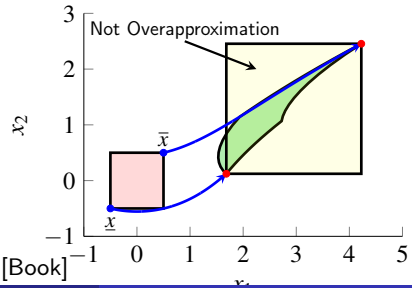
where $x_{\underline{w}}(\cdot)$ (resp. $x_{\overline{w}}(\cdot)$) is the trajectory with disturbance $\underline{w}(\cdot)$ (resp. $\overline{w}(\cdot)$) starting at $\underline{x}_0$ (resp. $\overline{x}_0$)

does not hold for non-monotone systems

Example:

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

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



[3] MW Hirsch, H Smith. "Monotone dynamical systems", 2006 [Book]

- **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

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

- **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: $(\underline{x}, \underline{w}) \mapsto \underline{d}_i(\underline{x}, \overline{x}, \underline{w}, \overline{w})$
3. competitive: $(\overline{x}, \overline{w}) \mapsto \underline{d}_i(\underline{x}, \overline{x}, \underline{w}, \overline{w})$
4. the same properties for $\overline{d}$

- **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: $(\underline{x}, \underline{w}) \mapsto \underline{d}_i(\underline{x}, \overline{x}, \underline{w}, \overline{w})$
3. competitive: $(\overline{x}, \overline{w}) \mapsto \underline{d}_i(\underline{x}, \overline{x}, \underline{w}, \overline{w})$
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 $\leq_{\mathrm{SE}}$:

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

In terms of cones, $\leq_{\mathrm{SE}}$ is induced by the cone $\mathbb{R}^n_{\geq 0} \times -\mathbb{R}^n_{\geq 0}$.

- **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: $(\underline{x}, \underline{w}) \mapsto \underline{d}_i(\underline{x}, \overline{x}, \underline{w}, \overline{w})$
3. competitive: $(\overline{x}, \overline{w}) \mapsto \underline{d}_i(\underline{x}, \overline{x}, \underline{w}, \overline{w})$
4. the same properties for $\overline{d}$

Computing decomposition function:

- close connection with **inclusion function** in Numerical Analysis[4]
- **Jacobian-based**: mean-value inequality and interval arithmetic[5]

[d] L. Jaulin, et al. "Applied Interval Analysis", 2001 [Book]
[e] A. Harapanahalli, **SJ**, S. Coogan, "A toolbox for fast interval arithmetic in Numpy", 2023

- **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: $(\underline{x}, \underline{w}) \mapsto \underline{d}_i(\underline{x}, \overline{x}, \underline{w}, \overline{w})$
3. competitive: $(\overline{x}, \overline{w}) \mapsto \underline{d}_i(\underline{x}, \overline{x}, \underline{w}, \overline{w})$
4. the same properties for $\overline{d}$

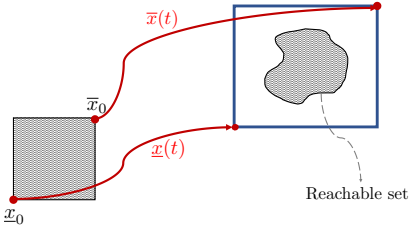
In this talk: mixed monotone theory for reachability analysis

Theorem[6]

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



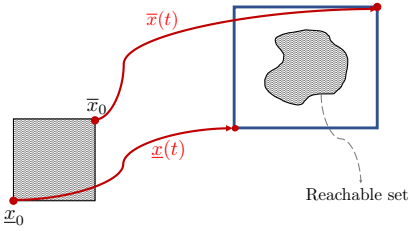
[6] S. Coogan and M. Arcak, "Efficient finite abstraction of mixed monotone systems", TAC 2015.

Theorem[6]

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



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

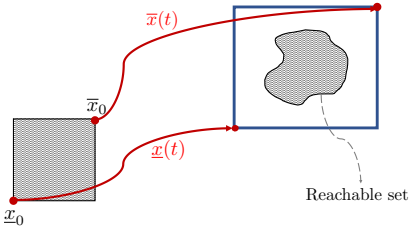
[6] S. Coogan and M. Arcak, "Efficient finite abstraction of mixed monotone systems", TAC 2015.

Theorem[6]

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



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

(Computational efficient): solve for one trajectory of embedding system

(Scalable): embedding system is $2n$-dimensional

[6] S. Coogan and M. Arcak, "Efficient finite abstraction of mixed monotone systems", TAC 2015.

Outline of this talk

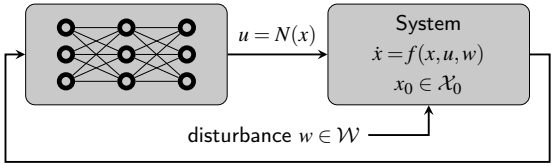
- Safety via Reachability Analysis
- Mixed Monotone Reachability
- **Neural Network-Controlled Systems**

An open-loop nonlinear system with a Neural Network (NN) controller

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

safety of the closed-loop system

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

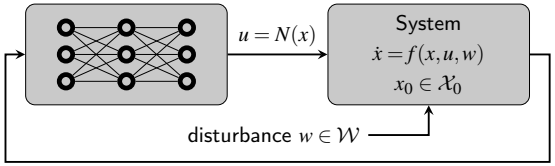


An open-loop nonlinear system with a Neural Network (NN) controller

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

safety of the closed-loop system

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



$u = N(x)$ is **pre-trained** feed-forward NN:

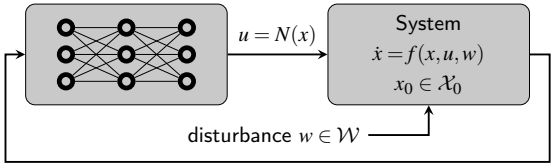
$$\xi^{(i)}(x) = \phi^{(i)}(W^{(i-1)}\xi^{(i-1)}(x) + b^{(i-1)})$$
$$x = \xi^{(0)}, \quad u = W^{(k)}\xi^{(k)}(x) + b^{(k)} := N(x),$$

An open-loop nonlinear system with a Neural Network (NN) controller

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

safety of the closed-loop system

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



$u = N(x)$ is **pre-trained** feed-forward NN:

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

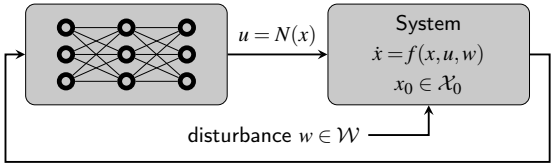
directly performing reachability on f^c is computationally challenging

An open-loop nonlinear system with a Neural Network (NN) controller

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

safety of the closed-loop system

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



$u = N(x)$ is **pre-trained** feed-forward NN:

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

Compositional Approach

Combine MM reachability with **neural network verification algorithms**

Linear Systems:

- M. Everett, et al. Reachability analysis of neural feedback loops, IEEE Access, 2021
- H. Hu, et al. Reach-SDP: Reachability analysis of closed-loop systems with neural network controllers via semidefinite programming, CDC, 2020

Nonlinear Systems:

- N. Rober, et al. Backward reachability analysis of neural feedback loops: Techniques for linear and nonlinear systems, OJCSYS, 2023
- C. Sidrane, et al. OVERT: An algorithm for safety verification of neural network control policies for nonlinear systems, JMLR, 2022.
- C. Huang, et al. ReachNN: Reachability analysis of neural-network controlled systems, TECS, 2019
- R. Ivanov, et al. Verisig 2.0: Verification of neural network controllers using taylor model preconditioning, CAV, 2021
- C. Huang, et al. POLAR: A polynomial arithmetic framework for verifying neural-network controlled systems, ATVA, 2022
- C. Schilling, et al. Verification of neural-network control systems by integrating Taylor models and zonotopes, AAAI, 2022

Compositional Approach

Combine MM reachability with **neural network verification algorithms**

Compositional Approach

Combine MM reachability with **neural network verification algorithms**

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

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

Compositional Approach

Combine MM reachability with **neural network verification algorithms**

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

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

Many neural network verification algorithms can produce such bounds, including:

Compositional Approach

Combine MM reachability with **neural network verification algorithms**

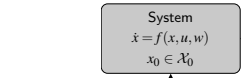
Input-output bounds: Given a neural network controller $u = N(x)$

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

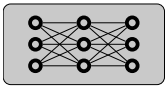
Many neural network verification algorithms can produce such bounds, including:

- Lipschitz-based estimates (Combettes et al., 2019),
- convex relaxation bounds (H. Zhang et al., 2018),
- interval arithmatics (S. Gowal et al., 2018),
- polyhedral bounds (M. Fazlyab et al., 2019)

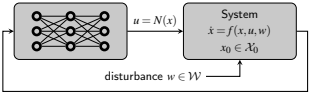
Reachability of open-loop system treating u as a parameter



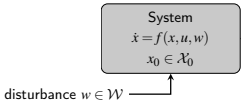
Neural network verification algorithm for bounds on u



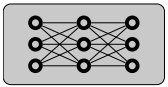
Reachability of open-loop system + Neural network verification bounds



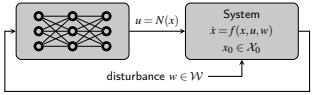
$$\dot{\underline{x}} = \underline{d}(\underline{x}, \overline{x}, \underline{u}, \overline{u}, \underline{w}, \overline{w})$$
$$\dot{\overline{x}} = \overline{d}(\underline{x}, \overline{x}, \underline{u}, \overline{u}, \underline{w}, \overline{w})$$



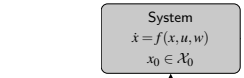
$$\underline{u}_{[\underline{x},\overline{x}]} \leq N(x) \leq \overline{u}_{[\underline{x},\overline{x}]} \quad \text{for every } x \in [\underline{x}, \overline{x}].$$



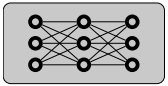
$$\dot{\underline{x}} = \underline{d}(\underline{x}, \overline{x}, \underline{u}_{[\underline{x},\overline{x}]}, \overline{u}_{[\underline{x},\overline{x}]}, \underline{w}, \overline{w})$$
$$\dot{\overline{x}} = \overline{d}(\underline{x}, \overline{x}, \underline{u}_{[\underline{x},\overline{x}]}, \overline{u}_{[\underline{x},\overline{x}]},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



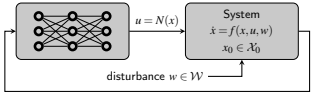
$$\dot{\underline{x}} = \underline{d}(\underline{x}, \overline{x}, \underline{u}, \overline{u}, \underline{w}, \overline{w})$$
$$\dot{\overline{x}} = \overline{d}(\underline{x}, \overline{x}, \underline{u}, \overline{u}, \underline{w}, \overline{w})$$



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

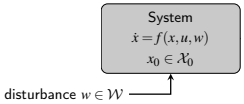


$$\dot{\underline{x}} = \underline{d}(\underline{x}, \overline{x}, \underline{u}_{[\underline{x},\overline{x}]}, \overline{u}_{[\underline{x},\overline{x}]}, \underline{w}, \overline{w})$$
$$\dot{\overline{x}} = \overline{d}(\underline{x}, \overline{x}, \underline{u}_{[\underline{x},\overline{x}]}, \overline{u}_{[\underline{x},\overline{x}]},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

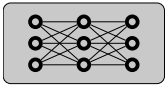


Input-output over-approximation:
$\mathcal{R}_{f^c}(t, \mathcal{X}_0, \mathcal{W}) \subseteq [\underline{x}(t), \overline{x}(t)]$

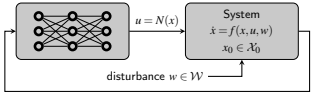
$$\dot{\underline{x}} = \underline{d}(\underline{x}, \overline{x}, \underline{u}, \overline{u}, \underline{w}, \overline{w})$$
$$\dot{\overline{x}} = \overline{d}(\underline{x}, \overline{x}, \underline{u}, \overline{u}, \underline{w}, \overline{w})$$



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



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



Input-output over-approximation:
$\mathcal{R}_{f^c}(t, \mathcal{X}_0, \mathcal{W}) \subseteq [\underline{x}(t), \overline{x}(t)]$

It lead to overly-conservative estimates of reachable set

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

Neural network controller as **disturbances** (worst-case scenario)
It 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 $1.5 = \underline{K} \leq K \leq \overline{K} = 3$.

Neural network controller as **disturbances** (worst-case scenario)
It 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 $1.5 = \underline{K} \leq K \leq \overline{K} = 3$.

Input-output bounds

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

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

This system is unstable.

Functional bounds

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

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

This system is stable.

Neural network controller as **disturbances** (worst-case scenario)
It 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 $1.5 = \underline{K} \leq K \leq \overline{K} = 3$.

Input-output bounds

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

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

This system is unstable.

Functional bounds

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

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

This system is stable.

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

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

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

[7] H. Zhang et al., "Efficient neural network robustness certification with general activation functions", NeurIPS 2018.

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

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

- the bounds $\underline{N}_{[\underline{x},\overline{x}]}(x)$ and $\underline{N}_{[\underline{x},\overline{x}]}(x)$ are functions of x.

[7] H. Zhang et al., "Efficient neural network robustness certification with general activation functions", NeurIPS 2018.

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

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

- the bounds $\underline{N}_{[\underline{x},\overline{x}]}(x)$ and $\underline{N}_{[\underline{x},\overline{x}]}(x)$ are functions of x.
- Example: CROWN[7] can provide functional bounds.

[7] H. Zhang et al., "Efficient neural network robustness certification with general activation functions", NeurIPS 2018.

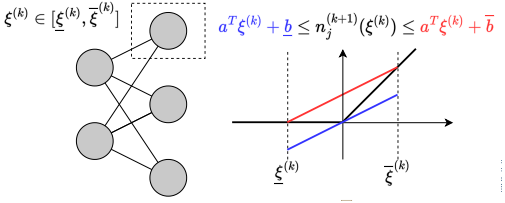
Functional bounds: Given a neural network controller $u = N(x)$

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

- the bounds $\underline{N}_{[\underline{x},\overline{x}]}(x)$ and $\underline{N}_{[\underline{x},\overline{x}]}(x)$ are functions of x.
- Example: CROWN[7] can provide functional bounds.

CROWN verification algorithm

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



[7] H. Zhang et al., "Efficient neural network robustness certification with general activation functions", NeurIPS 2018.

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

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

- the bounds $\underline{N}_{[\underline{x},\overline{x}]}(x)$ and $\underline{N}_{[\underline{x},\overline{x}]}(x)$ are functions of x.
- Example: CROWN[7] 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:

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

[7] H. Zhang et al., "Efficient neural network robustness certification with general activation functions", NeurIPS 2018.

Evolution of the system on $[\underline{x}, \overline{x}]$:

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

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

Evolution of the system on $[\underline{x}, \overline{x}]$:

$$\dot{x} = f(x, N(x), \underbrace{w}_{\text{disturbance}})$$

where

$$\underbrace{\underline{A}_{[\underline{x},\overline{x}]}x}_{\text{stabilizing term}} + \underbrace{\underline{b}}_{\text{disturbance}} \leq N(x) \leq \underbrace{\overline{A}_{[\underline{x},\overline{x}]}x}_{\text{stabilizing term}} + \underbrace{\overline{b}}_{\text{disturbance}}$$

Evolution of the system on $[\underline{x}, \overline{x}]$:

$$\dot{x} = f(x, N(x), \underbrace{w}_{\text{disturbance}})$$

where

$$\underbrace{\underline{A}_{[\underline{x},\overline{x}]}x}_{\text{stabilizing term}} + \underbrace{\underline{b}}_{\text{disturbance}} \leq N(x) \leq \underbrace{\overline{A}_{[\underline{x},\overline{x}]}x}_{\text{stabilizing term}} + \underbrace{\overline{b}}_{\text{disturbance}}$$

Evolution of the system on $[\underline{x}, \overline{x}]$:

$$\dot{x} = f(x, N(x), \underbrace{w}_{\text{disturbance}})$$

where

$$\underbrace{\underline{A}_{[\underline{x},\overline{x}]}x}_{\text{stabilizing term}} + \underbrace{\underline{b}}_{\text{disturbance}} \leq N(x) \leq \underbrace{\overline{A}_{[\underline{x},\overline{x}]}x}_{\text{stabilizing term}} + \underbrace{\overline{b}}_{\text{disturbance}}$$

Reachability analysis: treat b and w as disturbances and $A_{[\underline{x},\overline{x}]}x$ as a part of the system to obtain a decomposition function

Theorem[8]

Original system

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

Embedding system

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

$\underline{H}$ and $\overline{H}$ capture the stabilizing interactions between nonlinear system and neural network.

[8] **SJ** and A. Harapanahalli and S. Coogan, "Efficient interaction-aware interval analysis of neural network feedback loops", TAC, 2024

Theorem[8]

Original system

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

Embedding system

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

$\underline{H}$ and $\overline{H}$ capture the stabilizing interactions between nonlinear system and neural network.

Functional over-approximation: $\mathcal{R}_{f^c}(t, \mathcal{X}_0, \mathcal{W}) \subseteq [\underline{x}(t), \overline{x}(t)]$

[8] **SJ** and A. Harapanahalli and S. Coogan, "Efficient interaction-aware interval analysis of neural network feedback loops", TAC, 2024

Theorem[8]

Original system

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

Embedding system

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

$\underline{H}$ and $\overline{H}$ capture the stabilizing interactions between nonlinear system and neural network.

Functional over-approximation: $\mathcal{R}_{f^c}(t, \mathcal{X}_0, \mathcal{W}) \subseteq [\underline{x}(t), \overline{x}(t)]$

Computational aspects

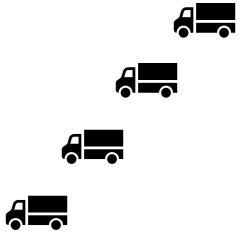
- one trajectory of $2n$-dim embedding system (fast using integration methods)
- implement NN verification algorithm per time-step (fast using CROWN)

[8] **SJ** and A. Harapanahalli and S. Coogan, "Efficient interaction-aware interval analysis of neural network feedback loops", TAC, 2024

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 $w_x^j, w_y^j \sim \mathcal{U}([-0.001, 0.001])$.



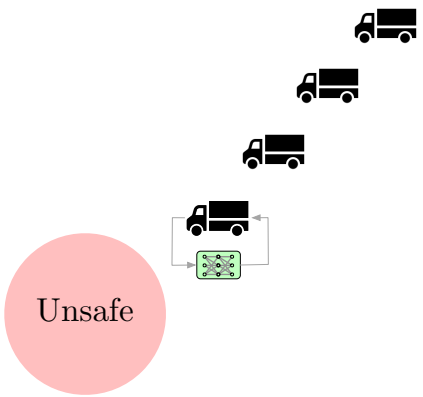
Dynamics of the jth vehicle

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

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

$$4 \times 100 \times 100 \times 2, \text{ with ReLU activations}$$

and is trained using trajectory data from an MPC controller for the first vehicle.



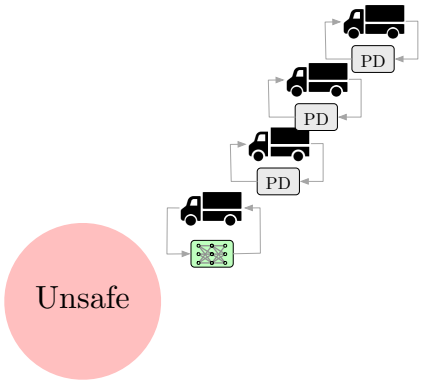
Dynamics of the jth vehicle

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

where $w_x^j, w_y^j \sim \mathcal{U}([-0.001, 0.001])$. 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 $d \in \{x, y\}$.



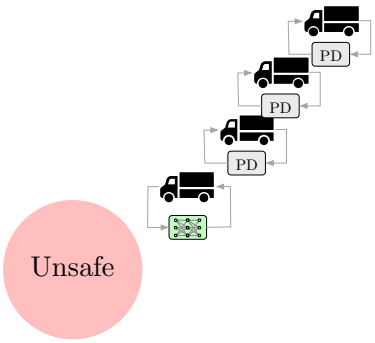
Dynamics of the jth vehicle

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

where $w_x^j, w_y^j \sim \mathcal{U}([-0.001, 0.001])$. 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 $d \in \{x, y\}$.



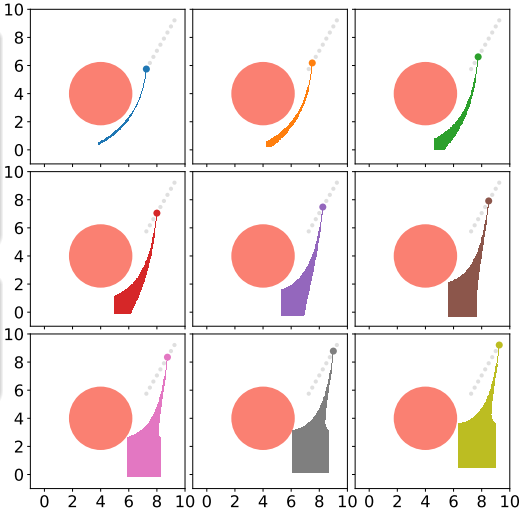
Safety: platoon avoids the obstacle

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 $w_x^j, w_y^j \sim \mathcal{U}([-0.001, 0.001])$.

- Input-output bounds
- a platoon of 9 vehicles
- reachable over-approximations for $t \in [0, 1.5]$



Dynamics of the jth vehicle

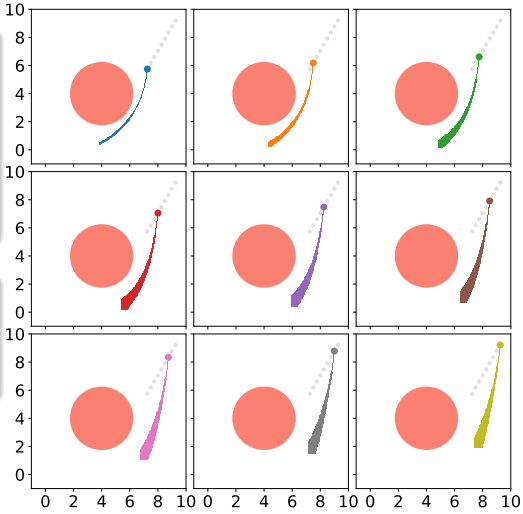
$$\dot{p}_x^j = v_x^j, \qquad \dot{v}_x^j = \tanh(u_x^j) + w_x^j,$$
$$\dot{p}_y^j = v_y^j, \qquad \dot{v}_y^j = \tanh(u_y^j) + w_y^j,$$

where $w_x^j, w_y^j \sim \mathcal{U}([-0.001, 0.001])$.

- Functional bounds
- a platoon of 9 vehicles
- reachable over-approximations for $t \in [0, 1.5]$

N (units)	# of states	Our Approach (s)	POLAR (s)	JuliaReach (s)
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



POLAR = C. Huang et al., ATVA 2022

JuliaReach = C. Schilling et al., AAAI 2022

Conclusions and Future Directions

Key takeaways

1. reachability as a framework for safety certification of autonomous systems
2. computationally efficient approach for reachability using mixed monotone theory
3. compositional approach for verification of neural network-controlled systems
4. capture stabilizing effect of learning-based components

1. reachability as a framework for safety certification of autonomous systems
2. computationally efficient approach for reachability using mixed monotone theory
3. compositional approach for verification of neural network-controlled systems
4. capture stabilizing effect of learning-based components

Future Research Directions:

- built-in safety: reachability guarantees in *training* of neural network controllers

1. reachability as a framework for safety certification of autonomous systems
2. computationally efficient approach for reachability using mixed monotone theory
3. compositional approach for verification of neural network-controlled systems
4. capture stabilizing effect of learning-based components

Future Research Directions:

- built-in safety: reachability guarantees in *training* of neural network controllers
- interactive NN verifier: optimize the NN verifier to obtain the tightest reachable set

1. reachability as a framework for safety certification of autonomous systems
2. computationally efficient approach for reachability using mixed monotone theory
3. compositional approach for verification of neural network-controlled systems
4. capture stabilizing effect of learning-based components

Future Research Directions:

- built-in safety: reachability guarantees in *training* of neural network controllers
- interactive NN verifier: optimize the NN verifier to obtain the tightest reachable set
- beyond stability: capture the effect of specifications such as invariance, optimality, logic

1. reachability as a framework for safety certification of autonomous systems
2. computationally efficient approach for reachability using mixed monotone theory
3. compositional approach for verification of neural network-controlled systems
4. capture stabilizing effect of learning-based components

Future Research Directions:

- built-in safety: reachability guarantees in *training* of neural network controllers
- interactive NN verifier: optimize the NN verifier to obtain the tightest reachable set
- beyond stability: capture the effect of specifications such as invariance, optimality, logic
- stochastic systems: capture the effect of stochastic noise in NN-controlled systems

Thank you for your attention!

Back up Slides

How to compute a decomposition function for a system?

How to compute a decomposition function for a system?

Different approaches for constructing decomposition functions

- linear systems
- polynomial systems

How to compute a decomposition function for a system?

Different approaches for constructing decomposition functions

- linear systems
- polynomial systems

Every **locally Lipschitz** system has at least one decomposition function

How to compute a decomposition function for a system?

Different approaches for constructing decomposition functions

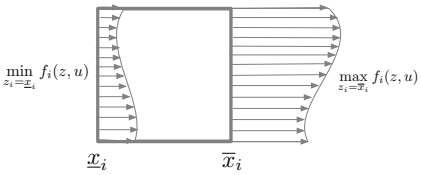
- linear systems
- polynomial systems

Every **locally Lipschitz** system has at least one decomposition function

The best (tightest) decomposition function is given by

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

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



How to compute a decomposition function for a system?

Different approaches for constructing decomposition functions

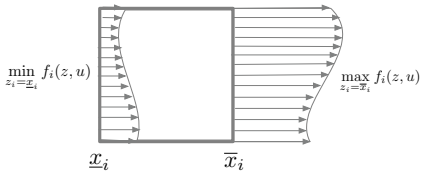
- linear systems
- polynomial systems

Every **locally Lipschitz** system has at least one decomposition function

The best (tightest) decomposition function is given by

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

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



obtaining this decomposition function is computationally complicated.

How to compute a decomposition function for a system?

How to compute a decomposition function for a system?

- Assume $f : \mathbb{R} \to \mathbb{R}$ is scalar:

Mean-value Inequality

$$f(\underline{x}) + \left[\min_{z\in[\underline{x},\overline{x}]} \frac{\partial f}{\partial x}\right](\overline{x} - \underline{x}) \leq f(x) \leq f(\underline{x}) + \left[\max_{z\in[\underline{x},\overline{x}]} \frac{\partial f}{\partial x}\right](\overline{x} - \underline{x})$$

where $[A]^+ = \max\{A, 0\}$ and $[A]^- = \min\{A, 0\}$.

How to compute a decomposition function for a system?

- Assume $f : \mathbb{R} \to \mathbb{R}$ is scalar:

Mean-value Inequality

$$\underbrace{f(\underline{x}) + \left[\min_{z \in [\underline{x}, \overline{x}]} \frac{\partial f}{\partial x}\right]^{-} (\overline{x} - \underline{x})}_{\underline{d}(\underline{x}, \overline{x})} \leq f(x) \leq \underbrace{f(\underline{x}) + \left[\max_{z \in [\underline{x}, \overline{x}]} \frac{\partial f}{\partial x}\right]^{+} (\overline{x} - \underline{x})}_{\overline{d}(\underline{x}, \overline{x})}$$

where $[A]^{+} = \max\{A, 0\}$ and $[A]^{-} = \min\{A, 0\}$.

How to compute a decomposition function for a system?

- Assume $f : \mathbb{R} \to \mathbb{R}$ is scalar:

Mean-value Inequality

$$\underbrace{f(\underline{x}) + \left[\min_{z \in [\underline{x}, \overline{x}]} \frac{\partial f}{\partial x}\right]^{-} (\overline{x} - \underline{x})}_{\underline{d}(\underline{x}, \overline{x})} \leq f(x) \leq \underbrace{f(\underline{x}) + \left[\max_{z \in [\underline{x}, \overline{x}]} \frac{\partial f}{\partial x}\right]^{+} (\overline{x} - \underline{x})}_{\overline{d}(\underline{x}, \overline{x})}$$

where $[A]^{+} = \max\{A, 0\}$ and $[A]^{-} = \min\{A, 0\}$.

The effect of $\overline{x}$ on $\underline{d}(\underline{x}, \overline{x})$ is competitive.

The effect of $\overline{x}$ on $\overline{d}(\underline{x}, \overline{x})$ is cooperative.

How to compute a decomposition function for a system?

- Assume $f : \mathbb{R} \to \mathbb{R}$ is scalar:

Mean-value Inequality

$$\underbrace{f(\underline{x}) + \left[\min_{z \in [\underline{x}, \overline{x}]} \frac{\partial f}{\partial x}\right]^{-} (\overline{x} - \underline{x})}_{\underline{d}(\underline{x}, \overline{x})} \leq f(x) \leq \underbrace{f(\underline{x}) + \left[\max_{z \in [\underline{x}, \overline{x}]} \frac{\partial f}{\partial x}\right]^{+} (\overline{x} - \underline{x})}_{\overline{d}(\underline{x}, \overline{x})}$$

where $[A]^+ = \max\{A, 0\}$ and $[A]^- = \min\{A, 0\}$.

The effect of $\overline{x}$ on $\underline{d}(\underline{x}, \overline{x})$ is competitive.

The effect of $\overline{x}$ on $\overline{d}(\underline{x}, \overline{x})$ is cooperative.

Punchline

sign pattern of $\frac{\partial f}{\partial x}$ separates cooperative and competitive effect of states.

How to compute a decomposition function for a system?

Theorem[9]

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

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

$\underline{x} \mapsto R_1 \mapsto R_2 \mapsto \ldots \mapsto R_n \mapsto \overline{x}$, then the i-th column of $\underline{M}$ is $\min_{z \in R_i, w \in [\underline{u},\overline{u}]} \frac{\partial f_i}{\partial x}(z,w)$

[4] **SJ** and A. Harapanahalli and S. Coogan, Efficient interaction-aware interval analysis of neural network feedback loops, TAC, 2023

How to compute a decomposition function for a system?

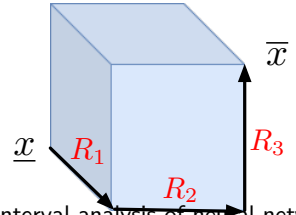
Theorem[9]

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

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

$\underline{x} \mapsto R_1 \mapsto R_2 \mapsto \ldots \mapsto R_n \mapsto \overline{x}$, then the i-th column of $\underline{M}$ is $\min_{z \in R_i, w \in [\underline{u},\overline{u}]} \frac{\partial f_i}{\partial x}(z,w)$

- Interval analysis for computing Jacobian bounds.
- Use tools and techniques from interval analysis.



[4] **SJ** and A. Harapanahalli and S. Coogan, Efficient interaction-aware interval analysis of neural network feedback loops, TAC, 2023

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

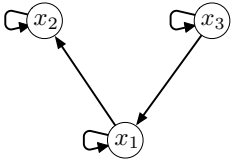
Linear systems

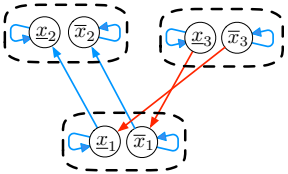
Original system

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

Embedding system

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





For a scalar vector field $f : \mathbb{R} \to \mathbb{R}$, we show that $\underline{d}(\underline{x}, \overline{x}) = f(\underline{x}) + \left[\min_{z \in [\underline{x}, \overline{x}]} \frac{\partial f}{\partial x}\right]^- (\overline{x} - \underline{x})$ is

1. cooperative in $\underline{x}$
2. competitive in $\overline{x}$

$$\frac{\partial}{\partial \underline{x}} \underline{d}(\underline{x}, \overline{x}) = \frac{\partial}{\partial \underline{x}} f(\underline{x}) - \left[\min_{z \in [\underline{x}, \overline{x}]} \frac{\partial f}{\partial x}\right]^- = \frac{\partial f}{\partial x}|_{x=\underline{x}} - \left[\min_{z \in [\underline{x}, \overline{x}]} \frac{\partial f}{\partial x}\right]^- \geq 0.$$

Similarly,

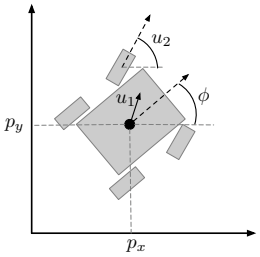
$$\frac{\partial}{\partial \overline{x}} \underline{d}(\underline{x}, \overline{x}) = \left[\min_{z \in [\underline{x}, \overline{x}]} \frac{\partial f}{\partial x}\right]^- \leq 0$$

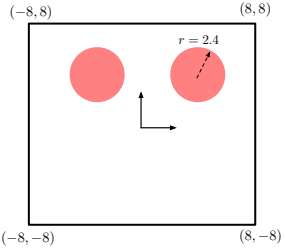
Dynamics of bicycle

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

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

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



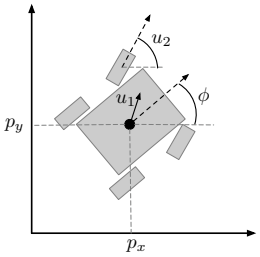


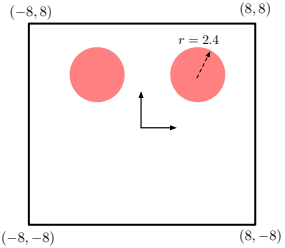
Dynamics of bicycle

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

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

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





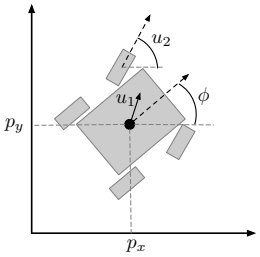
Goal: steer the bicycle to the origin avoiding the obstacles

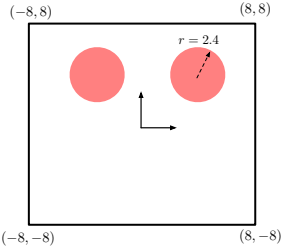
Dynamics of bicycle

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

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

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





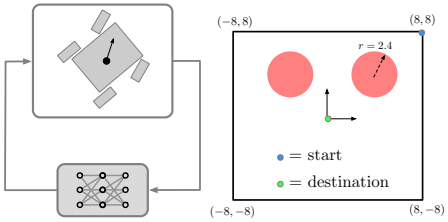
Goal: steer the bicycle to the origin avoiding the obstacles

- train a feedforward neural network $4 \mapsto 100 \mapsto 100 \mapsto 2$ 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:

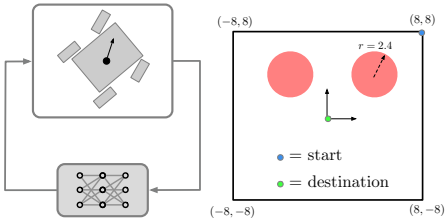
$$\dot{\underline{x}} = \underline{d}(\underline{x}, \overline{x}, \underline{u}, \overline{u}, \underline{w}, \overline{w})$$
$$\dot{\overline{x}} = \overline{d}(\underline{x}, \overline{x}, \underline{u}, \overline{u}, \underline{w}, \overline{w})$$

$\underline{u} \leq N(x) \leq \overline{u}$, 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

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

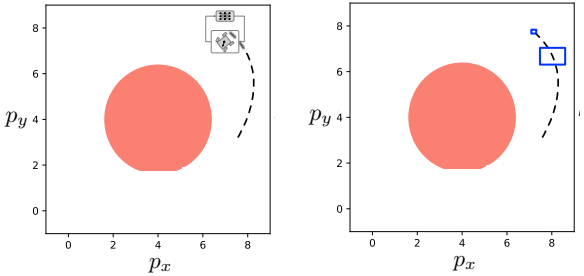
- CROWN for verification of neural network



Euler integration with step h:

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

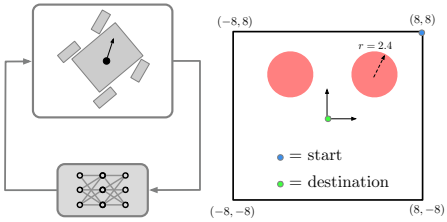
$\underline{u}_0 \leq N(x) \leq \overline{u}_0$, for every $x \in [\underline{x}_0, \overline{x}_0]$.



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

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

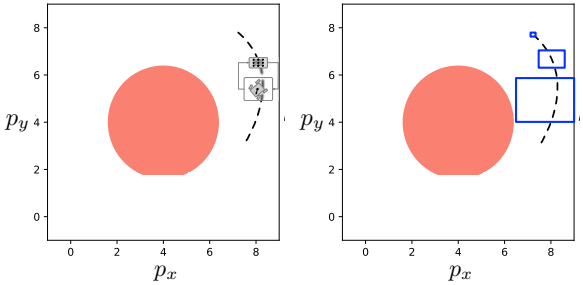
- CROWN for verification of neural network



Euler integration with step h:

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

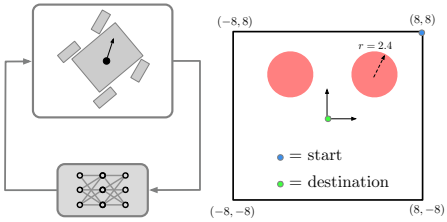
$\underline{u}_1 \leq N(x) \leq \overline{u}_1$, for every $x \in [\underline{x}_1, \overline{x}_1]$.



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

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

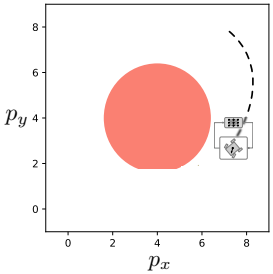
- CROWN for verification of neural network

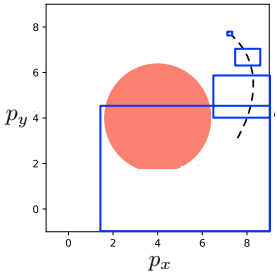


Euler integration with step h:

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

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

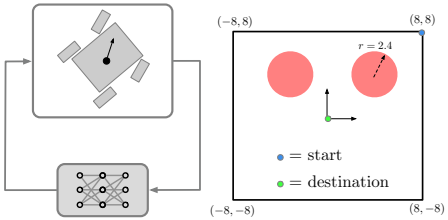


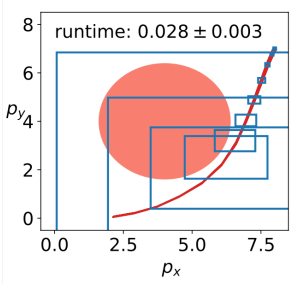


- start from $(8, 7)$ toward $(0, 0)$
- $\mathcal{X}_0 = [\underline{x}_0, \overline{x}_0]$ with

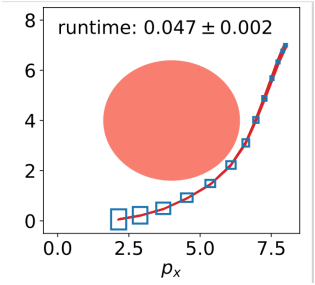
$$\underline{x}_0 = \begin{pmatrix} 7.95 & 6.95 & -\frac{2\pi}{3} - 0.01 & 1.99 \end{pmatrix}^\top$$
$$\overline{x}_0 = \begin{pmatrix} 8.05 & 7.05 & -\frac{2\pi}{3} + 0.01 & 2.01 \end{pmatrix}^\top$$

- CROWN for verification of neural network





Naive interconnection approach



interaction approach