# Reachability Analysis of Control Systems: A Mixed Monotone Approach

**Saber Jafarpour**

University of Colorado Boulder

Georgia Tech

September 11, 2024

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
## Introduction

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

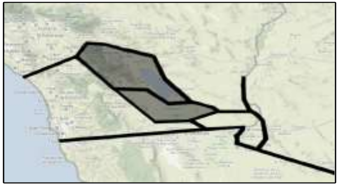
Autonomous systems in our societies are becoming more **interconnected** and **complex**.

# Modern Autonomous Systems

## Safety and Robustness guarantees

**A critical task**

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



2011 US Southwest blackout

Postal Drone hit the building

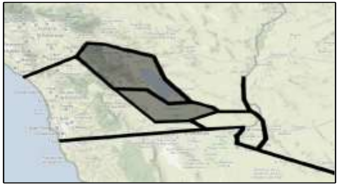
Self-driving car accident

# Modern Autonomous Systems

## Safety and Robustness guarantees

**A critical task**

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



2011 US Southwest blackout

Postal Drone hit the building

Self-driving car accident

**My Research**

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

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

**In this talk**: Autonomous Systems with Learning-based components

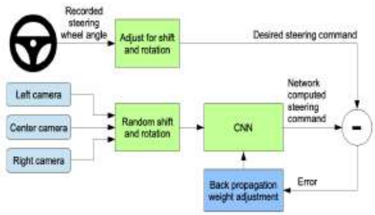
**In this talk**: Autonomous Systems with Learning-based components

- 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 reasons: computationally burdensome, executed by an expert, complicated representation.

Self driving vehicles:

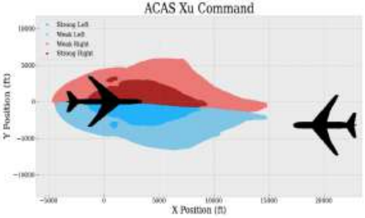



M. Bojarski, et al., NeurIPS, 2016.

Robotic motion planning:

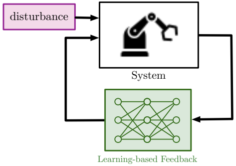
M. Everett, et. al., IROS, 2018.

Collision avoidance:



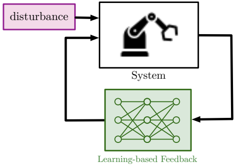

K. Julian, et. al., DASC, 2016.

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



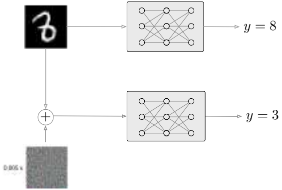

[1] C. Szegedy et. al. Intriguing properties of neural networks. In ICLR, 2014

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



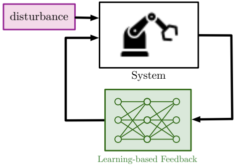

**Issues with learning algorithms:**

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



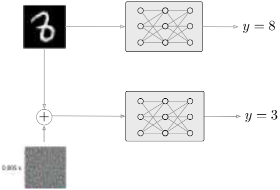

[1]C. Szegedy et. al. Intriguing properties of neural networks. In ICLR, 2014

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




**Issues with learning algorithms:**

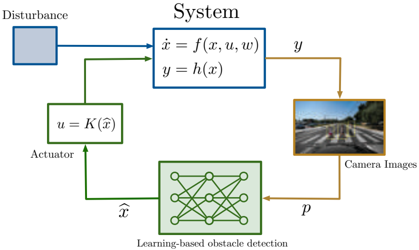
- large # of parameters with nonlinearity
- sensitive wrt to input perturbations[1]
- no safety guarantee in their training



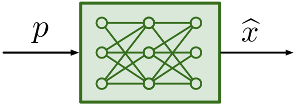

1. **Verification**: how safe is the closed-loop system?
2. **Training**: how to design the learning component to ensure safety?

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

## Perception-based Obstacle Avoidance

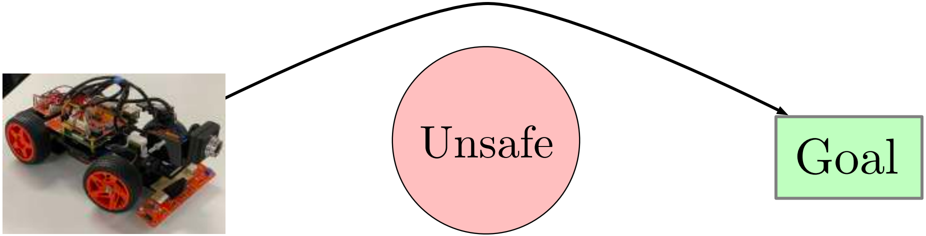



$$\dot{x} = f(x, u, w)$$
$$y = h(x)$$

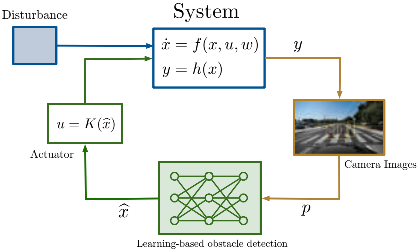



Learning-based obstacle detection

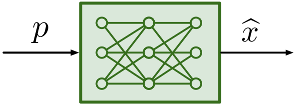
trained offline using images



## Perception-based Obstacle Avoidance




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



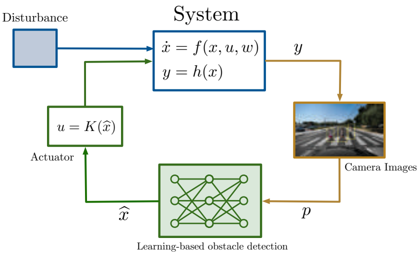

Learning-based obstacle detection

trained offline using images

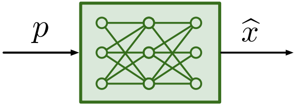
## No guarantee to avoid the obstacle:

- out of distribution images
- changes in the environment

## Perception-based Obstacle Avoidance

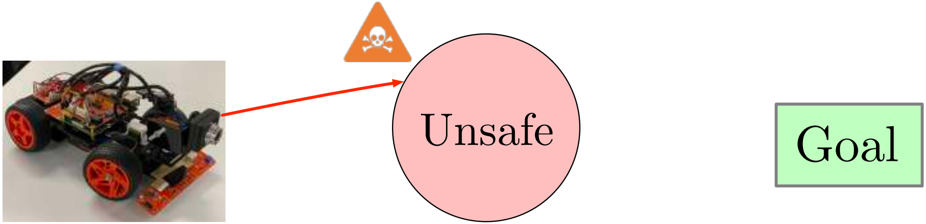



$$\dot{x} = f(x, u, w)$$
$$y = h(x)$$




Learning-based obstacle detection

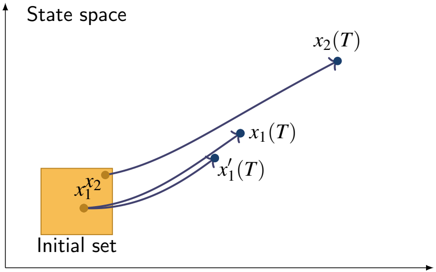
trained offline using images

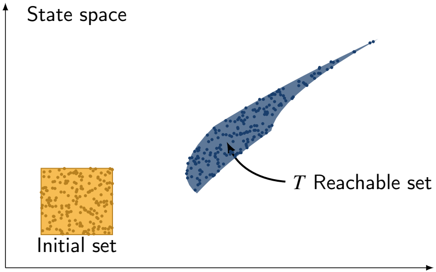


# Outline of this talk

- **Reachability Analysis**
- Mixed Monotone Theory
- Neural Network Controlled Systems

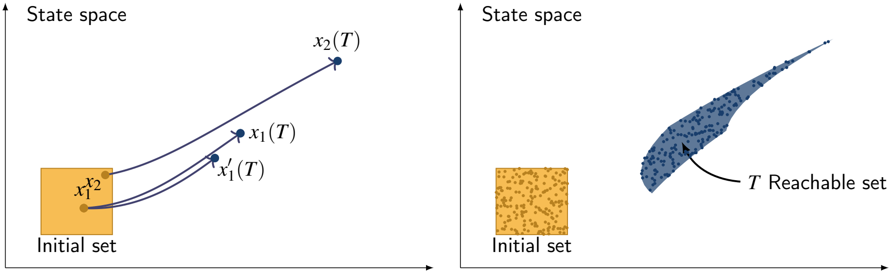
**System** : $\dot{x} = f(x, w)$ **State** : $x \in \mathbb{R}^n$ **Uncertainty** : $w \in \mathcal{W} \subseteq \mathbb{R}^m$




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 $T$?

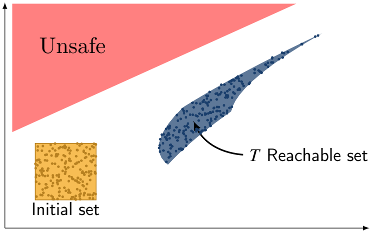
- $T$**-reachable sets** characterize evolution of the system

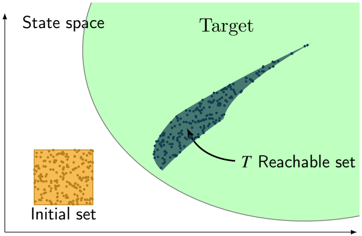
$$\mathcal{R}_f(T, \mathcal{X}_0, \mathcal{W}) = \{x_w(T) \mid x_w(\cdot) \text{ is a traj for some } w(\cdot) \in \mathcal{W} \text{ with } x_0 \in \mathcal{X}_0\}$$

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

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

- Example: Reach-avoid problem



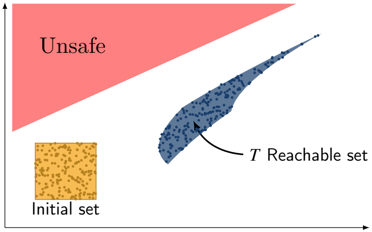



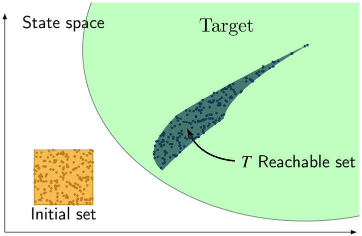

$$\mathcal{R}_f(T, \mathcal{X}_0, \mathcal{W}) \cap \text{ Unsafe set } = \emptyset$$

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

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

- Example: Reach-avoid problem



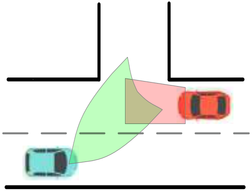




$\mathcal{R}_f(T, \mathcal{X}_0, \mathcal{W}) \cap$ Unsafe set $= \emptyset$

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

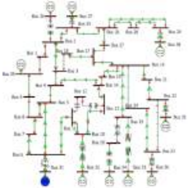
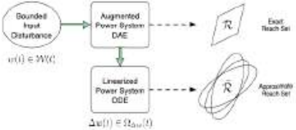
Combining different instantiation of Reach-avoid problem $\Longrightarrow$
**diverse range of specifications**
(complex planning using logics, invariance, stability)

## Autonomous Driving:

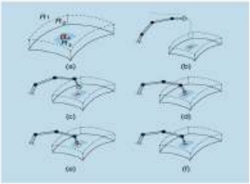
Althoff, 2014

## Power grids:

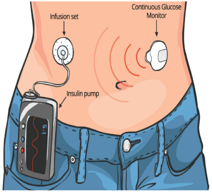



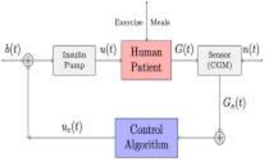

Chen and Domınguez-Garcıa, 2016

## 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

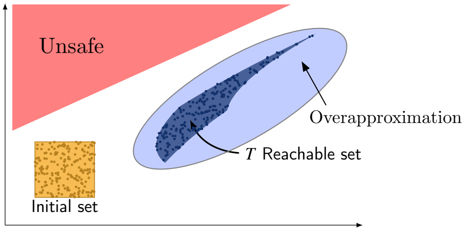
**Over-approximation**: $\mathcal{R}_f(T, \mathcal{X}_0, \mathcal{W}) \subseteq \overline{\mathcal{R}}_f(T, \mathcal{X}_0, \mathcal{W})$

Computing the $T$-reachable sets are computationally challenging

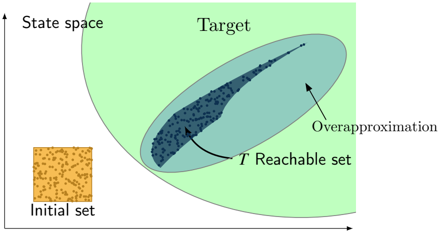
**Solution:** over-approximations and under-approximation of reachable sets

- 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].

[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].

**Reasons:**

- Impossible to completely characterize behavior of the system (human-in-the-loop)
- Lead to conservative design (stochastic environments)
- Simpler design with computationally efficiency (learning-based controllers)

[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].

**Reasons:**

- Impossible to completely characterize behavior of the system (human-in-the-loop)
- Lead to conservative design (stochastic environments)
- Simpler design with computationally efficiency (learning-based controllers)

**Run-time reachability**: In these applications, we need to compute reachable sets in run-time to verify safety 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: $\sim$ 1980

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

Different approaches for approximating reachable sets

- Bisimulations
- Linear, and piecewise linear systems (Ellipsoidal methods)
- Polynomial systems (Sum of Square)
- Optimization-based approaches (Hamilton-Jacobi, Level-set method)

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

Different approaches for approximating reachable sets

- Bisimulations
- Linear, and piecewise linear systems (Ellipsoidal methods)
- Polynomial systems (Sum of Square)
- Optimization-based approaches (Hamilton-Jacobi, Level-set method)

The classical and general approaches are computationally heavy and are not suitable for run-time reachability.

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

Different approaches for approximating reachable sets

- Bisimulations
- Linear, and piecewise linear systems (Ellipsoidal methods)
- Polynomial systems (Sum of Square)
- Optimization-based approaches (Hamilton-Jacobi, Level-set method)

The classical and general approaches are computationally heavy and are not suitable for run-time reachability.

**In this talk**: a mathematically rigorous and computationally efficient approach for run-time reachability

# Outline of this talk

- Reachability Analysis
- **Mixed Monotone Theory**
- Neural Network Controlled Systems

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

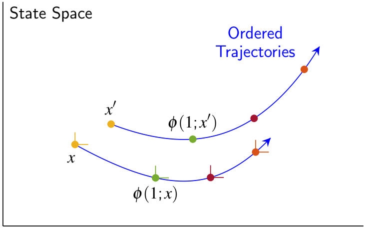
A dynamical system $\dot{x} = f(x, w)$ is monotone[3] if

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

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

**Monotonicity test**

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




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

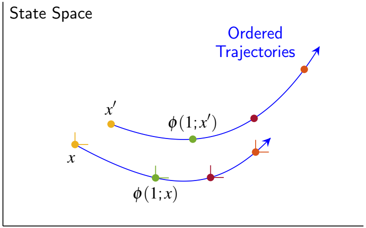
A dynamical system $\dot{x} = f(x, w)$ is monotone[3] if

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

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

## Monotonicity test

1. $\frac{\partial f}{\partial x}(x, w)$ is Metzler (off-diag $\geq 0$)
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 $\mathcal{W} = [\underline{w}, \overline{w}]$

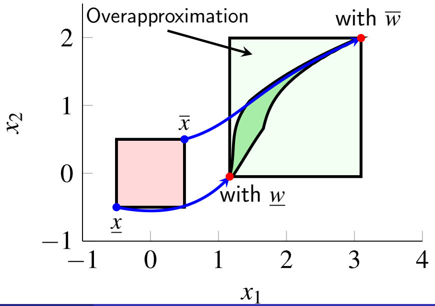
$$\mathcal{R}_f(t, [\underline{x}_0, \overline{x}_0]) \subseteq [x_{\underline{w}}(t), x_{\overline{w}}(t)]$$

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:**

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

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



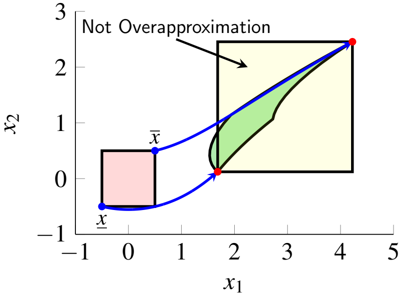
- For non-monotone dynamical systems the extreme trajectories do not provide any over-approximation of reachable sets

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

**Example:**

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

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



- **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 $\overline{d}$

- **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 $\leq_{\mathrm{SE}}$:

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

- $f$ 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:**

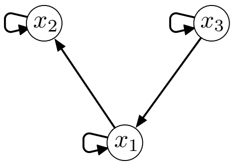
J-L. Gouze and L. P. Hadeler. Monotone flows and order intervals. Nonlinear World, 1994

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

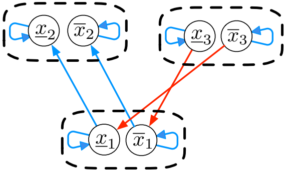



**Original system**

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

**Embedding system**

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

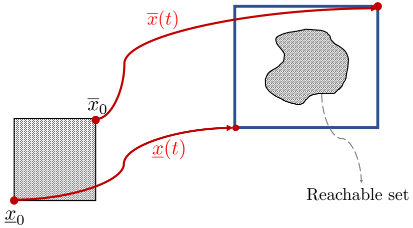


**Theorem**[4]

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



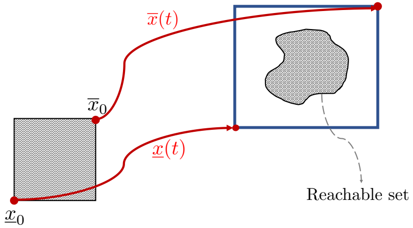

[4] Coogan and Arcak, "Efficient finite abstraction of mixed monotone systems", HSCC, 2015.

**Theorem**[4]

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

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

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




**(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:**

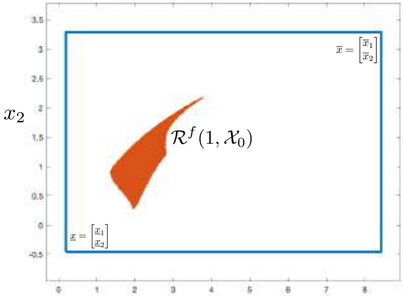
$$\frac{d}{dt}\begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \\ \overline{x}_1 \\ \overline{x}_2 \end{bmatrix} = \begin{bmatrix} \underline{x}_2^3 - \overline{x}_2 + \underline{w} \\ \underline{x}_1 \\ \overline{x}_2^3 - \underline{x}_2 + \overline{w} \\ \overline{x}_1 \end{bmatrix} \quad \begin{bmatrix}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

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

**Decomposition function**

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

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



# Outline of this talk

- Reachability Analysis
- Mixed Monotone Theory
- **Neural Network Controlled Systems**

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

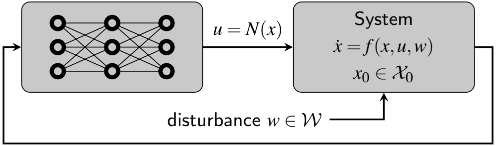
$$\dot{x} = f(x, u, w),$$
$$u = N(x),$$

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

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

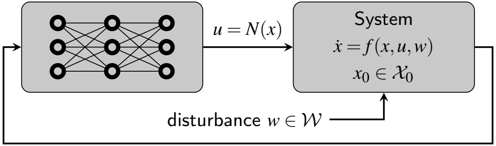


Given the open-loop nonlinear system with a neural network controller

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

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

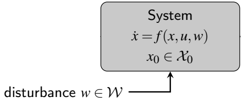
$N(x)$ is high dimensional and has a large # of parameters

Reachability of open-loop system treating $u$ as a parameter

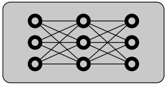
Reachability of open-loop system treating $u$ as a parameter

Neural network verification algorithm for bounds on $u$

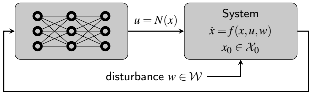
Reachability of open-loop system treating $u$ as a parameter



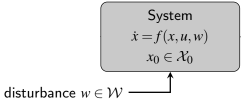

Neural network verification algorithm for bounds on $u$



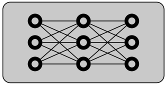
Reachability of open-loop system + Neural network verification bounds



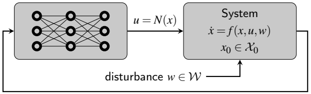
Reachability of open-loop system treating $u$ as a parameter




Neural network verification algorithm for bounds on $u$



Reachability of open-loop system + Neural network verification bounds




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

**In this talk**: how to suitably define this 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

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

[5]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

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

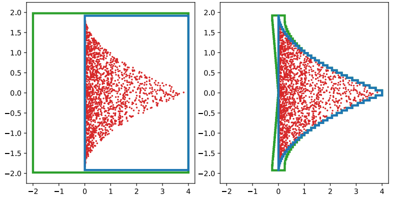
- Interval arithmetic allows computing Jacobian bounds efficiently.

[5] 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

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

- Interval arithmetic allows computing Jacobian bounds efficiently.
- `npinterval`[5]: 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.
$g(x_1,x_2) = [x_2^2 + 2x_1x_2 + x_2^2, 4\sin(x_1/4)\cos(x_2/4) - 4\cos(x_1/4)\sin(x_2/4)]^T$

[5] 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 $u = N(x)$

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

[6] 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)

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

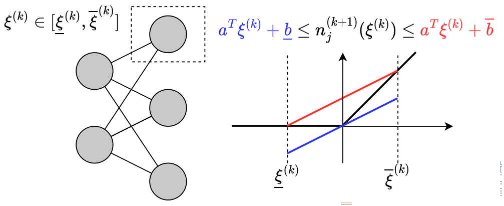
**Input-output bounds:** Given a neural network controller $u = N(x)$

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

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

**CROWN**[6]

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



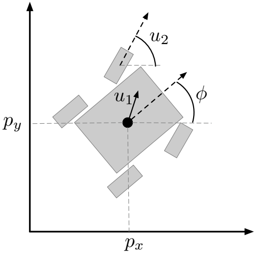

[6] Zhang, Weng, Chen, Hsieh, Daniel. "Efficient neural network robustness certification with general activation functions." NeurIPS, 2018.

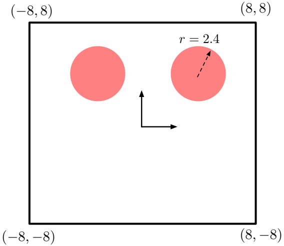
### Dynamics of bicycle

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

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

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



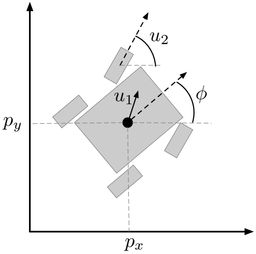



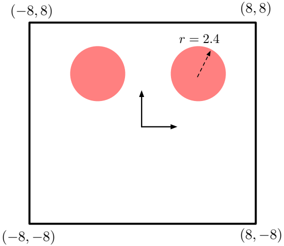
## Dynamics of bicycle

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

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

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



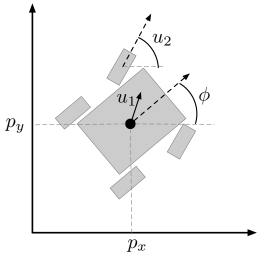

**Goal:** steer the bicycle to the origin avoiding the obstacles

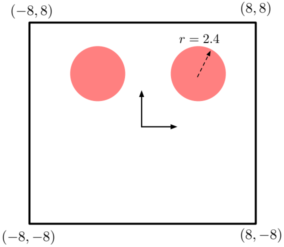
**Dynamics of bicycle**

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

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

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



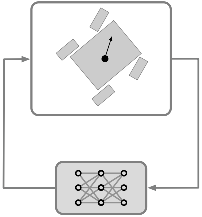

**Goal:** steer the bicycle to the origin avoiding the obstacles

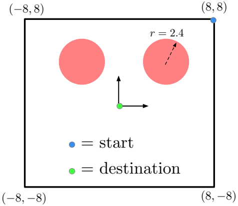
- train a feedforward neural network $4 \mapsto 100 \mapsto 100 \mapsto 2$ using data from model predictive control

- 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:

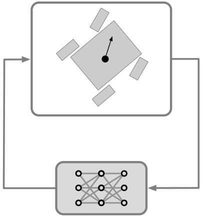
$$\dot{\underline{x}} = \underline{d}(\underline{x}, \overline{x}, \underline{u}, \overline{u}, \underline{w}, \overline{w})$$
$$\dot{\overline{x}} = \overline{d}(\underline{x}, \overline{x}, \underline{u}, \overline{u}, \underline{w}, \overline{w})$$

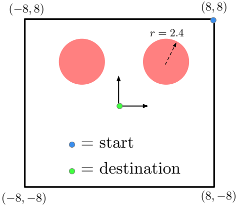
$\underline{u} \leq N(x) \leq \overline{u}$, for every $x \in [\underline{x}, \overline{x}]$.

- 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

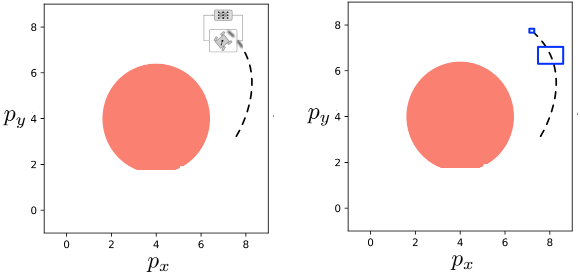





Euler integration with step $h$:

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

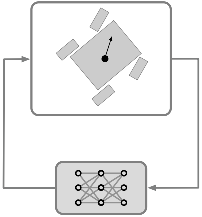
$\underline{u}_0 \leq N(x) \leq \overline{u}_0$, for every $x \in [\underline{x}_0, \overline{x}_0]$.

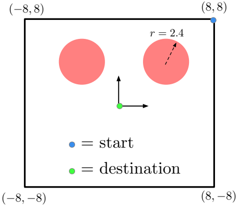


- start from $(8,8)$ toward $(0,0)$
- $\mathcal{X}_0 = [\underline{x}_0, \overline{x}_0]$ with

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

- CROWN for verification of neural network

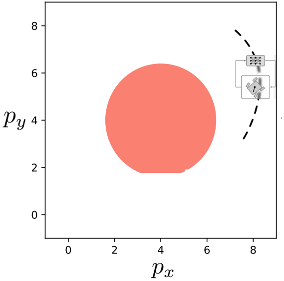


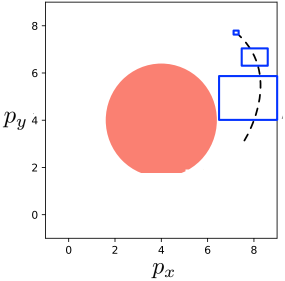



Euler integration with step $h$:

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

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

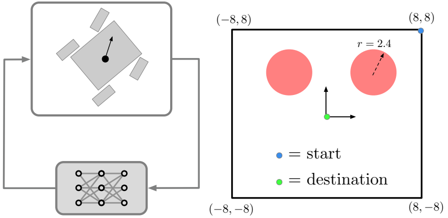


- start from $(8, 8)$ toward $(0, 0)$
- $\mathcal{X}_0 = [\underline{x}_0, \overline{x}_0]$ with

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

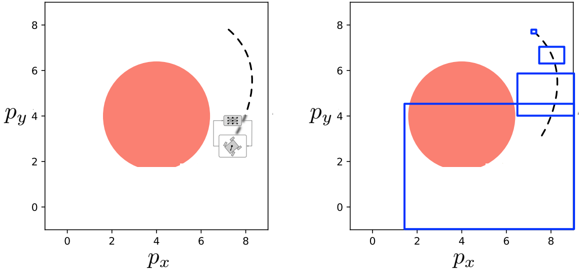
- CROWN for verification of neural network




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 $x \in [\underline{x}_2, \overline{x}_2]$.



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 $1 < K \leq 3$.

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 $1 < K \leq 3$.

### Naive interconnection approach

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.

### Interaction approach

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

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

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 $1 < K \leq 3$.

### Naive interconnection approach

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.

### Interaction approach

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

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

This system is stable.

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

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

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

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

**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[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:

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

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

**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**[8]

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

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

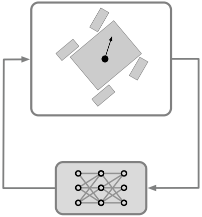
is a **decomposition function for the closed-loop system**.

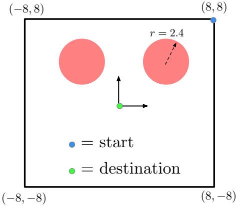
[8] Jafarpour, Harapanahalli, Coogan. "Efficient Interaction-aware Interval Reachability of Neural Network Feedback Loops", arXiv, 2003

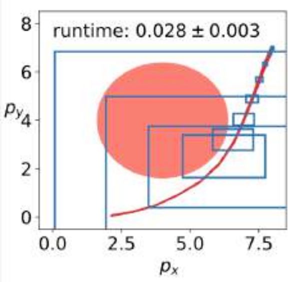
- start from $(8, 7)$ toward $(0, 0)$
- $\mathcal{X}_0 = [\underline{x}_0, \overline{x}_0]$ with

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

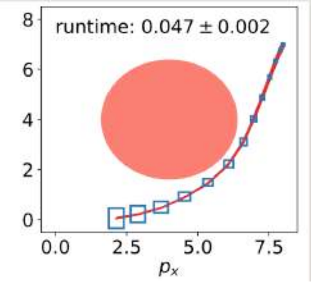
- CROWN for verification of neural network






Naive interconnection approach




interaction approach

# Conclusions

and follow-up work

- Reachability as a framework for safety certification
- Mixed monotone theory as a computationally efficient method for reachability
- Reachability of neural network controlled systems
- 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. arXiv, Sep 2023