Mixed-monotone Theory for Verification of Autonomous System

Saber Jafarpour

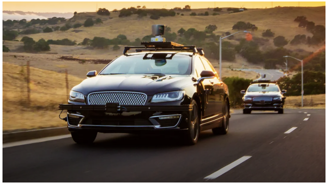
University of Colorado Boulder

April 8, 2024

Energy/power systems

Air mobility

Autonomous driving



Manufacturing

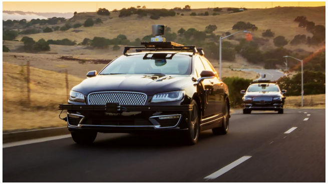
Transportation systems

Agriculture

Energy/power systems

Air mobility

Autonomous driving



Manufacturing

Transportation systems

Agriculture

An important goal (Safe Autonomy)

Perform their tasks while ensuring **safety** and **robustness** of the system.

In this talk: Autonomous systems with learning-enabled components

In this talk: Autonomous systems with learning-enabled components

Machine learning is a driving forces for developments in autonomous systems

In this talk: Autonomous systems with learning-enabled components

Machine learning is a driving forces for developments in autonomous systems

- availability of data and computation tools
- performance and efficiency

In this talk: Autonomous systems with learning-enabled components

Machine learning is a driving forces for developments in autonomous systems

- availability of data and computation tools
- performance and efficiency

Success stories and potential applications

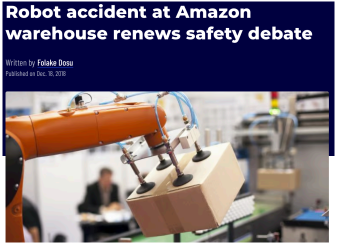
NVIDIA self driving car

Amazon fulfillment centers

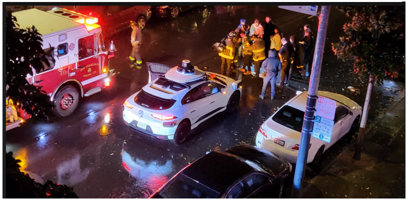
Manufacturing

But can we ensure their safety?

Tesla Slams Right Into Overturned Truck While on Autopilot

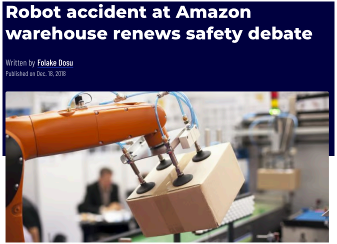


Waymo driverless car strikes bicyclist in San Francisco, causes minor injuries



But can we ensure their safety?

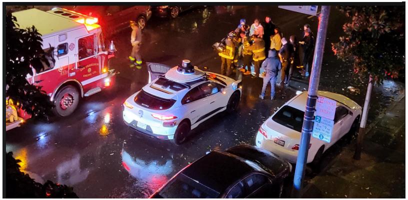
Tesla Slams Right Into Overturned Truck While on Autopilot



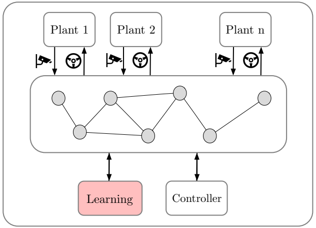
Robot accident at Amazon warehouse renews safety debate

Written by Folake Dosu
Published on Dec. 18, 2018

Waymo driverless car strikes bicyclist in San Francisco, causes minor injuries

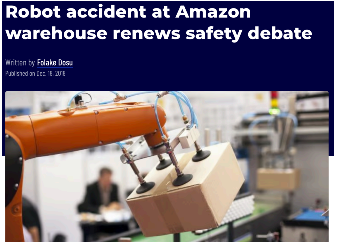


What is different with Learning-based components?



But can we ensure their safety?

Tesla Slams Right Into Overturned Truck While on Autopilot

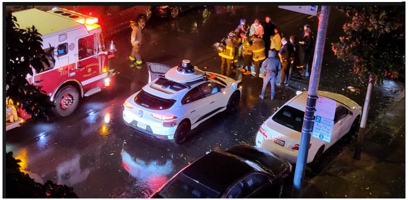


Robot accident at Amazon warehouse renews safety debate

Written by Folake Dosu

Published on Dec. 18, 2018

Waymo driverless car strikes bicyclist in San Francisco, causes minor injuries



- limited guarantee in their design

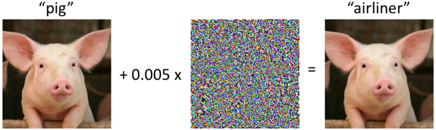
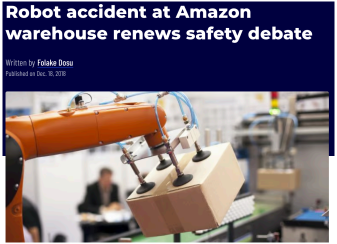


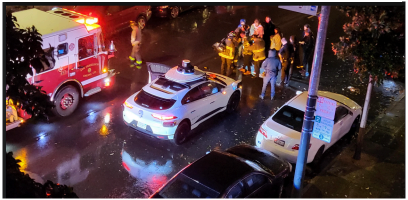
Image credit: MIT CSAIL

But can we ensure their safety?

Tesla Slams Right Into Overturned Truck While on Autopilot

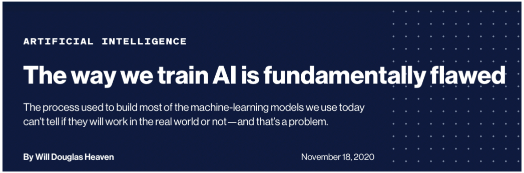


Waymo driverless car strikes bicyclist in San Francisco, causes minor injuries



- limited guarantee in their design

MIT Technology Review


ARTIFICIAL INTELLIGENCE

The way we train AI is fundamentally flawed

The process used to build most of the machine-learning models we use today can't tell if they will work in the real world or not—and that's a problem.

By Will Douglas Heaven

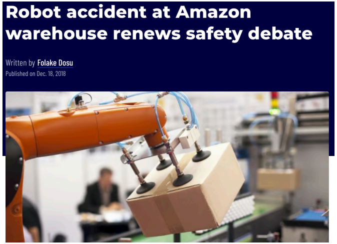
November 18, 2020

Learning-enabled Autonomous Systems

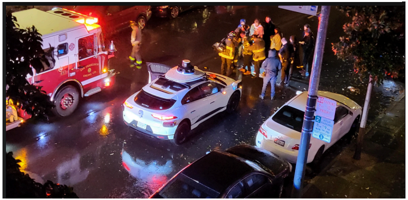
Safety Assurance as a Challenge

But can we ensure their safety?

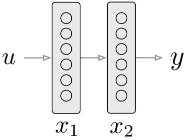
Tesla Slams Right Into Overturned Truck While on Autopilot



Waymo driverless car strikes bicyclist in San Francisco, causes minor injuries



- limited guarantee in their design
- large # of parameters with nonlinearity

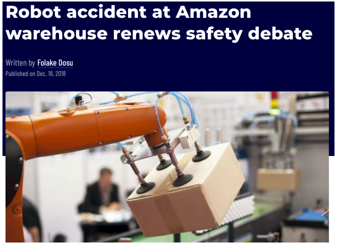


$478 \times 100 \times 100 \times 10$

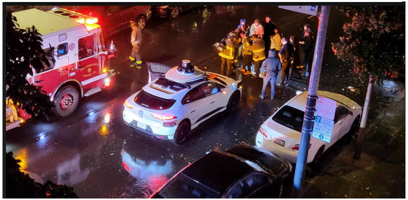
$\#$ of parameters ~ 90000
$\#$ of activation patterns $\sim 10^{60}$

But can we ensure their safety?

Tesla Slams Right Into Overturned Truck While on Autopilot



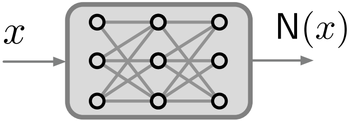
Waymo driverless car strikes bicyclist in San Francisco, causes minor injuries



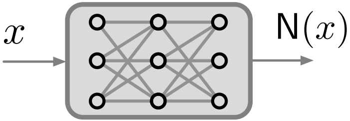
- limited guarantee in their design
- large # of parameters with nonlinearity

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

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



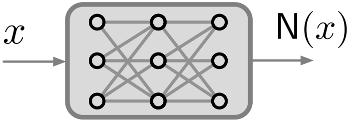
ML focus on safety and robustness of **stand-alone** learning algorithms



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

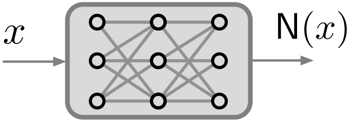


Different approaches:

- 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 part of the system** (controller, motion planner, obstacle detection)

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



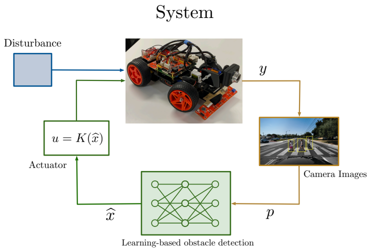
Different approaches:

- 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)

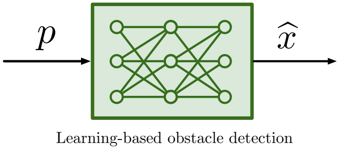
In autonomous systems, learning algorithms are **a part of the system** (controller, motion planner, obstacle detection)

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

Perception-based Obstacle Avoidance



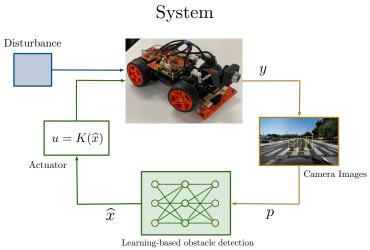
In-the-loop



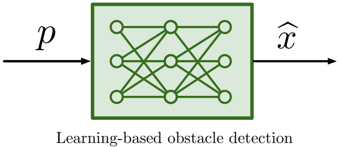
trained offline using images

Stand-alone

Perception-based Obstacle Avoidance



In-the-loop

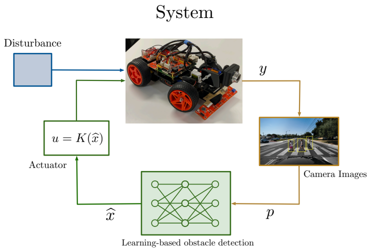


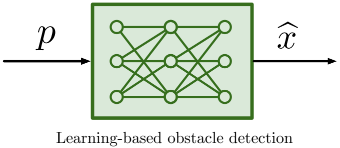
trained offline using images

Stand-alone

- **stand-alone**: estimation of states using learning algorithm
- **in-the-loop**: closed-loop system avoid the obstacle

Perception-based Obstacle Avoidance





trained offline using images

In-the-loop

Stand-alone

- **stand-alone**: estimation of states using learning algorithm
- **in-the-loop**: closed-loop system avoid the obstacle

> **In-the-loop**: how the autonomous system perform with the learning algorithm as a part of it.

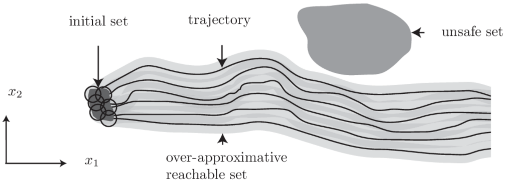
Ensure safety of the autonomous system with learning algorithms **in-the-loop**

Ensure safety of the autonomous system with learning algorithms **in-the-loop**

Safety of autonomous system using **reachability analysis**

Ensure safety of the autonomous system with learning algorithms **in-the-loop**

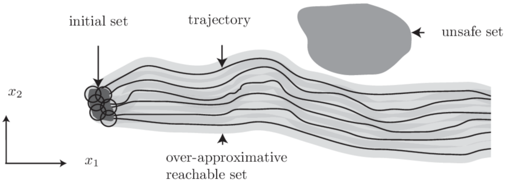
Safety of autonomous system using **reachability analysis**



Reachability analysis estimates the evolution of the autonomous system

Ensure safety of the autonomous system with learning algorithms **in-the-loop**

Safety of autonomous system using **reachability analysis**



Reachability analysis estimates the evolution of the autonomous system

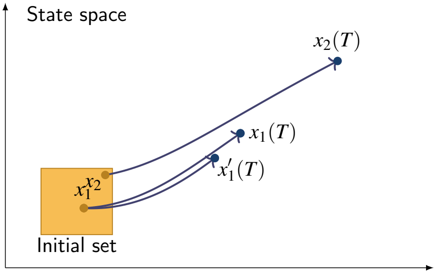
In this talk:

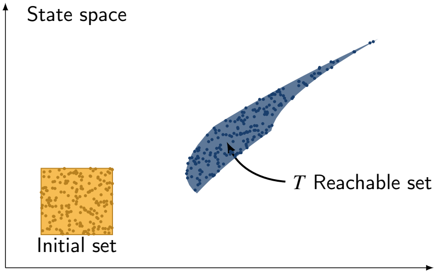
1. control-theoretic tools for efficient and scalable reachability
2. applications to safety assurance of learning-enabled systems

Outline of this talk

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

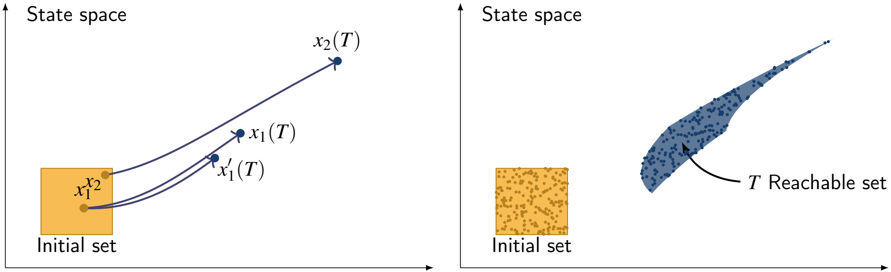
System : $\dot{x} = f(x, w)$ **State** : $x \in \mathbb{R}^n$ **Uncertainty** : $w \in \mathcal{W} \subseteq \mathbb{R}^m$





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?

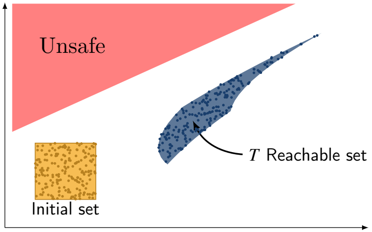
- T**-reachable sets** characterize evolution of the system

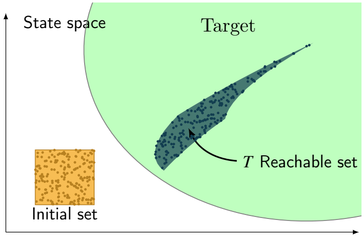
$$\mathcal{R}_f(T, \mathcal{X}_0, \mathcal{W}) = \{x_w(T) \mid x_w(\cdot) \text{ is a traj for some } w(\cdot) \in \mathcal{W} \text{ with } x_0 \in \mathcal{X}_0\}$$

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



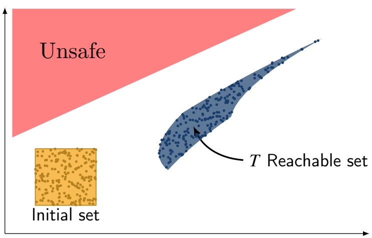


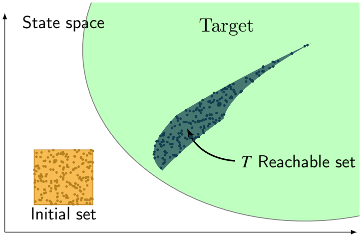
$$\mathcal{R}_f(T, \mathcal{X}_0, \mathcal{W}) \cap \text{ Unsafe set } = \emptyset$$

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

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

- Example: Reach-avoid problem





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

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

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

Computing the T-reachable sets are computationally challenging

Computing the T-reachable sets are computationally challenging

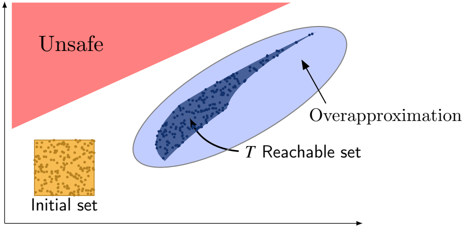
Solution: over-approximations of reachable sets

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

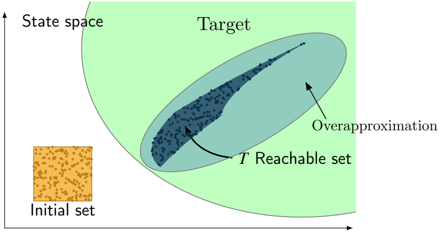
Computing the T-reachable sets are computationally challenging

Solution: over-approximations of reachable sets

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

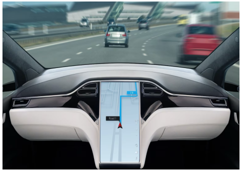


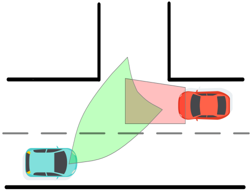
$\overline{\mathcal{R}}_f(T, \mathcal{X}_0, \mathcal{W}) \cap \text{Unsafe set} = \emptyset$



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

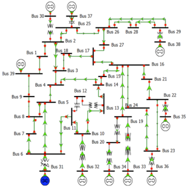
Autonomous Driving:

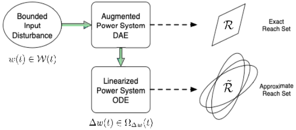




Althoff, 2014

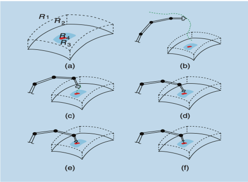
Power grids:



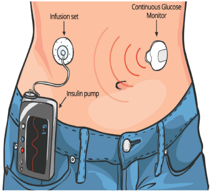


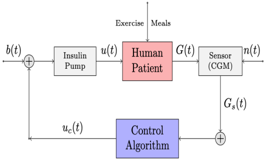
Chen and Domınguez-Garcıa, 2016

Robot-assisted Surgery:



Drug Delivery:





Chen, Dutta, and Sankaranarayanan, 2017

Reachability of dynamical system is an old problem: ~ 1980

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

Different approaches for approximating reachable sets

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

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

Different approaches for approximating reachable sets

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

Most of the classical reachability approaches are computationally heavy and not scalable to large-size systems

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

Different approaches for approximating reachable sets

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

Most of the classical reachability approaches are computationally heavy and not scalable to large-size systems

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

Outline of this talk

- Reachability Analysis
- **Monotone System Theory**
- 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.

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

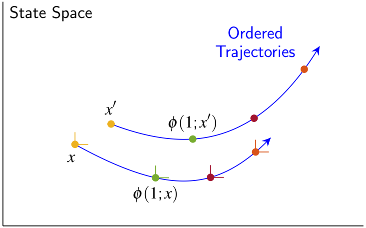
A dynamical system $\dot{x} = f(x, w)$ is monotone if

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

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

Theorem[1]: Monotonicity test

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



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

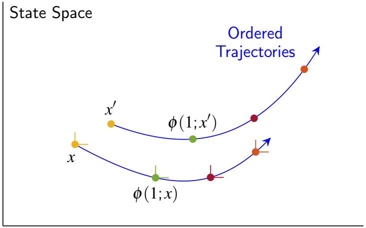
A dynamical system $\dot{x} = f(x, w)$ is monotone if

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

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

Theorem[1]: Monotonicity test

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



In this talk: monotone system theory for reachability analysis

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

Monotone System

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

Non-monotone System

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

Theorem (classical result)

For a monotone system $\dot{x} = f(x,w)$ with $w \in \mathcal{W} = [\underline{w}, \overline{w}]$

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

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

Theorem (classical result)

For a monotone system $\dot{x} = f(x, w)$ with $w \in \mathcal{W} = [\underline{w}, \overline{w}]$

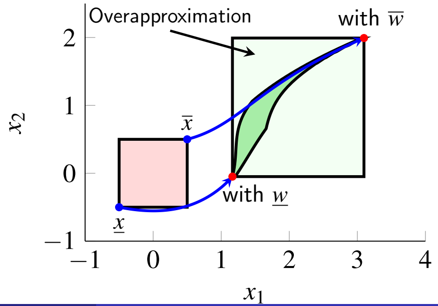
$$\mathcal{R}_f(t, [\underline{x}_0, \overline{x}_0], [\underline{w}, \overline{w}]) \subseteq [x_{\underline{w}}(t), x_{\overline{w}}(t)]$$

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

Example:

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

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



A large number of the dynamical systems are **not** monotone

A large number of the dynamical systems are **not** monotone

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

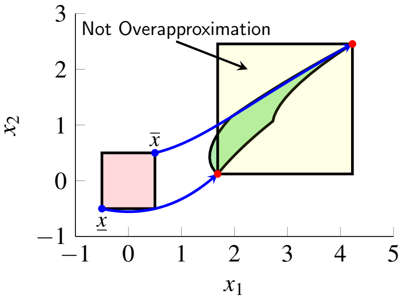
A large number of the dynamical systems are **not** monotone

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

Example:

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

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



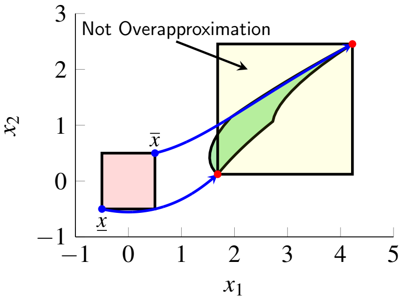
A large number of the dynamical systems are **not** monotone

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

Example:

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

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



How to over-approximate the reachable sets of non-monotone systems?

- **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 $\overline{d}$

f locally Lipschitz $\Longrightarrow$ a decomposition function exists

Southeast partial order $\leq_{\mathrm{SE}}$:

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

Southeast partial order $\leq_{\mathrm{SE}}$:

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

Theorem (Classical Result)

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

$$\begin{bmatrix} \underline{x}_0 \\ \overline{x}_0 \end{bmatrix} \leq_{\mathrm{SE}} \begin{bmatrix} \underline{y}_0 \\ \overline{y}_0 \end{bmatrix}, \quad \begin{bmatrix} \underline{u} \\ \overline{u} \end{bmatrix} \leq_{\mathrm{SE}} \begin{bmatrix} \underline{w} \\ \overline{w} \end{bmatrix} \quad \Longrightarrow \quad \begin{bmatrix} \underline{x}_{[\underline{u},\overline{u}]}(t) \\ \overline{x}_{[\underline{u},\overline{u}]}(t) \end{bmatrix} \leq_{\mathrm{SE}} \begin{bmatrix} \underline{y}_{[\underline{w},\overline{w}]}(t) \\ \overline{y}_{[\underline{w},\overline{w}]}(t) \end{bmatrix}$$

Southeast partial order $\leq_{\mathrm{SE}}$:

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

Theorem (Classical Result)

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

$$\begin{bmatrix} \underline{x}_0 \\ \overline{x}_0 \end{bmatrix} \leq_{\mathrm{SE}} \begin{bmatrix} \underline{y}_0 \\ \overline{y}_0 \end{bmatrix}, \quad \begin{bmatrix} \underline{u} \\ \overline{u} \end{bmatrix} \leq_{\mathrm{SE}} \begin{bmatrix} \underline{w} \\ \overline{w} \end{bmatrix} \quad \Longrightarrow \quad \begin{bmatrix} \underline{x}_{[\underline{u},\overline{u}]}(t) \\ \overline{x}_{[\underline{u},\overline{u}]}(t) \end{bmatrix} \leq_{\mathrm{SE}} \begin{bmatrix} \underline{y}_{[\underline{w},\overline{w}]}(t) \\ \overline{y}_{[\underline{w},\overline{w}]}(t) \end{bmatrix}$$

Key idea: use monotonicity of the embedding system to study the original dynamical system

A short (and incomplete) Literature review:

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

S. Coogan and M. Arcak. Stability of traffic flow networks with a polytree topology. Automatica, 2016

A short (and incomplete) Literature review:

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

S. Coogan and M. Arcak. Stability of traffic flow networks with a polytree topology. Automatica, 2016

In this talk: use embedding system to study reachability of the original system

Original System:

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

$$\mathcal{W} = [2.2\ ,\ 2.3]$$

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:

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

$$\mathcal{W} = [2.2\ ,\ 2.3]$$

blue = cooperative, red = competitive

Embedding System:

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

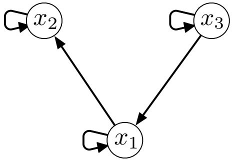
Decomposition function

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

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

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

Linear systems

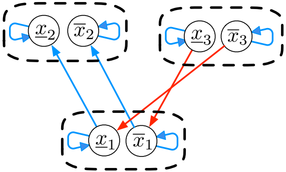


Original system

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

Embedding system

$$\dot{\underline{x}} = [A]^{\mathrm{Mzl}}\underline{x} + [A]^{\mathrm{n-Mzl}}\overline{x} + B^{+}\underline{w} + B^{-}\overline{w}$$
$$\dot{\overline{x}} = [A]^{\mathrm{Mzl}}\overline{x} + [A]^{\mathrm{n-Mzl}}\underline{x} + B^{+}\overline{w} + B^{-}\underline{w}$$



How to compute a decomposition function for a system?

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

Mean-value Inequality

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

Then

$$\begin{bmatrix} \underline{d}(\underline{x},\overline{x}) \\ \overline{d}(\underline{x},\overline{x}) \end{bmatrix} = \begin{bmatrix} \left[\min_{z\in[\underline{x},\overline{x}]} \frac{\partial f}{\partial x}\right]^{+} & \left[\min_{z\in[\underline{x},\overline{x}]} \frac{\partial f}{\partial x}\right]^{-} \\ \left[\max_{z\in[\underline{x},\overline{x}]} \frac{\partial f}{\partial x}\right]^{-} & \left[\max_{z\in[\underline{x},\overline{x}]} \frac{\partial f}{\partial x}\right]^{+} \end{bmatrix} \begin{bmatrix} \underline{x} \\ \overline{x} \end{bmatrix}$$

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

How to compute a decomposition function for a system?

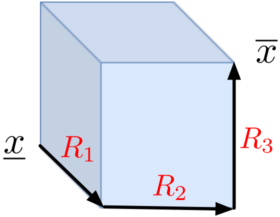
Theorem[2]

Jacobian-based: $\dot{x} = f(x, w)$ with differentiable f, then

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

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

- Interval analysis for computing Jacobian bounds.
- `immrax`: Toolbox that implements interval analysis in `JAX`.



[2] **SJ** and A. Harapanahalli and S. Coogan, L4DC, 2023

Decomposition Functions

A Jacobian-based approach

How to compute a decomposition function for a system?

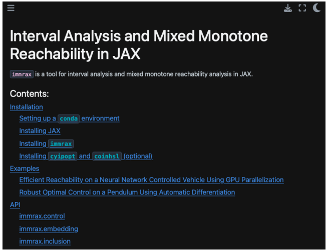
Theorem[2]

Jacobian-based: $\dot{x} = f(x, w)$ with differentiable f, then

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

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

- Interval analysis for computing Jacobian bounds.
- `immrax`: Toolbox that implements interval analysis in `JAX`.



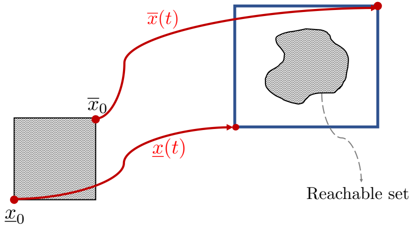
[2] **SJ** and A. Harapanahalli and S. Coogan, L4DC, 2023

Theorem[3]

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



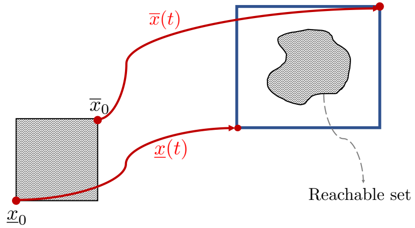
[3] H. Smith, Journal of Difference Equations and Applications, 2008

Theorem[3]

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

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

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



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

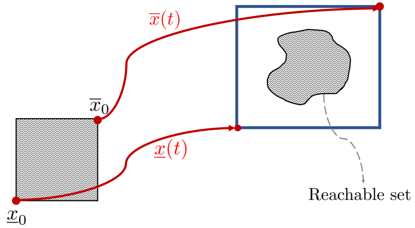
[3] H. Smith, Journal of Difference Equations and Applications, 2008

Theorem[3]

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

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

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

[3] H. Smith, Journal of Difference Equations and Applications, 2008

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

Decomposition function

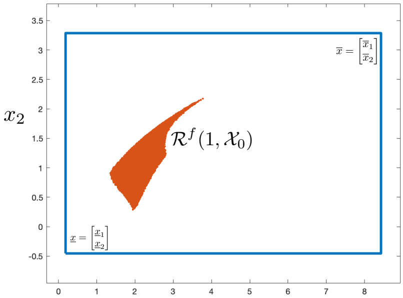
$$\underline{d}(\underline{x}, \overline{x}, \underline{w}, \overline{w}) = \begin{bmatrix} \underline{x}_2^3 + \underline{w} \\ \underline{x}_1 \end{bmatrix} + \begin{bmatrix} -\overline{x}_2 \\ 0 \end{bmatrix}$$

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

Embedding System:

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

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

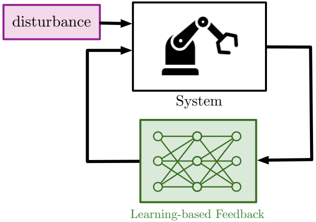


Outline of this talk

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

- **In this part:** Learning-based component as a controller

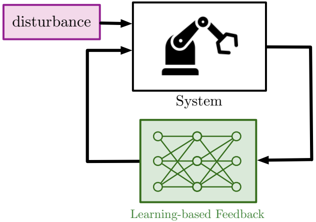
- **In this part:** Learning-based component as a controller



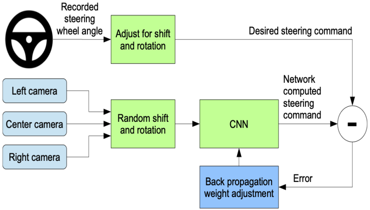
- **In this part:** Learning-based component as a controller

Issues with traditional controllers:

1. computationally burdensome
2. interaction with human
3. complicated representation



Self driving vehicles:

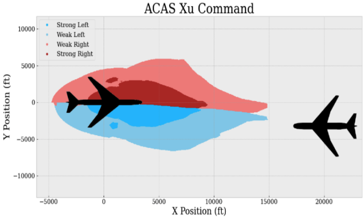


M. Bojarski, et al., NeurIPS, 2016.

Robotic motion planning:

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

Collision avoidance:

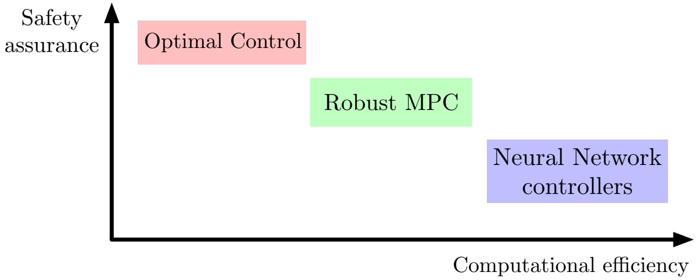


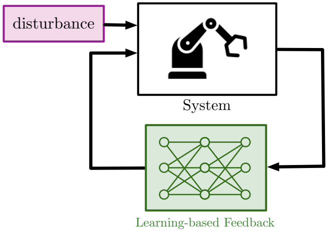
K. Julian, et. al., DASC, 2016.

Safety of learning-enabled autonomous systems **cannot be completely ensured** at the design level[4]

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

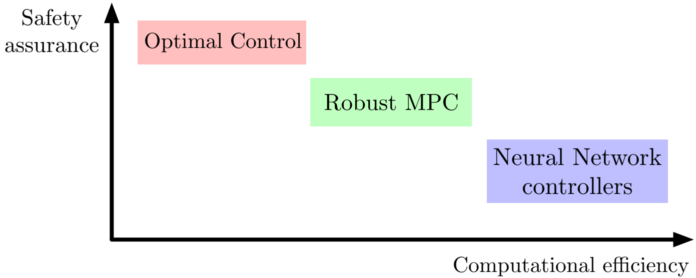
Safety of learning-enabled autonomous systems **cannot be completely ensured** at the design level[4]

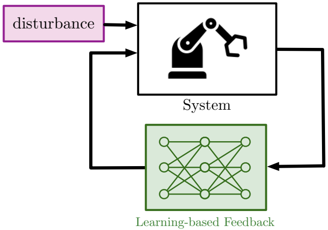




[4] Institute for Defense Analysis, The Status of Test, Evaluation, Verification, and Validation of Autonomous Systems, 2018

Safety of learning-enabled autonomous systems **cannot be completely ensured** at the design level[4]





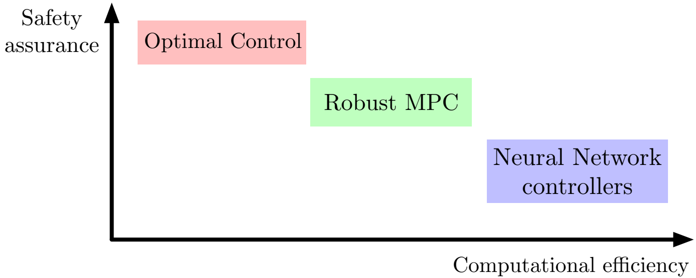
Design a mechanism that can do **run-time** safety verification

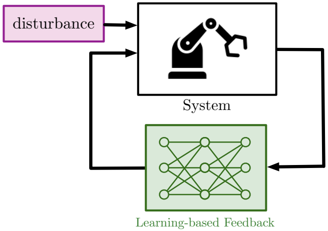
[4] Institute for Defense Analysis, The Status of Test, Evaluation, Verification, and Validation of Autonomous Systems, 2018

Analysis of Learning-based Controllers

Safety Verification

Safety of learning-enabled autonomous systems **cannot be completely ensured** at the design level[4]





Design a mechanism that can do **run-time** safety verification

Our approach: reachable set over-approximations for some time in future.

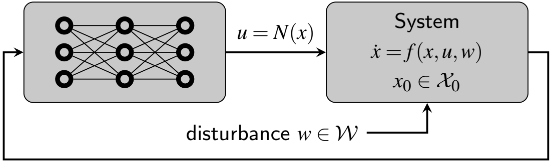
[4] Institute for Defense Analysis, The Status of Test, Evaluation, Verification, and Validation of Autonomous Systems, 2018

An open-loop nonlinear system with a neural network controller

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

safety of the closed-loop system

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

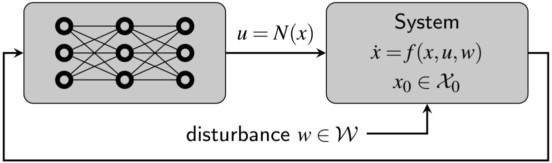


An open-loop nonlinear system with a neural network controller

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

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 neural network with k-layer:

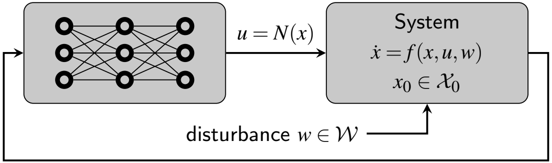
$$\xi^{(i)}(x) = \phi^{(i)}(W^{(i-1)}\xi^{(i-1)}(x) + b^{(i-1)})$$
$$x = \xi^{(0)}, \quad u = W^{(k)}\xi^{(k)}(x) + b^{(k)} := N(x),$$

An open-loop nonlinear system with a neural network 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 neural network with k-layer:

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

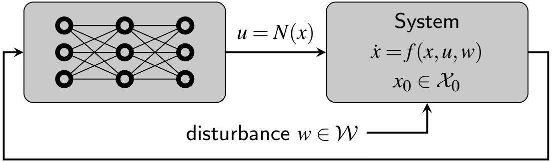
directly performing reachability on f^c is computationally challenging

An open-loop nonlinear system with a neural network 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 neural network with k-layer:

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

Rigorousness of control tools + effectiveness of ML tools

Combine our reachability frameworks with neural network verification methods

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

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

[5] H. Zhang et al., NeurIPS 2018.

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

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

Many neural network verification algorithms can produce these bounds.

ex. CROWN (H. Zhang et al., 2018), LipSDP (M. Fazlyab et al., 2019), IBP (S. Gowal et al., 2018).

[5] H. Zhang et al., NeurIPS 2018.

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

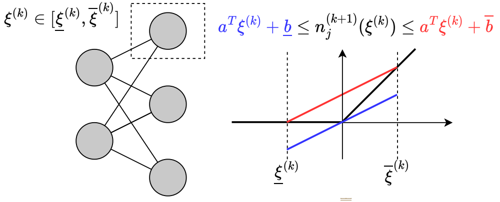
$$\underline{u}_{[\underline{x},\overline{x}]} \leq N(x) \leq \overline{u}_{[\underline{x},\overline{x}]}, \qquad \text{for all } x \in [\underline{x}, \overline{x}]$$

Many neural network verification algorithms can produce these bounds.

ex. CROWN (H. Zhang et al., 2018), LipSDP (M. Fazlyab et al., 2019), IBP (S. Gowal et al., 2018).

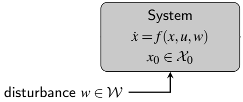
CROWN[5]

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

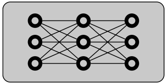


[5] H. Zhang et al., NeurIPS 2018.

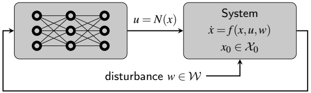
Reachability of open-loop system treating u as a parameter



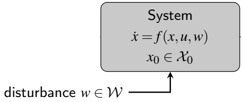
Neural network verification algorithm for bounds on u



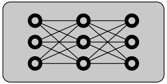
Reachability of open-loop system + Neural network verification bounds



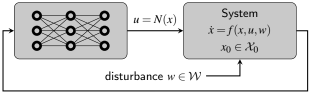
$$\dot{\underline{x}} = \underline{d}(\underline{x}, \overline{x}, \underline{u}, \overline{u}, \underline{w}, \overline{w})$$
$$\dot{\overline{x}} = \overline{d}(\underline{x}, \overline{x}, \underline{u}, \overline{u}, \underline{w}, \overline{w})$$



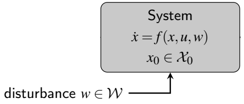
$$\underline{u}_{[\underline{x},\overline{x}]} \leq N(x) \leq \overline{u}_{[\underline{x},\overline{x}]} \quad \text{for every } x \in [\underline{x}, \overline{x}].$$



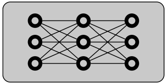
$$\dot{\underline{x}} = \underline{d}(\underline{x}, \overline{x}, \underline{u}_{[\underline{x},\overline{x}]}, \overline{u}_{[\underline{x},\overline{x}]}, \underline{w}, \overline{w})$$
$$\dot{\overline{x}} = \overline{d}(\underline{x}, \overline{x}, \underline{u}_{[\underline{x},\overline{x}]}, \overline{u}_{[\underline{x},\overline{x}]},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



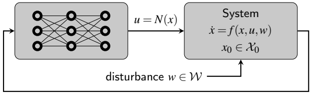
$$\dot{\underline{x}} = \underline{d}(\underline{x}, \overline{x}, \underline{u}, \overline{u}, \underline{w}, \overline{w})$$
$$\dot{\overline{x}} = \overline{d}(\underline{x}, \overline{x}, \underline{u}, \overline{u}, \underline{w}, \overline{w})$$



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

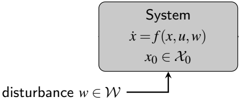


$$\dot{\underline{x}} = \underline{d}(\underline{x}, \overline{x}, \underline{u}_{[\underline{x},\overline{x}]}, \overline{u}_{[\underline{x},\overline{x}]}, \underline{w}, \overline{w})$$
$$\dot{\overline{x}} = \overline{d}(\underline{x}, \overline{x}, \underline{u}_{[\underline{x},\overline{x}]}, \overline{u}_{[\underline{x},\overline{x}]},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

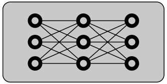


Composition approach over-approximation:
$$\mathcal{R}_{f^c}(t, \mathcal{X}_0, \mathcal{W}) \subseteq [\underline{x}(t), \overline{x}(t)]$$

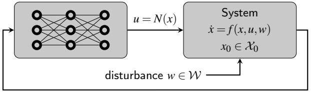
$$\dot{\underline{x}} = \underline{d}(\underline{x}, \overline{x}, \underline{u}, \overline{u}, \underline{w}, \overline{w})$$
$$\dot{\overline{x}} = \overline{d}(\underline{x}, \overline{x}, \underline{u}, \overline{u}, \underline{w}, \overline{w})$$



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



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



Composition approach 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

Case Study: Bicycle Model

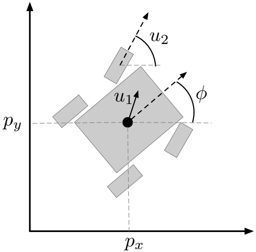
A naive compositional approach

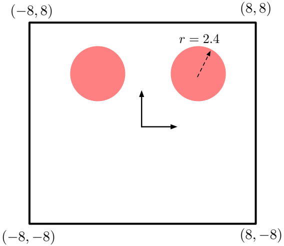
Dynamics of bicycle

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

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

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



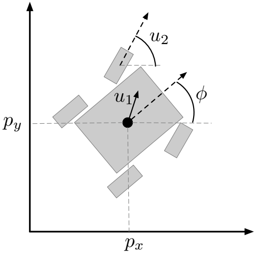


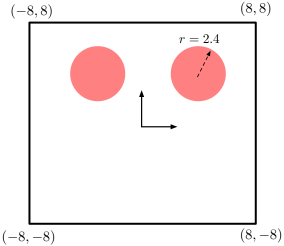
Dynamics of bicycle

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

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

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





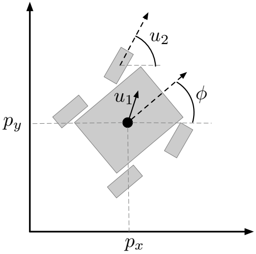
Goal: steer the bicycle to the origin avoiding the obstacles

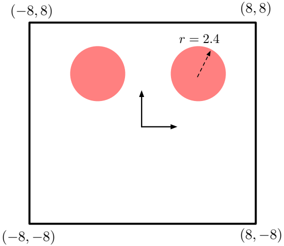
Dynamics of bicycle

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

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

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





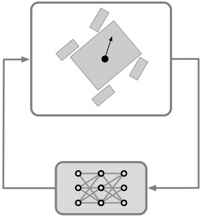
Goal: steer the bicycle to the origin avoiding the obstacles

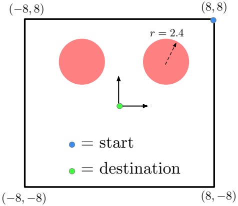
- train a feedforward neural network $4 \mapsto 100 \mapsto 100 \mapsto 2$ using data from model predictive control

- 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:

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

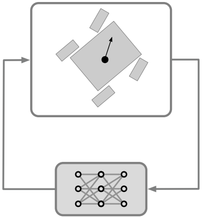
$\underline{u} \leq N(x) \leq \overline{u}$, for every $x \in [\underline{x}, \overline{x}]$.

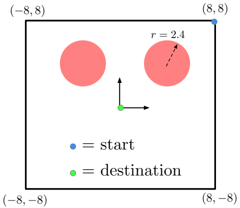
- start from $(8,8)$ toward $(0,0)$
- $\mathcal{X}_0 = [\underline{x}_0, \overline{x}_0]$ with

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

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

- CROWN for verification of neural network



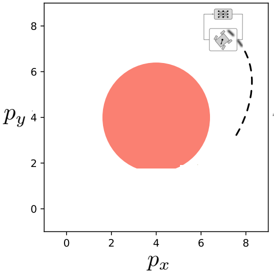


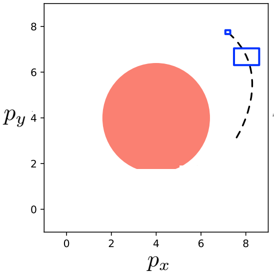
Euler integration with step h:

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

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

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

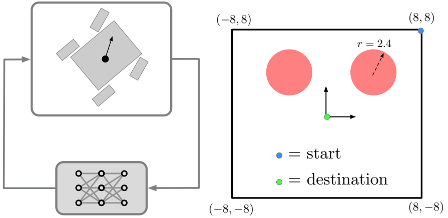




- start from $(8,8)$ toward $(0,0)$
- $\mathcal{X}_0 = [\underline{x}_0, \overline{x}_0]$ with

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

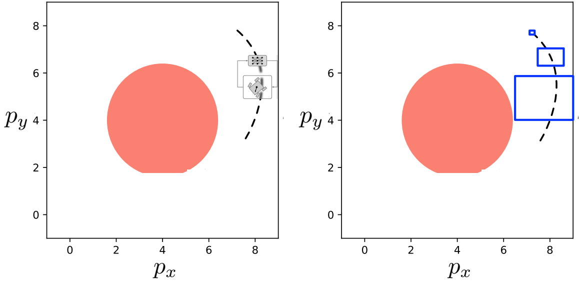
- CROWN for verification of neural network



Euler integration with step h:

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

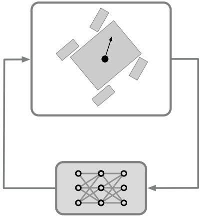
$\underline{u}_1 \leq N(x) \leq \overline{u}_1$, for every $x \in [\underline{x}_1, \overline{x}_1]$.

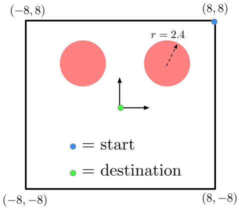


- start from $(8,8)$ toward $(0,0)$
- $\mathcal{X}_0 = [\underline{x}_0, \overline{x}_0]$ with

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

- CROWN for verification of neural network

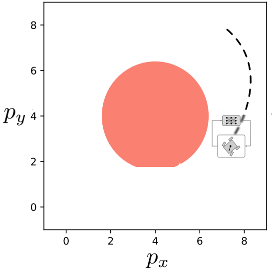


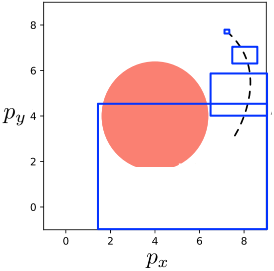


Euler integration with step h:

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

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





Stabilizing Effect of Neural Network Controllers

Issues with the compositional approach

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 $1 < K \leq 3$.

Compositional 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-aware 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 $1 < K \leq 3$.

Compositional 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-aware 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 $u = N(x)$

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

[6] H. Zhang et al., 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[6] can provide functional bounds.

CROWN functional bounds:

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

CROWN input-output bounds:

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

[6] H. Zhang et al., NeurIPS 2018.

Theorem[7]

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 effect of interactions between nonlinear system and neural network.

Interaction-aware over-approximation:

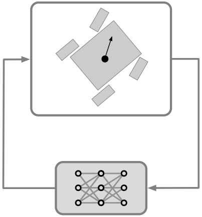
$$\mathcal{R}_{f^c}(t, \mathcal{X}_0, \mathcal{W}) \subseteq [\underline{x}(t), \overline{x}(t)]$$

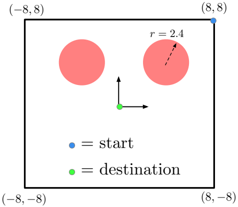
[7] **SJ** and A. Harapanahalli and S. Coogan, under review, 2023

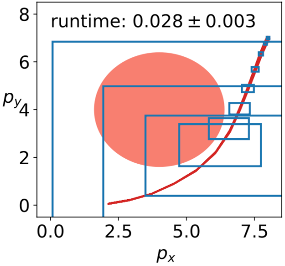
- start from $(8,7)$ toward $(0,0)$
- $\mathcal{X}_0 = [\underline{x}_0, \overline{x}_0]$ with

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

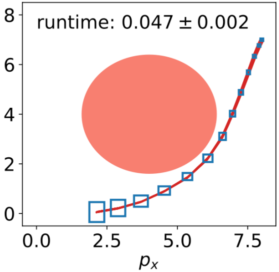
- CROWN for verification of neural network







Composition approach

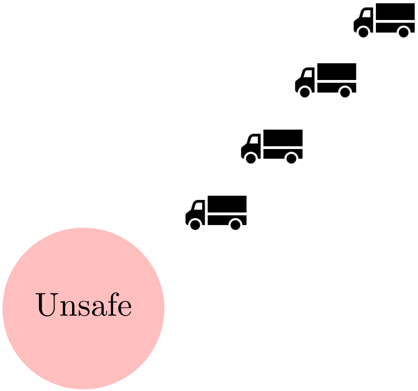


Interaction-aware approach

Dynamics of the jth vehicle

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

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



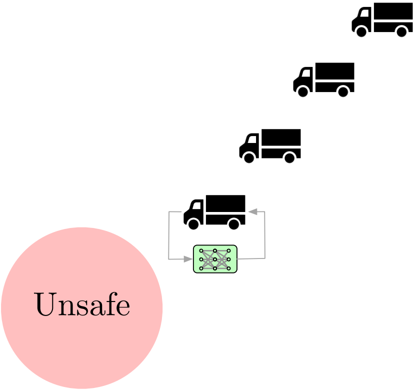
Dynamics of the jth vehicle

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

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

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

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



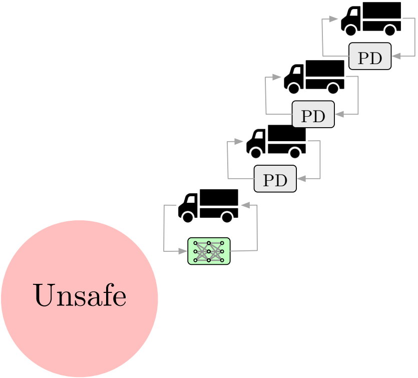
Dynamics of the jth vehicle

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

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

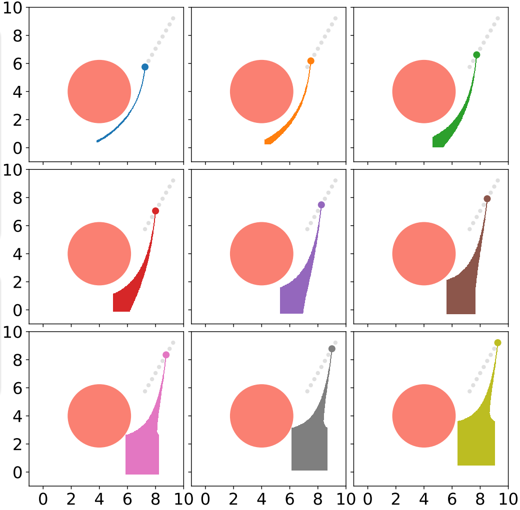


Dynamics of the jth vehicle

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

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

- compositional approach
- a platoon of 9 vehicles
- reachable overapproximations for $t \in [0, 1.5]$



Dynamics of the jth vehicle

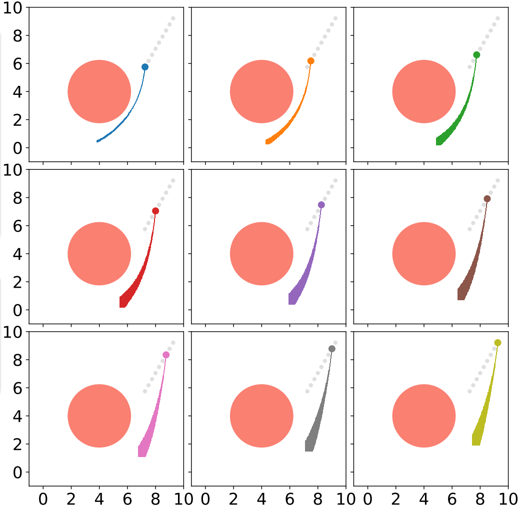
$$\dot{p}_x^j = v_x^j, \qquad \dot{v}_x^j = \tanh(u_x^j) + w_x^j,$$
$$\dot{p}_y^j = v_y^j, \qquad \dot{v}_y^j = \tanh(u_y^j) + w_y^j,$$

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

- interaction-aware approach
- 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

Key takeaways

- reachability as a framework for safety certification of autonomous systems
- developed computationally efficient and scalable approaches for reachability using monotone system theory
- run-time verification of neural network controlled systems
- capture stabilizing effect of learning-based components