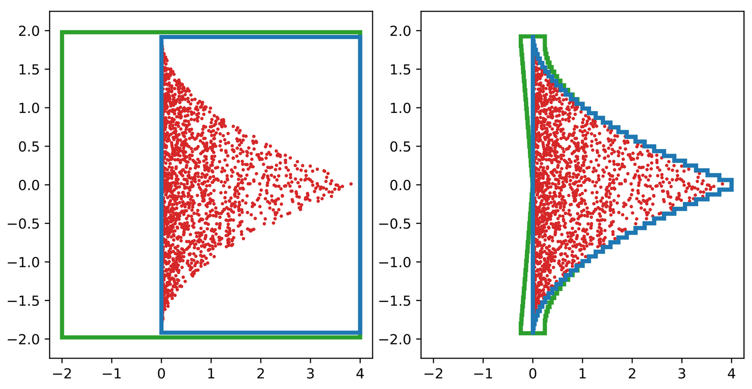
# A Toolbox for Fast Interval Arithmetic in `numpy` with an Application to Formal Verification of Neural Network Controlled Systems


**Akash Harapanahalli[1], Saber Jafarpour[1] and Samuel Coogan[1]**

(1) Georgia Institute of Technology, {aharapan,saber,sam.coogan}@gatech.edu


## Inclusion Functions

**Goal:** over-approximate the output of a mapping using intervals.



- $[f] : \mathbb{IR}^n \to \mathbb{IR}^m$ is an *inclusion function* for $f : \mathbb{R}^n \to \mathbb{R}^m$ if for every $x \in [\underline{x}, \overline{x}] = [x]$,
$$f(x) \in [f]([x]).$$
- Inclusion functions can capture localized behaviors of functions—they preserve the structure when the intervals are small.

### Tight Inclusion Function

The inclusion function with the tightest input-output interval

$$[f]([x]) = \left[ \inf_{x\in[x]} f(x), \ \sup_{x\in[x]} f(x) \right]$$

- For a general function $g$, finding its tight inclusion function is computationally intractable.

## Efficient Natural Inclusion Functions Using `npinterval`

### Natural Inclusion Functions

Given $f = e_1 \circ \cdots \circ e_n$, and inclusion functions $[e_1], \ldots, [e_n]$,

$$[f] = [e_1] \circ \cdots \circ [e_n]$$

is a natural inclusion function for $f$.

- `npinterval` defines a new `interval` dtype for `numpy`.
- Standard elementary `ufuncs` map to their tight inclusion function in compiled C.
- Familiar interface, support for $n$-dimensional arrays, matrix operations, vectorization.

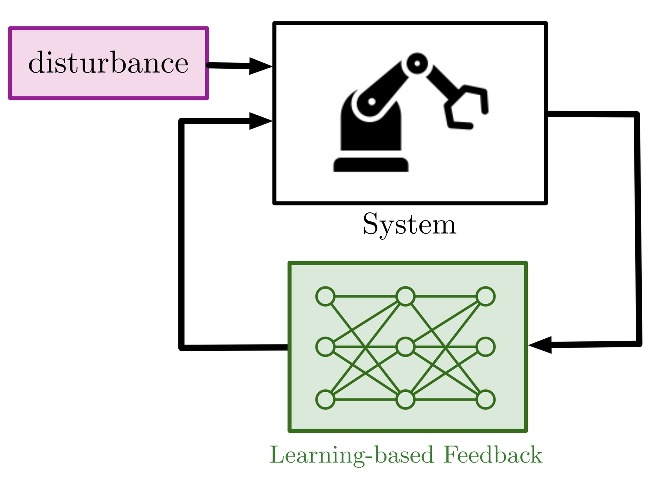
```
import numpy as np, interval
i = np.interval(1, 2)
a = np.array([i+2, np.exp(i)])
print(np.sqrt(a))
 >> [([1.732, 2])
     ([1.649, 2.718])]
print(a.dtype)
 >> interval
```

## Application: Neural Network Controlled Systems

- Neural networks are deployed as controllers in safety-critical applications (self driving vehicle and mobile robots).

### Problem Statement

Under uncertainty, ensure safety of the closed-loop system.

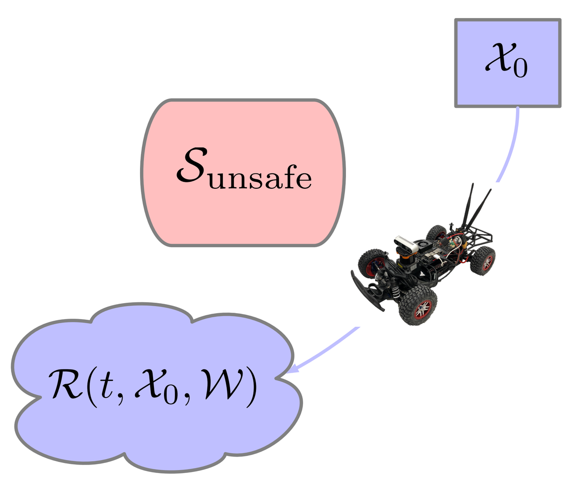



**Challenges**:

1. Neural networks are brittle with respect to input perturbations
2. The error can compound in the closed-loop interconnection.

**Our approach**: Safety verification via interval reachability

- Disturbance $\mathcal{W}$ and initial uncertainty $\mathcal{X}_0$
- The *reachable set*
$$\mathcal{R}(t, \mathcal{X}_0, \mathcal{W}) = \{x(t) \text{ is a trajectory}\}$$
- Unsafe set $\mathcal{S}_{\text{unsafe}} \subseteq \mathbb{R}^n$




Find an over-approximation $\overline{\mathcal{R}}(t, \mathcal{X}_0, \mathcal{W})$, and check if

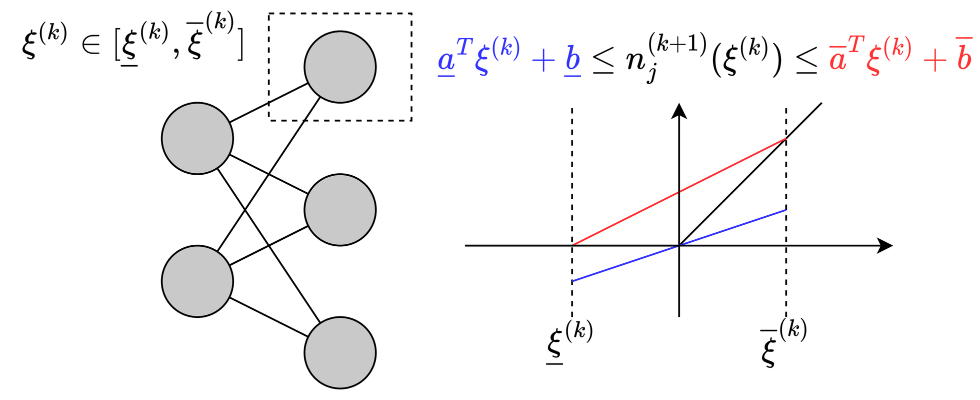
$$\overline{\mathcal{R}}(t, \mathcal{X}_0, \mathcal{W}) \bigcap \mathcal{S}_{\text{unsafe}} = \varnothing$$

## Inclusion Functions for Neural Networks

Find $\underline{N}, \overline{N}$ such that for every $x \in [\underline{x}, \overline{x}] \subseteq [\underline{y}, \overline{y}]$,

$$\underline{N}_{[y]}([x]) \leq N(x) \leq \overline{N}_{[y]}([x]).$$

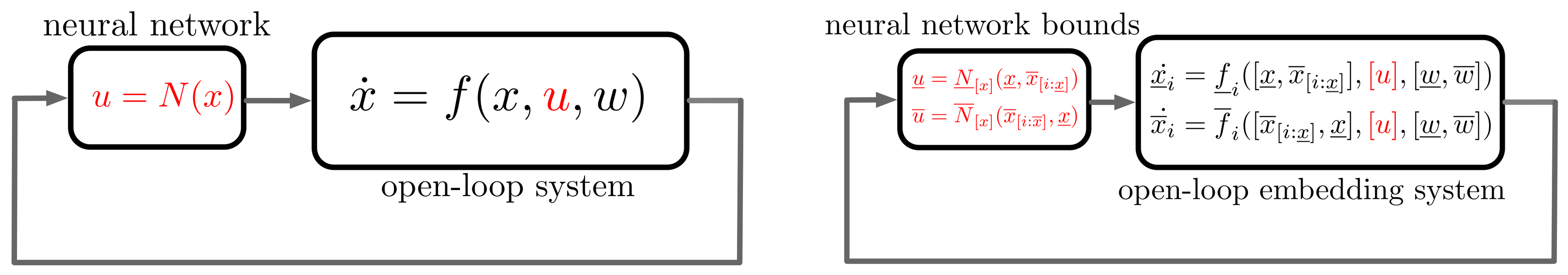
- $\underline{N}, \overline{N}$: neural network verification algorithms such as CROWN, IBP, LipSDP.
- CROWN [1] provides linear bounds $\underline{N}_{[y]}$ and $\overline{N}_{[y]}$.



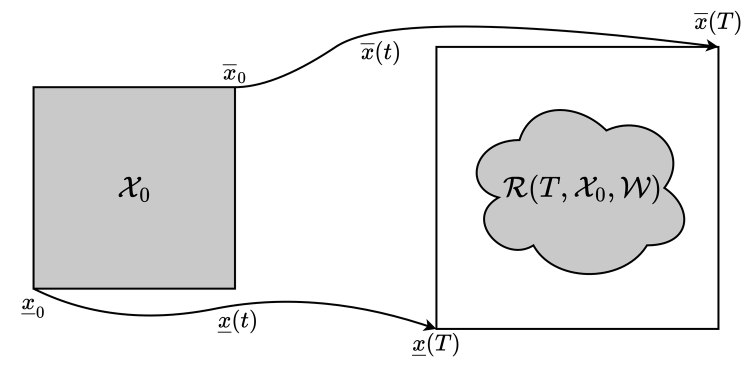

## Interval Reachability of Neural Network Controlled Systems

Consider $\dot{x} = f(x, N(x), w)$ with open-loop inclusion function $[f]$, and inclusion function $[N]_{[y]}$. The *closed-loop embedding system* is

$$\dot{\underline{x}}_i = \underline{f}_i([\underline{x}, \overline{x}_{i:\underline{x}}], [N]_{[x]}([\underline{x}, \overline{x}_{i:\underline{x}}]), [\underline{w}, \overline{w}]),$$
$$\dot{\overline{x}}_i = \overline{f}_i([\underline{x}_{i:\overline{x}}, \overline{x}], [N]_{[x]}([\underline{x}_{i:\overline{x}}, \overline{x}]), [\underline{w}, \overline{w}])$$

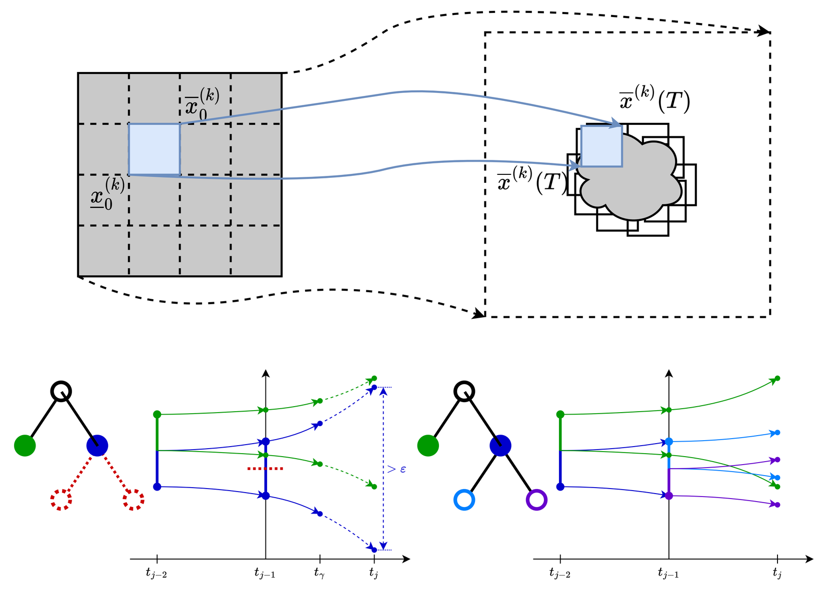


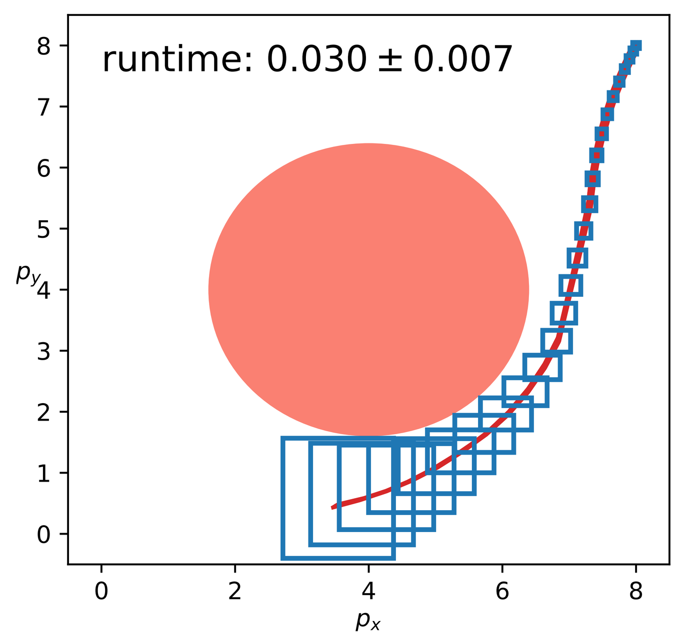

A single trajectory of the embedding system provides lower bound $\underline{x}$ and upper bound $\overline{x}$ on reachable set of original system at time $t$.




## Numerical Experiments

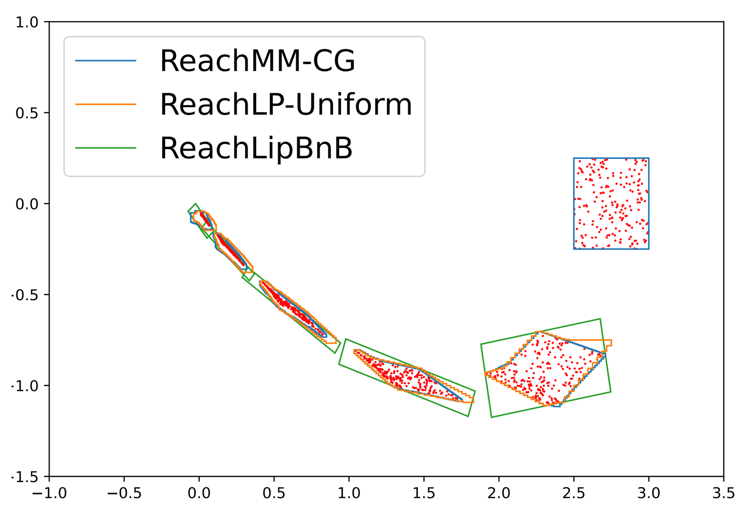
- Partitioning improves the accuracy of interval analysis.
- separation between i) partitions that query neural network verification algorithm, and ii) partitions that only do integration.






**Vehicle Model**:
Kinematic bicycle model, controlled by a $4 \times 100 \times 100 \times 2$ ReLU neural network, trained to stabilize to the origin while avoiding an obstacle.

**Double Integrator Model**:
Controlled by a $2 \times 10 \times 5 \times 1$ ReLU neural network, compare to [2,3].




| Method | Runtime (s) | Area |
|---|---|---|
| ReachMM-CG | $\mathbf{1.762 \pm 0.026}$ | $\mathbf{9.9 \cdot 10^{-3}}$ |
| ReachLP-Unif | $3.149 \pm 0.004$ | $1.0 \cdot 10^{-2}$ |
| ReachLP-GSG | $2.164 \pm 0.031$ | $8.8 \cdot 10^{-2}$ |
| ReachLipBnB | $3.681 \pm 0.100$ | $1.2 \cdot 10^{-2}$ |

## References


(1) H. Zhang et al., *Efficient neural network robustness certification with general activation function*, NeurIPS, 2018

(2) M. Everett et al., *Reachability analysis of neural feedback loops*, IEEE Access, 2021

(3) T. Entesari et al., *ReachLipBnB: A branch-and-bound method for reachability analysis of neural autonomous systems using Lipschitz bounds*, ICRA, 2023