Verification of Neural-Network Control Systems

Christian Schilling

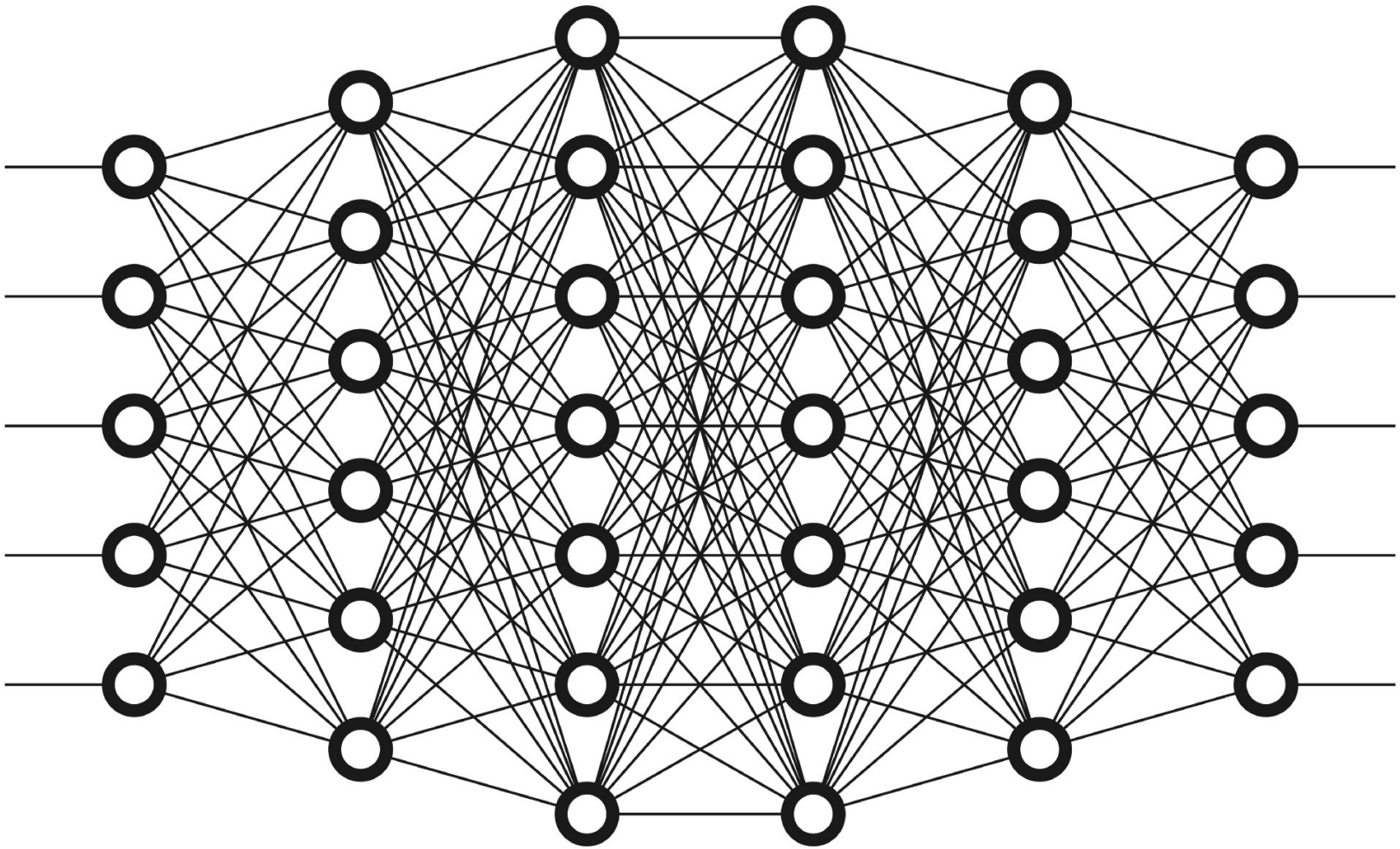
AALBORG UNIVERSITET

DIREC Seminar 2022 in Aarhus, September 26

based on joint work presented at AAAI 2022 with
Marcelo Forets and Sebastián Guadalupe
from Universidad de la República, Uruguay

Verification of neural networks

- Is the following function correct?



Verification of programs

- Is the following function correct?

```
def foo(A):
    i = 1
    while i < len(A):
        x = A[i]
        j = i - 1
        while j >= 0 and A[j] > x:
            A[j+1] = A[j]
            j = j - 1
        A[j+1] = x
        i = i + 1
```

No verification without specification

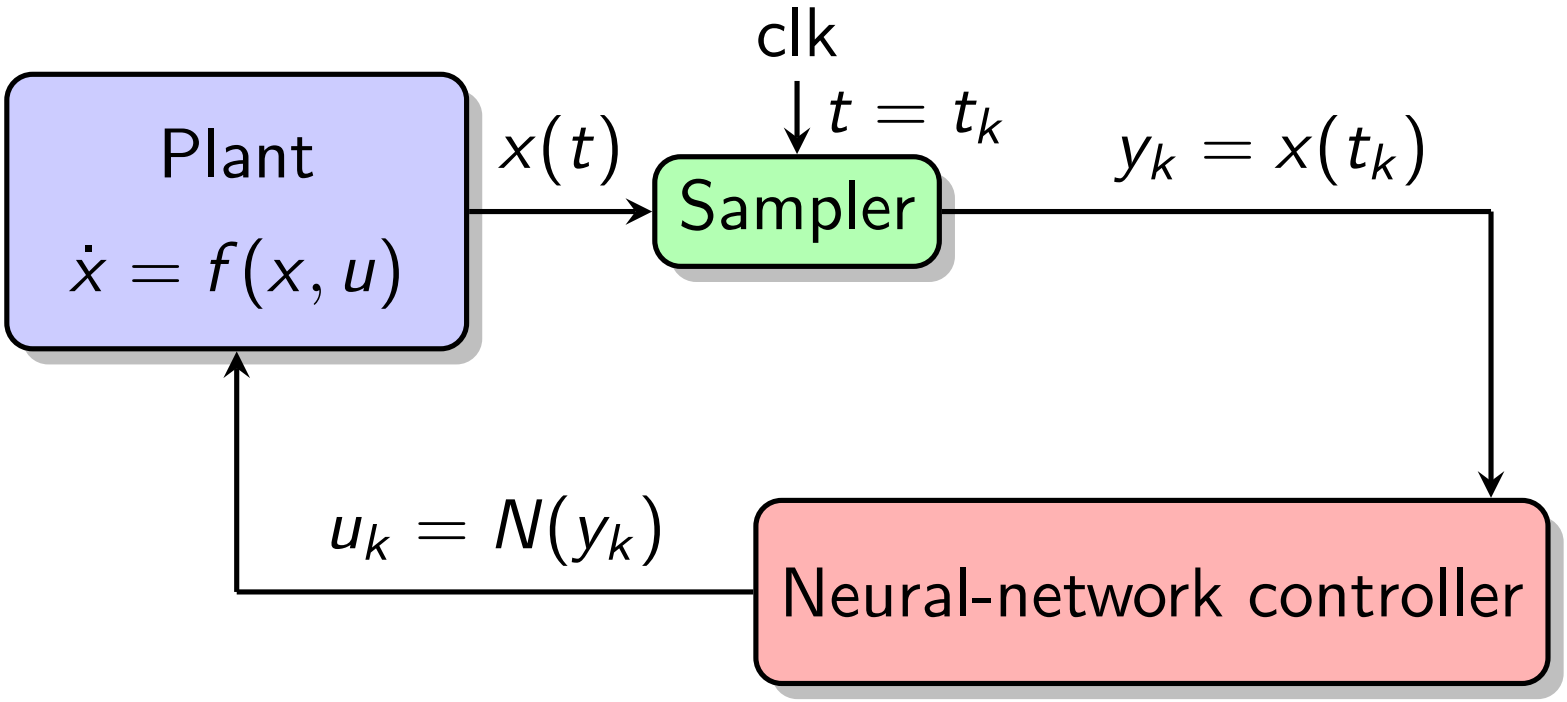
- **Verification** requires a **specification**
- We may not understand how a **neural network** works

  But this does not mean we cannot **verify** it

- The problem is **not** the **neural network**

  (Disclaimer: ignoring scalability)

Neural-network control system



Unicycle model

Plant:

$\dot{x} = v\cos(\theta)$

$\dot{y} = v\sin(\theta)$

$\dot{\theta} = u_2$

$\dot{v} = u_1 + w$

Specification:

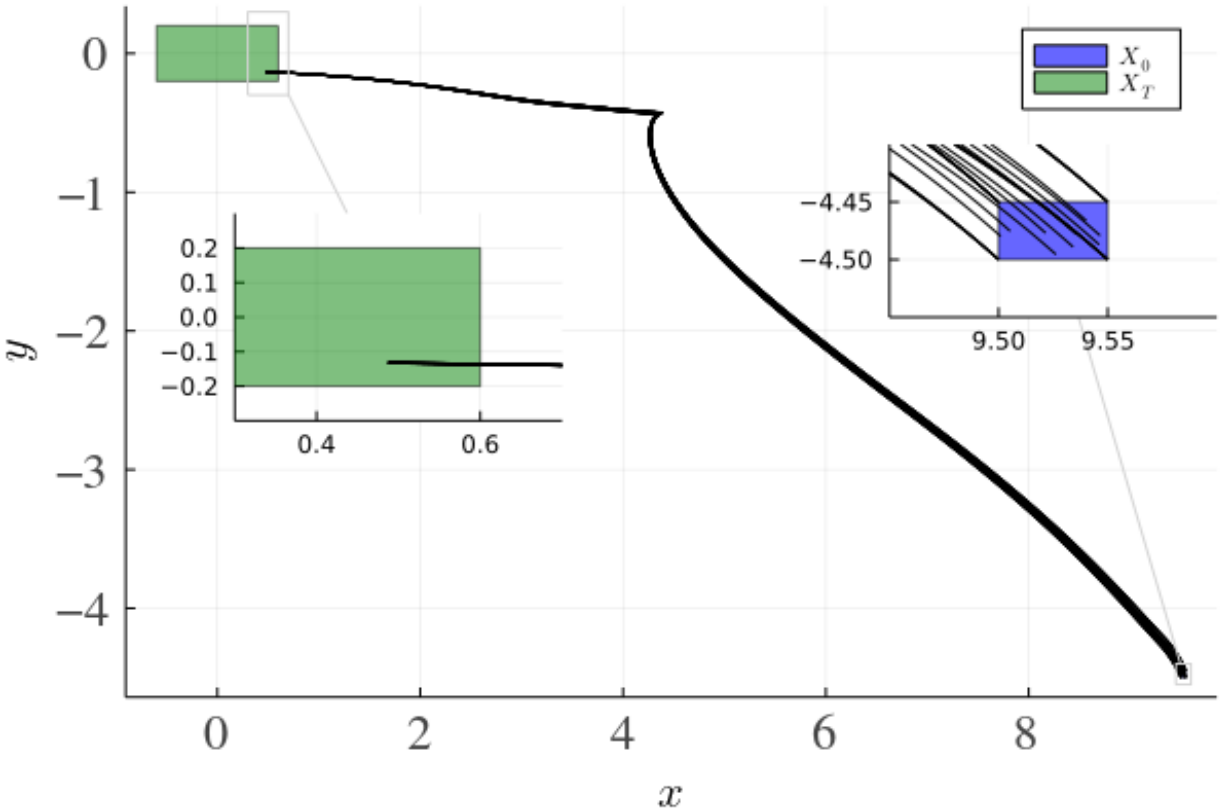
$x(0) \in \mathcal{X}_0$

$x(10) \overset{!}{\in} \mathcal{X}_T$

Controller:

500 hidden units

0.2 s period



42 simulations

Unicycle model

Plant:

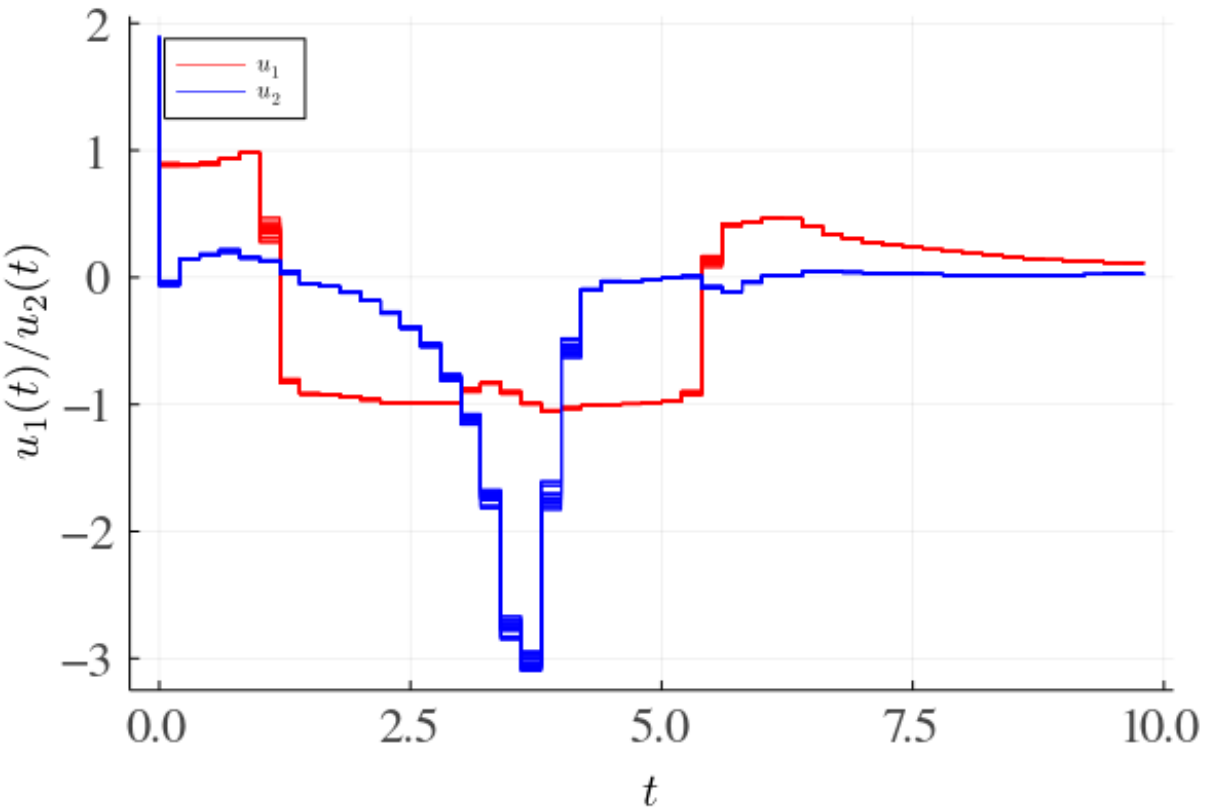
$\dot{x} = v\cos(\theta)$
$\dot{y} = v\sin(\theta)$
$\dot{\theta} = u_2$
$\dot{v} = u_1 + w$

Specification:

$x(0) \in \mathcal{X}_0$
$x(10) \overset{!}{\in} \mathcal{X}_T$

Controller:

500 hidden units
0.2 s period



control signals (42 simulations)

Unicycle model

Plant:

$\dot{x} = v\cos(\theta)$

$\dot{y} = v\sin(\theta)$

$\dot{\theta} = u_2$

$\dot{v} = u_1 + w$

Specification:

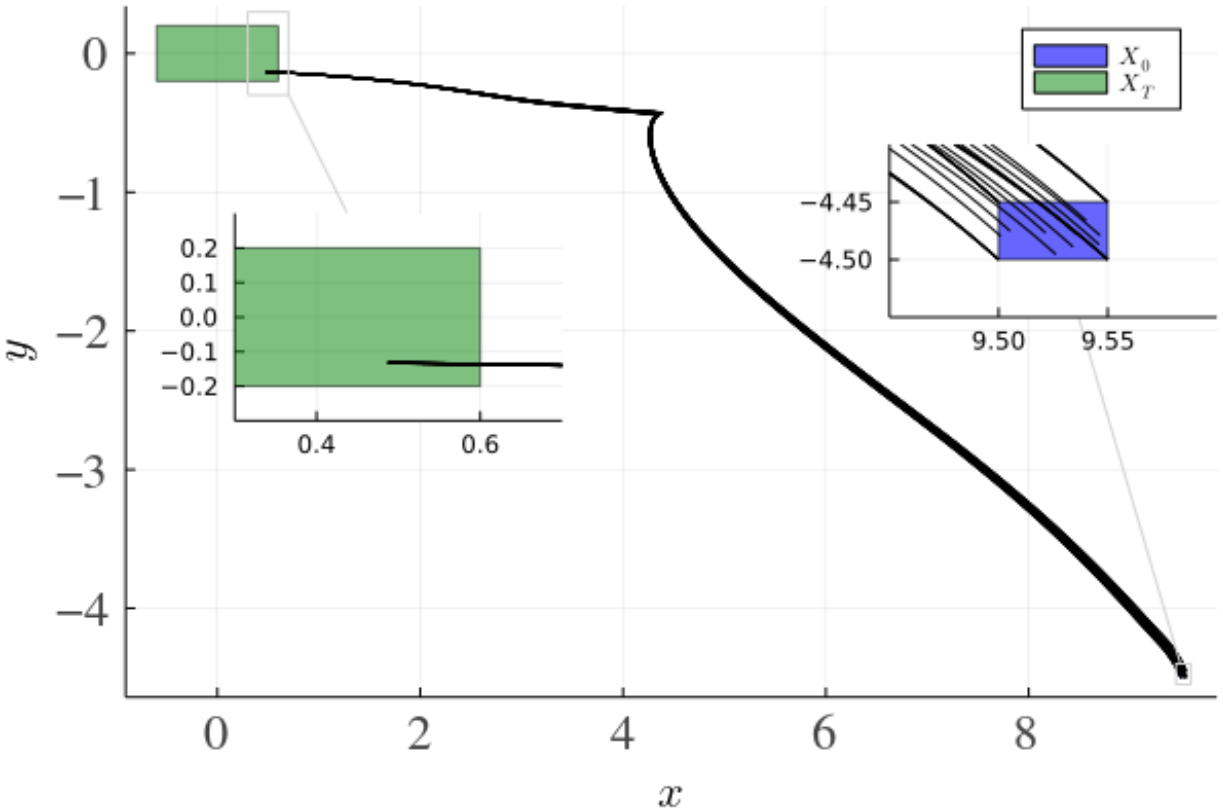
$x(0) \in \mathcal{X}_0$

$x(10) \overset{!}{\in} \mathcal{X}_T$

Controller:

500 hidden units

0.2 s period



42 simulations

Unicycle model

Plant:

$\dot{x} = v\cos(\theta)$

$\dot{y} = v\sin(\theta)$

$\dot{\theta} = u_2$

$\dot{v} = u_1 + w$

Specification:

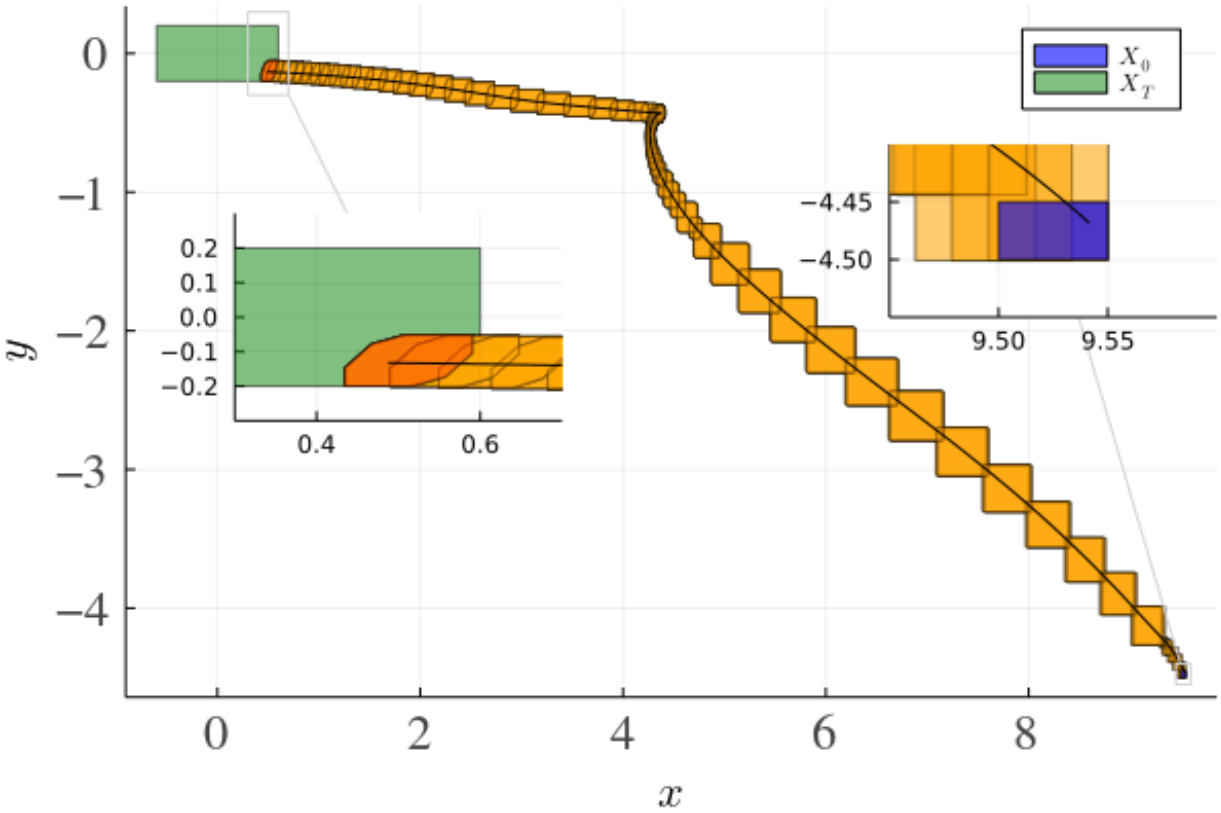
$x(0) \in \mathcal{X}_0$

$x(10) \overset{!}{\in} \mathcal{X}_T$

Controller:

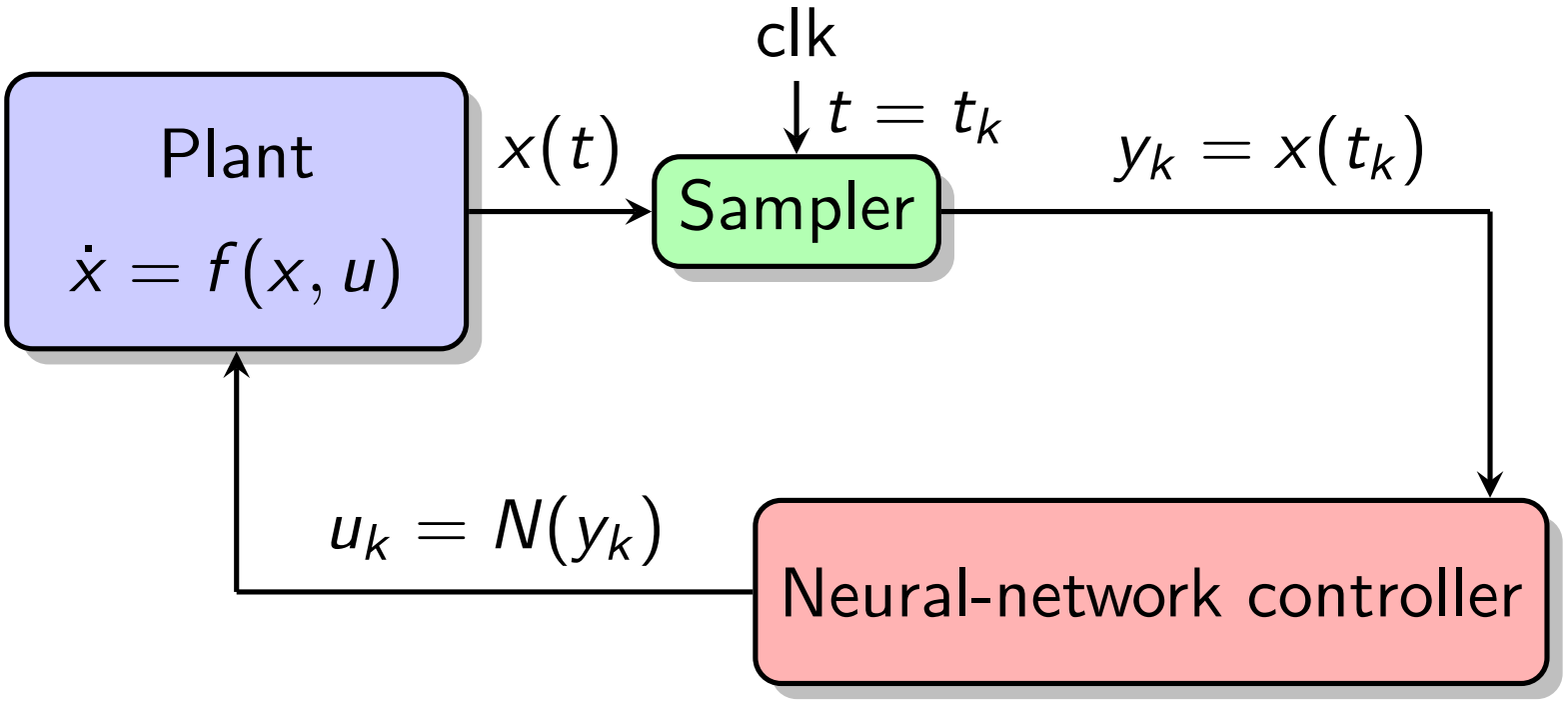
500 hidden units

0.2 s period

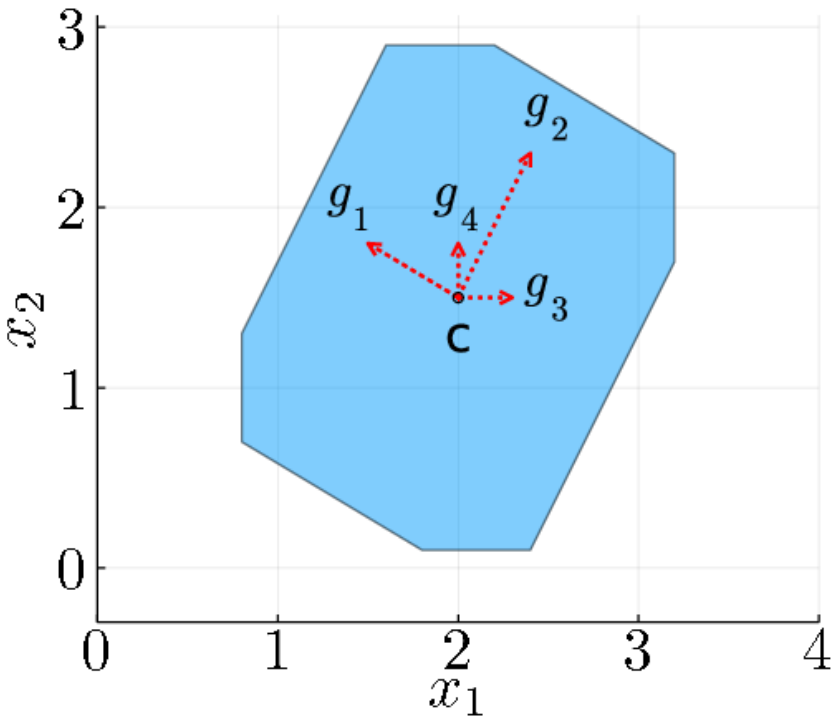


set-based simulations/reachability analysis

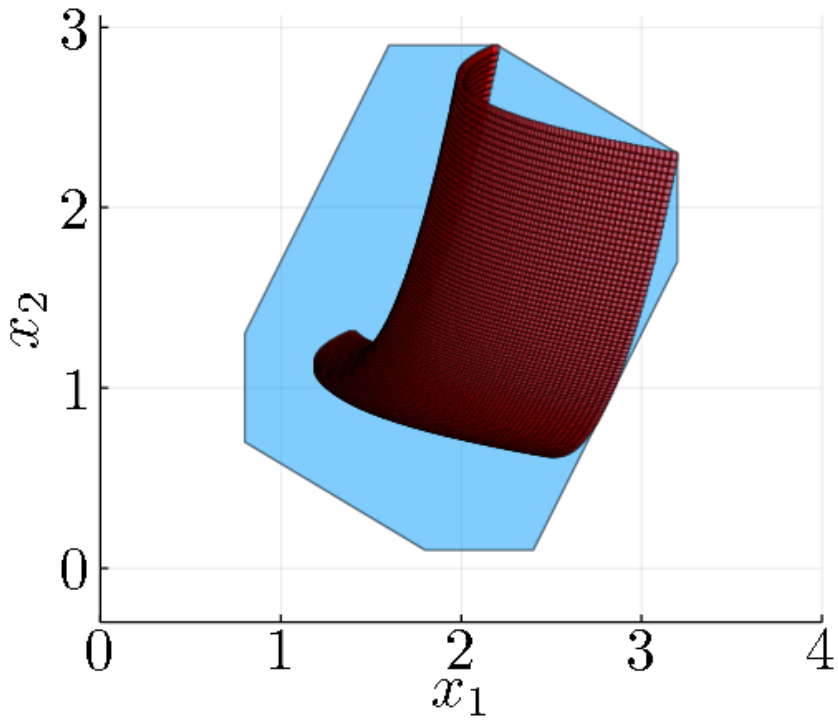
Neural-network control system



Zonotopes and Taylor models

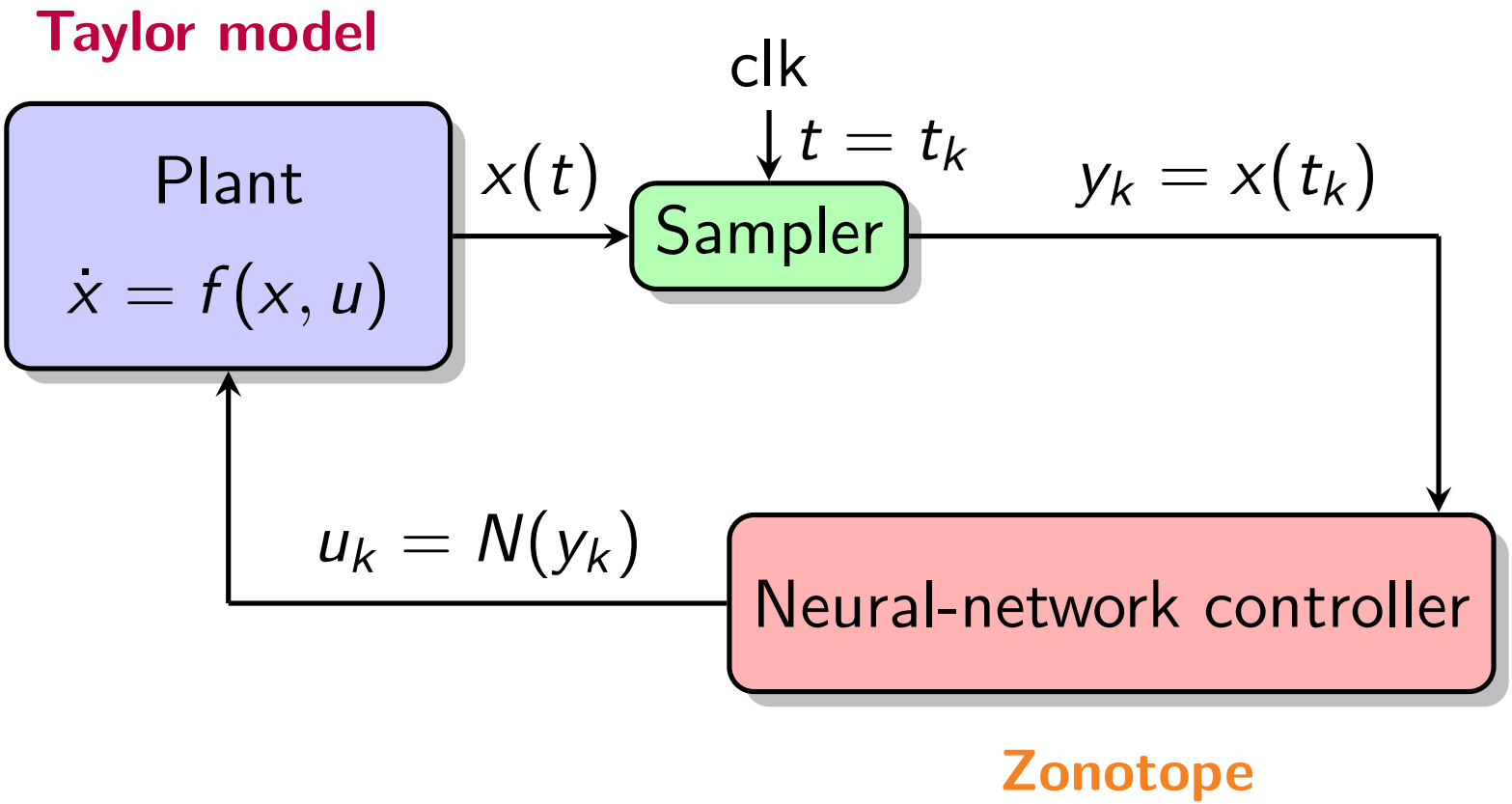


structured zonotope

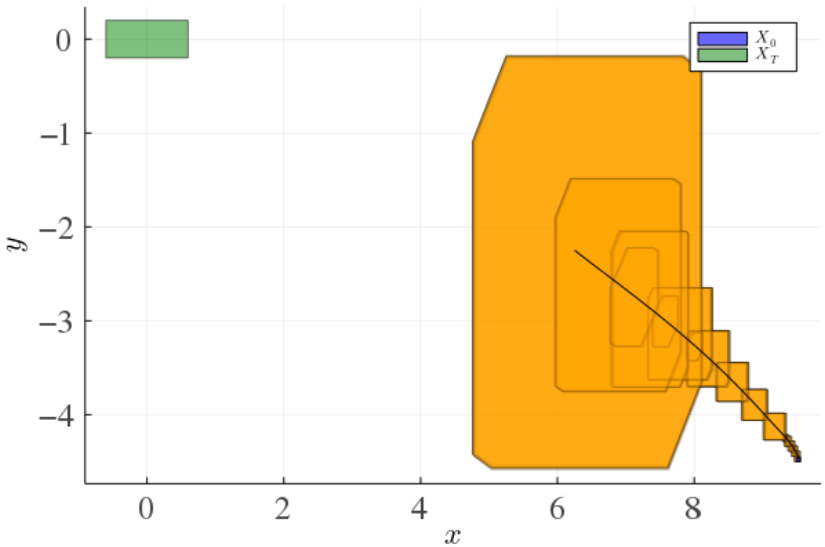


Taylor model (red) enclosed by **structured zonotope** (blue)

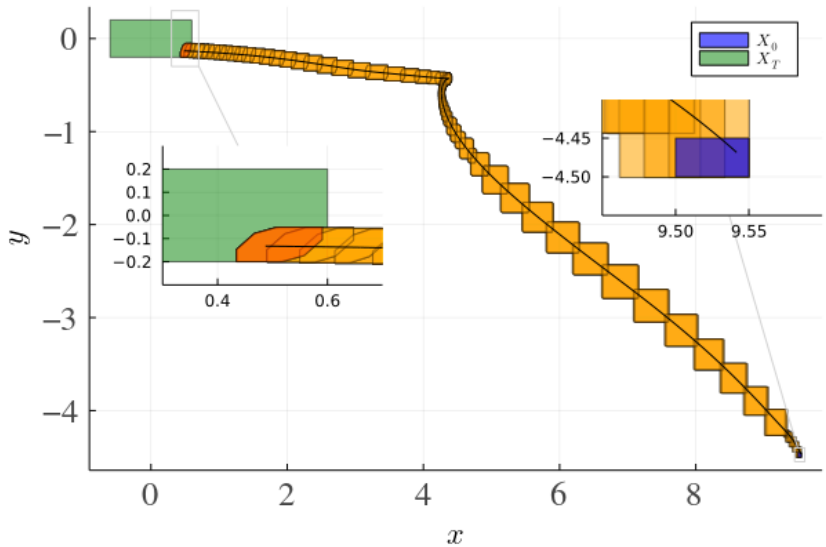
Trivial problem?



Trivial problem?

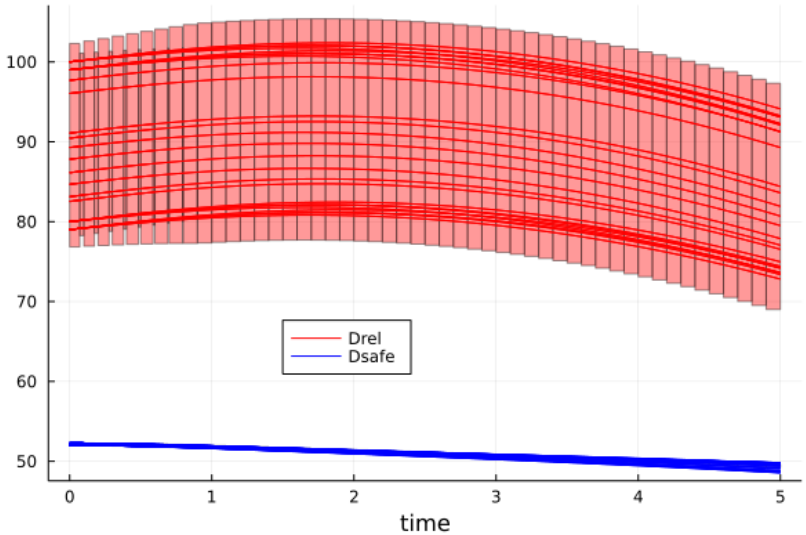


naive combination

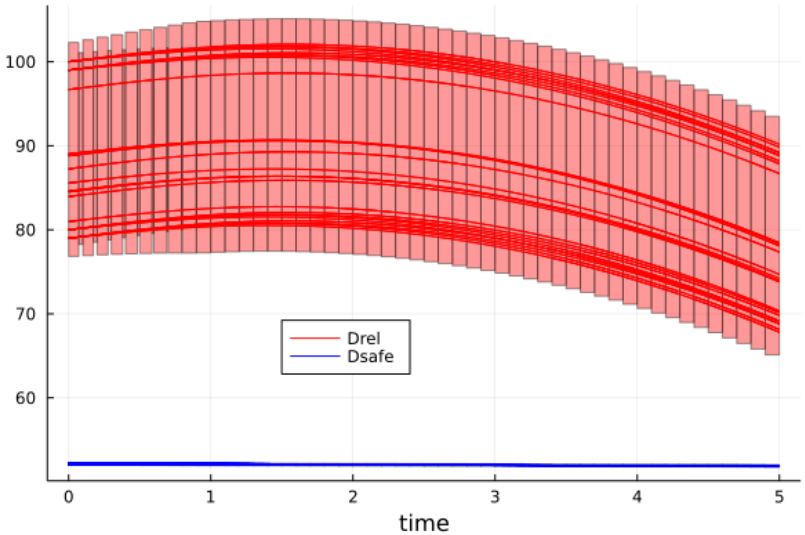


careful combination

Adaptive cruise control

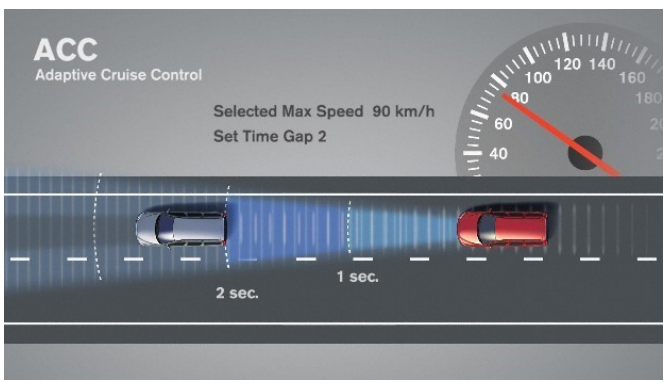


ReLU activations

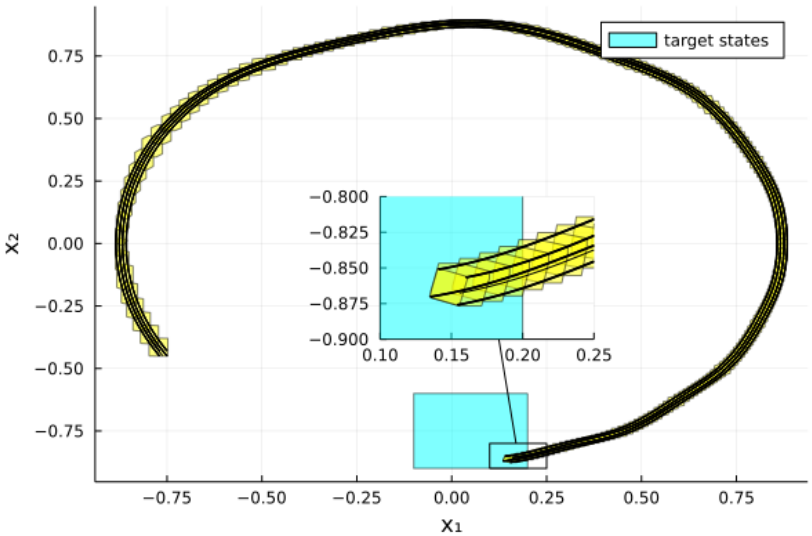


tanh activations

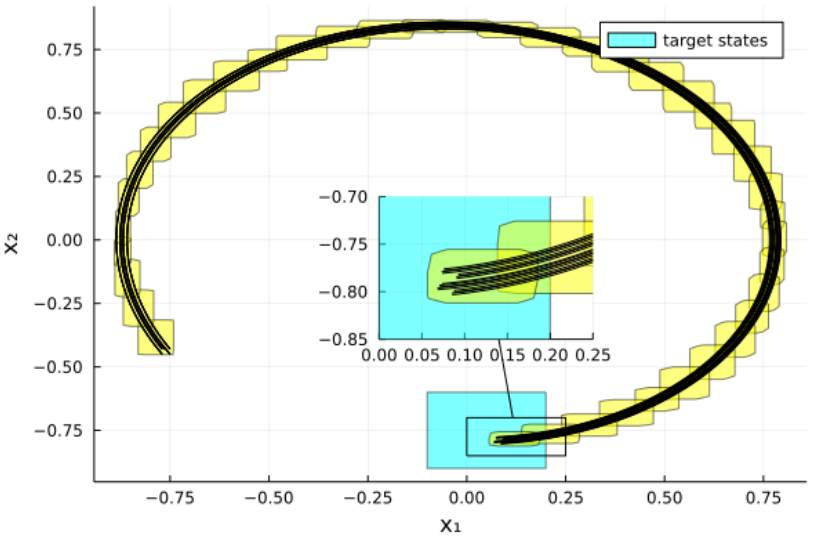
- Braking maneuver of lead car
  Show that $D_{\text{rel}} \geq D_{\text{safe}}$
- 6 state dimensions,
  1 control dimension



Translational oscillations by a rotational actuator (TORA)

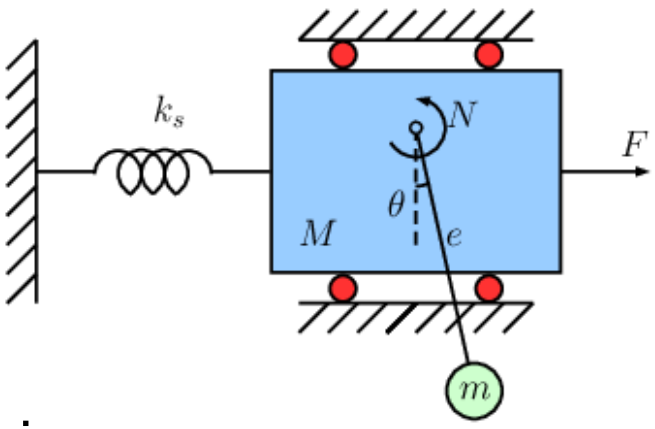


Sigmoid activations

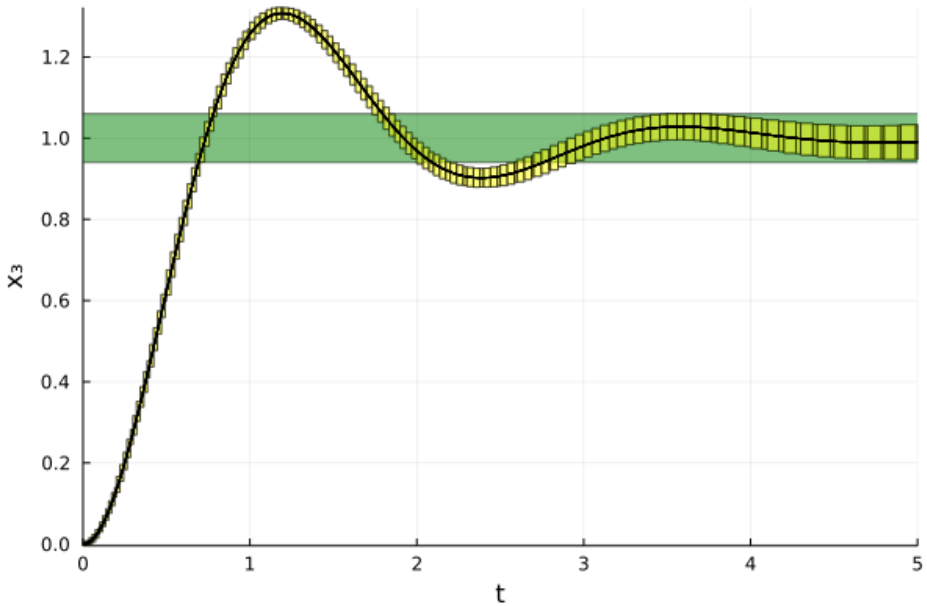


ReLU and tanh activations

- Cart attached to wall via spring with rotating weight attached
- Reach a target set within 5 s
- 4 state dimensions, 1 control dimension



Quadrotor model



Sigmoid activations

- Stabilize within 5 s
- 12 state dimensions,
  3 control dimension

Conclusion

- Reachability approach for **neural-network control systems**
- Challenge: repeated **set conversion** incurs precision loss
- Mitigated by **Taylor models** and **structured zonotopes**
- Solved all competition problems in 2021 for first time
- Publicly available implementation in **JuliaReach**[1]

[1] https://github.com/JuliaReach/.