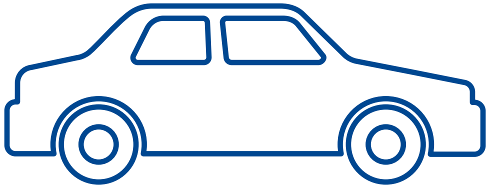
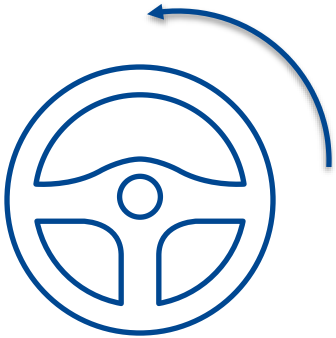
# Solving Difficult Reachability Problems in JuMP.jl

Chelsea Sidrane, PhD
Postdoctoral Research Fellow
KTH Royal Institute of Technology

# Background: What is a control system?

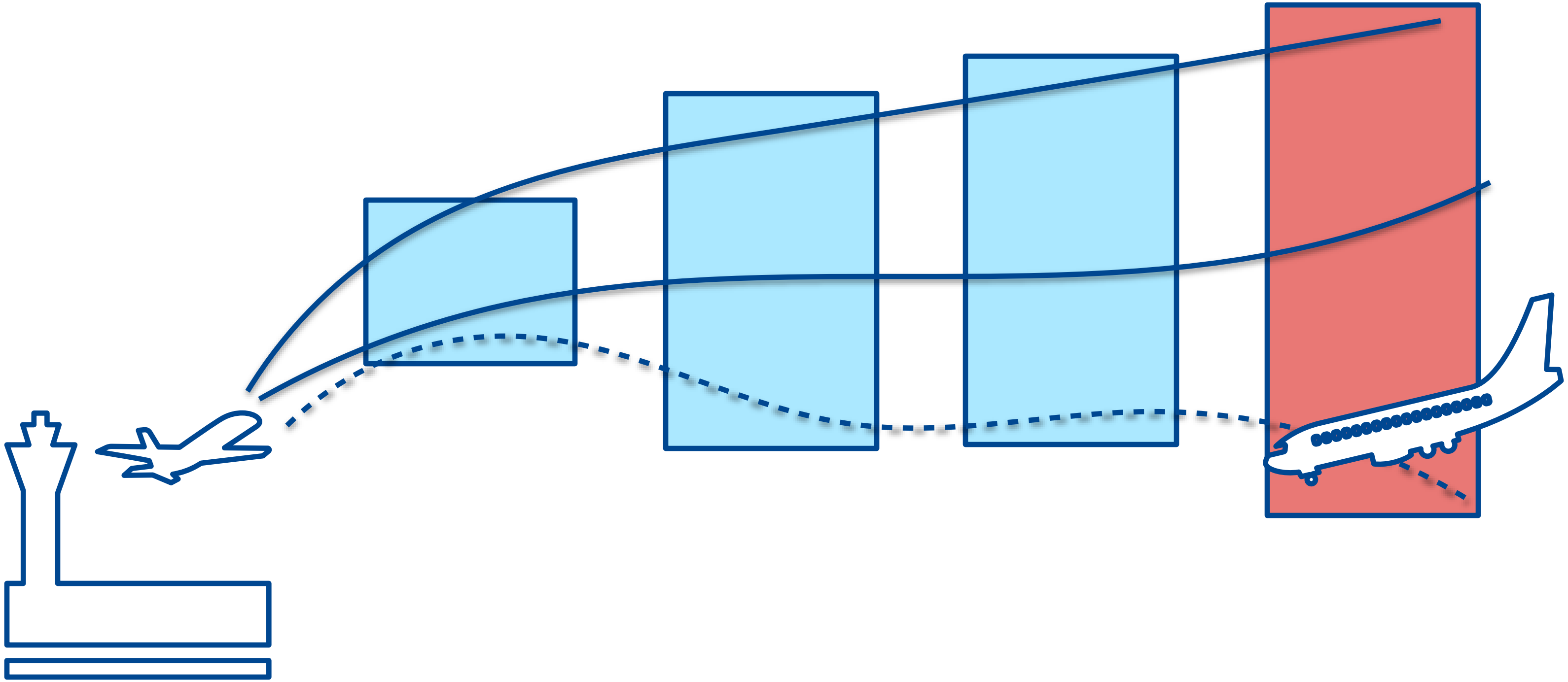


Dynamical system



that takes control inputs
(In this case from a computer, not a human)

# Reachability Analysis is Important for Control Systems



*Reachability analysis for nonlinear dynamical systems with neural network control policies*

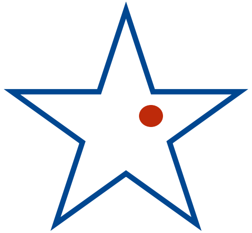
# OVERTVerify.jl

https://github.com/sisl/OVERTVerify.jl

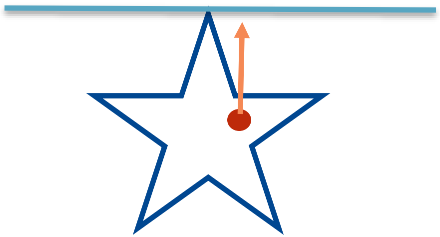
Sidrane, Chelsea, Amir Maleki, Ahmed Irfan, and Mykel J. Kochenderfer. "OVERT: An algorithm for safety verification of neural network control policies for nonlinear systems." *Journal of Machine Learning Research* 23, no. 117 (2022): 1-45.

# Optimization To Solve Reachability Problems

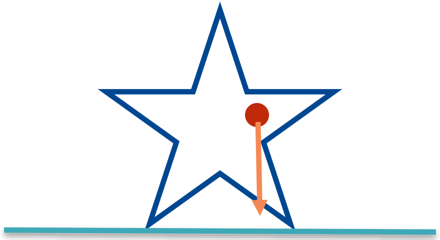
Method 1: Explicit Computation



Encode system
constraints and find
initial feasible point



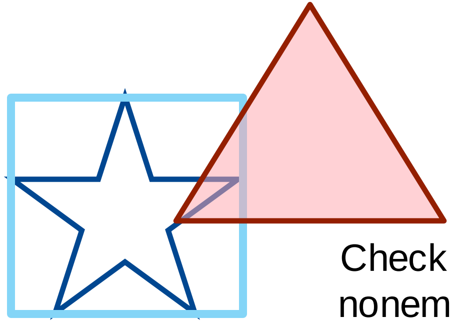
Maximize



Minimize

# Optimization To Solve Reachability Problems

Method 1: Explicit Computation, e.g., and an avoid set



Check for nonempty intersection

Compute minimal enclosing hull of reachable set with optimizer

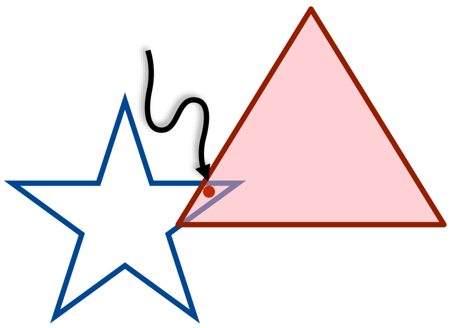
*Ability to change objective in JuMP.jl while keeping constraints is helpful*

LazySets.jl, of course

*Repeat for every timestep*

# Optimization To Solve Reachability Problems

Method 2: Feasibility Check

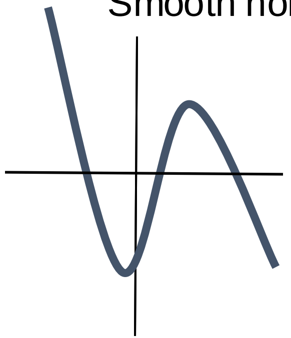


Encode system constraints and avoid set
into optimizer; find feasible point

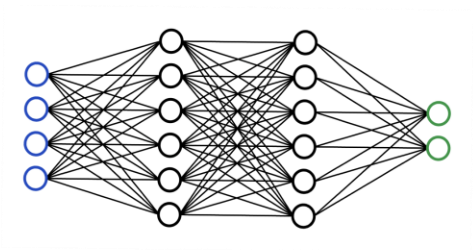
*Repeat for every timestep*

# Nonlinear Functions Make Reachability Difficult

Smooth nonlinearity



Neural network



Adding them naively would make the optimization problem non-convex

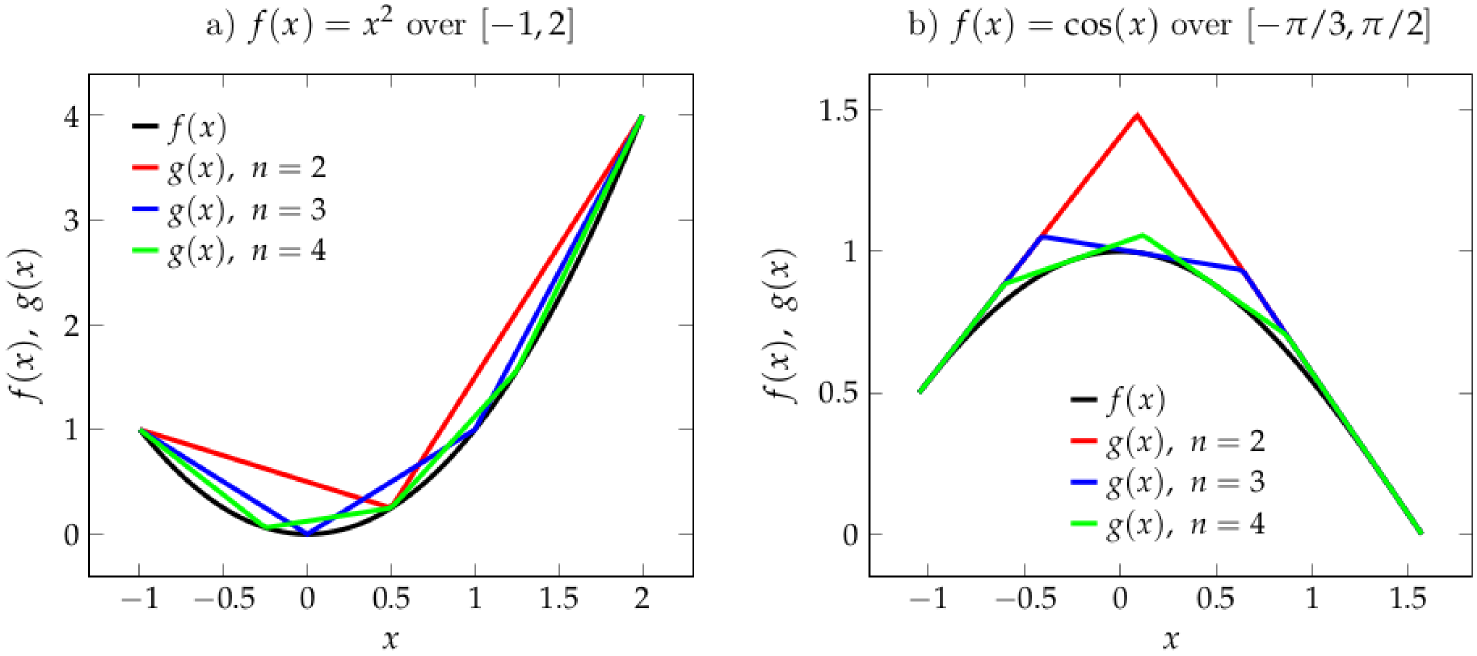
And speed + optimality are essential

# OVERT.jl

https://github.com/sisl/OVERT.jl

Sidrane, Chelsea, Amir Maleki, Ahmed Irfan, and Mykel J. Kochenderfer. "OVERT: An algorithm for safety verification of neural network control policies for nonlinear systems." *Journal of Machine Learning Research* 23, no. 117 (2022): 1-45.

# Area Minimal Piecewise Linear Bounds

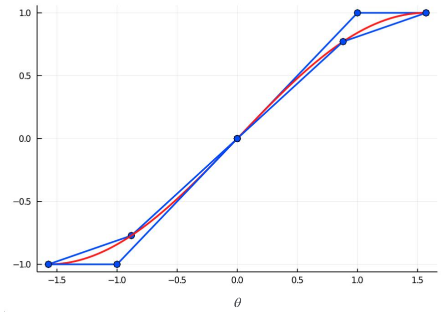


# OVERTVerify.jl

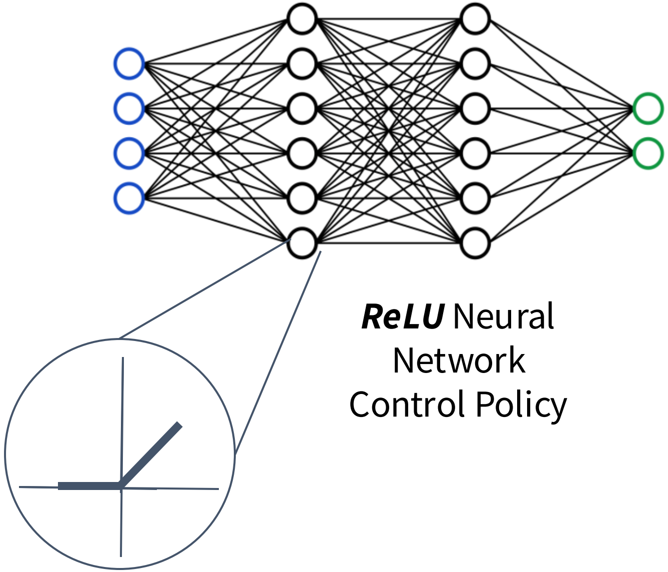
https://github.com/sisl/OVERTVerify.jl

Sidrane, Chelsea, Amir Maleki, Ahmed Irfan, and Mykel J. Kochenderfer. "OVERT: An algorithm for safety verification of neural network control policies for nonlinear systems." *Journal of Machine Learning Research* 23, no. 117 (2022): 1-45.

# Handling Two Kinds of Nonlinearities




Piecewise linear inclusions of smooth nonlinear functions in dynamics



***ReLU*** Neural Network Control Policy

**ReLU Function**

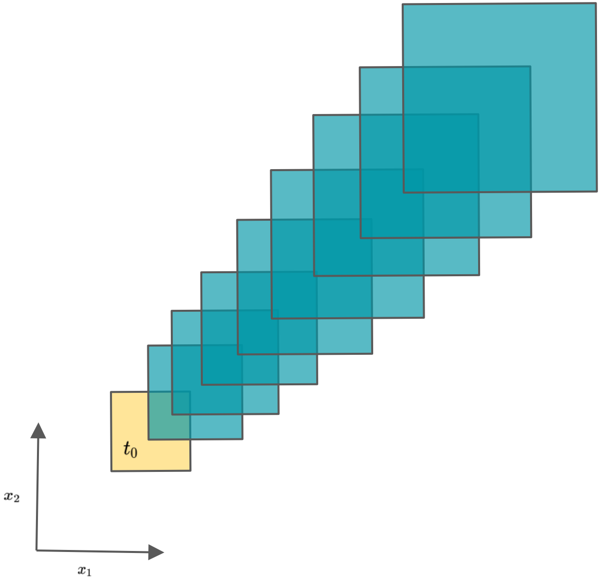
Everything is piecewise-linear!

We can solve an MILP!

# Keeping Reachability Tight + Tractable

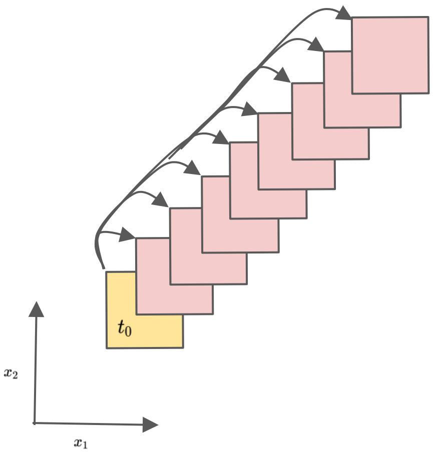
Pure 1-step

*Too Conservative*



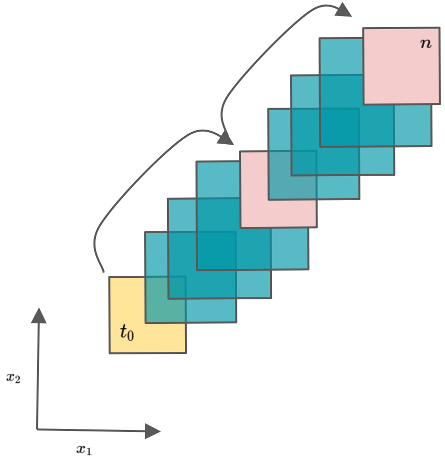

Pure n-step

*Intractable*




Hand-Tuned Schedule

*Impractical*

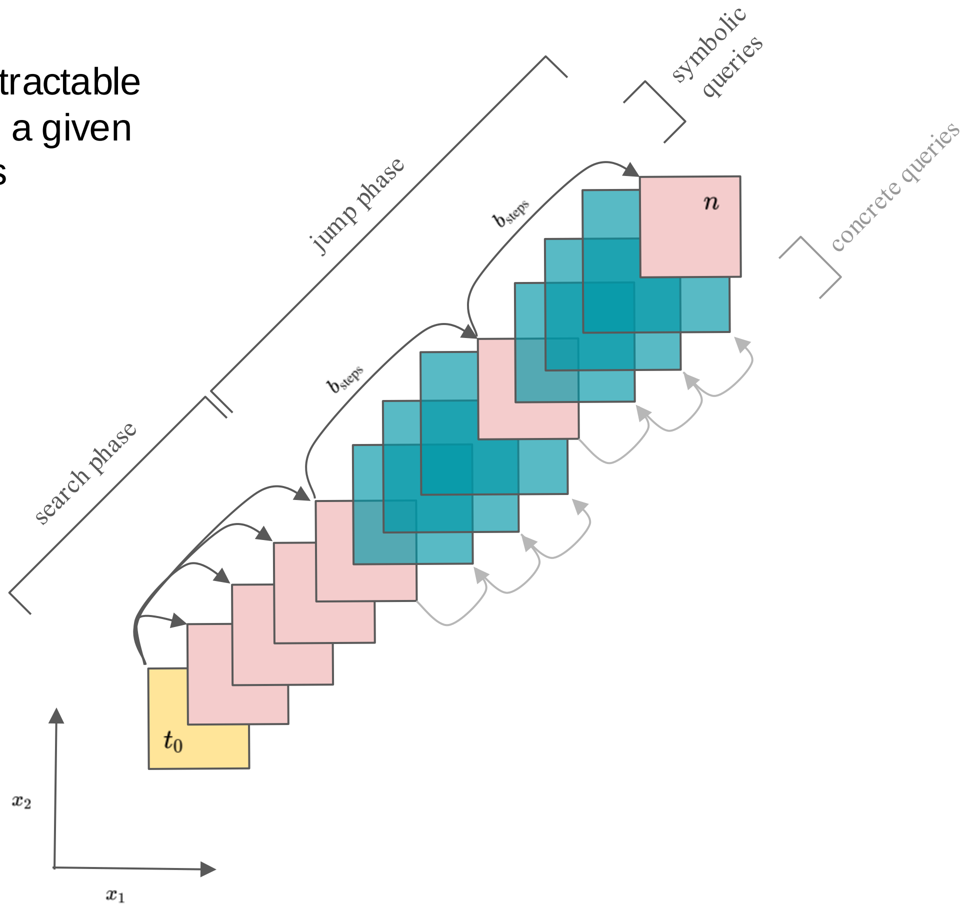


# **TTT:** A **Temporal** Refinement Heuristic for **Tenuously Tractable** Discrete Time Reachability Problems

My newest project

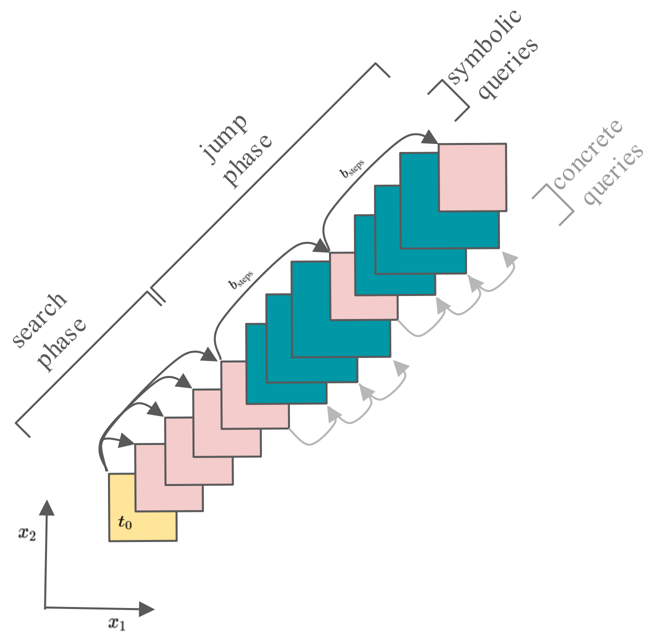
# Automatic Hybrid-Symbolic Reachability

Search for the longest tractable n-step query subject to a given time budget in seconds



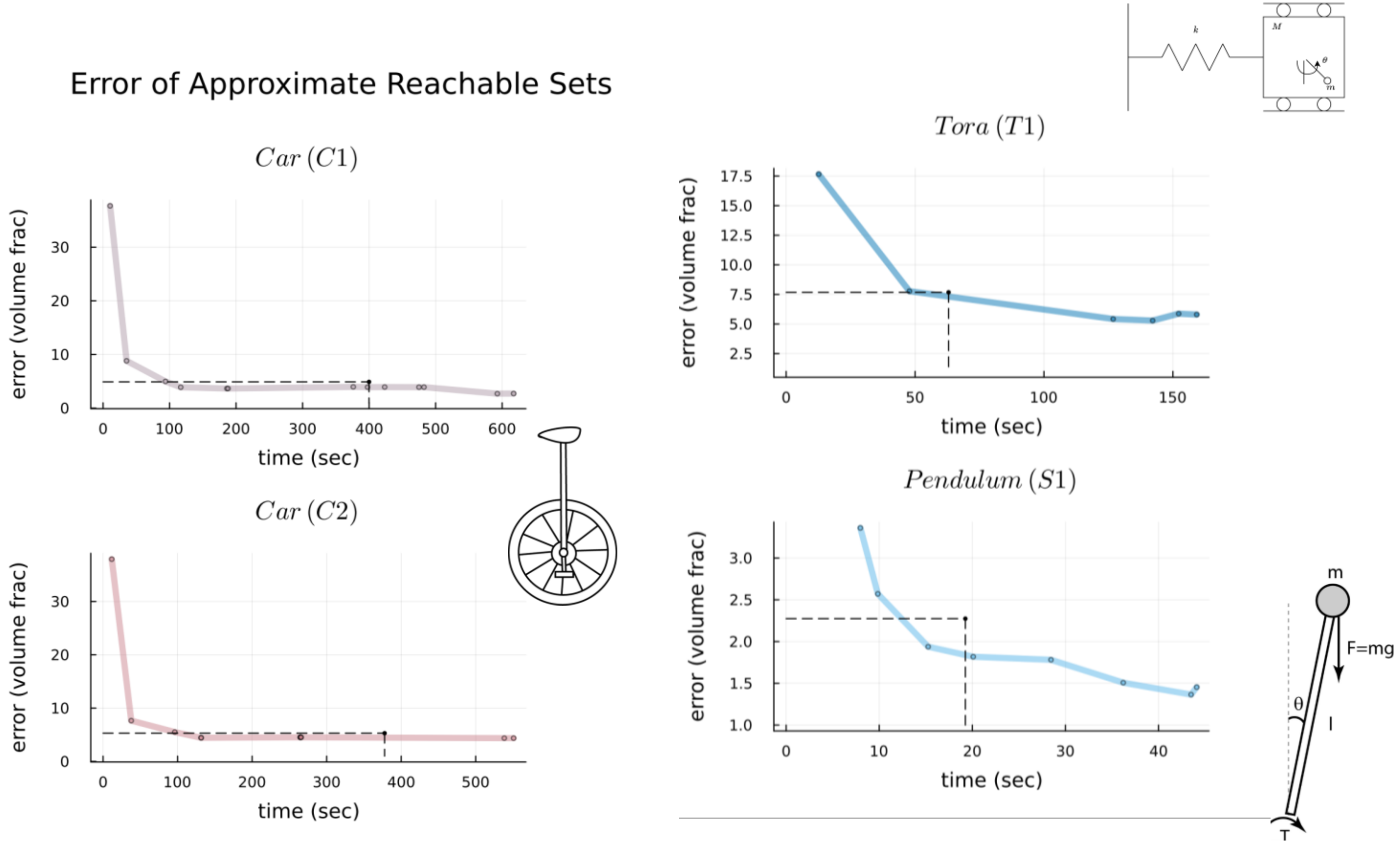
# Automatic Hybrid-Symbolic Reachability

- Use a linear estimate of query time as function of number of steps
- At each solve, enforce a time limit through early stopping
- When using early stopping, must use objective bound to ensure soundness
- If !isfinite(objective_bound(model)) or relative_gap(model) > 0.50, extend the time limit and call optimize!(model) again

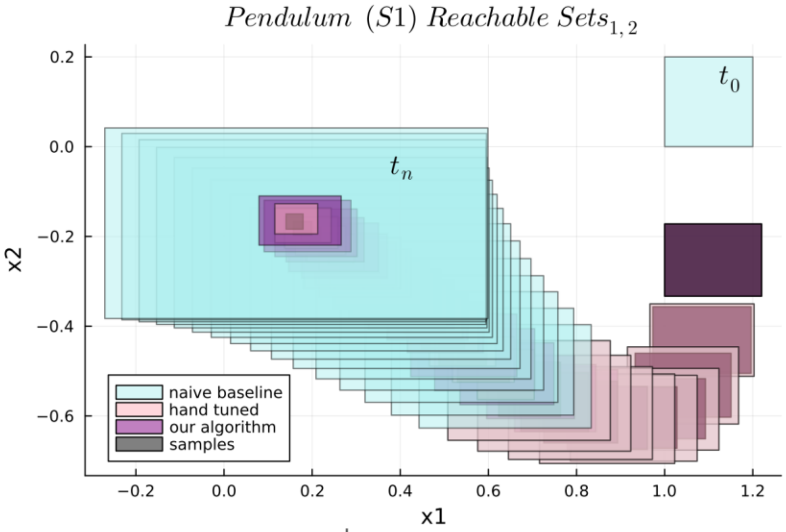


# Results

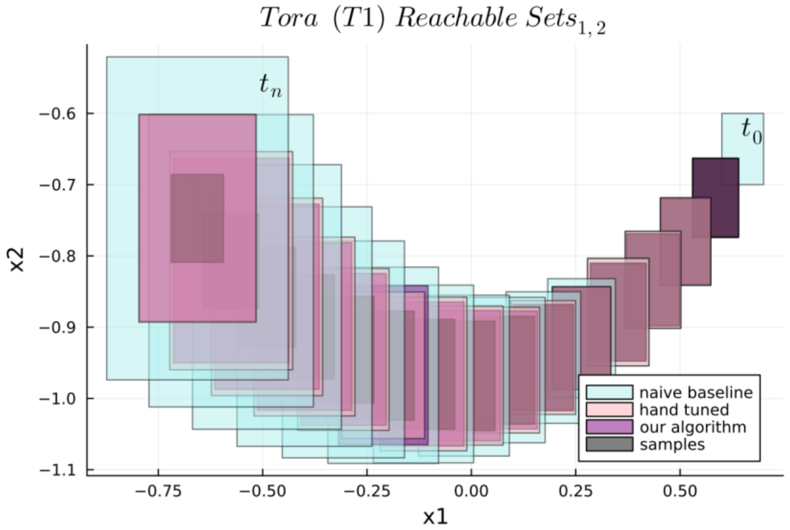
- Can produce reachable sets of varying fidelity given varying time
  - *No more hand-tuning needed!*
- For similar amounts of error as a hand-tuned approach, we are 20-70% faster



# Some Reachable Sets




| | Naive | Hand-Tuned | Ours |
|---|---|---|---|
| Error (volume) | 11.9 | 2.27 | 1.94 |
| Error (radius) | 3.79 | 1.61 | 1.59 |
| Time | 7.84s | 19.3s | **15.3s** |




| | Naive | Hand-Tuned | Ours |
|---|---|---|---|
| Error (volume) | 31.7 | 7.67 | 7.78 |
| Error (radius) | 1.81 | 1.41 | 1.38 |
| Time | 6.62s | 63.0s | **47.8s** |

# Julia Package Summary

**<u>My Packages Discussed in this Talk</u>**

OVERT.jl https://github.com/sisl/OVERT.jl

*Overapproximations of nonlinear functions*

OVERTVerify.jl https://github.com/sisl/OVERTVerify.jl

*Reachability analysis for nonlinear dynamical systems with neural network control policies*

*AutomaticRefinement.jl (potentially coming soon to a GitHub near you)*

*The automatic temporal refinement work discussed here*

**<u>Other Useful Related Packages</u>**

Expr2MIP.jl https://github.com/chelseas/Expr2MIP.jl

*Encode arbitrary expressions of type* Expr *into JuMP MILP models (depends on OVERT.jl for smooth nonlinear functions) (My package)*

NeuralVerification.jl https://github.com/sisl/NeuralVerification.jl

*Pedagogical implementations of various neural network verification algorithms (Written by collaborators)*

( == *depends on* JuMP.jl)

# Info & Links

Chelsea Sidrane, PhD

chelse@kth.se

website:

# Thanks for listening!

**Reach out if you want to discuss :)**

# Reserve Slides

# OVERT.jl

# OVERT's Overapproximation

$$\dot{\theta}_{t+1} = \dot{\theta}_t + c_1 \sin(\theta_t) + c_2\ u_t$$

1) Re-write nonlinear multi-dimensional functions as one-dimensional or affine functions

$$v_1 = \sin(\theta_t)$$
$$\dot{\theta}_{t+1} = \dot{\theta}_t + c_1\ v_1 + c_2\ u_t$$

Made much easier by Julia's Expr type and easy symbolic manipulation

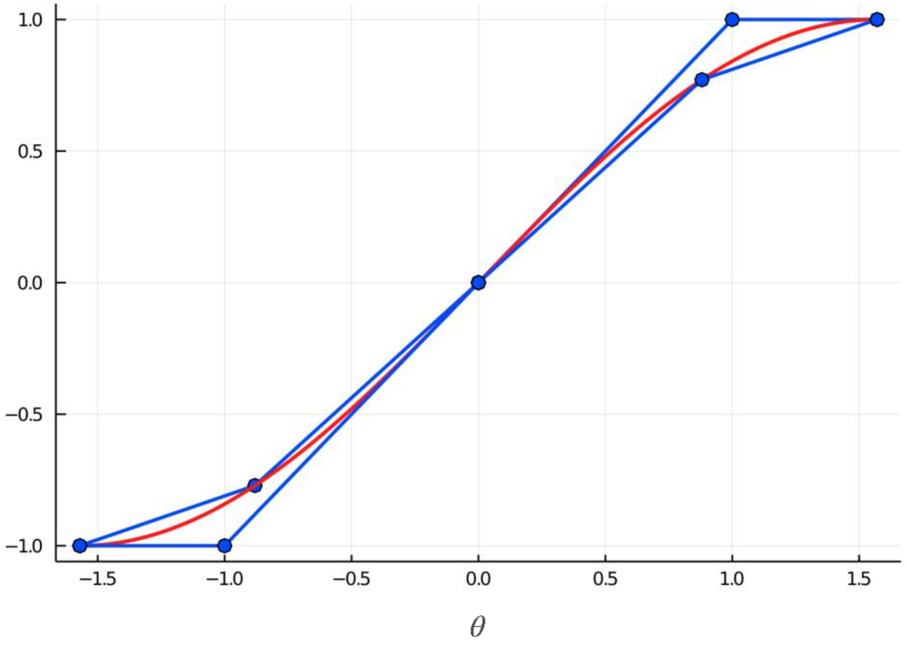
1) Overapproximate each nonlinear one-dimensional function

$$\sin_{LB}(\theta_t) \leq v_1 \leq \sin_{UB}(\theta_t)$$

# Area Minimal Piecewise Linear Bounds

$\sin$

$\sin_{LB}$, $\sin_{UB}$



# Implementation of OVERT.jl Minimum Area Bounds

solve system of equations = 0 to optimize bound points x_i using NLsolve.jl □

Optimality not needed but always a perk

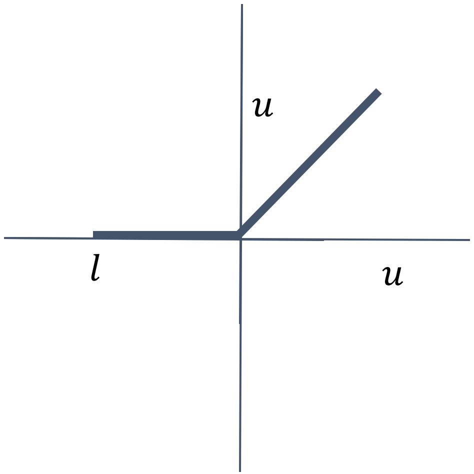
$$
\begin{aligned}
x_0 &= a \\
x_1 &= h\left(a, \frac{x_1 + x_2}{2}\right) \\
x_i &= h\left(\frac{x_{i-1} + x_i}{2}, \frac{x_i + x_{i+1}}{2}\right), \quad i \in \{2, \cdots, n-2\} \\
x_{n-1} &= h\left(\frac{x_{n-1} + x_n}{2}, b\right) \\
x_n &= b
\end{aligned}
$$

# OVERTVerify.jl

# Encoding the Problem in an MILP

- All piecewise linear functions can be written in terms of max and min
- Max can be encoded as shown using a unique upper and lower bound for each instance instead of `big-M`

Constraints




$$z = max(\hat{z}, 0)$$

$$\begin{aligned} z &\geq \hat{z} \\ z &\geq 0 \\ z &\leq \hat{z} - l(1-\delta) \\ z &\leq u\delta \\ \text{for } \hat{z} &\in [\mathrm{l}, \mathrm{u}] \end{aligned}$$


Tjeng, V., Xiao, K. and Tedrake, R., 2017. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*.

# How to Get Bounds?

- Dynamics with Smooth Nonlinearities
  - From OVERT.jl which uses interval arithmetic, and from reach sets
- Neural Network
  - Using MaxSens
    - [W. Xiang, H. Tran, and T. T. Johnson, “Output reachable set estimation and verification for multilayer neural networks,” pp. 5777–5783, IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 11, Nov. 2018.]

# Concrete Vs. Symbolic Reachability