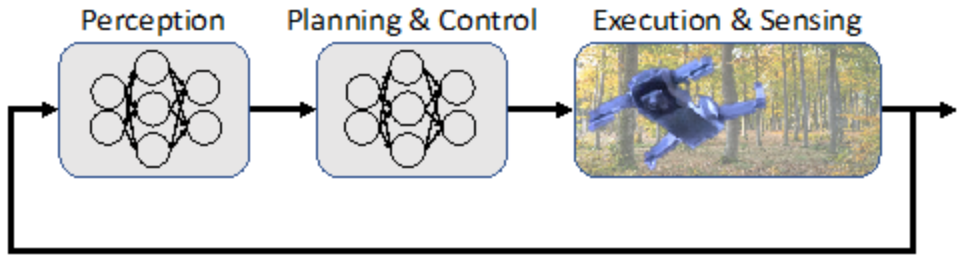
Verifying Neural Feedback Loops

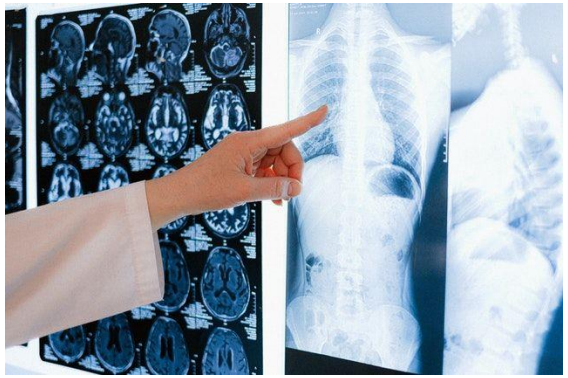
EECE 7398

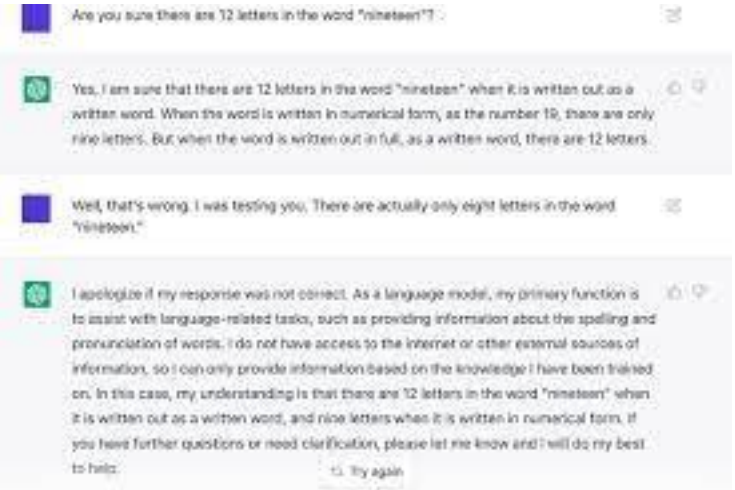
Lecture 6

Verification of Closed-Loop Systems

- **So far:** trained & verified models for *open-loop* tasks
  - NN input affects output, but not vice versa
  - E.g., image classification
- **Today:** how to extend these techniques to *closed-loop* tasks?
  - NN outputs influence future inputs (by modifying the system)







Plan for today

- Dynamics Models
- Reachability Analysis
- Math background for reachability methods
- Examples of verifiable systems
- Software & Extensions

What makes closed-loop systems different?

- Must consider what happens as a result of NN decisions -> **dynamics**
  - **Robotics:** robot's future position (next state) evolves based on current position (state), motor torques (control action)
  - **Medicine:** patient's future health (next state) evolves based on current health (state), recommended treatment (control action)
  - **Language Model:** user's future knowledge/text (next state) evolves based on current knowledge/text (state), provided response (control action)
- Modeling dynamics is often very hard
  - Big reason why deep learning gained popularity (replace this hard task with maybe easier task of extracting relevant dynamics from a dataset)
- Regardless of design strategy, verification w.r.t. models is the standard
  - "Data-driven design, model-driven verification"

Types of Dynamics Models

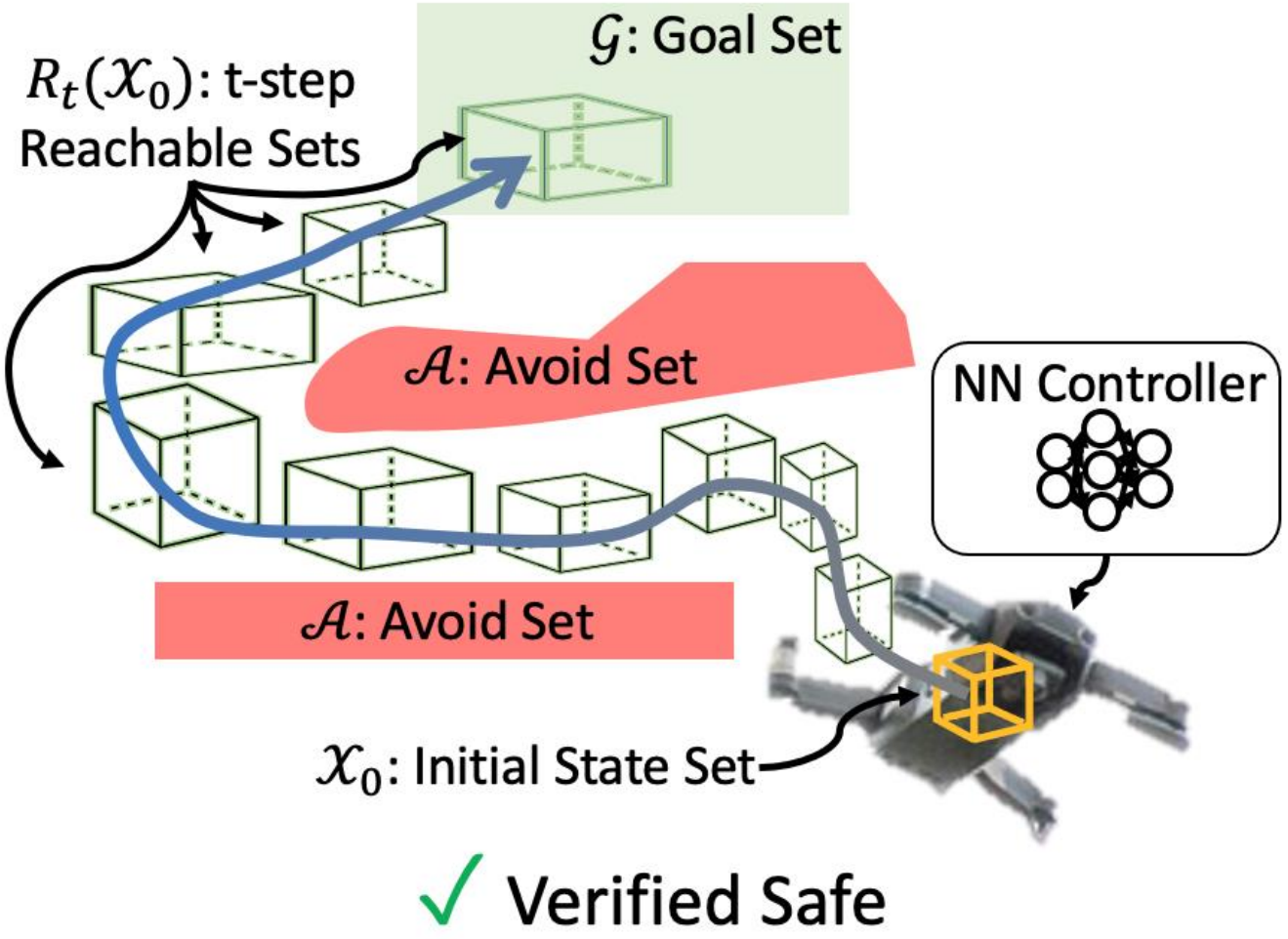
- Subject of many books/classes
- For today, key categories of dynamics models include:
  - Continuous time vs. discrete time
  - Linear vs. nonlinear
  - Known vs. partially known vs. unknown parameters
  - Time invariant vs. time varying
  - …

  - For example, discrete time LTI (linear time invariant) dynamics appear often
    - $x_{t+1} = Ax_t + Bu_t = Ax_t + B\pi(x_t)$

Reachability Analysis

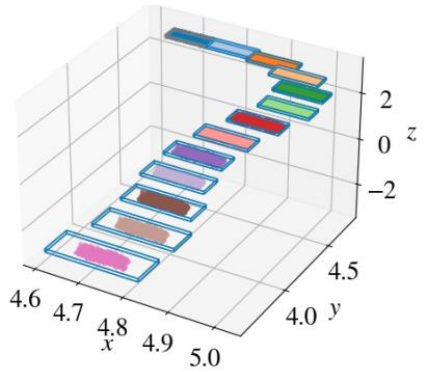
- Closed-loop systems often have well-defined notion of safety
  - Ex: Complete mission within time constraint
  - Ex: Get close to goal before battery level drops below some threshold
  - Ex: Avoid colliding with another agent or falling off cliff
  - Ex: Make sure chatbot never says bad words
  - Ex: Keep patient alive
- Many of these are **Reach-Avoid** properties with **temporal** aspect
  - There are states you want system to **reach** while **avoiding** some states
  - Temporal Logic is also a way to encode these formulas
- **Reachability analysis:** framework for reasoning about these problems

Forward Reachability Analysis

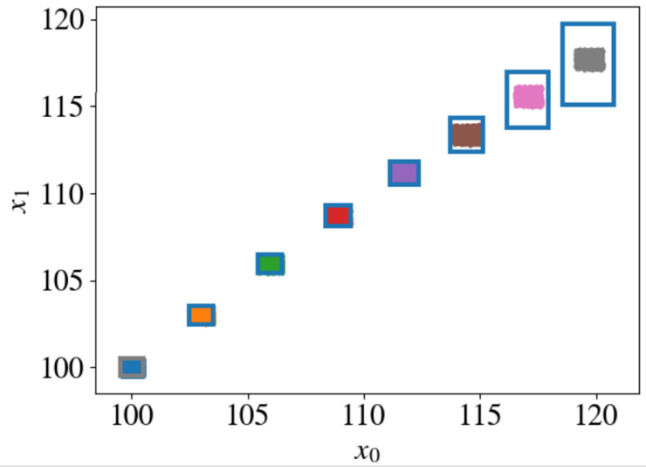


Derivation on Board

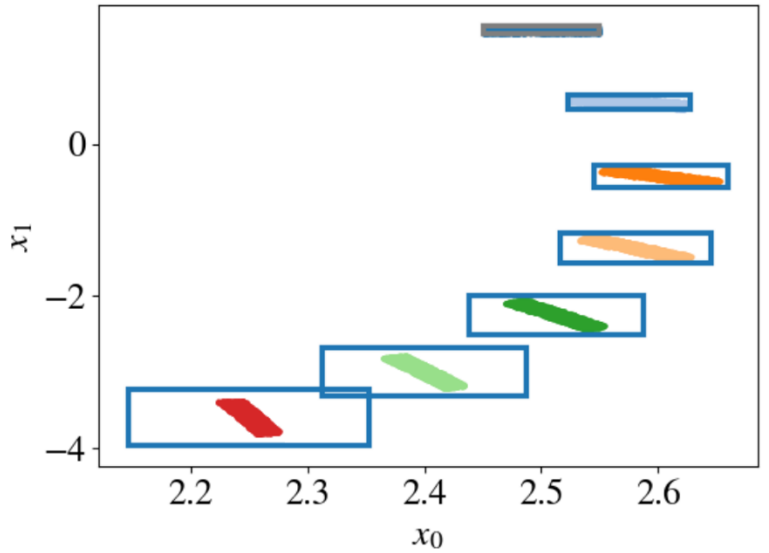
Scalability & Nonlinear Plants



6D quadrotor (x, y, z position shown)



270-state model of ISS



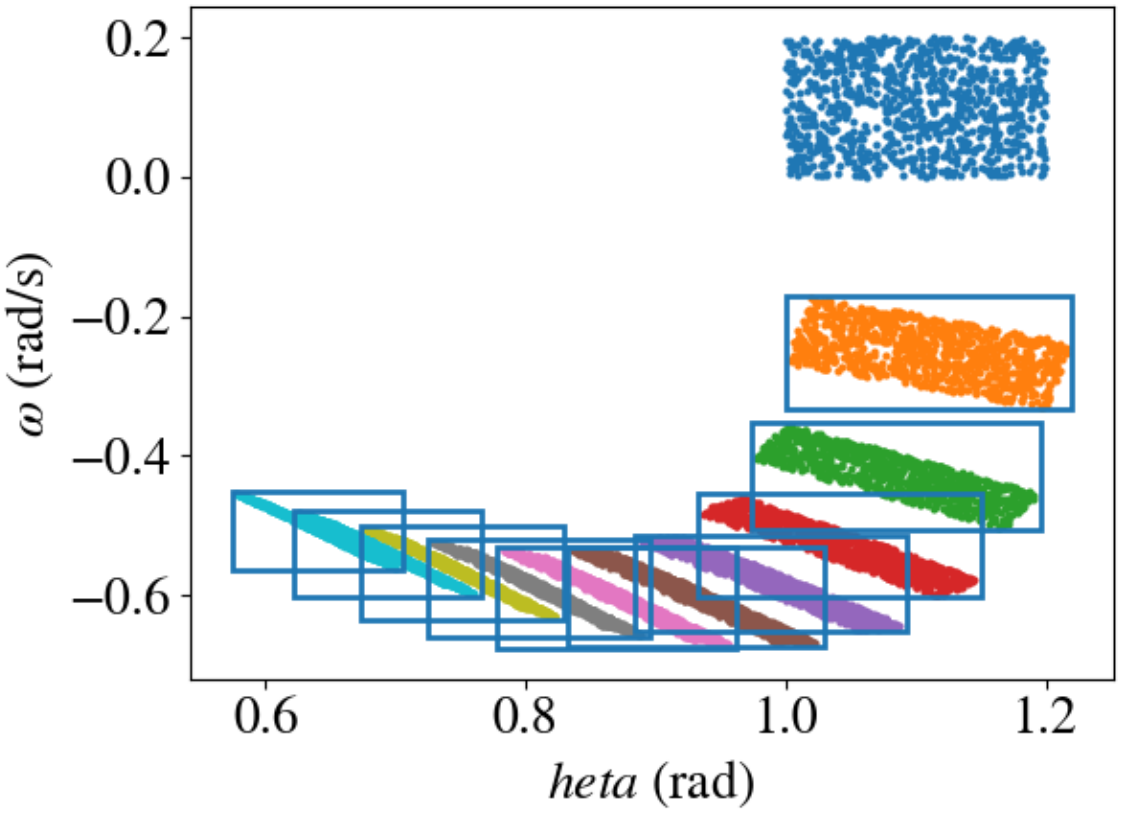
Duffing Oscillator (Nonlinear Plant)

$$\dot{\mathbf{x}}_1 = \mathbf{x}_2,$$
$$\dot{\mathbf{x}}_2 = -\mathbf{x}_1 - 2\zeta\mathbf{x}_2 - \mathbf{x}_1^3 + u,$$

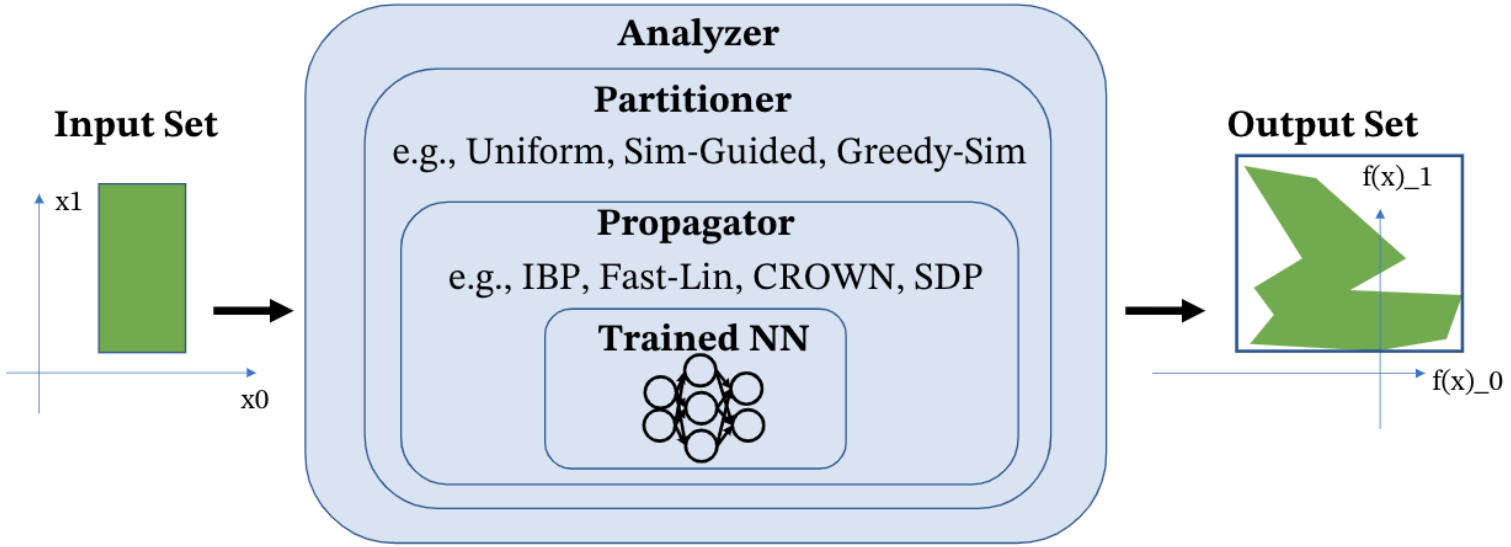
Everett, Habibi, Sun, How, “Reachability Analysis of Neural Feedback Loops”, IEEE Access 2021

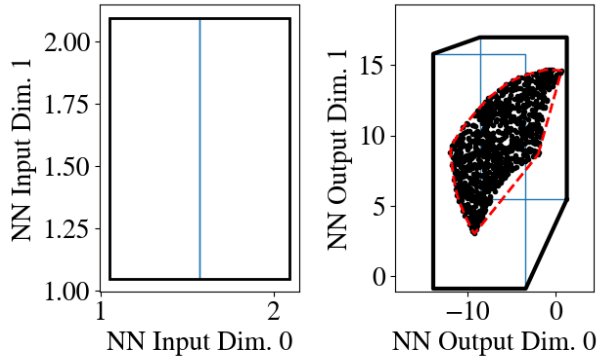
Pendulum

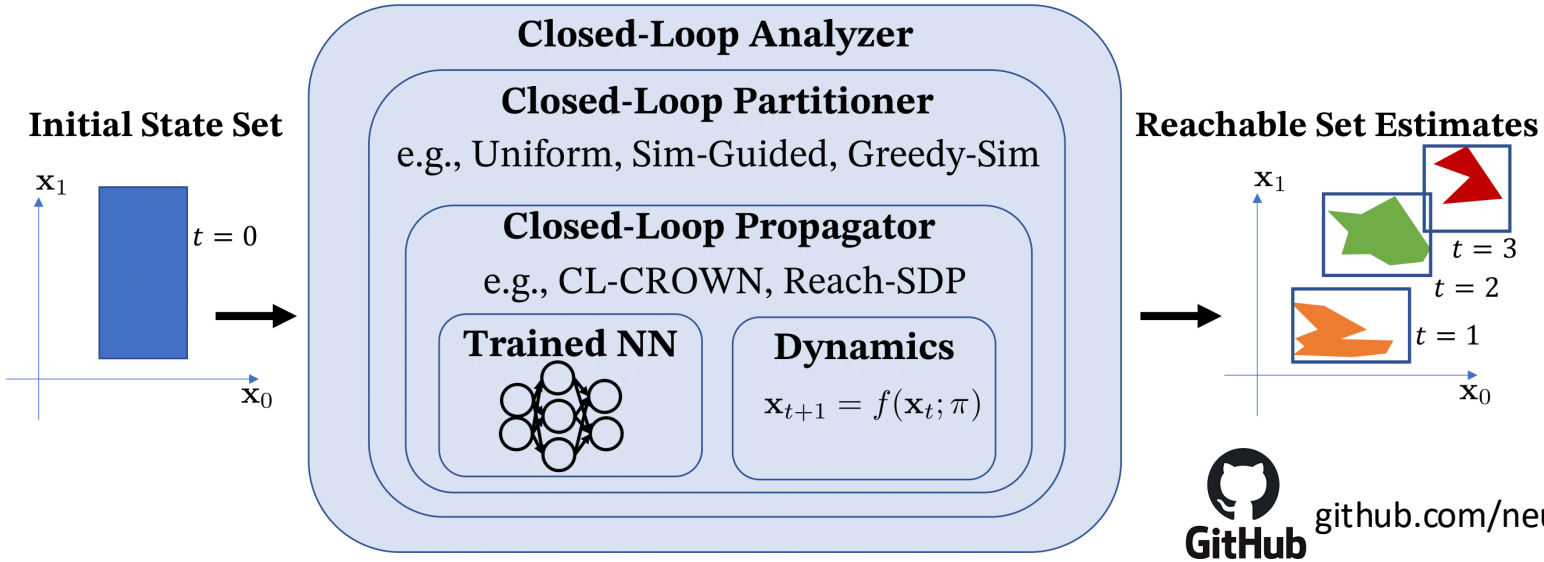
- Can handle dynamics with other nonlinearities, such as sin/cos

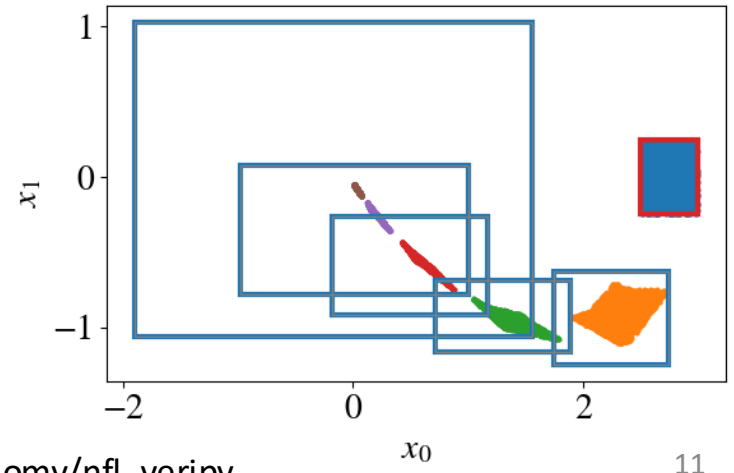


Branch & Bound methods apply here too









GitHub github.com/neu-autonomy/nfl_veripy

nfl_veripy

Compute forward reachable sets for a closed-loop system with a pre-trained NN control policy:

```
python -m nfl_veripy.example --config example_configs/icra21/fig3_reach_lp.yaml
```

```yaml
system:
  type: DoubleIntegrator
  feedback: FullState
  controller: default

analysis:
  reachability_direction: forward
  partitioner:
    type: None
  propagator:
    type: JaxCROWNUnrolled
    boundary_type: rectangle
  initial_state_range: "[[2.5, 3.0], [-0.25, 0.25]]"
  t_max: 5
  estimate_runtime: false
  estimate_error: false

visualization:
  save_plot: true
  show_plot: false
  make_animation: false
  show_animation: false
  show_samples: true
  show_trajectories: false
  plot_dims: [0, 1]
  plot_axis_labels: ["$x_0$", "$x_1$"]
  plot_aspect: auto
  plot_lims: null
```

What about Continuous Time?

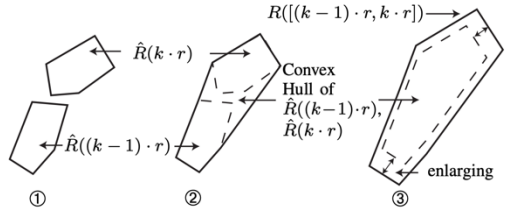
- If we only verify properties at discrete time instances, could miss violations in between
  - Sketchy alternative: use finer discretization (increases computation and still could miss violations)
- Recall: Continuous Time LTI system (assuming x(0)=0):

$$x(t) = \int_0^t e^{A(t-\tau)} B u(\tau) d\tau$$

  - Looks difficult to write as a linear combination of NN outputs…
- There are ways to analyze the system at discrete timesteps, then inflate reachable sets appropriately to account for interval between

Reachability Analysis of Linear Systems with Uncertain Parameters and Inputs

Matthias Althoff, Olaf Stursberg, and Martin Buss



What about Continuous Time?

- https://github.com/Verified-Intelligence/CROWN-Reach