Safe and efficient control using neural networks: An interior point approach

Baosen Zhang
ECE, University of Washington

NREL Autonomous Energy Systems Workshop
July 15th , 2022

Joint work with
Daniel Tabas

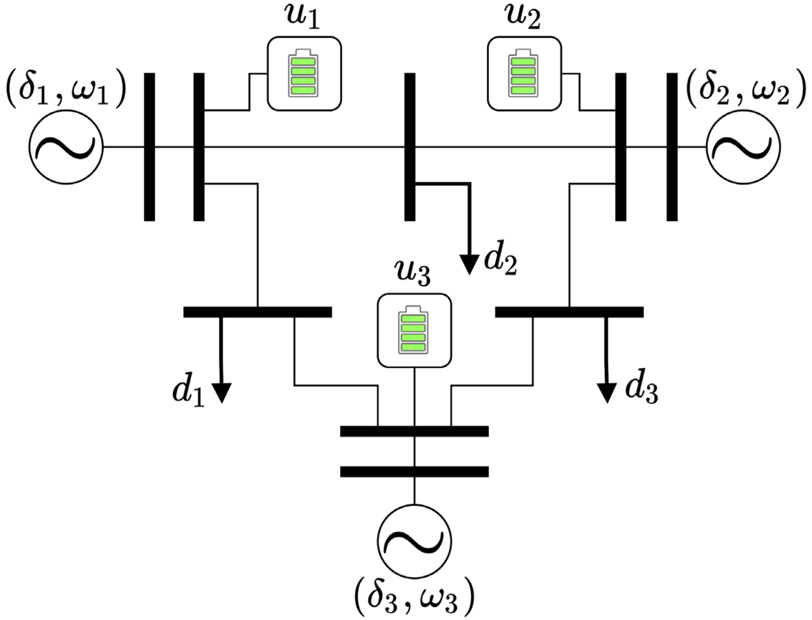
Safe Decision Making

- Power systems have several regulated quantities that must stay within prescribed ranges:
  - Bus voltage
  - Generator frequencies
  - Line flows
- Currently we have wide operating margins, but this is becoming increasing difficult
- Adaptive control and decisions are useful, and we focus on the computational aspects

Real-time Control

- Make fast, efficient, and safe decisions

  E.g., Frequency regulation with storage



- Frequencies and angles should stay within some bound
- Storage have actuation constraints

- Safety: all constraints should be satisfied for a set of disturbances
- Efficiency: minimize cost, low computation time

Planning

$$\min_x c(x) + \boxed{o(x)}$$

subject to:

Energy adequacy constraints

Technology constraints

Transmission constraints

$$\min_y g(x, y)$$

subject to:

Power balance constraints

Generation constraints

- The problem size can get very large, especially if storage and multiple scenarios are considered

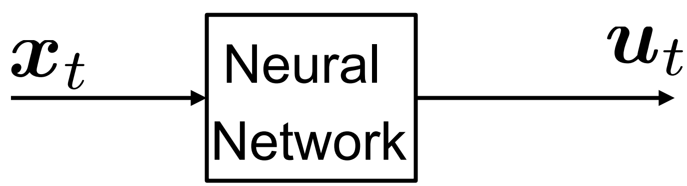
Computation Bottlenecks

- Common problem structure:

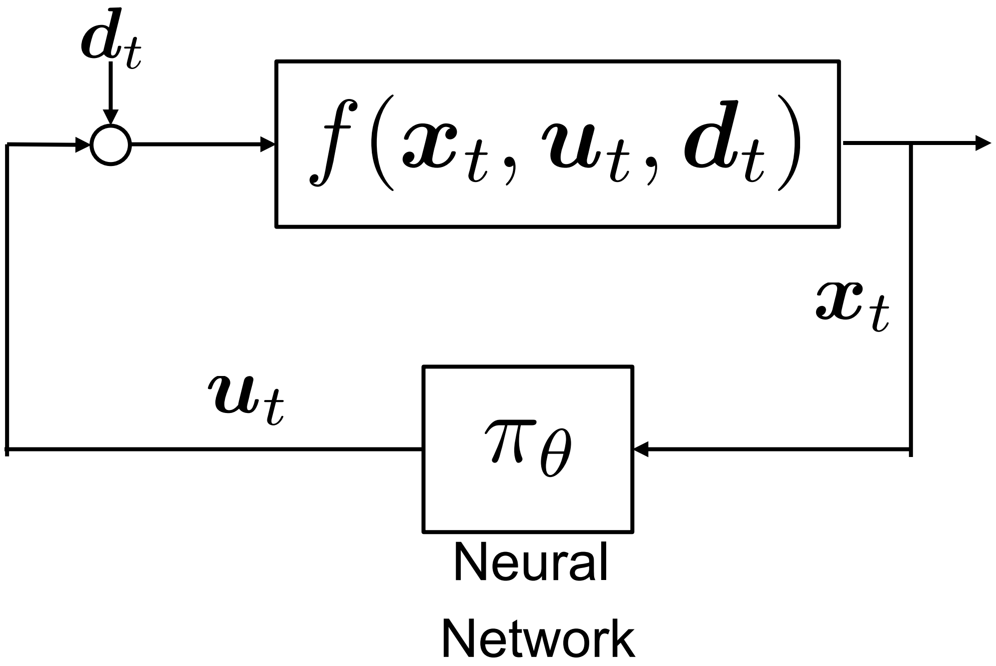
$$\boldsymbol{x}_{t+1} = f(\underbrace{\boldsymbol{x}_t}_{\text{state}}, \underbrace{\boldsymbol{u}_t}_{\text{control}}, \underbrace{\boldsymbol{d}_t}_{\text{disturbance}})$$

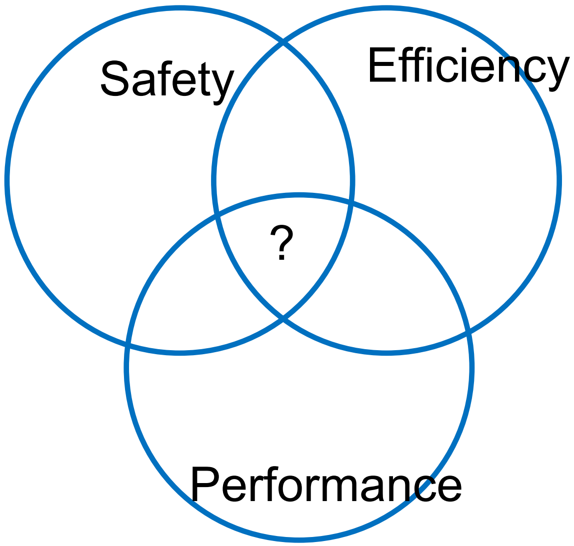
Find a control sequence that minimize cost and satisfy all state and control constraints

- Sometimes explicitly solving the problem is too computationally expensive
- Use a neural network as a proxy



Key Issues and Design Goals





Safe: x and u stay in their constraints

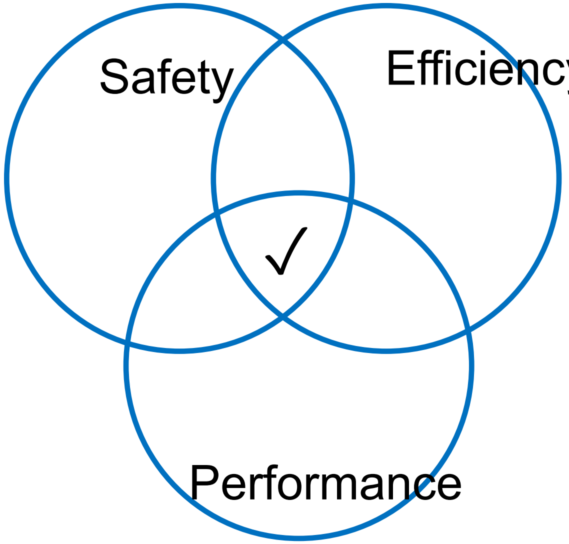
Computationally efficient

Performance: minimize some cost

Contributions

For a class of systems (e.g., linear)

- We provide a way to design neural networks that are always safe
- Easy to train, simple algebraic operations
- Based on geometry of convex sets and interior point algorithms



Model

$$\boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t + \boldsymbol{E}\boldsymbol{d}_t$$

State: $\boldsymbol{x} \in \mathcal{X}$
Control: $\boldsymbol{u} \in \mathcal{U}$
Disturbance: $\boldsymbol{d} \in \mathcal{D}$

convex polytopes

$$\mathcal{X} = \{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\}$$

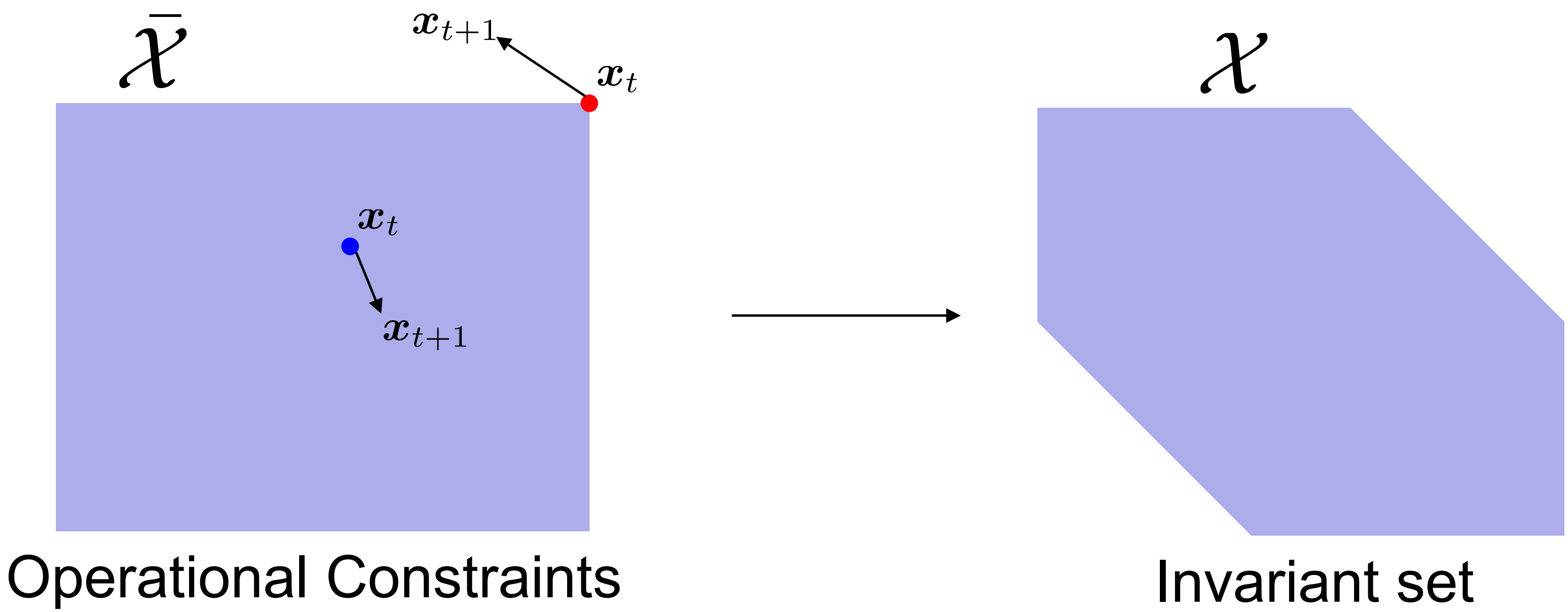
Static Feedback Policy: $\boldsymbol{u}_t = \pi_{\boldsymbol{\theta}}(\boldsymbol{x}_t)$

Safety: For all *t*, $\boldsymbol{u}_t \in \mathcal{U}$ and $\boldsymbol{x}_{t+1} \in \mathcal{X}$

Performance: $\min_{\boldsymbol{\theta}} \mathrm{cost}(\boldsymbol{x}, \pi_{\boldsymbol{\theta}}(\boldsymbol{u}))$

Safe Sets

$$\boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t + \boldsymbol{E}\boldsymbol{d}_t$$

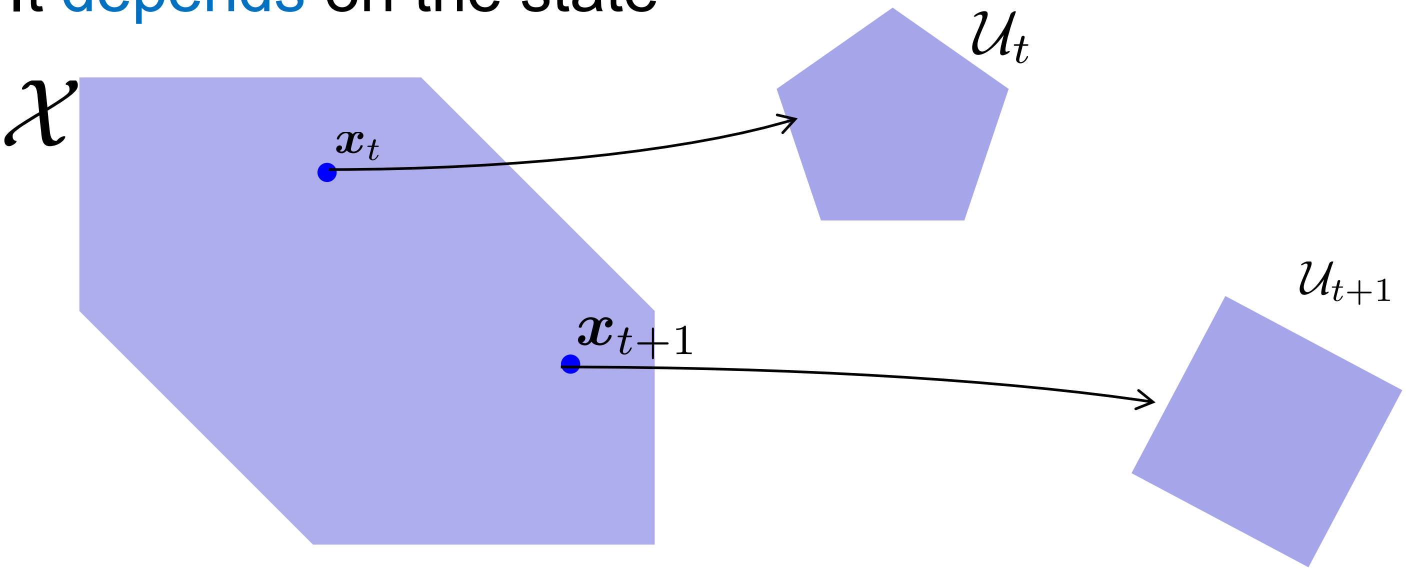


$$\mathcal{X} = \{\boldsymbol{x} : \exists \boldsymbol{u}, \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{E}\boldsymbol{d} \in \mathcal{X} \; \forall \boldsymbol{d}, \quad \mathcal{X} \in \bar{\mathcal{X}}\}$$

- Invariant set can is computed offline

Safe Actions

- Safe action set: $\mathcal{U}_t = \{\boldsymbol{u} \in \mathcal{U} : \boldsymbol{x}_{t+1} \in \mathcal{X}\}$
- It depends on the state



- Main challenge: how to get $\pi(\boldsymbol{x}_t)$ to be in these shifting polytopes?

Safe Actions

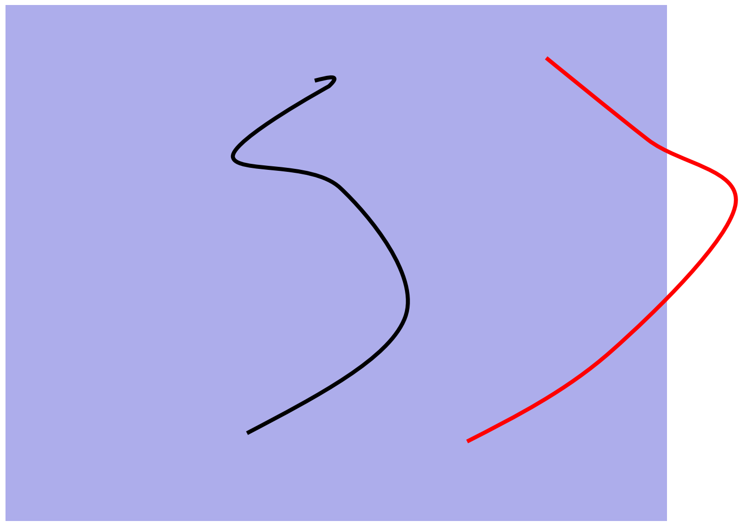
- It's not hard to get the output of a neural network to be in a fixed polytope
- The safe action set changes at every timestep

Current strategy: penalty or projection

Penalty-Based Methods

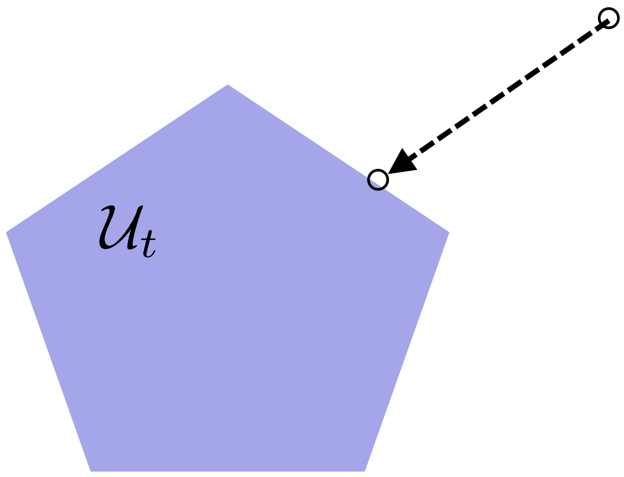
- High training cost for trajectories that violates the constraints
- Hard to balance safety and exploration
- At best probabilistic guarantees

State Constraints



Projection-Based Methods

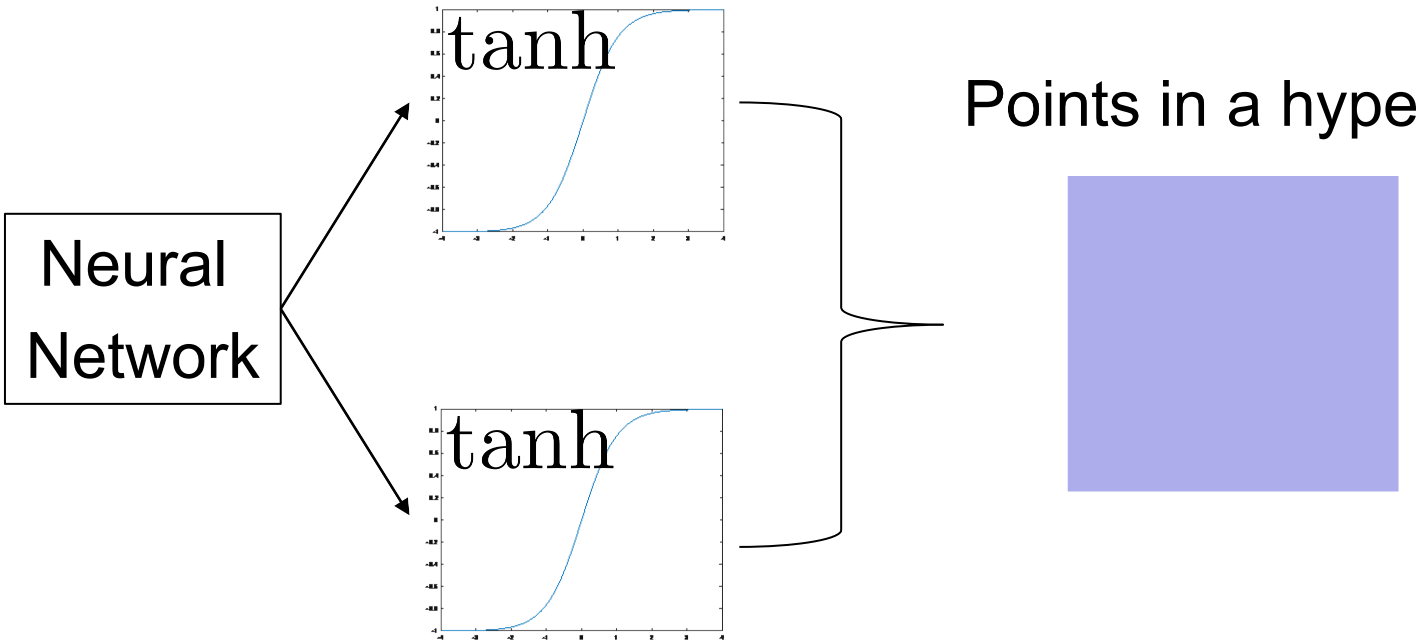
- Train a controller, if the control action falls outside, project it back



- Needs to solve an optimization problem
- Can over-explore the boundary

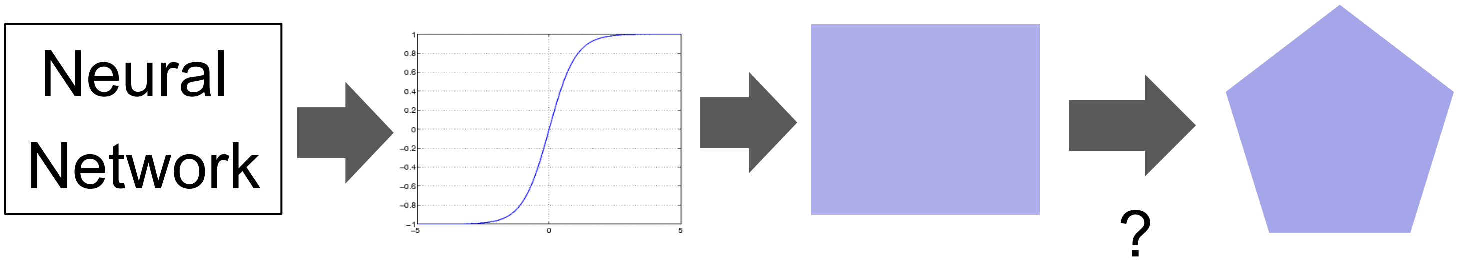
Restricting the Output

- Not all sets are difficult: Axis-aligned rectangles are easy



- Mapping a box to polytopes

Safe Control Actions



- We want a mapping between polytopes that is
  - Bijective
  - Easy to compute
  - (sub) Differentiable
- We use the gauge map

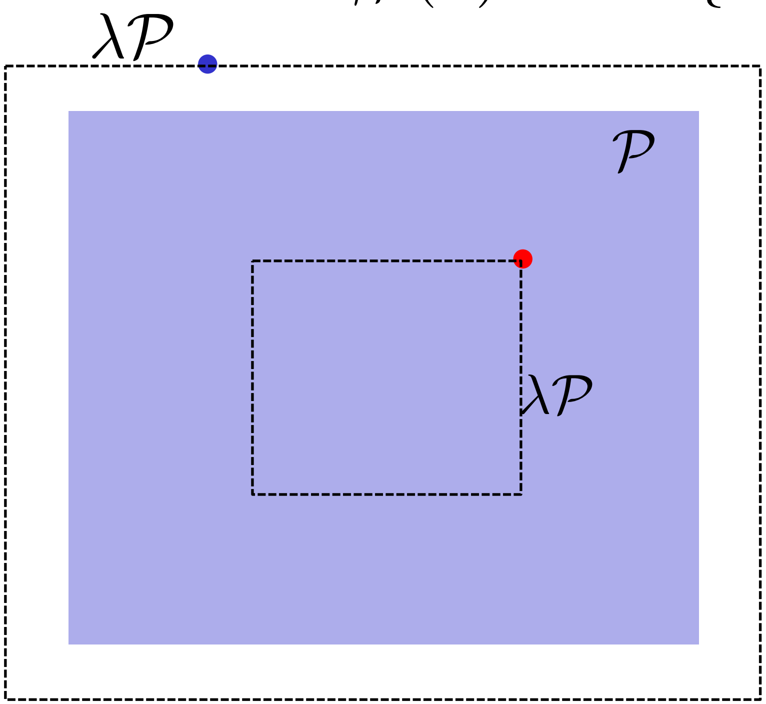
Mapping Between Polytopes

- Given a polytope that contains the origin,

$$\mathcal{P} = \{\boldsymbol{z} \in \mathbb{R}^n : \boldsymbol{F}\boldsymbol{z} \leq \boldsymbol{g}\}$$

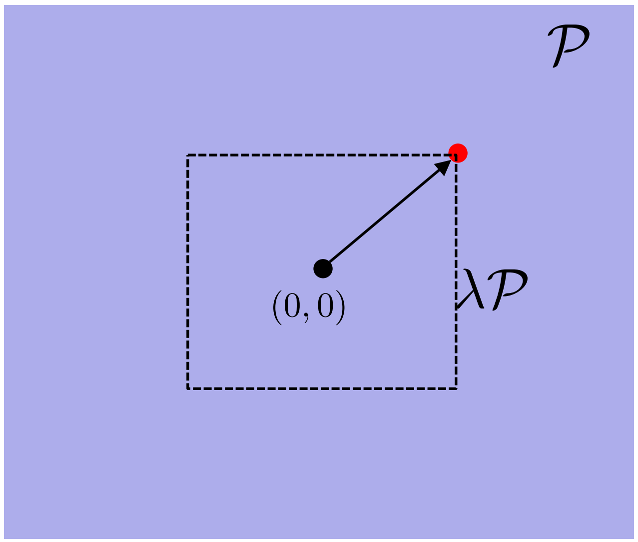
- The gauge function

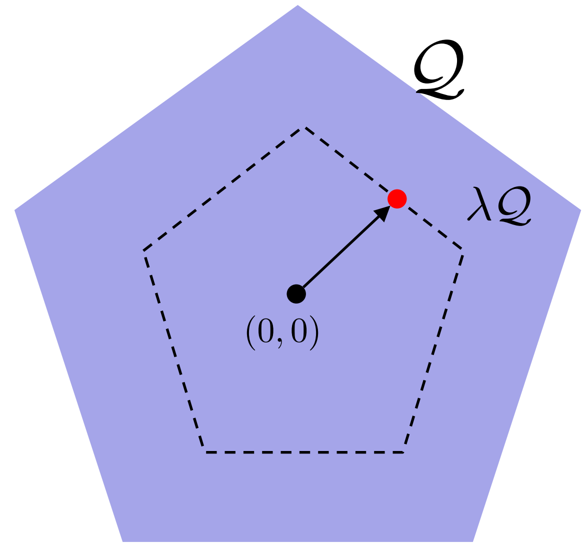
$$\gamma_{\mathcal{P}}(\boldsymbol{z}) = \min\{\lambda \geq 0 : \boldsymbol{z} \in \lambda\mathcal{P}\}$$



$$\begin{aligned} \gamma_{\mathcal{P}}(\boldsymbol{z}) &= \min\{\lambda \geq 0 : \boldsymbol{F}\boldsymbol{z} \leq \lambda\boldsymbol{g}\} \\ &= \min\{\lambda \geq 0 : \lambda \geq \frac{\boldsymbol{F}_i^T\boldsymbol{z}}{g_i}, \ \forall i\} \\ &= \max_i \frac{\boldsymbol{F}_i^T\boldsymbol{z}}{g_i} \end{aligned}$$

Gauge Map

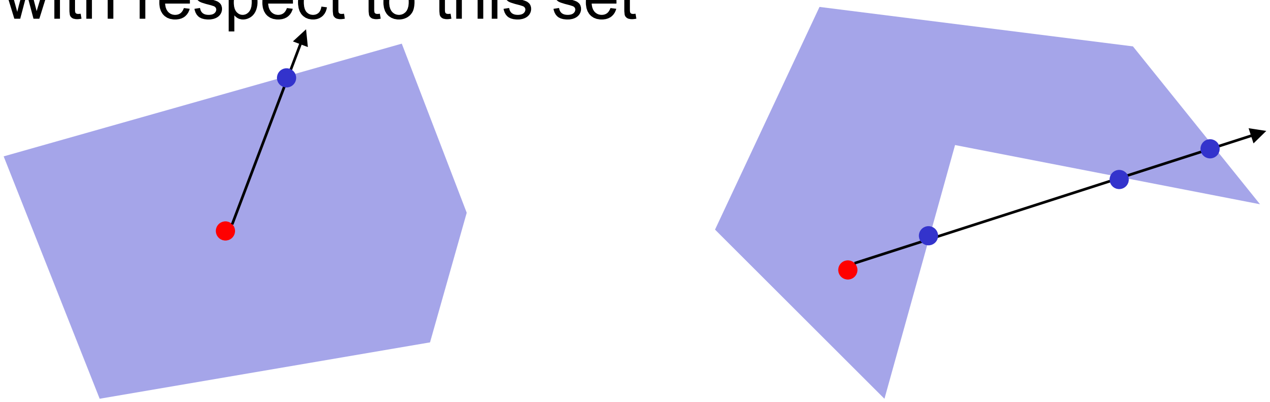




- Bijection between the sets
- Closed-form formula
- Sub-differentiable in the parameters

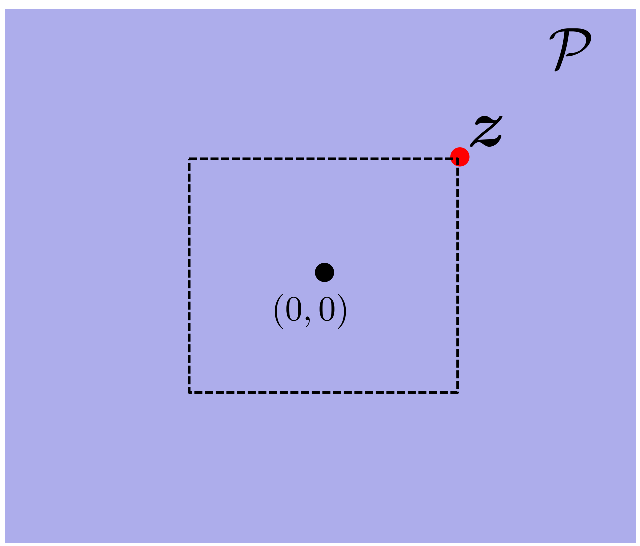
Convex Sets

- A ray from the origin can only cross the boundary of a convex set once
- This provides a unique way of identifying the the points with respect to this set

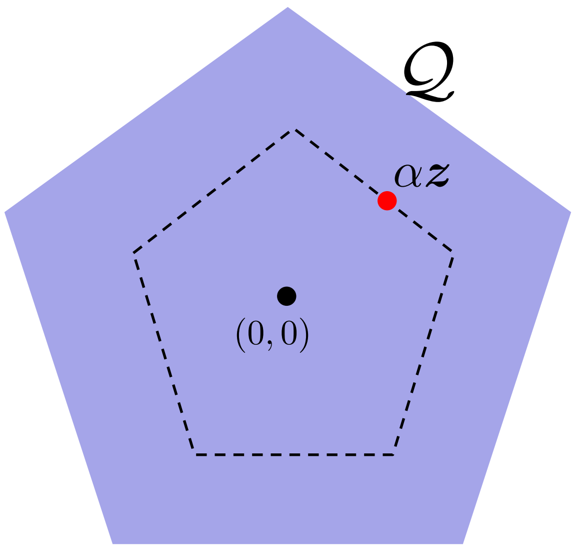


- For convex polytopes, the gauge function is easy to compute

Gauge Map



$\mathcal{P} = \{\boldsymbol{z} : \boldsymbol{F}\boldsymbol{z} \leq \boldsymbol{g}\}$



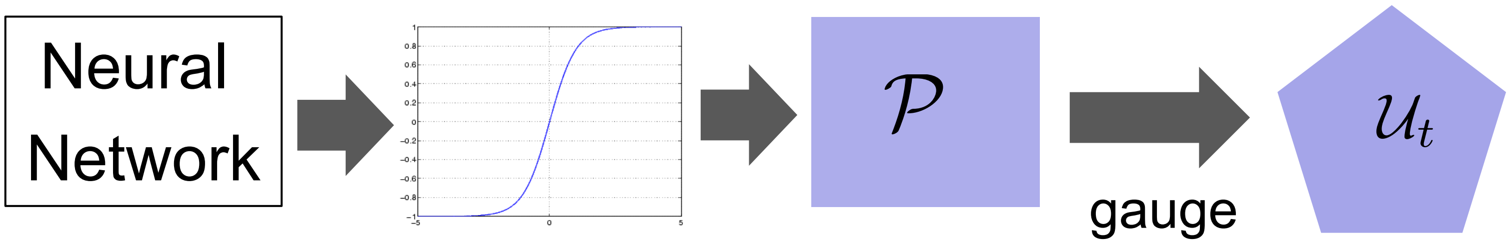
$\mathcal{Q} = \{\boldsymbol{z} : \boldsymbol{H}\boldsymbol{z} \leq \boldsymbol{k}\}$

Find $\alpha \geq 0$ such that $\gamma_{\mathcal{Q}}(\alpha\boldsymbol{z}) = \gamma_{\mathcal{P}}(\boldsymbol{z})$

$$\alpha = \frac{\gamma_{\mathcal{P}}(\boldsymbol{z})}{\gamma_{\mathcal{Q}}(\boldsymbol{z})} = \frac{\max_i \boldsymbol{F}_{\boldsymbol{i}}^{\boldsymbol{T}}\boldsymbol{z}/g_i}{\max_i \boldsymbol{H}_{\boldsymbol{i}}^{\boldsymbol{T}}\boldsymbol{z}/k_i}$$

Only requires the half-space representation of polytopes

Gauge Policy Networks



- $\mathcal{P}$: hypercube
- $\mathcal{U}_t$: half-space representation $\mathcal{U}_t = \{\boldsymbol{u} : \boldsymbol{H}\boldsymbol{u} \leq \boldsymbol{h}(\boldsymbol{x}_t)\}$
- Control policy:

$$\pi_\theta(\boldsymbol{x}_t) = G_t \circ \tanh \circ \phi_\theta(\boldsymbol{x}_t)$$

gauge map

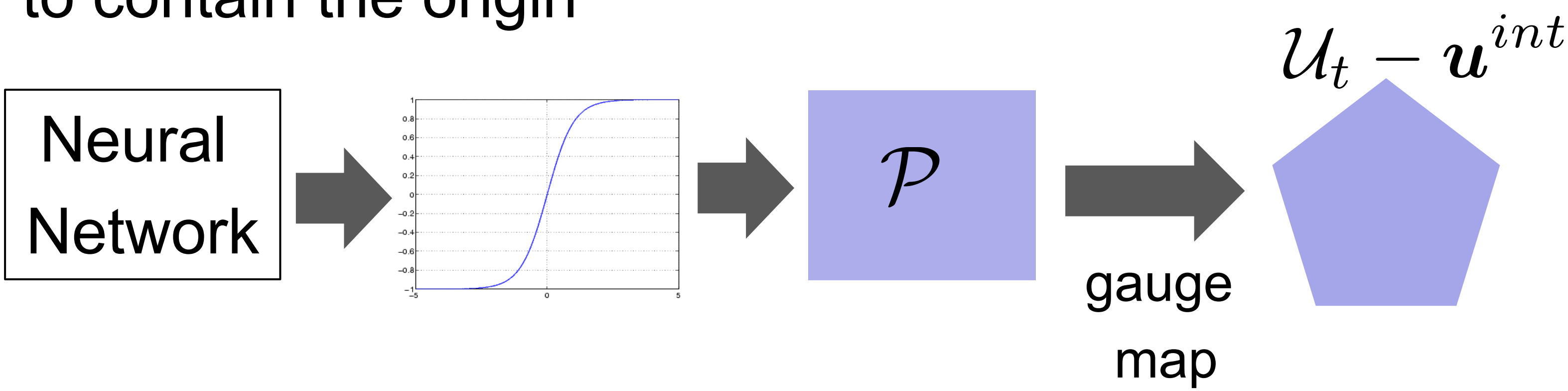
clamping function

feedforward neural network

- Sub-differentiable, easy to train
- But $\mathcal{U}_t$ may not contain the origin

Interior Points

- If we know an interior point, we can always shift a set to contain the origin



- $\boldsymbol{u}^{int}$ some interior point

$$\pi_\theta(\boldsymbol{x}_t) = G_t \circ \tanh \circ \phi_\theta(\boldsymbol{x}_t) + \boldsymbol{u}^{int}$$

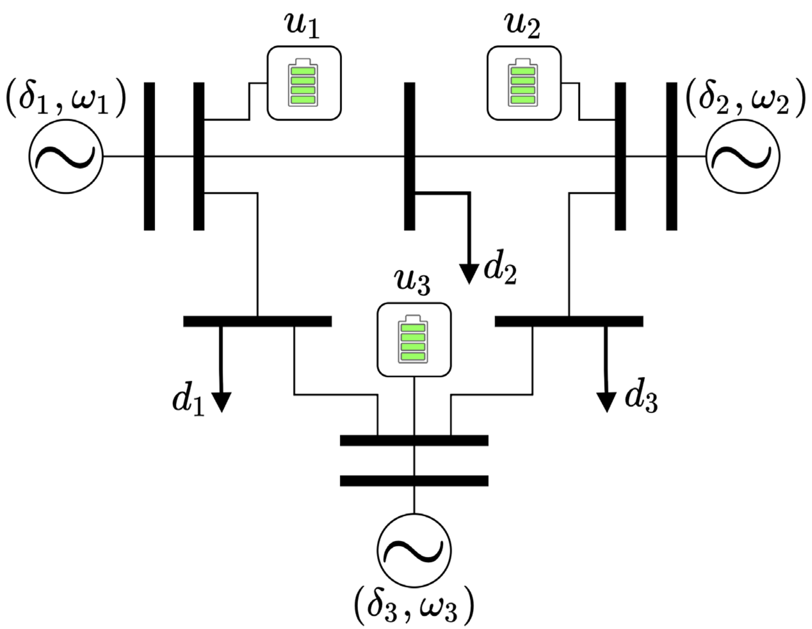
gauge map to $\mathcal{U}_t - \boldsymbol{u}^{int}$

Finding Interior Points

- In planning, there are often trivial, high cost, but feasible solutions
- In real-time control, there are often easy ways to find feasible control actions
  - There is a baseline feasible controller
  - A few iterations suffices

Frequency Control

- States: angle/frequency
- Control: battery output
- Cost: freq dev+bat deg

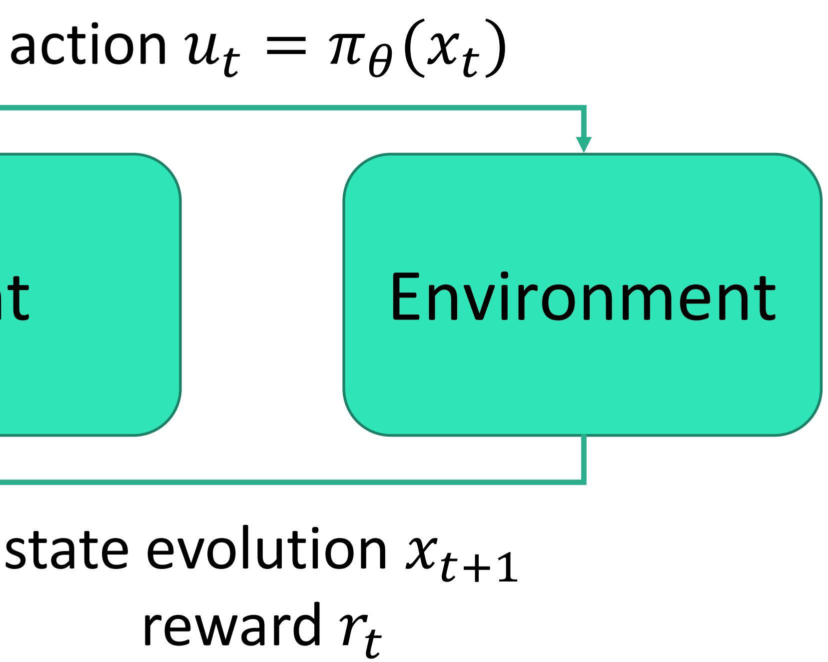


- There exist a linear safe controller:

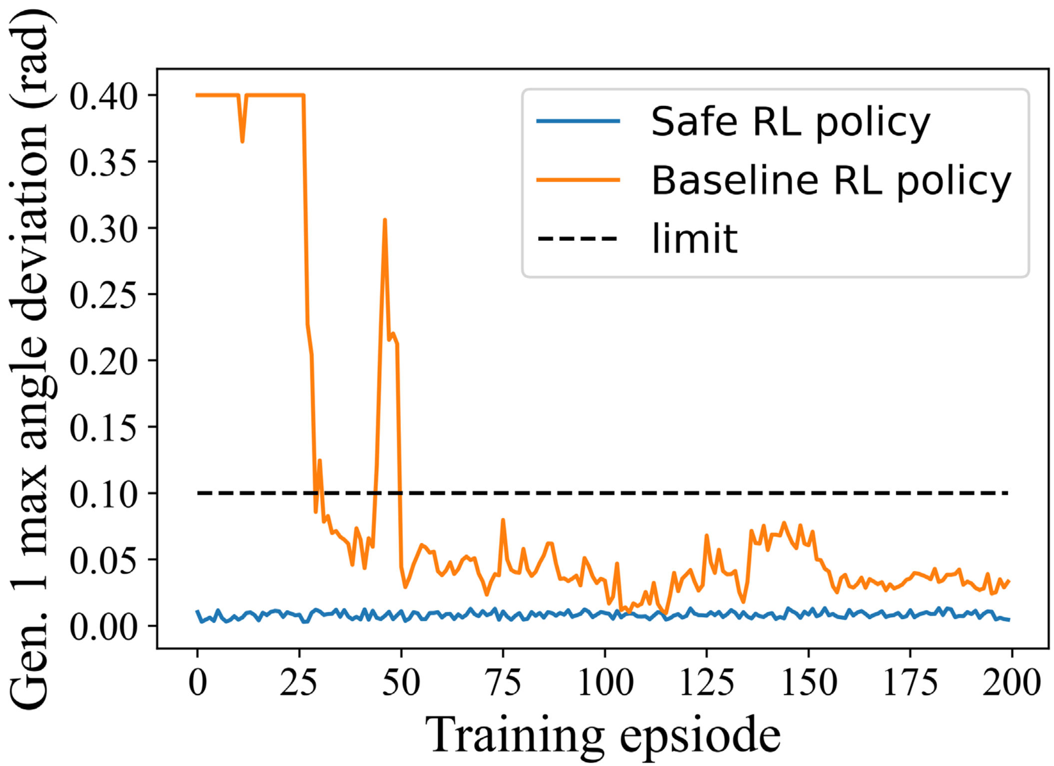
$$\boldsymbol{u}_t^{int} = \boldsymbol{K}\boldsymbol{x}_t$$

Policy Network

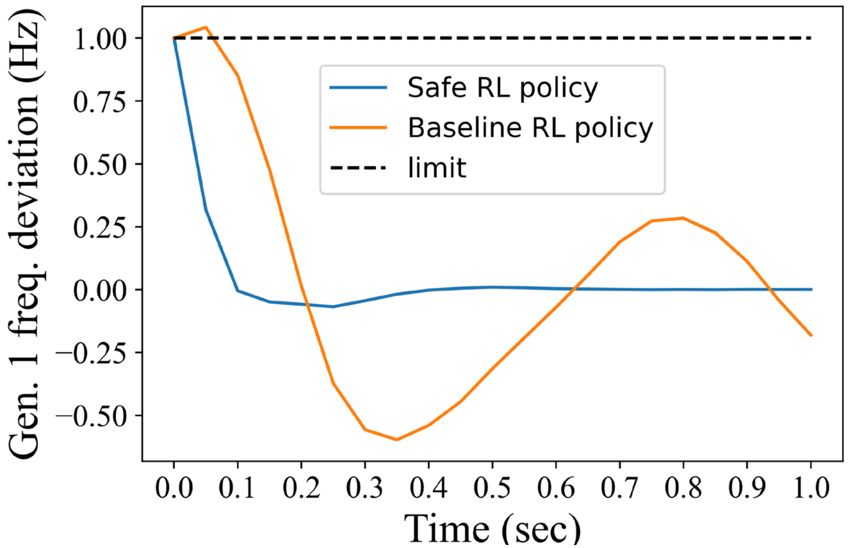
- Training: safely interact with environment to choose θ (policy gradient algorithms)
- Testing: θ fixed
- Benchmark: penalty-based approach



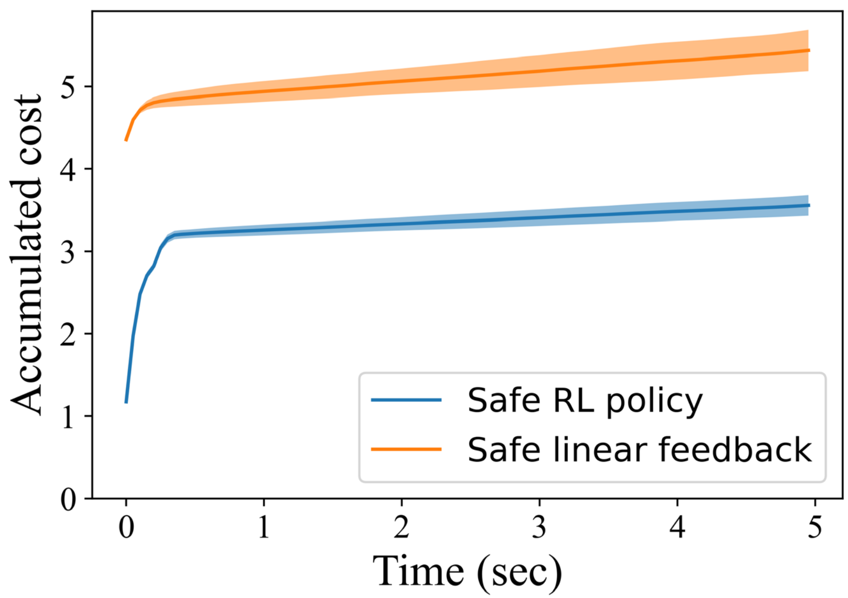
Safety During Training



Testing and Cost



Safety during testing



Lower operational costs than (safe) linear feedback

Explicit Model Predictive Control

- MPC is a very popular control strategy
- Exploit known dynamics and cost
- Constraint satisfaction and treat disturbances less conservatively
- Solving online MPC can be computationally intensive

$$\min_{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_T} \sum_{t=1}^{T} c(\boldsymbol{x}_t, \boldsymbol{u}_t)$$

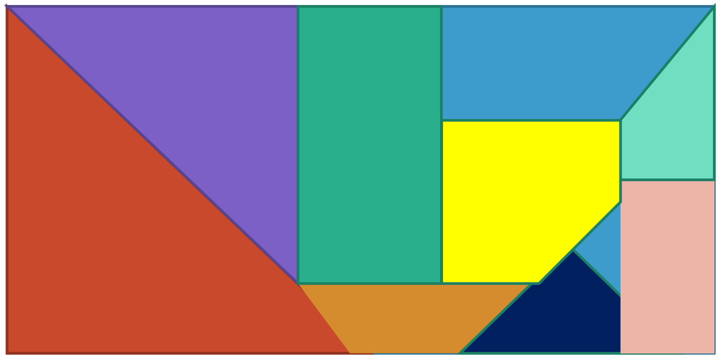
$$\text{s.t. } \boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t$$

$$\boldsymbol{x}_t \in \mathcal{X}, \boldsymbol{u}_t \in \mathcal{U} \ \forall t$$

Explicit MPC

- If the system is linear and the cost quadratic, then the optimal action are piecewise affine functions

$$\boldsymbol{u}_1^*, \ldots, \boldsymbol{u}_T^* = f(\boldsymbol{x}_0)$$



- The number of pieces scales exponentially
- Lots of work on using neural network to approximate
- Projection is used to provide hard guarantees

Solution Concept

- Use gauge neural network to choose a sequence $u_0, u_1, \ldots, u_{T-1}$ inside the feasible set $\mathcal{F}(\boldsymbol{x}_0)$
- Guarantees safety without oracles or projections

How do we get an interior point in $\mathcal{F}(\boldsymbol{x}_0)$?

Same approach as interior point algorithms used in optimization.

Interior Point Algorithms

- Suppose we want to solve

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$
$$\text{s.t. } g_i(\boldsymbol{x}) \leq 0, \; i = 1, \ldots, m$$

- Path-following interior point algorithm solves a sequence of problems

$$\boldsymbol{x}_\mu = \arg\min f(x) + \mu \sum_i^m \log(-g_i(\boldsymbol{x}))$$

Barrier function

$$\boldsymbol{x}_\mu \to \boldsymbol{x}^* \text{ as } \mu \to 0$$

- But this path requires a strictly feasible point to get started

Phase 1 of Interior Point Algorithms

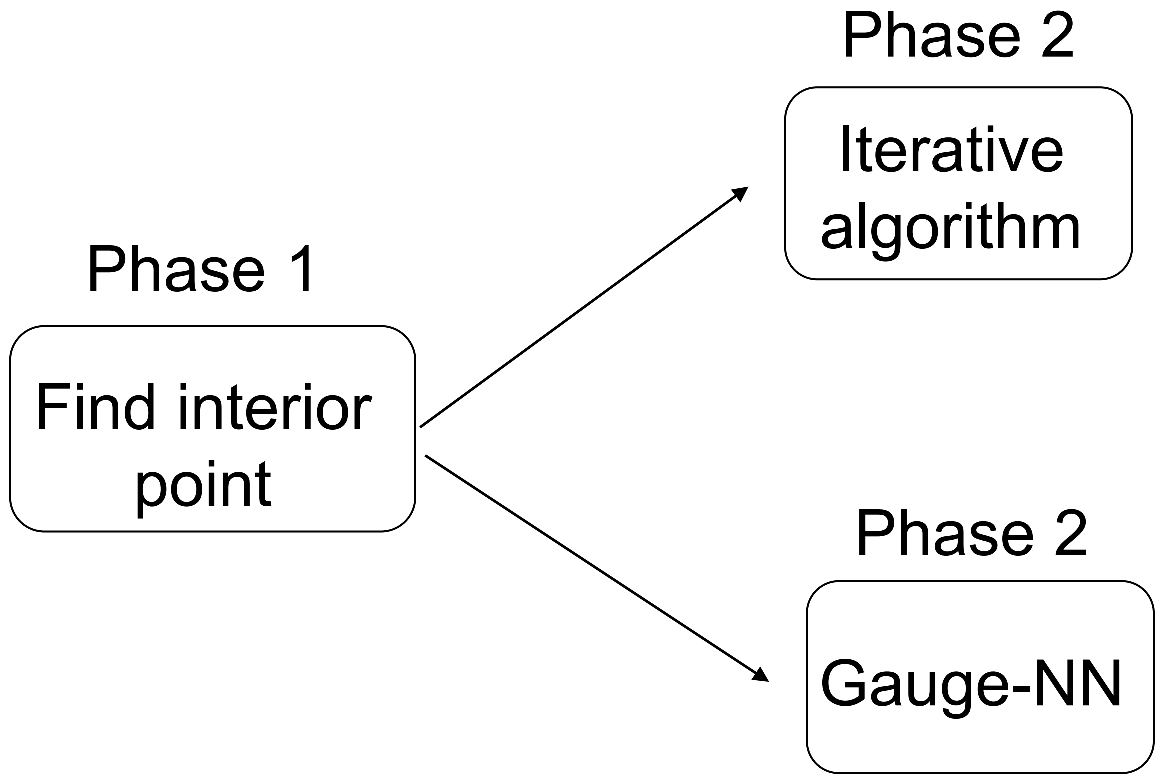
$$\begin{aligned} &\min_{\boldsymbol{x}} \ f(\boldsymbol{x}) \\ &\text{s.t.} \ \ g_i(\boldsymbol{x}) \leq 0, \ i = 1, \ldots, m \end{aligned}$$

- Sometimes a feasible point is easy to find
- If not, a phase 1 problem is solved

$$\begin{aligned} &\min_{\boldsymbol{x}, s} \ s \\ &\text{s.t.} \ \ g_i(\boldsymbol{x}) \leq s, \ i = 1, \ldots, m \end{aligned} \qquad \left.\begin{aligned} &\boldsymbol{x} = \boldsymbol{0} \\ &s = \max_i g_i(\boldsymbol{0}) \end{aligned}\right\} \text{ is feasible}$$

- This problem can be terminated once $s \leq 0$
- Finding a feasible point usually takes two or two iterations

Learning for MPC



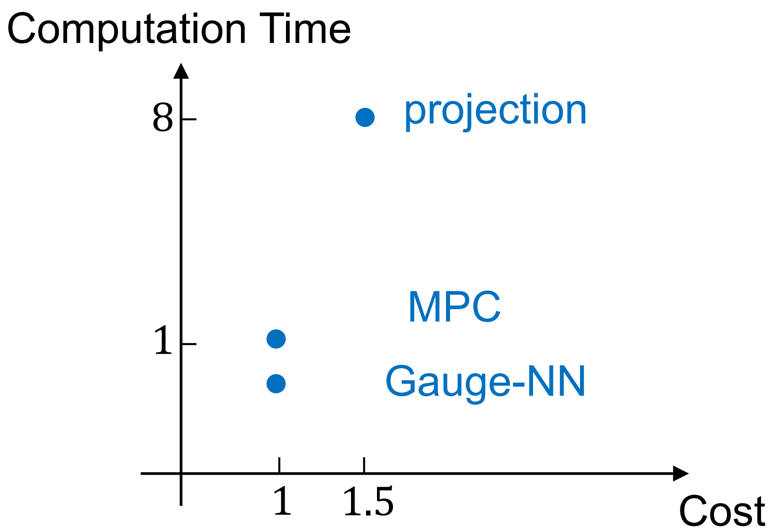
- Gauge-NN can lead to significant speedup if phase 2 is more expensive than phase 1

Simulation Setting

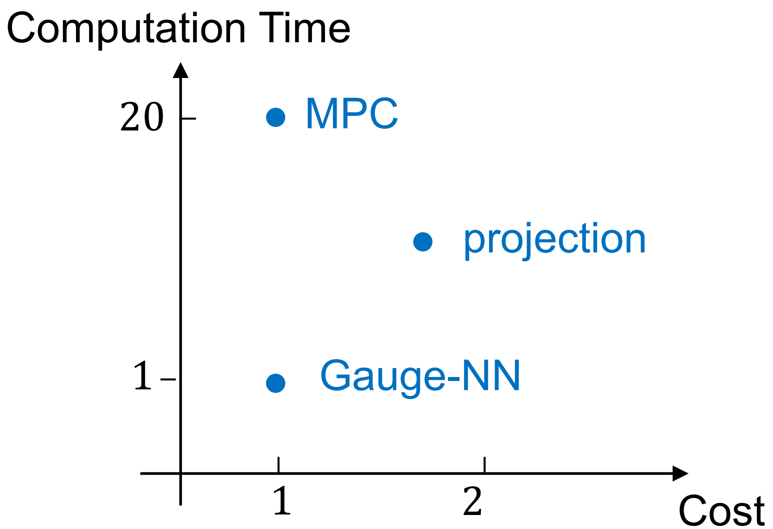
- MPC with linear constraints and quadratic cost function
- Training:
  - Gauge neural network $u_\theta: R^n \to R^{mT}$
  - Data: $\{x_0^k\}_{k=1}^N$
  - Predict trajectories generated by u_θ
  - Training loss: average MPC cost over sample trajectories
- Testing:
  - Close the loop: policy $\pi_\theta(x_t)$ given by first action from $u_\theta(x_t)$

Simulation Results

Deterministic MPC



Scenario-based MPC



Conclusion and Future Work

- A way to find safe policies through mapping between convex sets
- Can learn nonlinear policies
- Performance and computation speedup

Future work

- Convex but non-polytopic sets
- Non-convex costs (e.g., robust optimization)
- Multi-agent systems