



Data-Driven Safe Iterative Control: Learning from **Successful Trajectories** and **Failed Executions**

Ahmad Amine & Rahul Mangharam

Motivation

The real world is uncertain and never matches simulation:

Dynamics are uncertain:

- Unstructured environments: varying terrain, friction, temperature
- System degradation over time: wear and tear

Yet we expect performance to keep improving:

- Manufacturing: faster cycle times
- Racing: lower lap times

Learning cannot stop at deployment.



1. Learning from **Successful** Trajectories: Safe Information Theoretic Learning Model Predictive Control (**SIT-LMPC**)
2. Learning from **Failure**: Failure-Aware Iterative Learning (**FAIL**) of State-Control Invariant Sets

SIT-LMPC: Safe Information-Theoretic Learning Model Predictive Control for Iterative Tasks

Zirui Zang*, **Ahmad Amine***, Nick-Marios T. Kokolakis, Truong X. Nghiem, Ugo Rosolia, and Rahul Mangharam

IEEE RA-L 2026 - presented in IEEE ICRA 2026



IEEE ROBOTICS & AUTOMATION
LETTERS

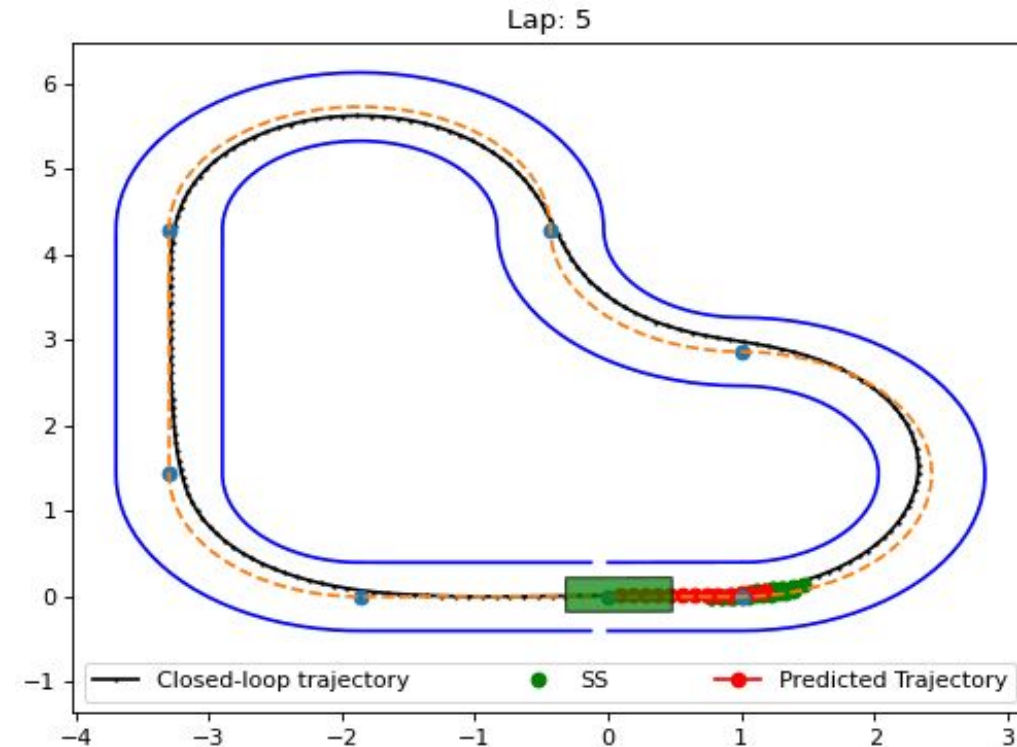
A PUBLICATION OF THE IEEE ROBOTICS AND AUTOMATION SOCIETY

SIT-LMPC - Problem Motivation

Goal: We seek a policy that

- a. Iteratively improves with experience
- b. Guarantees safety by complying with:
 - a. State constraints in horizon.
 - b. Terminate in the set of previous trajectories (safe fallback).
- c. Works for nonlinear stochastic dynamical systems with any constraints

Existing approaches sacrifice at least one point.



SIT-LMPC - Problem Statement

Discrete-time nonlinear *stochastic* dynamical system

$$x(k+1) = f(x(k), u(k), w(k)), \quad x(0) \stackrel{\text{a.s.}}{=} x_0,$$

Assumptions:

- i.i.d bounded disturbance

- Require:

- Require:



Infinite Horizon Optimal Control

Minimize expected running cost $r(x(k), u(k))$ for all time.

$$J(x_0, u(\cdot)) \triangleq \mathbb{E} \left[\sum_{k=0}^{\infty} r(x(k), u(k)) \right]$$

Infinite Horizon Optimal Control Problem

$$J^*(x_0) \triangleq \min_{u(\cdot) \in \mathcal{F}(x_0)} J(x_0, u(\cdot)) \quad (3a)$$

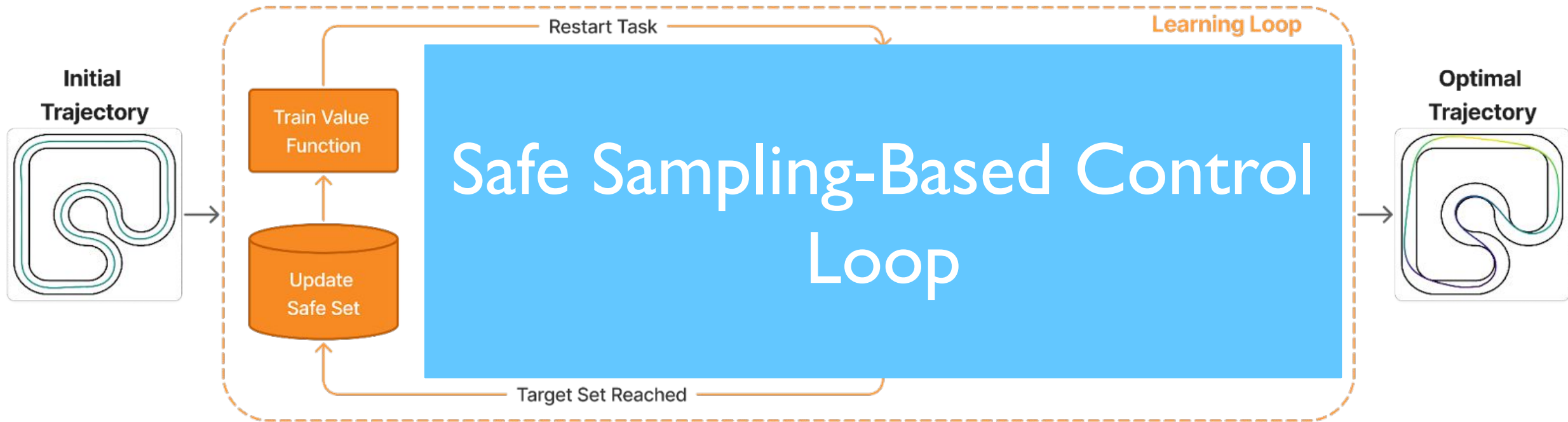
subject to

$$x(k+1) = f(x(k), u(k), w(k)), \quad k \in \overline{\mathbb{Z}}_+, \quad (3b)$$

$$x(0) \stackrel{\text{a.s.}}{=} x_0 \in \mathcal{X}, \quad (3c)$$

$$x(k) \in \mathcal{X}, \quad u(k) \in \mathcal{U}, \quad k \in \overline{\mathbb{Z}}_+, \quad (3d)$$

Proposed Approach - Overview



Convert to stochastic FTOCP using robust safe-sets, Rosolia et al. (2022):

1. Robust control invariant set w.r.t the dynamics and disturbance

$$\mathcal{S}_{t+1}(x_0^j) = \left\{ x_{t+1} \in \mathcal{S} \mid \begin{array}{l} \forall x_t \in \mathcal{S}_t(x_0^j), \forall w_t \in \mathcal{W}, \\ \exists u_t \in \mathcal{U}, x_{t+1} = f(x_t, u_t, w_t) \end{array} \right\}$$

2. Sampled safe set defined by the union over successful trajectories

$$\mathcal{S}^l \triangleq \{ x^i(k) : i \in \mathcal{I}^l \text{ and } k \in \overline{\mathbb{Z}}_+ \}$$

3. Practically, we use the convex safe set $\mathcal{CS}^l$

Finite Horizon Reformulation

Resulting stochastic FTOCP:

$$J_{\text{LMPC}}^l(x^l(t)) \triangleq \min_{u_{\cdot|t}^l} \mathbb{E} \left[\sum_{k=t}^{t+T-1} h(x_{k|t}^l, u_{k|t}^l) + V^l(x_{t+T|t}^l) \right]$$

subject to

$$x_{k+1|t}^l = f(x_{k|t}^l, u_{k|t}^l, w^l(k)), \quad k \in \{t, \dots, t+T-1\}$$

$$x_{t|t}^l \stackrel{\text{a.s.}}{=} x^l(t),$$

$$x_{k|t}^l \in \mathcal{X}, \quad u_{k|t}^l \in \mathcal{U} \quad \text{a.s.}, \quad k \in \{t, \dots, t+T-1\},$$

$$x_{t+T|t}^l \in \mathcal{CS}^{l-1} \quad \text{a.s.},$$

Cost-to-go $V^l(x_{t+T|t}^l)$ learnt from the safe-set

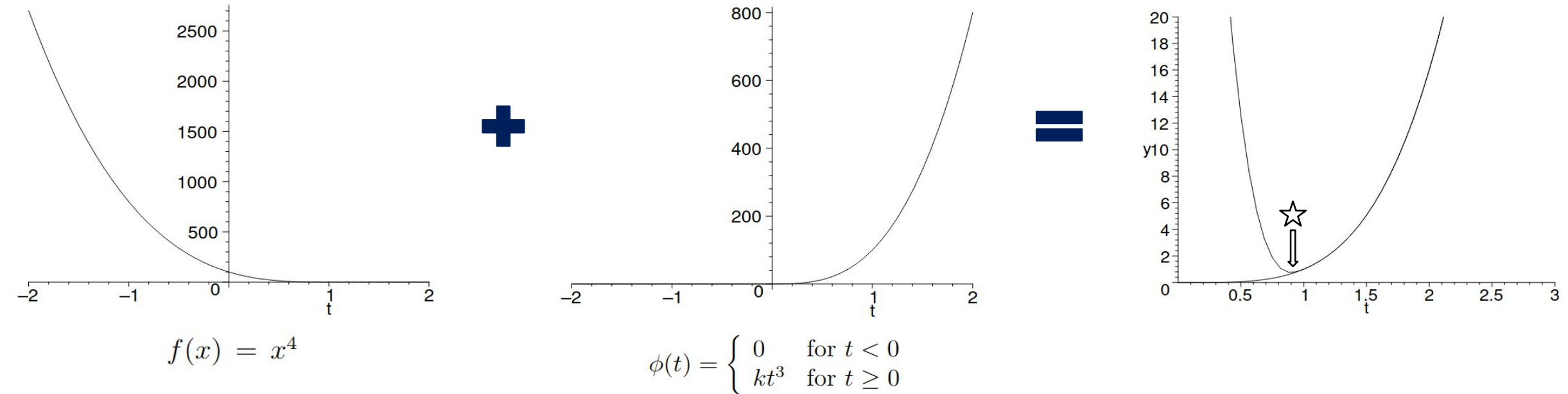
We solve this problem using sampling-based MPC - MPPI specifically

Constrained MPPI Approaches

1. Projection trajectories and control onto constraint sets
 - Expensive in higher dimensions
2. Utilize differential flatness for fast projection
 - Constricts the problem to specific class of dynamical systems
3. Iterative sampling, fixed high-cost for violations
 - Computationally expensive, conservative
4. Control Barrier Functions as safety filters and a cost term
 - Need a valid CBF or to learn one

Penalty MPPI

We convert the problem into an unconstrained problem using exterior penalty functions on the constraints:



The resulting cost function changes to:

$$\min_{u_{\cdot|t}^{l_r}} \mathbb{E} \left[\sum_{k=t}^{t+T-1} \left(r \left(x_{k|t}^{l_r}, u_{k|t}^{l_r} \right) + \lambda_{\mathcal{X}} d_{\mathcal{X}} \left(x_{k|t}^{l_r} \right) \right) + V^{l_r} \left(x_{t+T|t}^{l_r} \right) + \lambda_{\mathcal{S}^l} d_{\mathcal{S}^l} \left(x_{t+T|t}^{l_r} \right) \right]$$

Penalty Functions

- Define the set-distance function

$$\text{dist}(x, C) \triangleq \inf_{y \in C} \{\|x - y\|_2\}$$

- Define distance to each of the constraint sets

$$d_{\mathcal{CS}^{l-1}}(x) \triangleq \text{dist}(x, \mathcal{CS}^{l-1})$$

$$d_{\mathcal{X}}(x) \triangleq \text{dist}(x, \mathcal{X})$$

- Assign each a corresponding non-negative penalty multiplier $\lambda_{\mathcal{X}}$ and $\lambda_{\mathcal{S}^l}$

Penalized Problem Formulation

- Resulting penalized problem

$$J_{\text{SIT-MPC}}^{l_r} \left(t, x^{l_r}(t), u_{\cdot|t}^{l_r} \right) \\ \triangleq \min_{u_{\cdot|t}^{l_r}} \mathbb{E} \left[\sum_{k=t}^{t+T-1} \left(r \left(x_{k|t}^{l_r}, u_{k|t}^{l_r} \right) + \lambda_{\mathcal{X}} d_{\mathcal{X}}(x_{k|t}^{l_r}) \right) + V^{l_r} \left(x_{t+T|t}^{l_r} \right) + \lambda_{\mathcal{S}^l} d_{\mathcal{S}^l} \left(x_{t+T|t}^{l_r} \right) \right]$$

subject to $x_{k+1|t}^{l_r} = f \left(x_{k|t}^{l_r}, u_{k|t}^{l_r}, w^{l_r}(k) \right), \quad k \in [t, \dots, t+T-1],$

$$x_{t|t}^{l_r} = x^{l_r}(t)$$

$$u_{k|t}^{l_r} \in \mathcal{U}, \quad k \in [t, \dots, t+T-1].$$

- Key question:** How do we select $\lambda_{\mathcal{X}}$ and $\lambda_{\mathcal{S}^l}$?

Optimizing for Penalty Parameters

- Idea: sample $\lambda \in \Lambda \triangleq [0, \lambda_{\mathcal{X}}^{\max}] \times [0, \lambda_{\mathcal{CS}^{l-1}}^{\max}]$, solve P-MPPI problems in parallel
- Each MPPI solves the penalized MPC problem

$$\min_{u_{\cdot|t}^{l_r}} \mathbb{E} \left[\sum_{k=t}^{t+T-1} \left(r \left(x_{k|t}^{l_r}, u_{k|t}^{l_r} \right) + \lambda_{\mathcal{X}} d_{\mathcal{X}}(x_{k|t}^{l_r}) \right) + V^{l_r} \left(x_{t+T|t}^{l_r} \right) + \lambda_{\mathcal{S}^l} d_{\mathcal{S}^l} \left(x_{t+T|t}^{l_r} \right) \right]$$

- Filter out only feasible solutions from the P-MPPI solutions

$$\mathcal{F}_t^l = \{ \lambda \in \Lambda : \hat{x}_{k|t,\lambda}^{\star l} \in \mathcal{X}, k \in \{t, \dots, t+T\}, \hat{x}_{t+T|t,\lambda}^{\star l} \in \mathcal{CS}^{l-1} \}, \quad l \in \mathbb{Z}_+, \quad t \in \overline{\mathbb{Z}}_+.$$

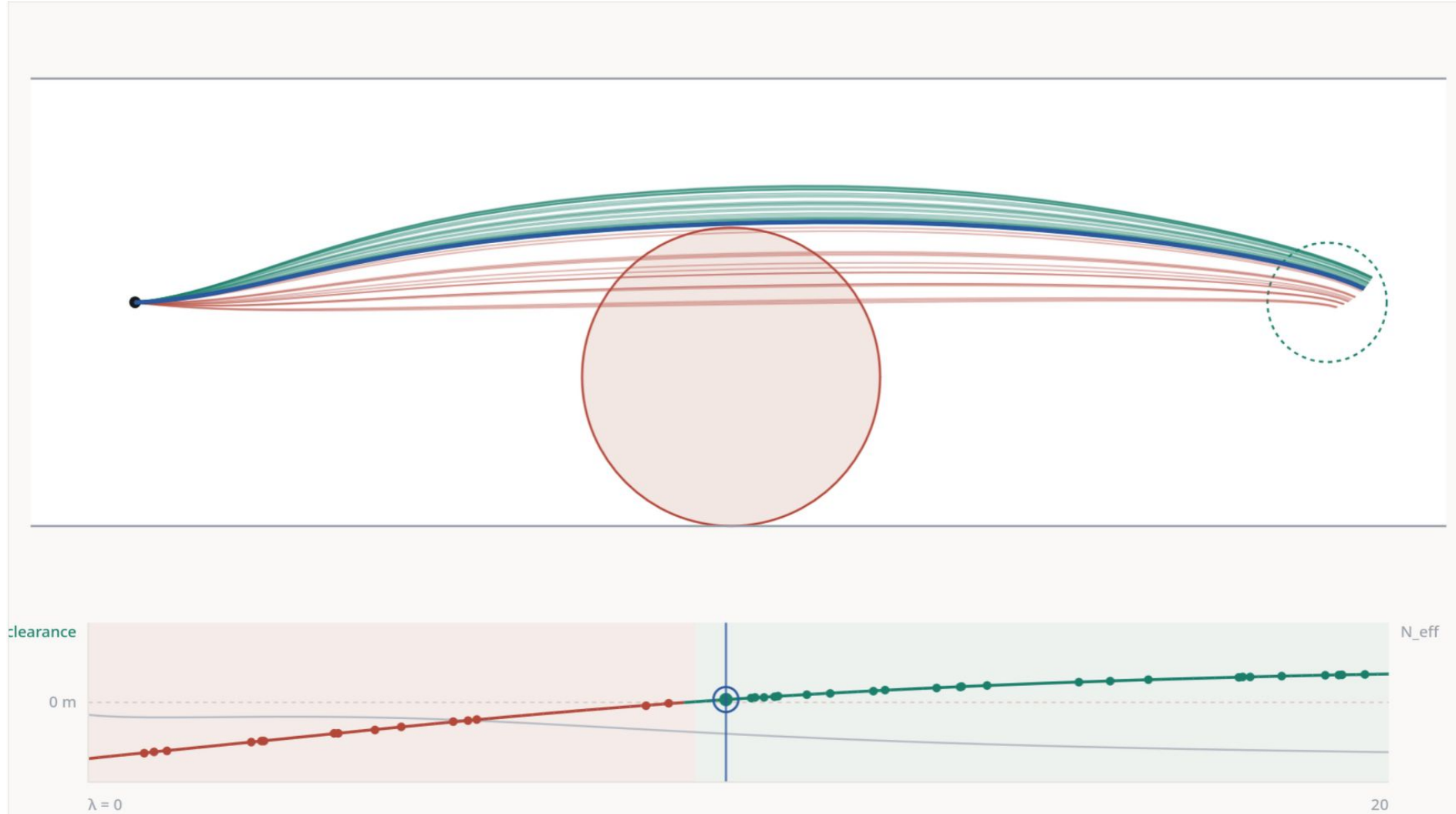
- Of feasible solutions, select one that minimizes the cost

$$\operatorname{argmin}_{\lambda \in \mathcal{F}_t^l} \sum_{k=t}^{t+T-1} \left(r \left(x_{k|t}^{l_r}, u_{k|t}^{l_r} \right) + \lambda_{\mathcal{X}} d_{\mathcal{X}}(x_{k|t}^{l_r}) \right) + V^{l_r} \left(x_{t+T|t}^{l_r} \right) + \lambda_{\mathcal{S}^l} d_{\mathcal{S}^l} \left(x_{t+T|t}^{l_r} \right)$$

- Fully parallelizable on the GPU while balancing optimality and safety

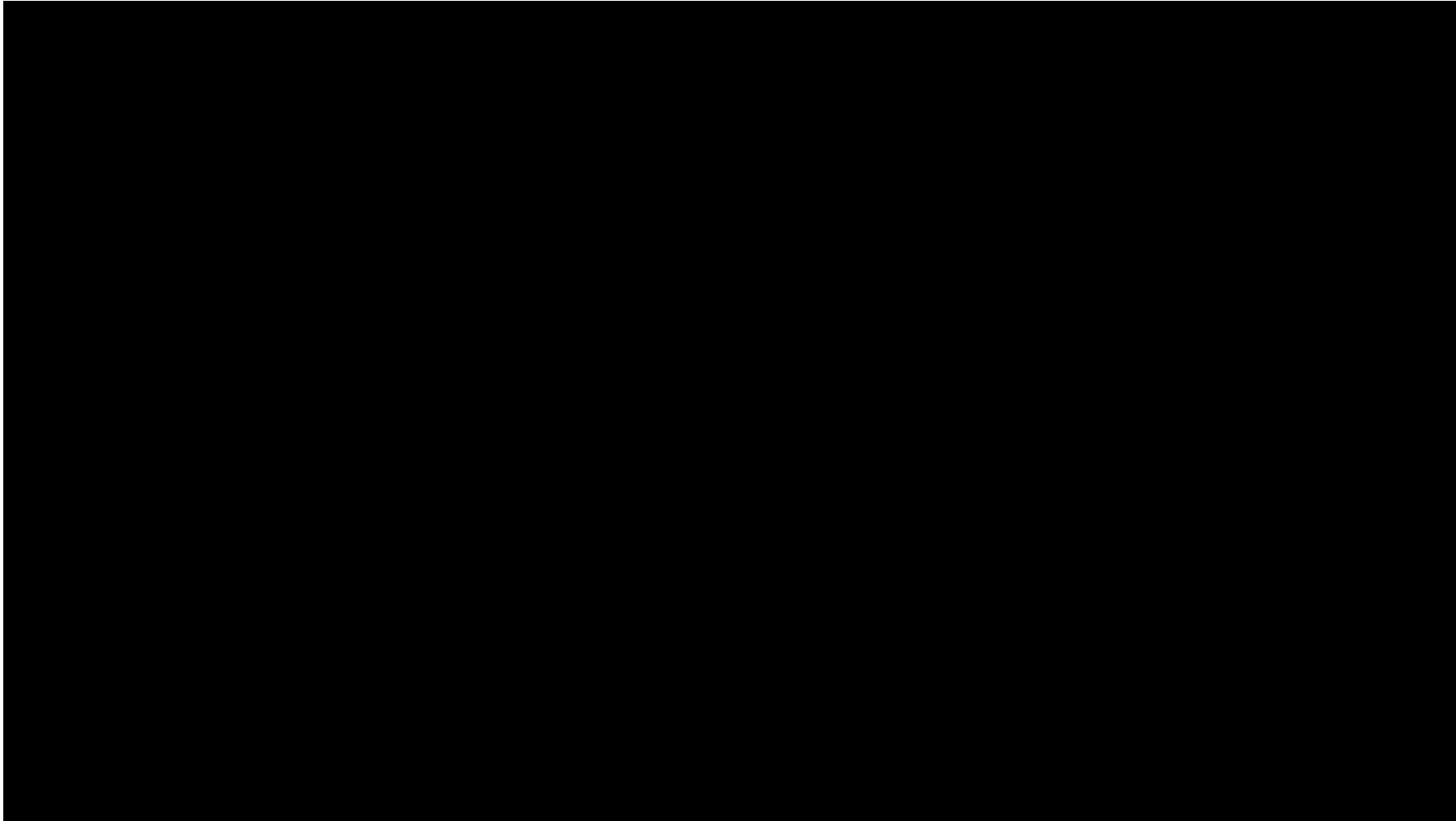
AP-MPPI in Action

- Idea: sample $\lambda \in \Lambda \triangleq [0, \lambda_{\mathcal{X}}^{\max}] \times [0, \lambda_{\mathcal{CS}^{l-1}}^{\max}]$ solve P-MPPI problems in parallel



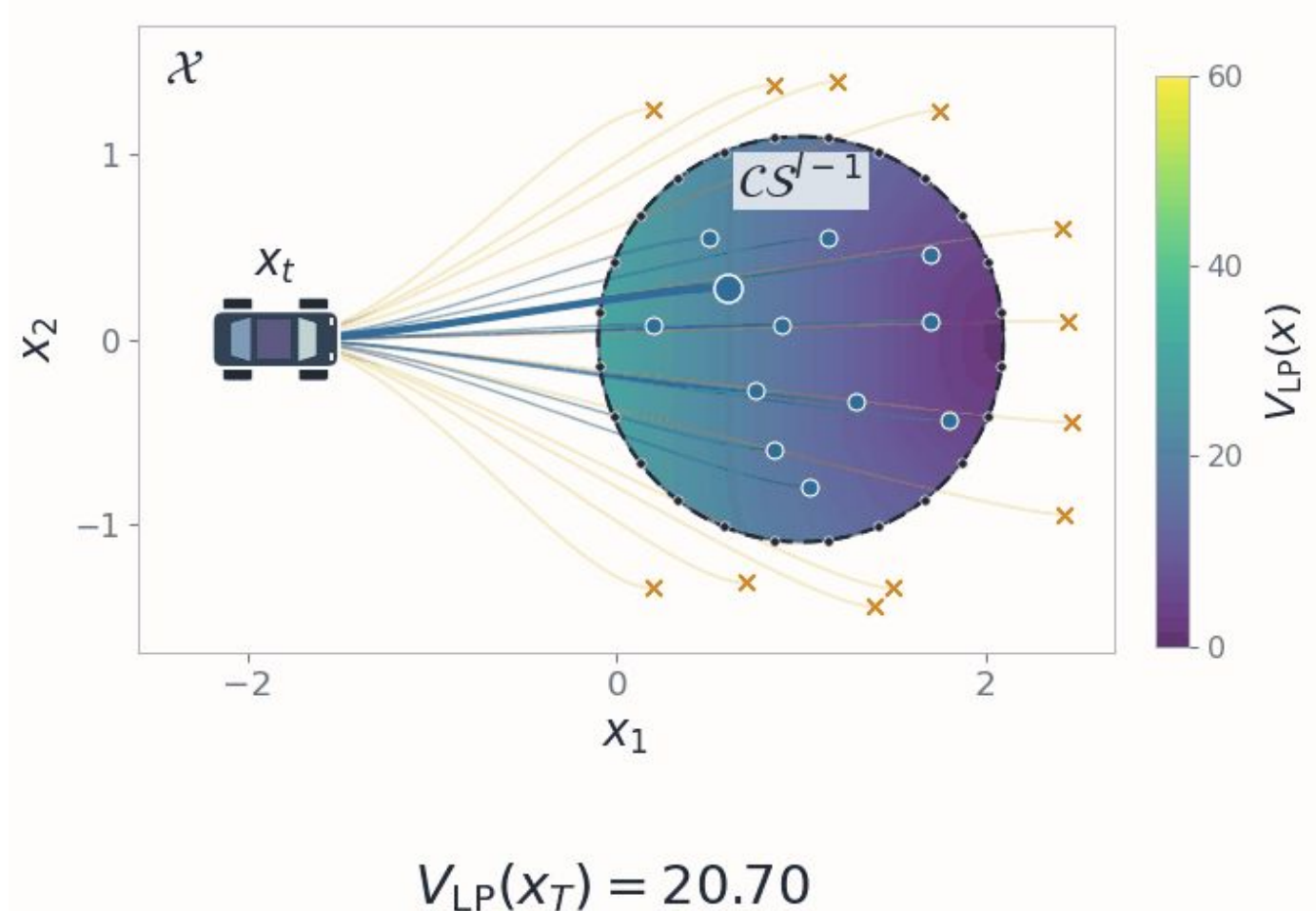
Adaptive Penalty Parameters

- Visualizing the individual costs and resulting trajectory



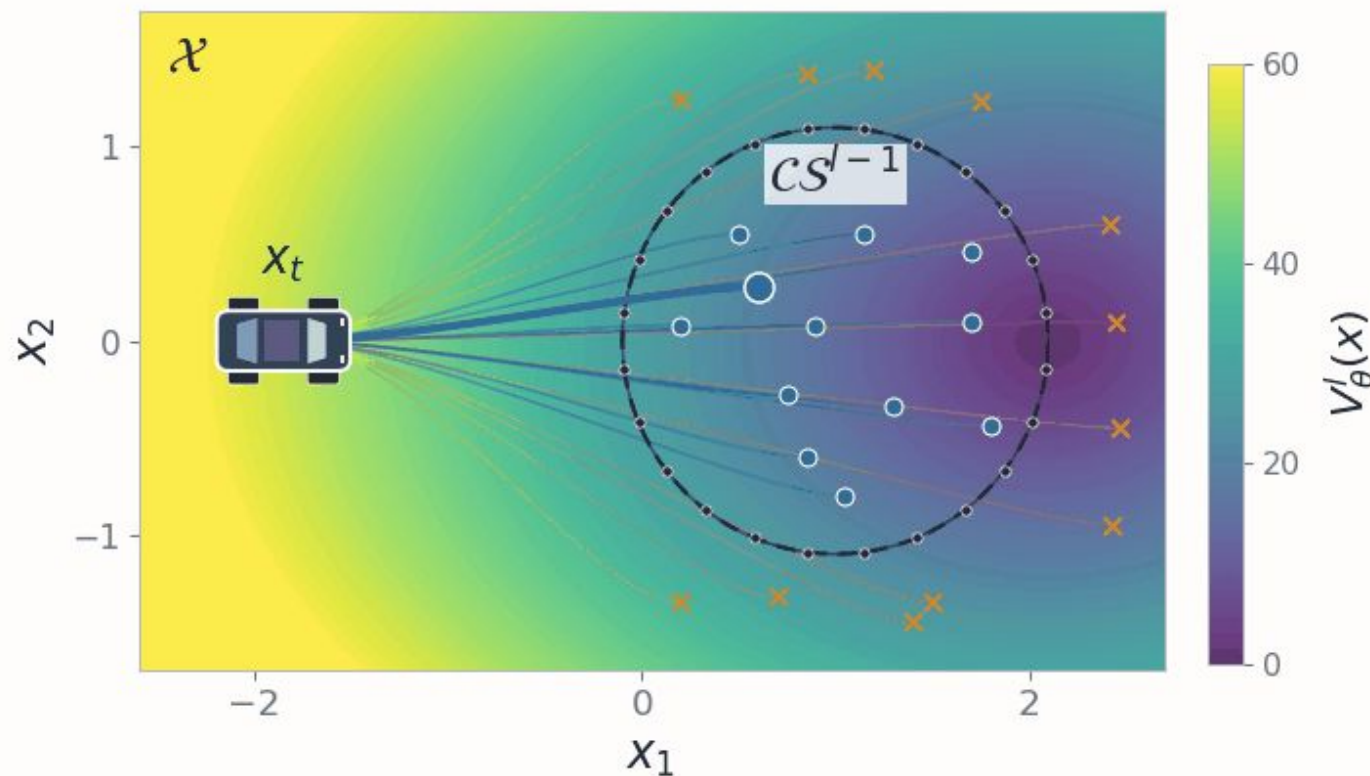
Cost-to-go of Sampled Trajectories

- How do we evaluate $V^{l_r}(x, u^{l_r}(\cdot))$ for sampled trajectories?
 - Can't use convex combination of values in the safe-set



Cost-to-go of Sampled Trajectories

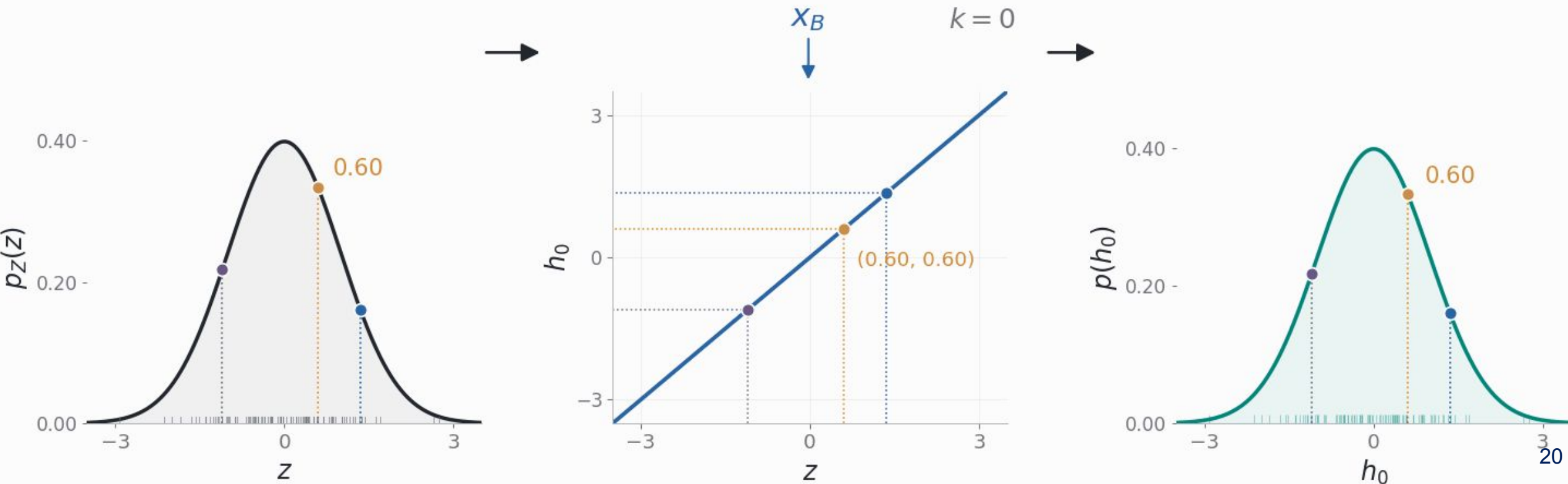
- We need to extrapolate $V^{l_r}(x, u^{l_r}(\cdot))$ outside the safe-set



$$\begin{aligned} & V_\theta^l(x_T) + 10 d_{CS^{l-1}}(x_T) \\ & \approx 20.24 + 10(0.000) \approx 20.24 \end{aligned}$$

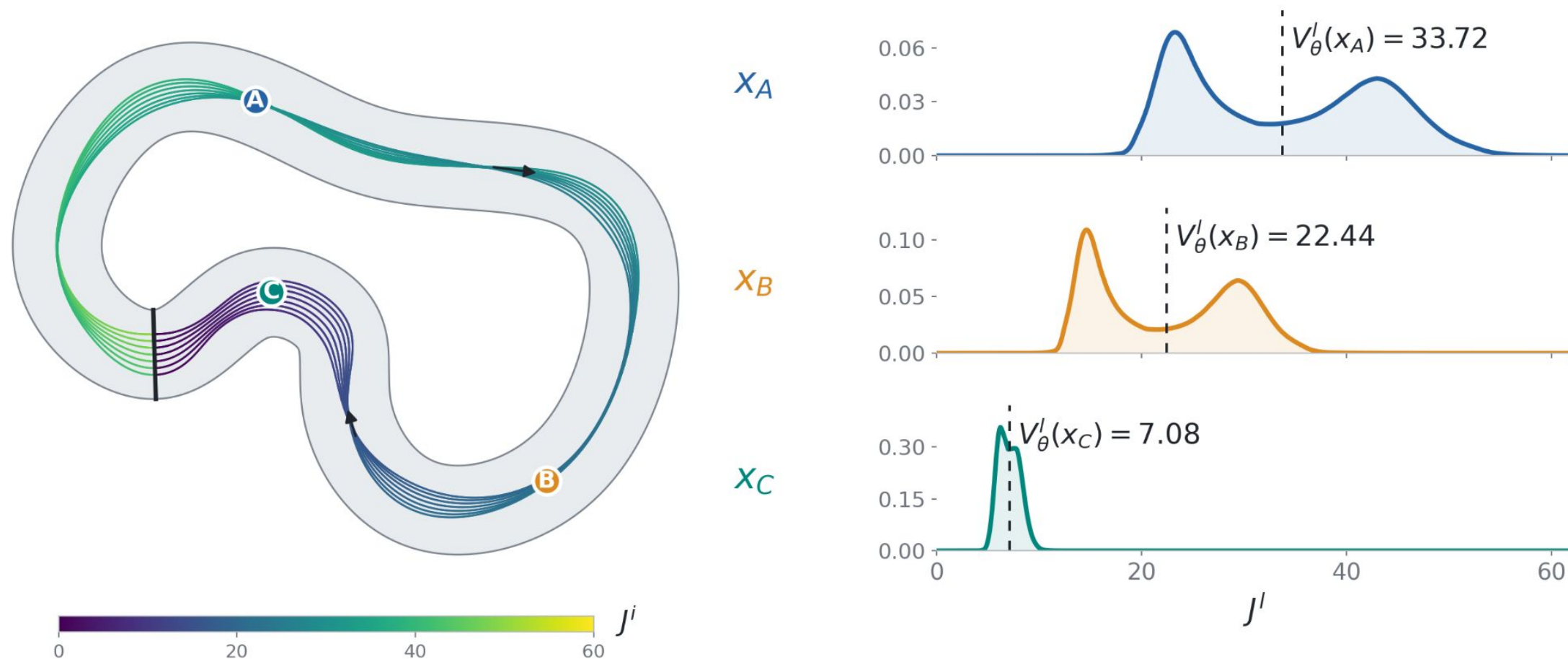
Value Function Learning

- Due to stochasticity of the system, $V^{l_r}(x, u^{l_r}(\cdot))$ is a random variable
 - Learn the distribution using a Normalizing Flow (NF)
 - Richer modelling of distribution compared to gaussian priors



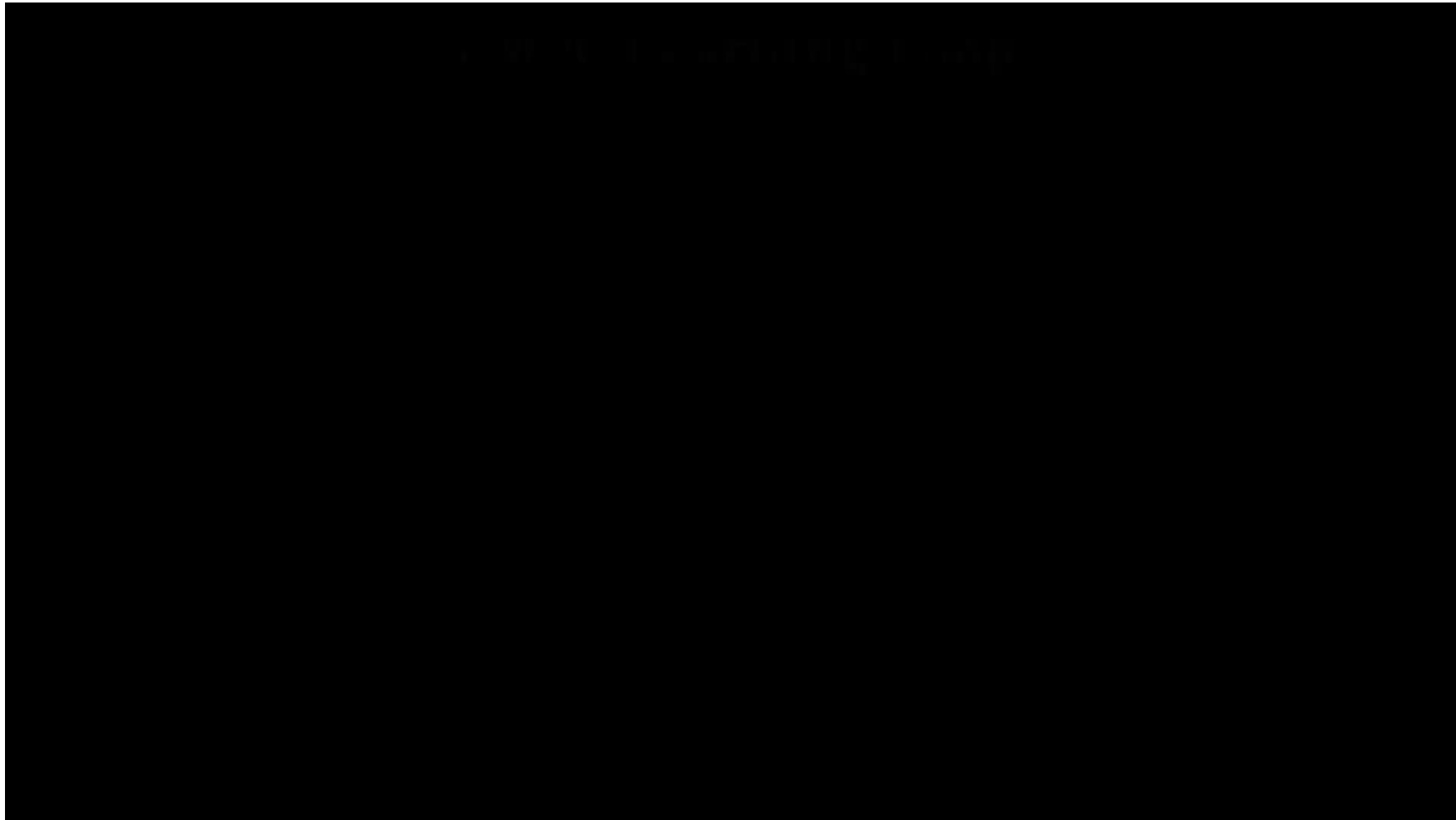
Learning - Visualized

- Due to stochasticity of the system, $V^{l_r}(x, u^{l_r}(\cdot))$ is a random variable
 - Learn the distribution using a Normalizing Flow (NF)



Learning - Visualized

- Visualizing the learnt value function



Experiments

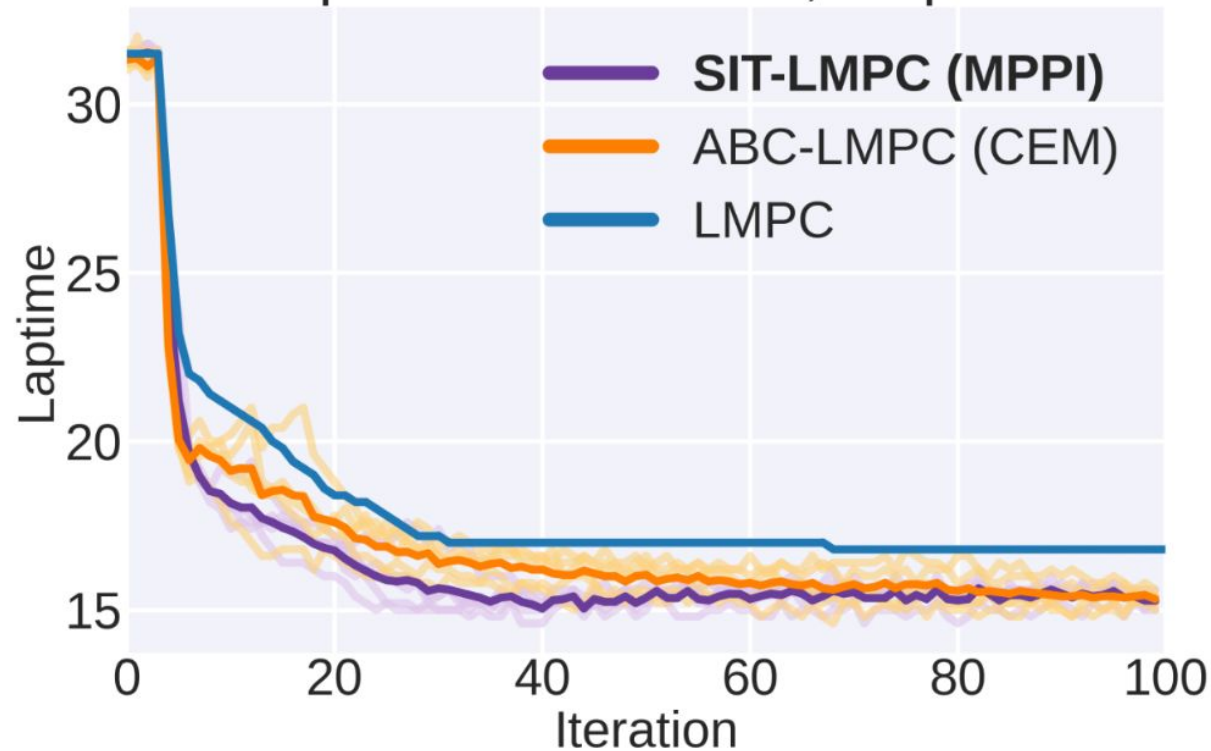
We compare with two baseline LMPC Algorithms

	System Type	Optimization Method	Constraint Enforcement
LMPC	Deterministic	Gradient-based Solver	Solver Enforced
ABC-LMPC	Stochastic	CEM Sampling	Constant Penalty Parameter
SIT-LMPC (our method)	Stochastic	Importance Sampling (MPPI)	Online Adaptive Penalty Parameter

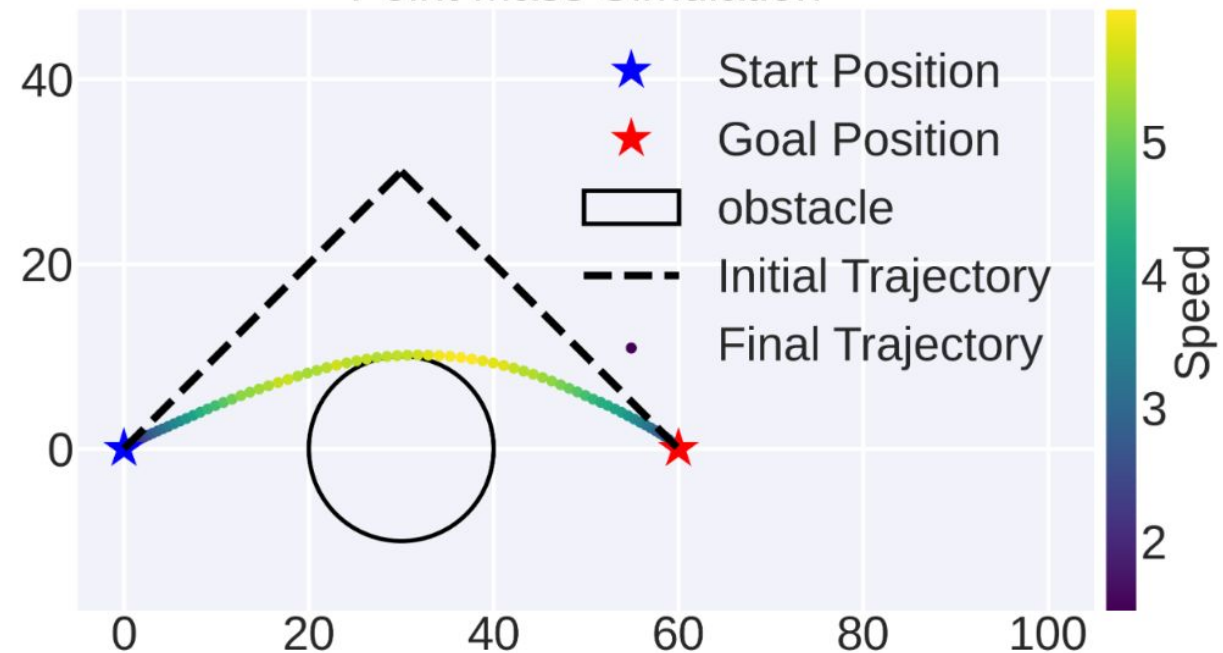
SIT-LMPC Sim Results

Outperforms Gradient-Based LMPC even for linear systems

Laptime vs. Iteration, Map 4

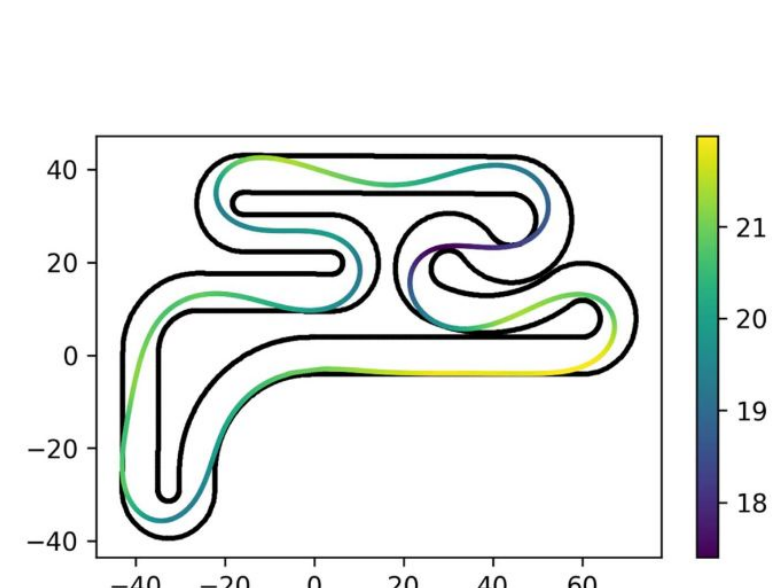
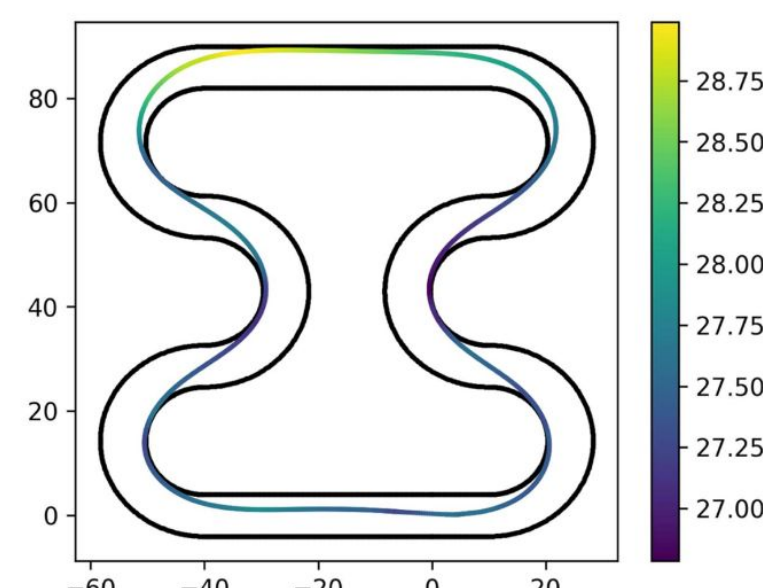
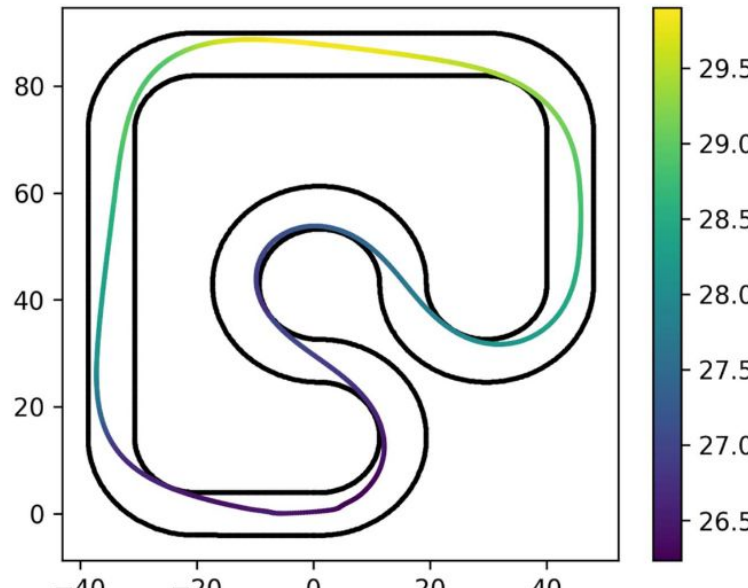
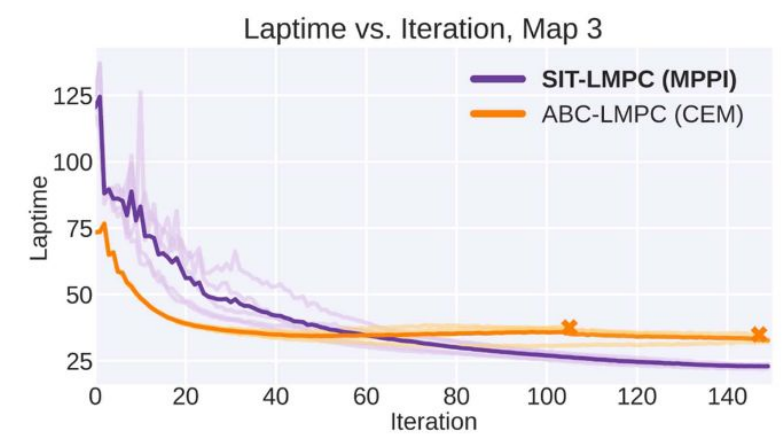
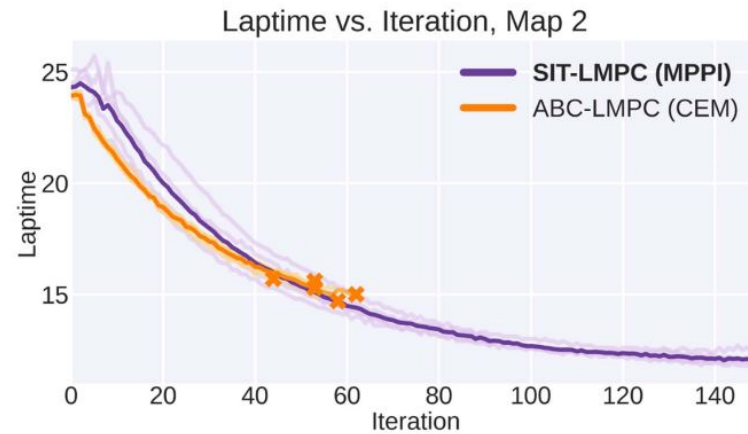
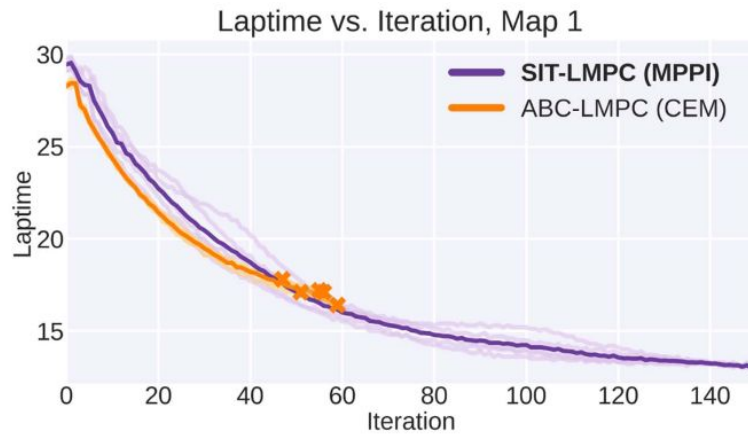


Point Mass Simulation



SIT-LMPC Sim Results - Continued

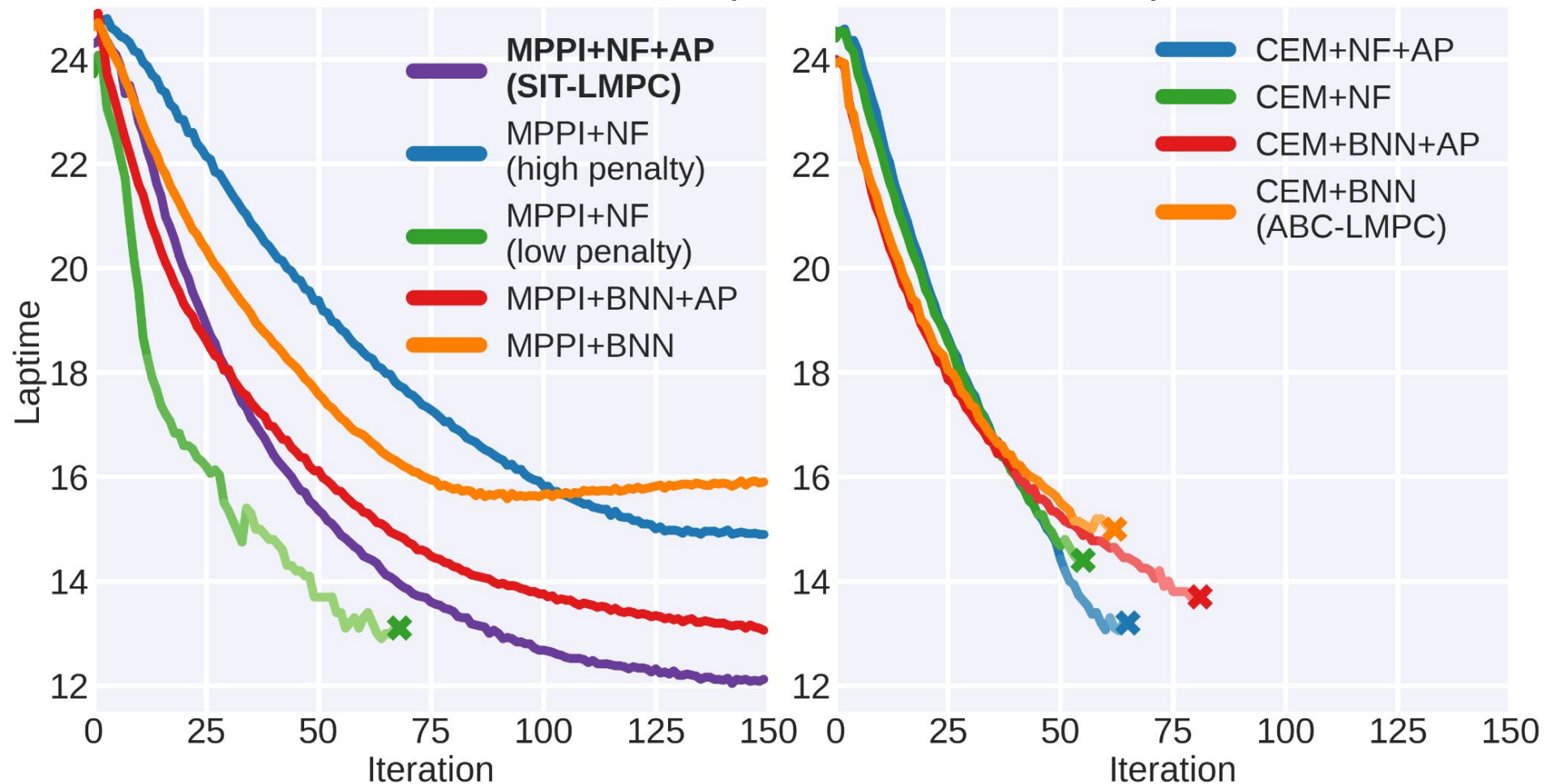
Outperforms CEM-Sampling LMPC



Ablation Studies

Outperforms all other variants

Ablation Studies, Laptime vs. Iteration, Map 2



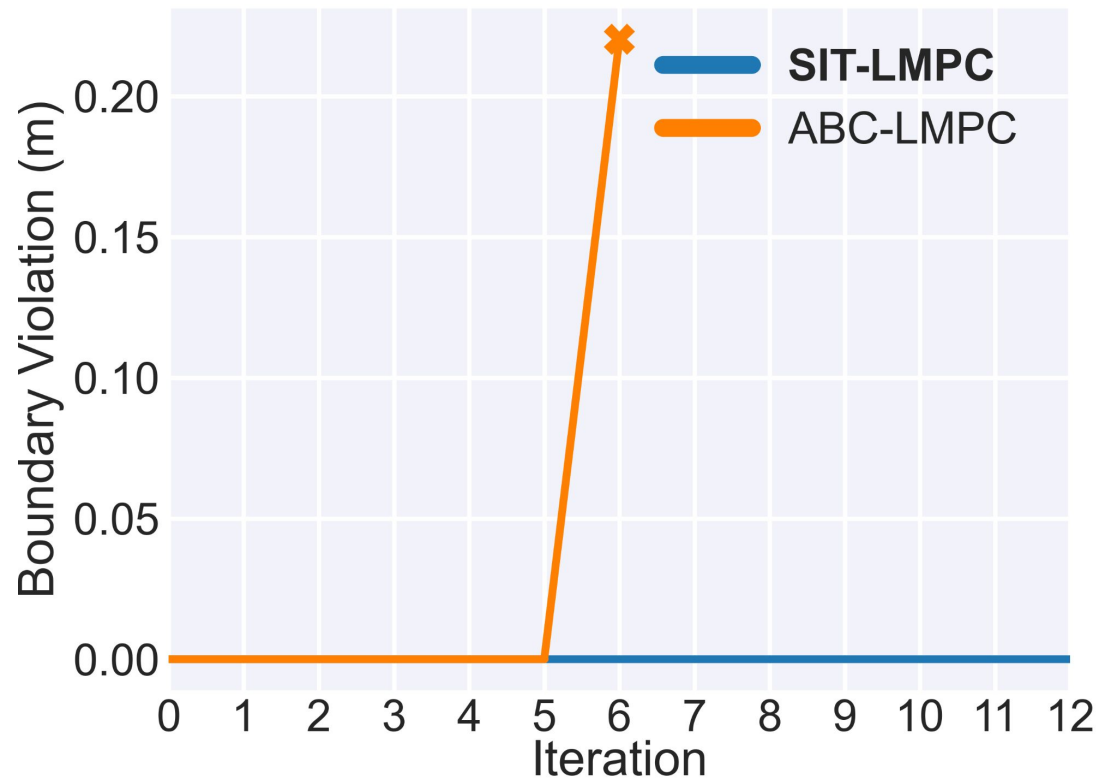
SIT-LMPC - Real World Experiment

GNSS Localization



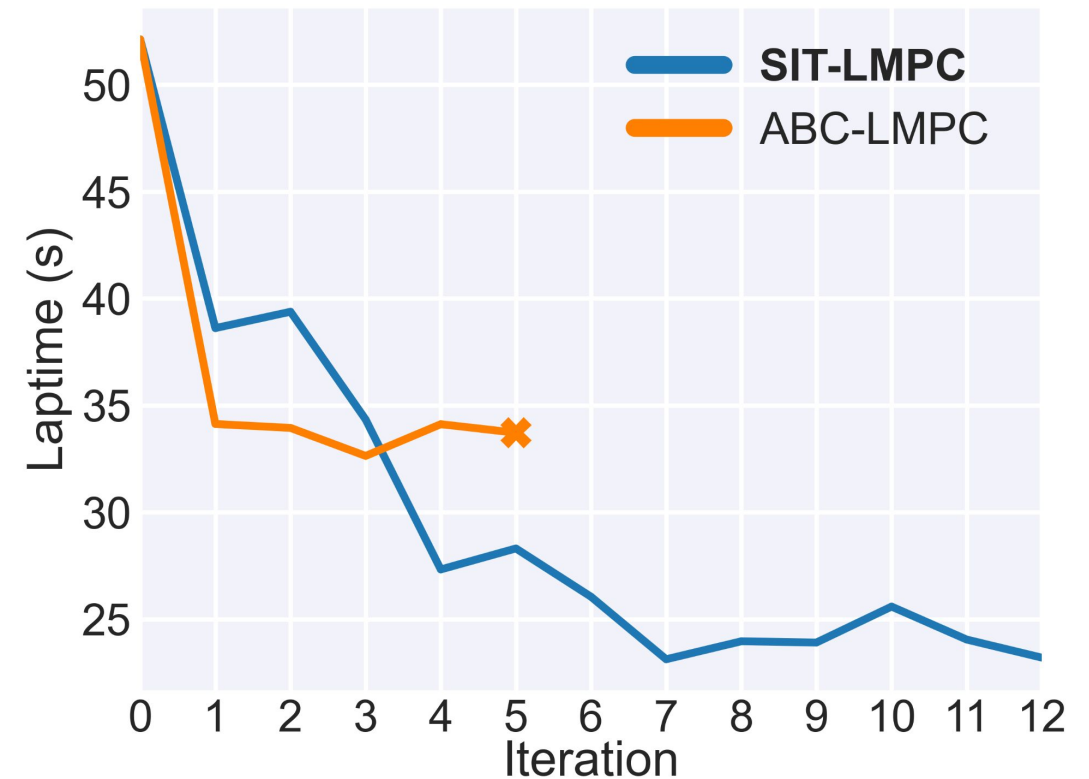
SIT-LMPC - Experimental Results

Boundary Violation vs. Iteration, Real World



Safe

Laptime vs Iteration, Real World



Performant

ABC-LMPC - Experimental Results

SIT-LMPC
Real-World Experiments

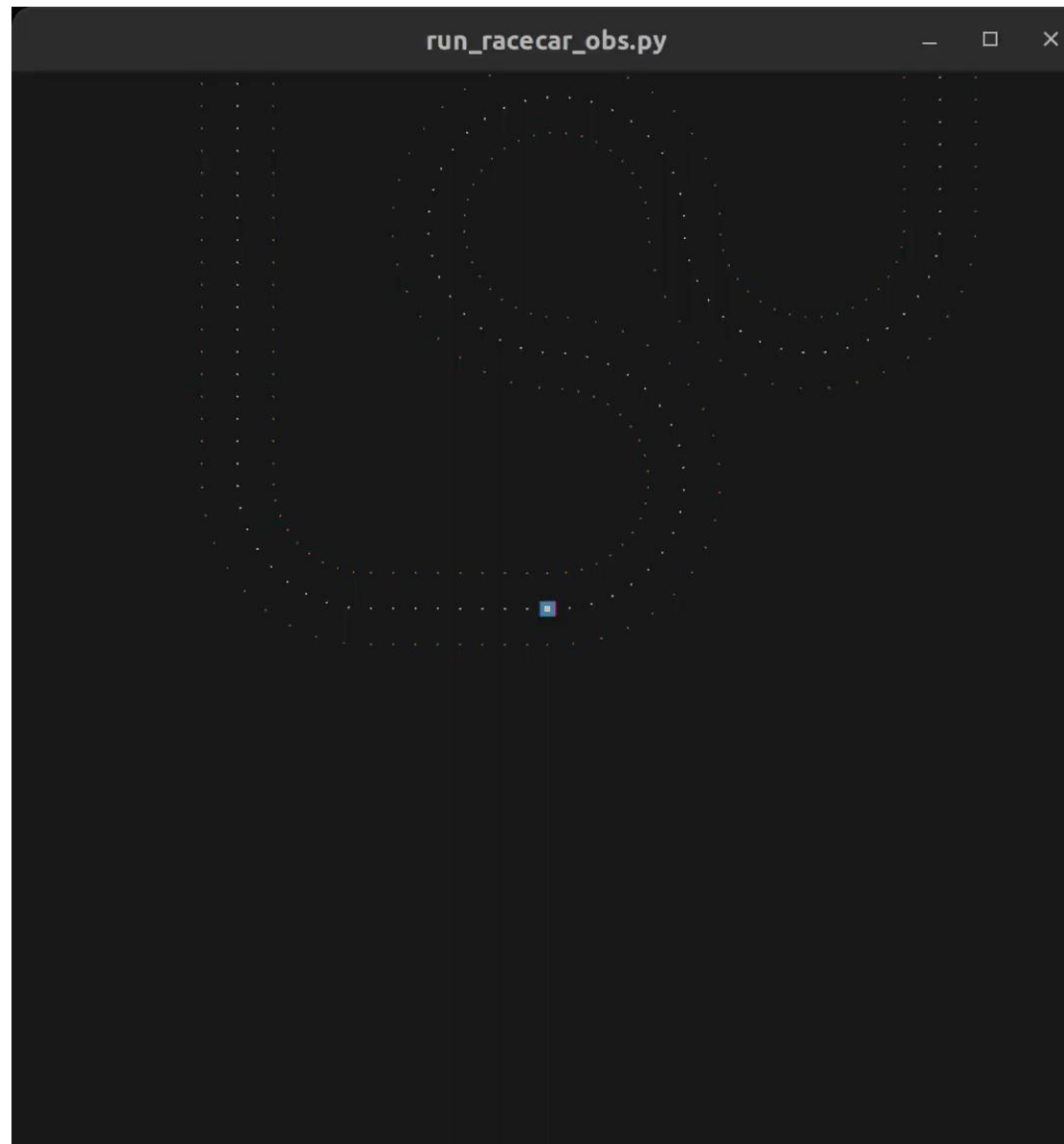
SIT-LMPC - Experimental Results

SIT-LMPC
Real-World Experiments

SIT-LMPC - Numerical Results

SIT-LMPC
Real-World Experiments

Bonus - Obstacle Avoidance



SIT-LMPC Assumption

- SIT-LMPC assumes knowledge of model's dynamics

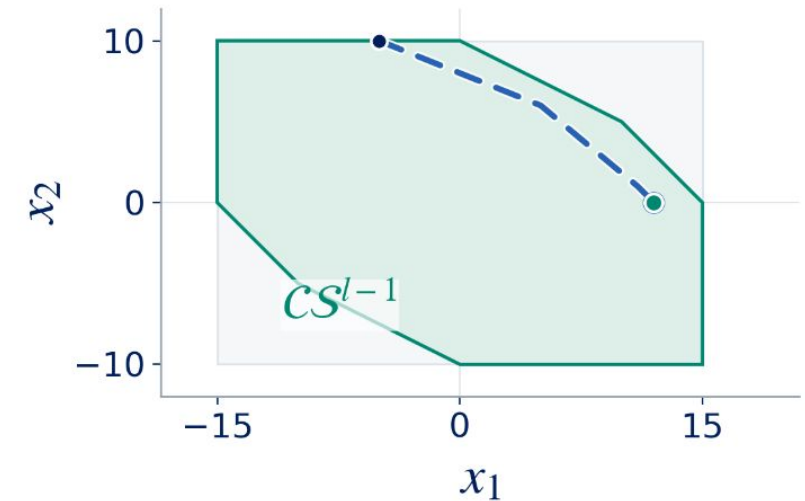
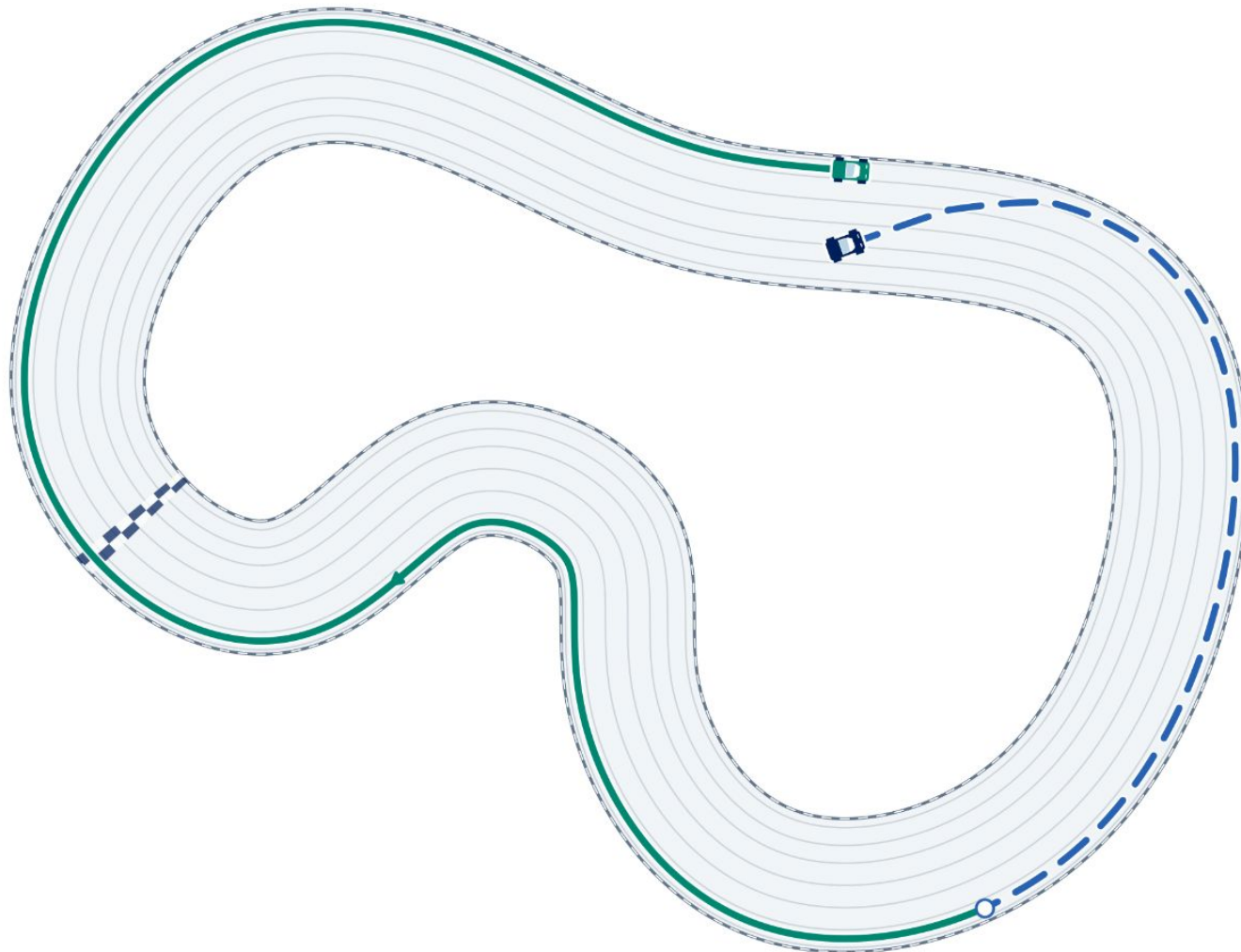
$$\begin{aligned} J_{\text{SIT-LMPC}}^{l_r} \left(t, x^{l_r}(t), u_{\cdot|t}^{l_r} \right) \\ \triangleq \min_{u_{\cdot|t}^{l_r}} \mathbb{E} \left[\sum_{k=t}^{t+T-1} \left(r \left(x_{k|t}^{l_r}, u_{k|t}^{l_r} \right) + \lambda_{\mathcal{X}} d_{\mathcal{X}}(x_{k|t}^{l_r}) \right) + V^{l_r} \left(x_{t+T|t}^{l_r} \right) + \lambda_{\mathcal{S}^l} d_{\mathcal{S}^l} \left(x_{t+T|t}^{l_r} \right) \right] \\ \text{subject to } \boxed{x_{k+1|t}^{l_r} = f \left(x_{k|t}^{l_r}, u_{k|t}^{l_r}, w^{l_r}(k) \right)}, \quad k \in [t, \dots, t+T-1], \\ x_{t|t}^{l_r} = x^{l_r}(t) \\ u_{k|t}^{l_r} \in \mathcal{U}, \quad k \in [t, \dots, t+T-1]. \end{aligned}$$

- **Key question:** What happens when we do not know the dynamics?
 - We resort to using estimates of the dynamics

$$\hat{x}_{k+1|t}^l = \hat{f} \left(\hat{x}_{k|t}^l, u_{k|t}^l, w^l(k) \right),$$

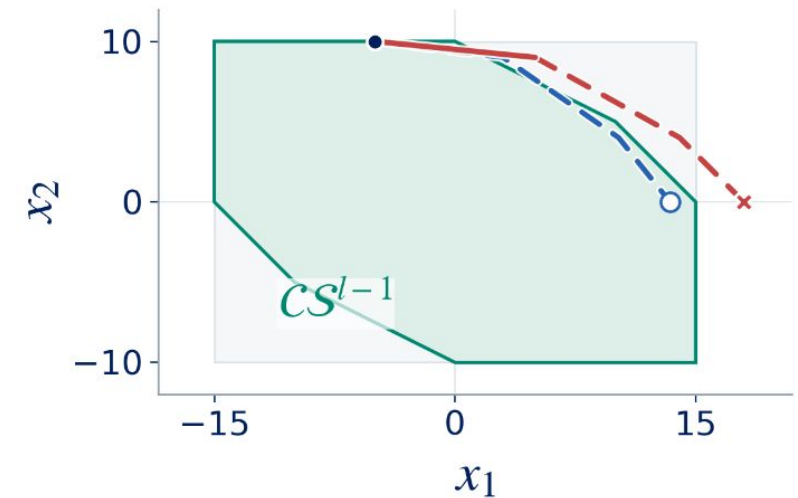
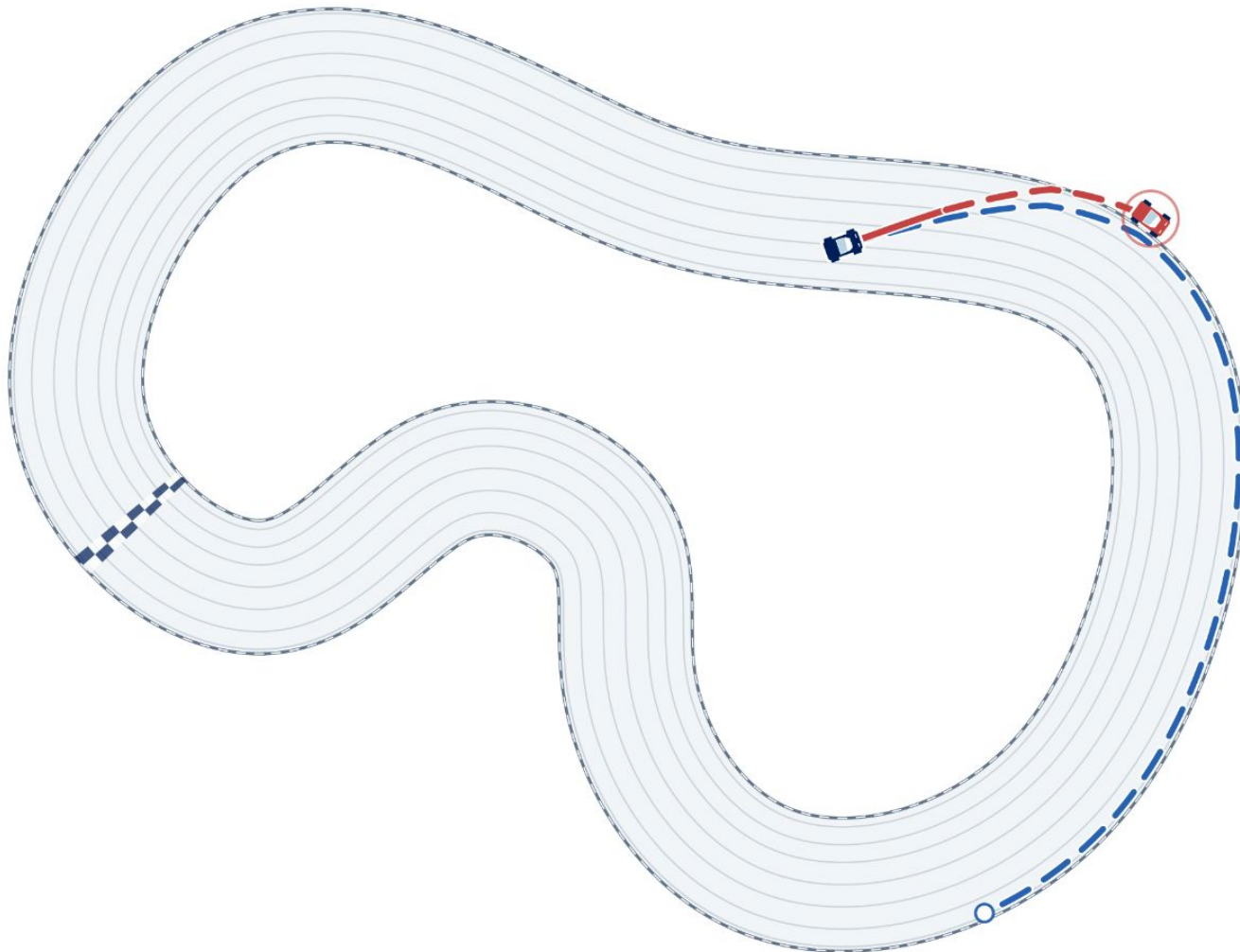
SIT-LMPC Assumption

- Will solve for controls that terminate in the safe-set given the model



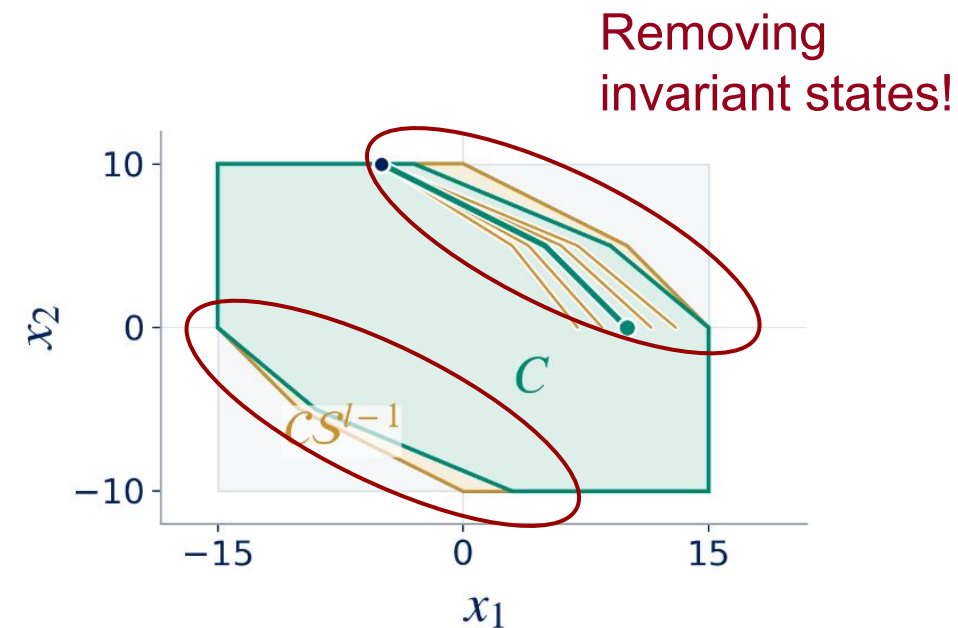
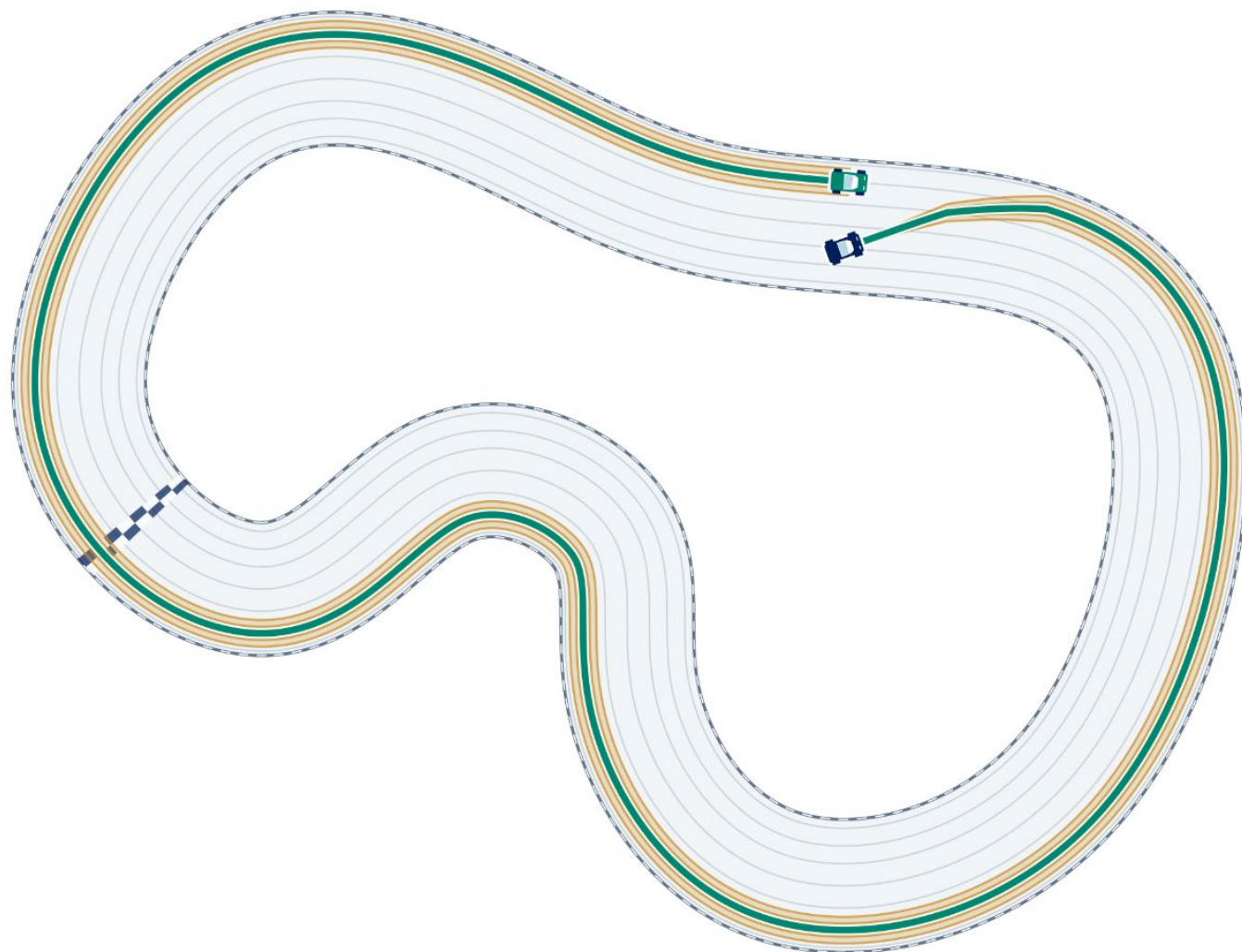
Model Mismatch

- Model mismatch results in a realized trajectory that violates constraints



Parametric Uncertainty

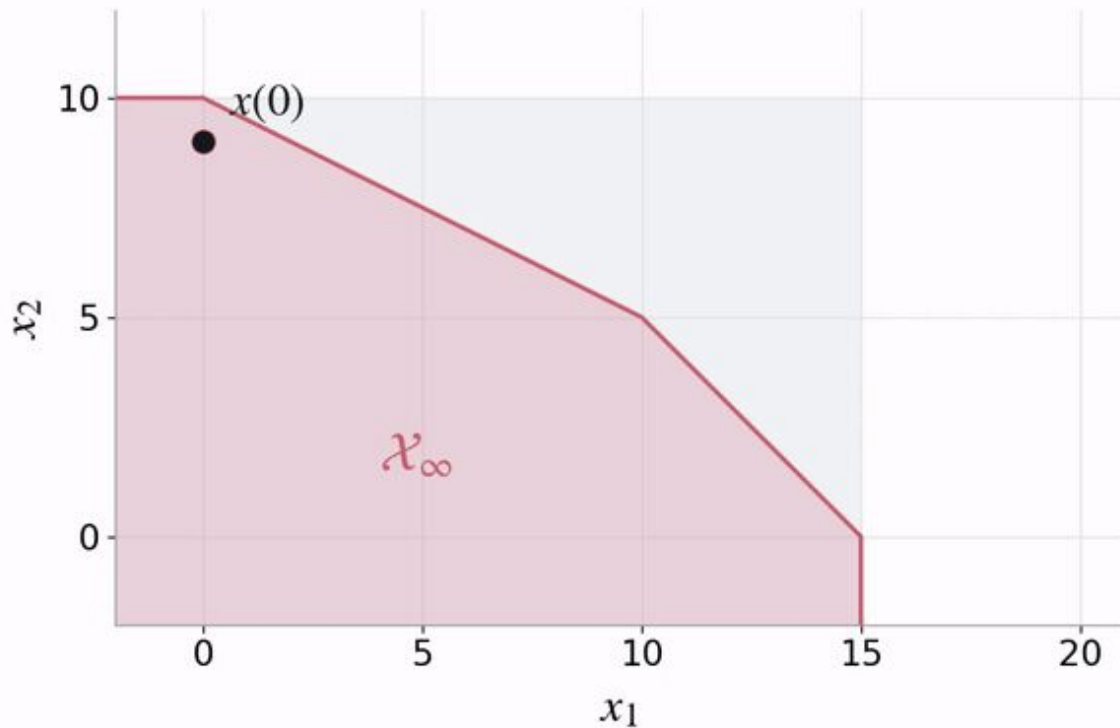
- Potential remedy is to further robustify against parametric uncertainty



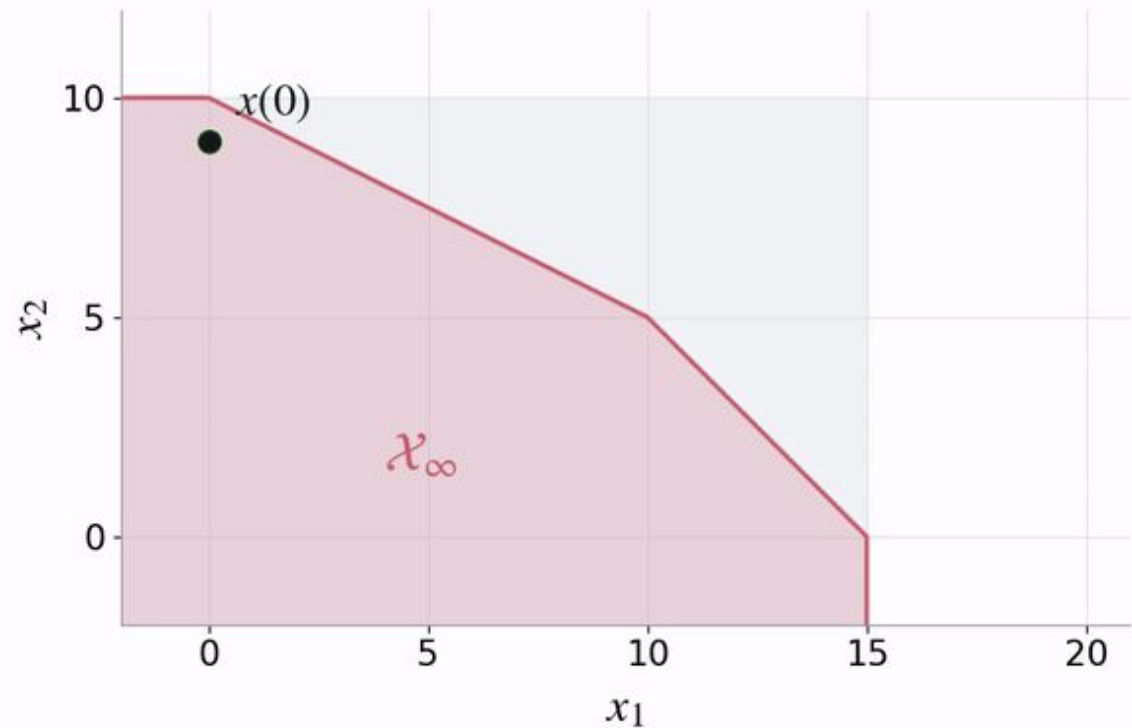
Ambiguity of Failure

- Can we classify “unsafe” set of states that lead to constraint violation?
 - Ambiguous – for the same state, some actions are safe, others no
 - Need a way to distinguish between states **and** controls

$$u(0) = -3$$

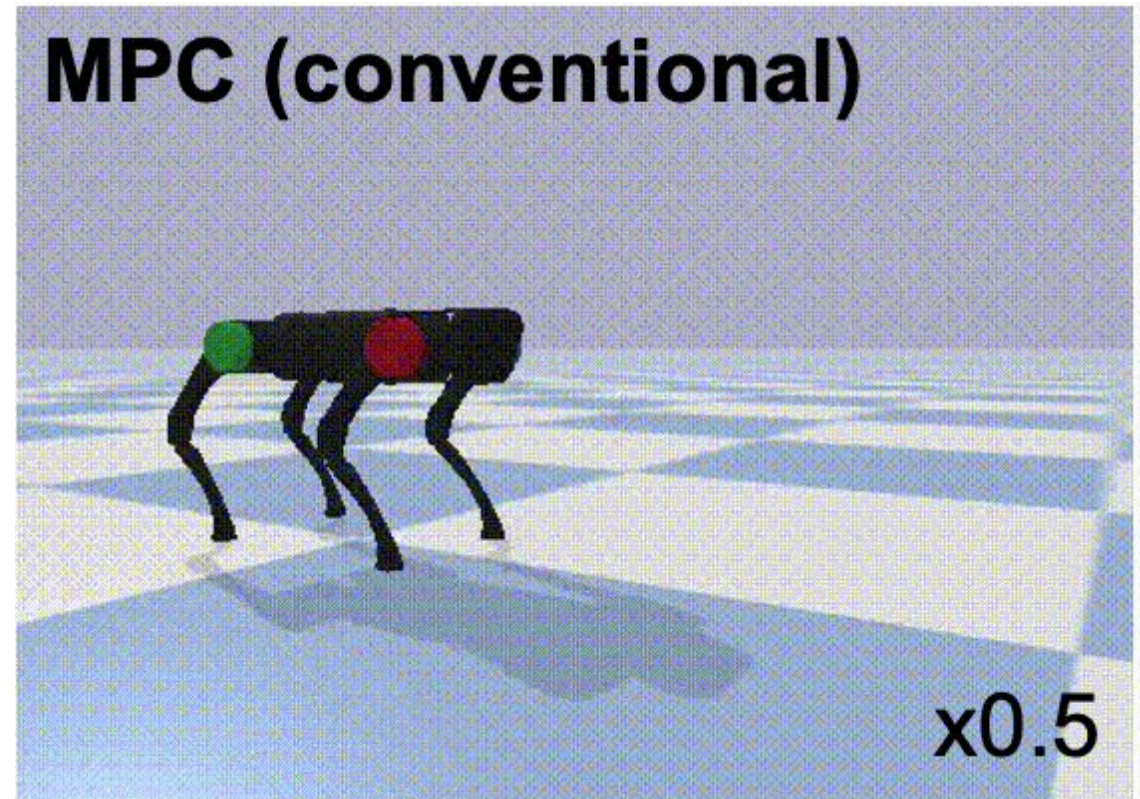


$$u(0) = -5$$



Failure

- The result can be ugly...



- Can we learn to avoid repeating these *failures*?



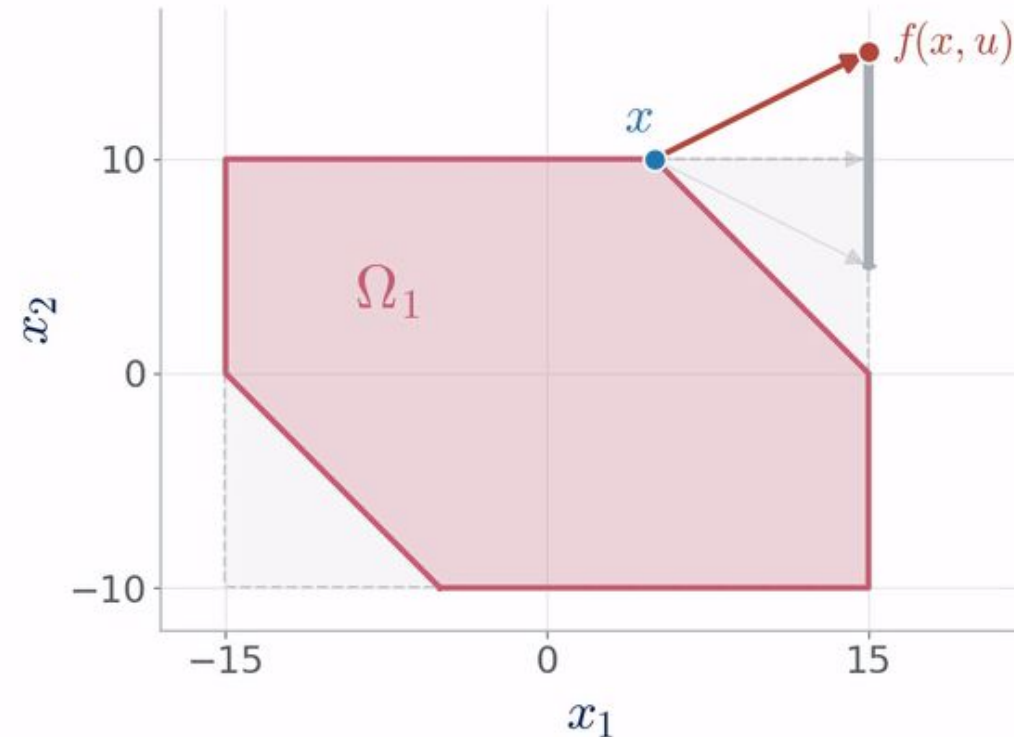
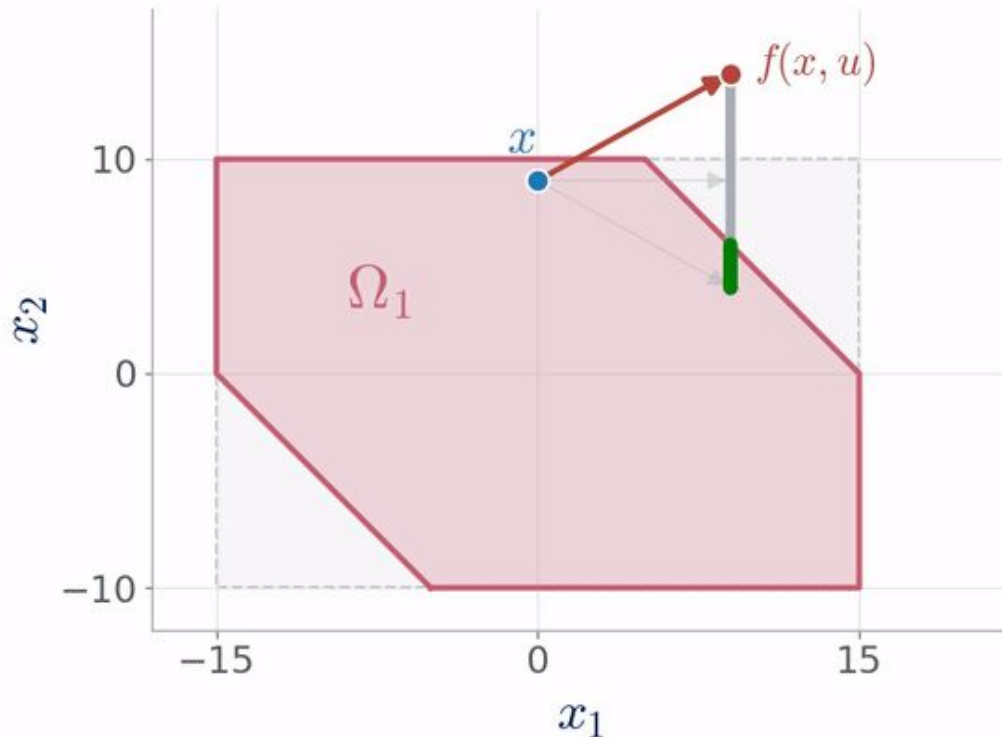
Failure-Aware Iterative Learning of State-Control Invariant Sets

Ahmad Amine, Nick-Marios T. Kokolakis, Truong X. Nghiem, Ugo Rosolia, and Rahul Mangharam

Accepted to IEEE CDC 2026

A Look Back - Control Invariance

- A set $\mathcal{C} \subseteq \mathcal{X}$, is *control invariant* if $\forall x \in \mathcal{C}, \exists u \in \mathcal{U}(x)$ such that $f(x, u) \in \mathcal{C}$.

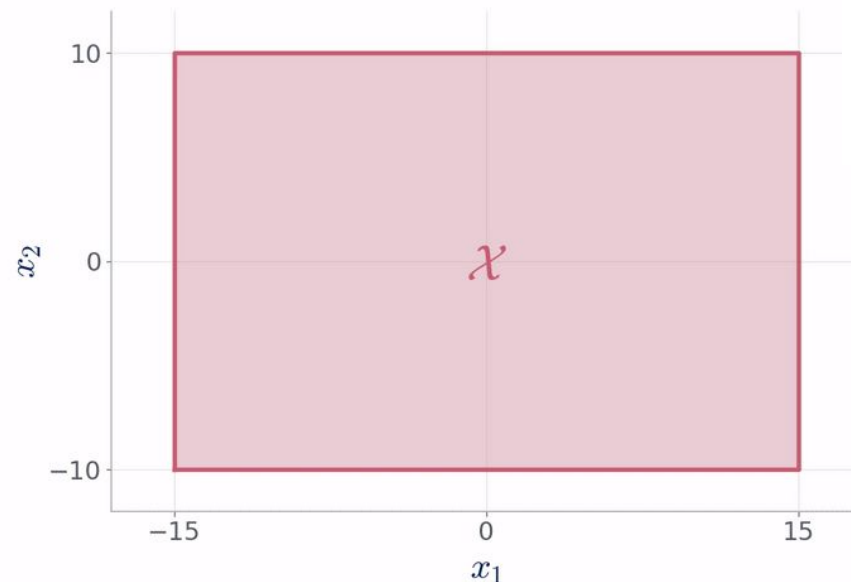


Maximal Control Invariant Set

- The Maximal Control Invariant Set (MCI) is the largest such set.
 - Can be computed by recursion with the predecessor operator:

$$\text{Pre}(\Omega) \triangleq \left\{ x \in \mathbb{R}^{n_x} : \exists u \in \mathcal{U}(x) \text{ s.t. } f(x, u) \in \Omega \right\}.$$

$$\Omega_{k+1} = \text{Pre}(\Omega_k) \cap \Omega_k, \quad k \in \bar{\mathbb{Z}}_+.$$



can only tell us that a control input exists, not which ones are safe!

Start from the state constraints

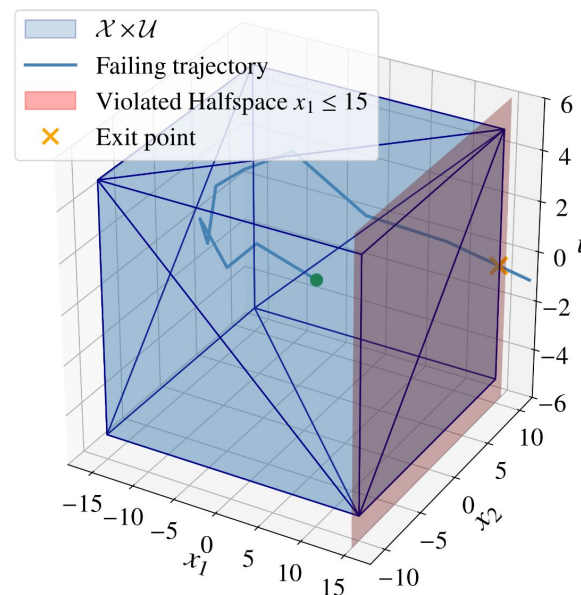
Repeat the predecessor calculation.

State-Control Safety

- Safe sets of states only state that a safe control input exists at a state
 - Need to quantify *which* control inputs are safe and which fail
- Consider constraints in state-control space

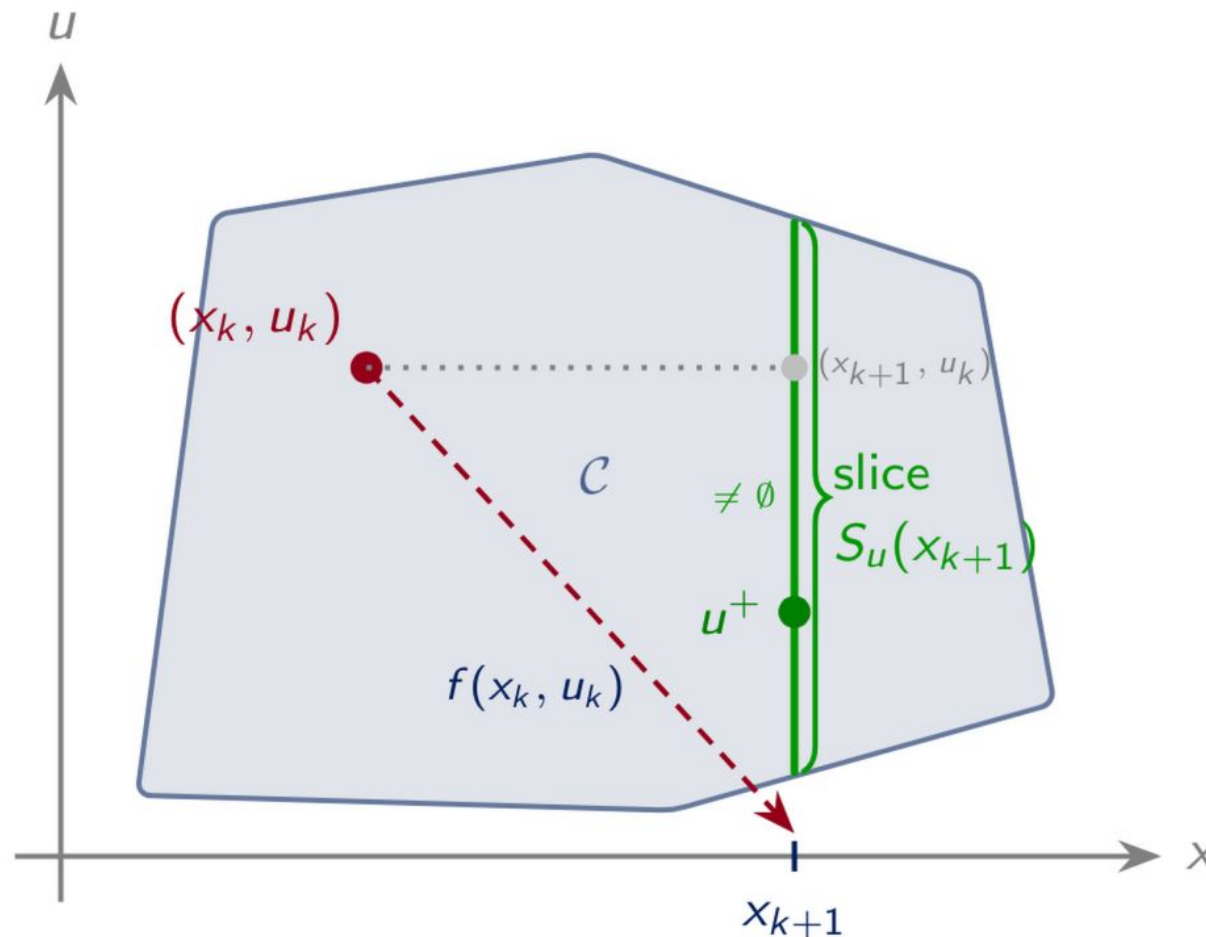
$$\mathcal{Z} \triangleq \{z \in \mathbb{R}^{n_x + n_u} : x \in \mathcal{X}, u \in \mathcal{U}(x)\}$$

- A *failure* is a state-control pair whose next state violates constraints



State-Control Invariance

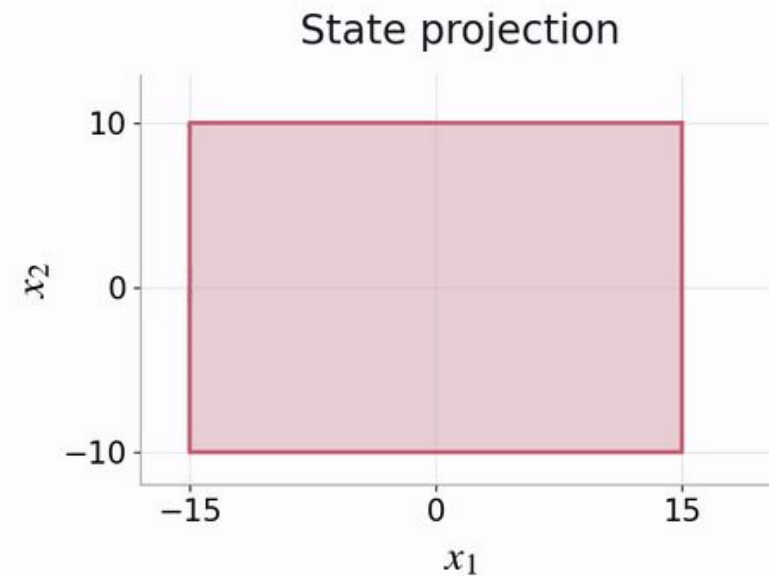
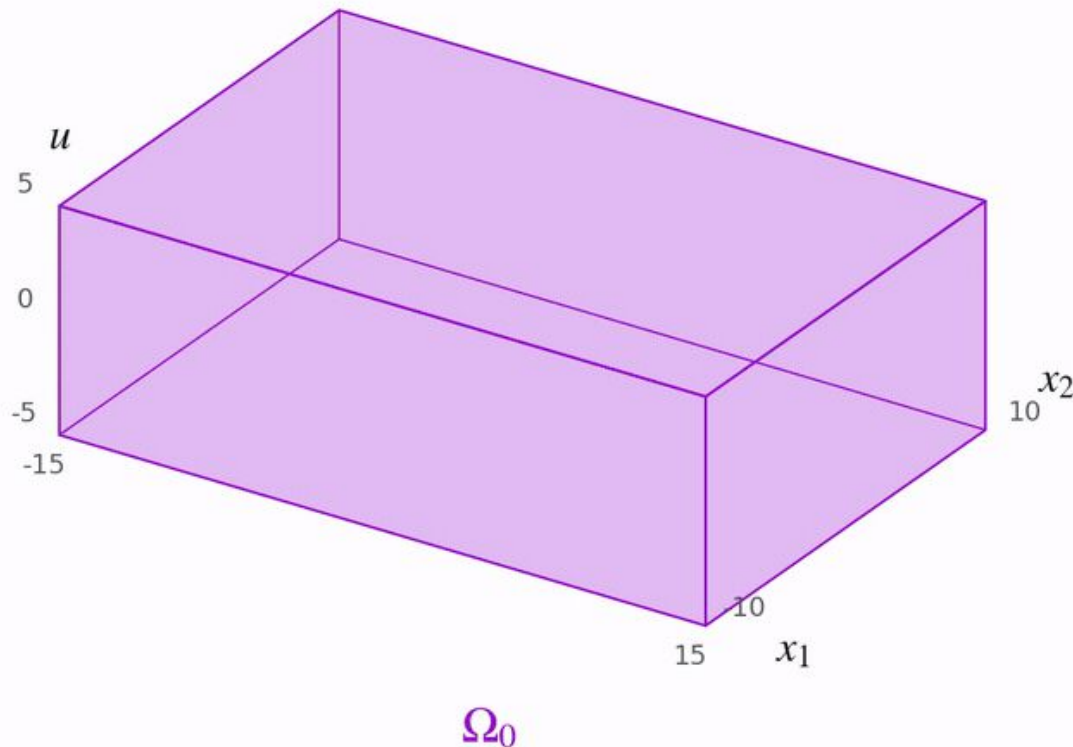
- A set $\mathcal{C} \subseteq \mathcal{Z}$ is *state-control invariant* if for every $z \in \mathcal{C}$, there exists a $u^+ \in \mathbb{R}^{n_u}$ such that $[f(x, u)^\top \quad (u^+)^\top]^\top \in \mathcal{C}$



Maximal State-Control Invariance

- The Maximal State-Control Invariant Set (MSCI) is the largest such set.
 - Can be computed by recursion with the predecessor operator:

$$\text{Pre}_z(\Omega) \triangleq \left\{ z \in \mathbb{R}^{n_x+n_u} : \exists u^+ \in \mathbb{R}^{n_u} \text{ s.t. } \begin{bmatrix} f(x, u)^\top & (u^+)^\top \end{bmatrix}^\top \in \Omega \right\},$$



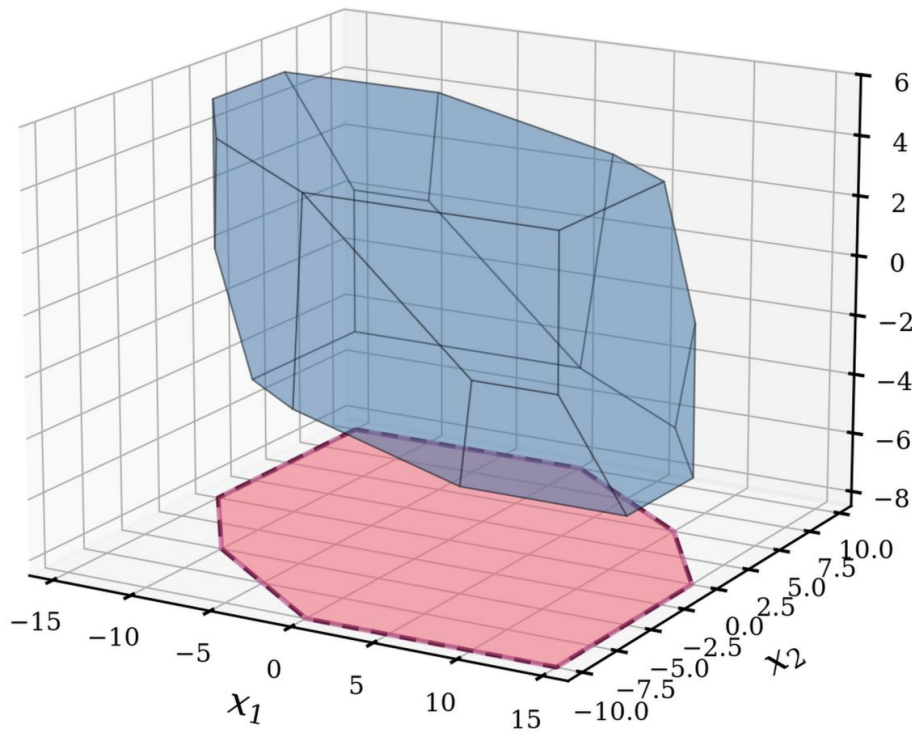
6 halfspaces

MSCI Includes the MCI and More

Safe states

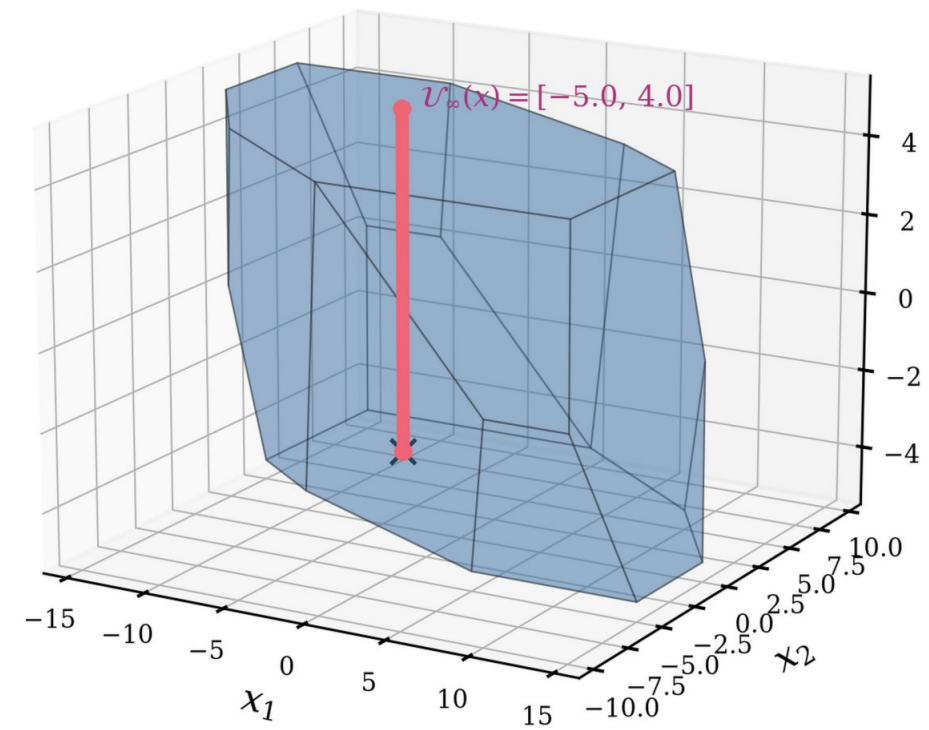
- Projecting away the control recovers the largest region from which safety is achievable (MCI).

$\mathcal{Z}_\infty$
 $\Pi_x(\mathcal{Z}_\infty)$
--- MPT3 $\mathcal{X}_\infty$



Safe inputs

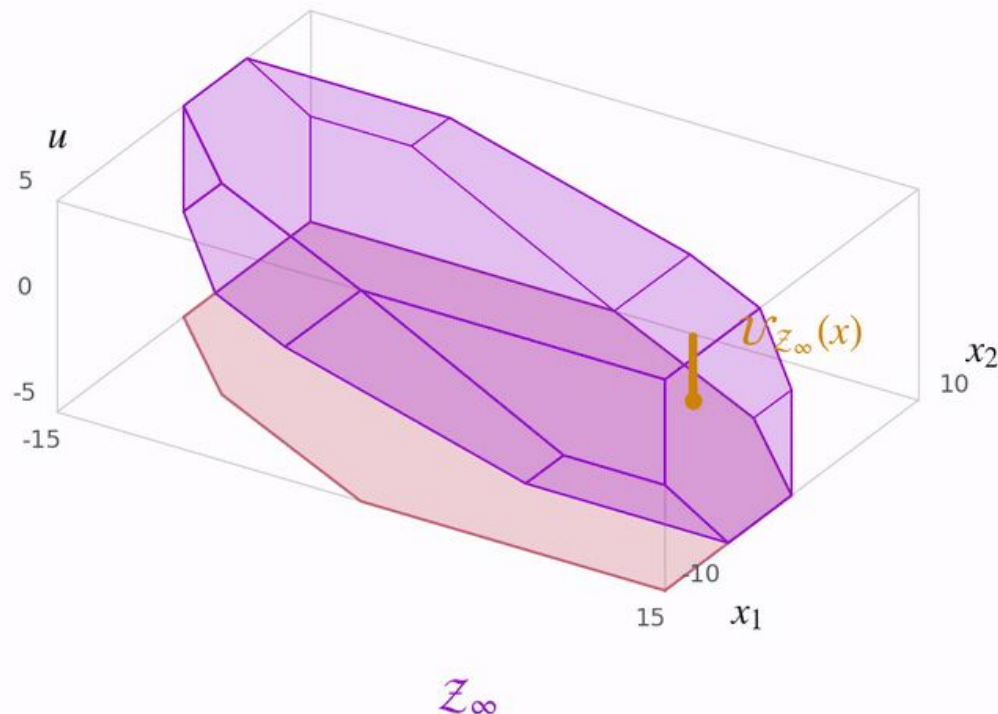
- At each safe state, the section lists every control that preserves invariance, not just that one exists.



MSCI Includes the MCI and More

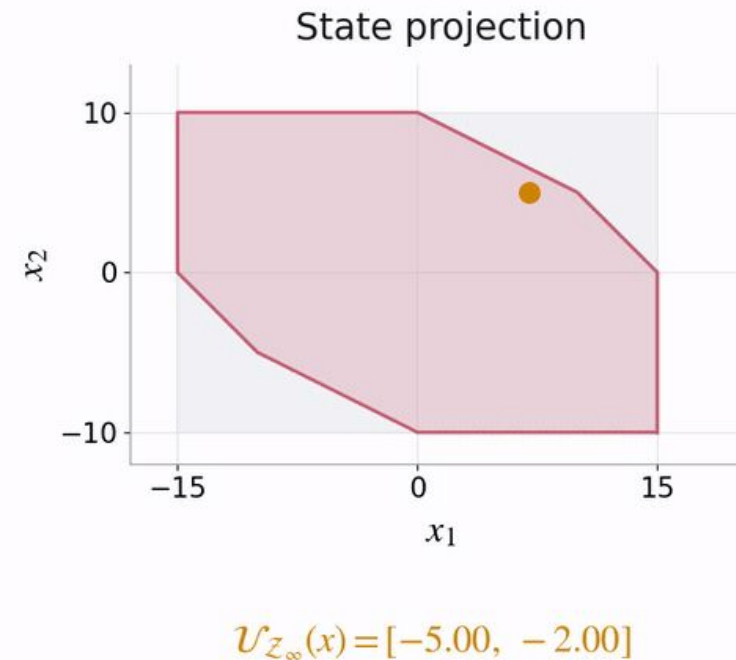
Safe states

- Projecting away the control recovers the largest region from which safety is achievable (MCI).



Safe inputs

- At each safe state, the section lists every control that preserves invariance, not just that one exists.

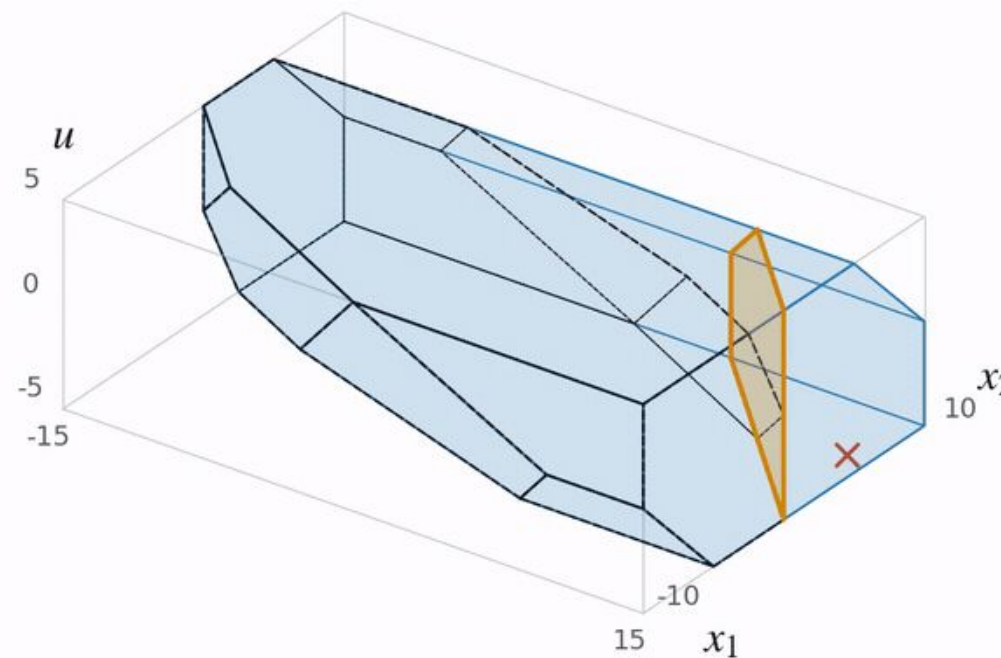
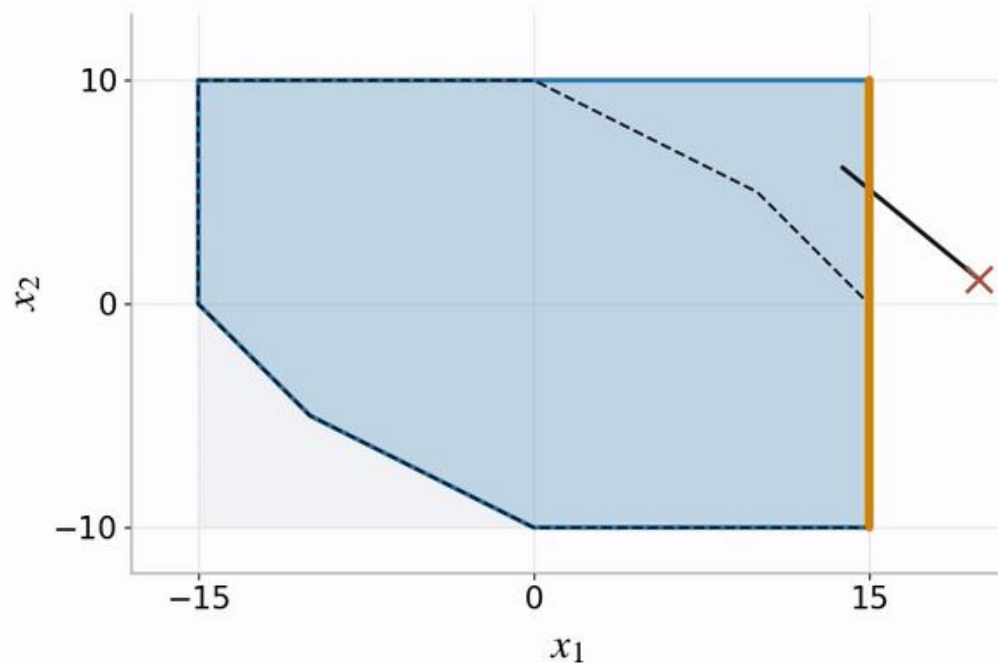


The state projection gives invariant states; sections give admissible invariance-preserving inputs.

Learning from Failure

- MSCl computation still requires a dynamics model $f(x_{i|k}, u_{i|k})$,
 - Can we avoid that?
 - Yes - by *learning* from **failing** trajectories/demonstrations

State projection and stored trajectory



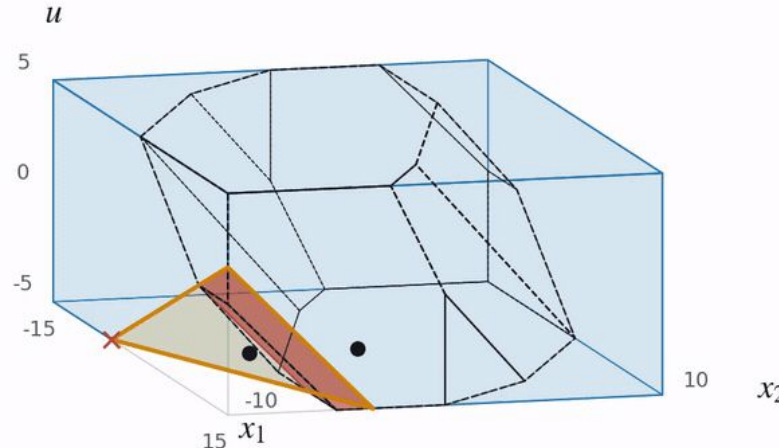
11 halfspaces in the current polytope

Learning from Failure

- From failed trajectories, we can learn, model-free constraints that separate failing state-controls from safe state-controls
 - Guaranteed safety for linear systems with polytopic constraints

1. Adjust the offset

MSCI region wrongly removed



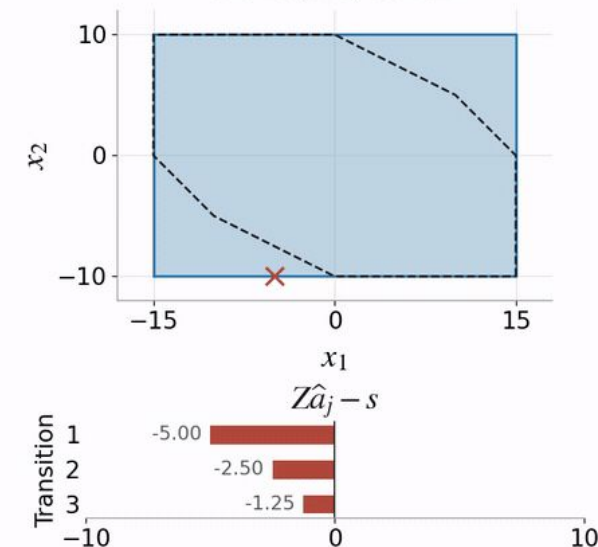
$$\hat{a}_j^\top z \leq \tilde{g}$$

$$\hat{a}_j = [0.25, -0.75, -0.75]^\top$$

Changing the offset translates the boundary; the normal is held fixed.

Blue: candidate set Dashed: MSCI / MCI reference
Red: wrongly removed region

State projection



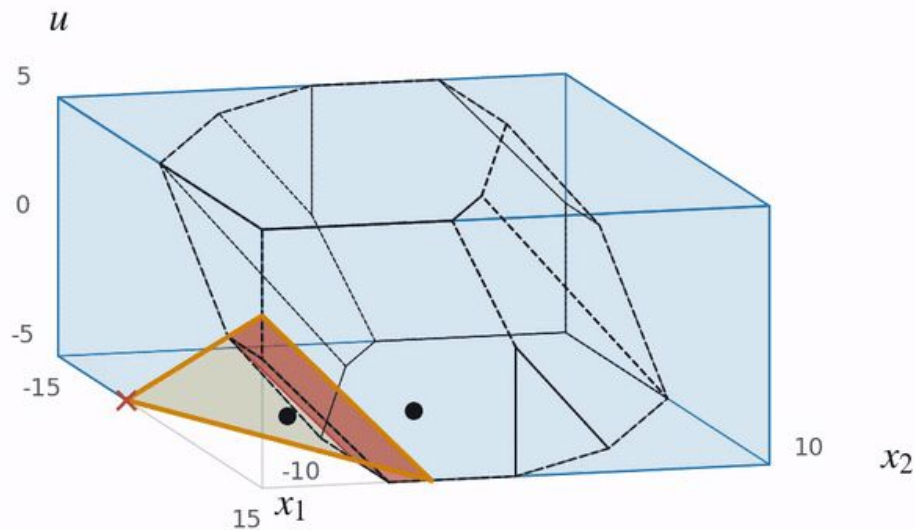
Trial offset: 10.00

Remove Just Enough

- Practically - remove the “right” amount of state-action pairs

1. Adjust the offset

MSCI region wrongly removed

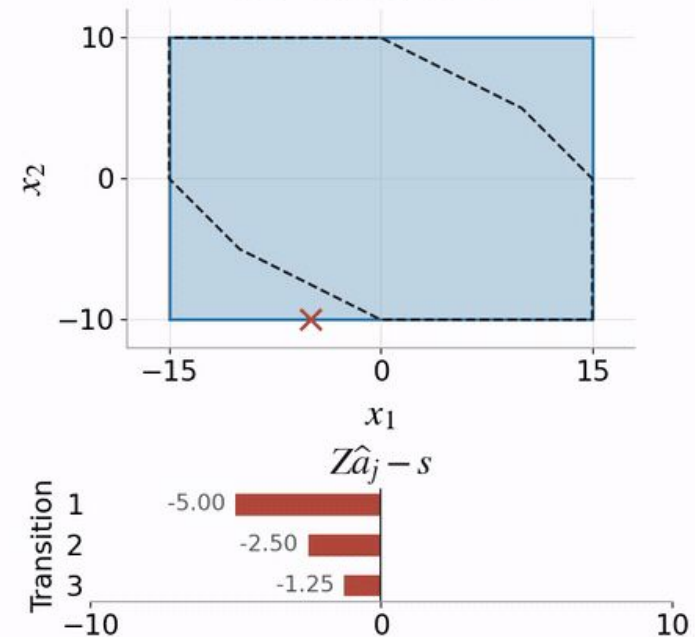


$$\hat{a}_j^\top z \leq \tilde{g}$$
$$\hat{a}_j = [0.25, -0.75, -0.75]^\top$$

Changing the offset translates the boundary; the normal is held fixed.

Blue: candidate set Dashed: MSCI / MCI reference
Red: wrongly removed region

State projection



Trial offset: 10.00

The Linear System Case

- For the rest, we will consider a linear system with polytopic constraints

$$x(k+1) = Ax(k) + Bu(k),$$

$$\mathcal{P} = \{z \in \mathbb{R}^{n_x+n_u} : H_z z \leq h_z\},$$

- The predecessor operator is then

$$\text{Pre}_z(\Omega) = \{z \in \mathbb{R}^{n_x+n_u} : \exists u^+ \text{ s.t. } [(Ax + Bu)^\top \quad (u^+)^\top]^\top \in \Omega\},$$

- or equivalently

$$\text{Pre}_z(\Omega) = \{z \in \mathbb{R}^{n_x+n_u} : \begin{bmatrix} A & B \end{bmatrix} z \in \Pi_x(\Omega)\}$$

Iterative Learning

- Learn the halfspaces of the predecessor of the current polytope
 - Start by defining the current iteration's polytope of constraints

$$\mathcal{P}_\ell \triangleq \{z \in \mathbb{R}^{n_x + n_u} : H_z^\ell z \leq h_z^\ell\},$$

with $\mathcal{X}_\ell \triangleq \Pi_x(\mathcal{P}_\ell) = \{x \in \mathbb{R}^{n_x} : H_{\text{proj}}^\ell x \leq g_{\text{proj}}^\ell\},$

- Our goal is then to learn $\text{Pre}_z(\mathcal{P}_\ell)$ from failing trajectories
- For the j -th constraint in $\text{Pre}_z(\mathcal{P}_\ell)$, our target is then

known!

$$a_j^* \left(H_{\text{proj}}^{\ell-1} \right)_j \begin{bmatrix} A & B \end{bmatrix} z \leq \left(g_{\text{proj}}^{\ell-1} \right)_j$$

Need

Failure Aware Iterative Learning

- Collect a trajectory ending with a failure, such that

$$z^\ell \in \mathcal{P}_{\ell-1} \quad \text{and} \quad z^\ell \notin \text{Pre}_z(\mathcal{P}_{\ell-1}),$$

- From this trajectory (and others), we have

$$Z = \begin{bmatrix} z(t_1)^\top \\ z(t_2)^\top \\ \vdots \\ z(t_p)^\top \end{bmatrix} \in \mathbb{R}^{p \times p}, \quad s = \begin{bmatrix} (H_{\text{proj}}^{\ell-1})_j x(t_1+1) \\ (H_{\text{proj}}^{\ell-1})_j x(t_2+1) \\ \vdots \\ (H_{\text{proj}}^{\ell-1})_j \underbrace{x(t_p+1)} \end{bmatrix} \in \mathbb{R}^p,$$
$$(H_{\text{proj}}^{\ell-1})_j \begin{bmatrix} A & B \end{bmatrix} Z = s$$

A Linear Regression

- We can rewrite $a_j^{*\top} z(t) = (H_{\text{proj}}^{\ell-1})_j [A \ B] z(t)$
 $= (H_{\text{proj}}^{\ell-1})_j x(t+1),$

- So $s = Z a_j^*$

- Therefore, we can learn the halfspace with the linear regression

$$\hat{a}_j = a_j^* = Z^{-1} s$$

- Then, we can update our polytope

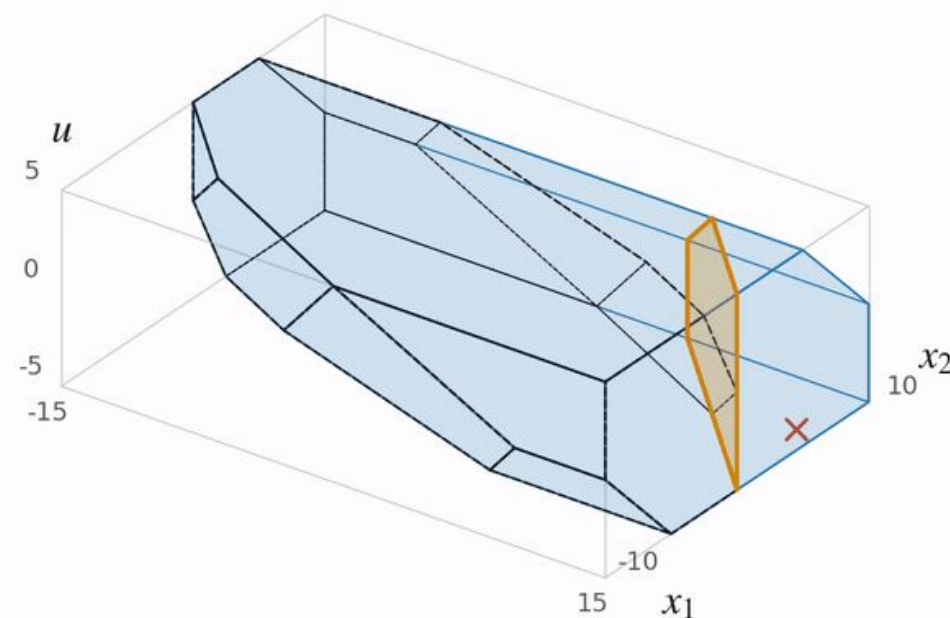
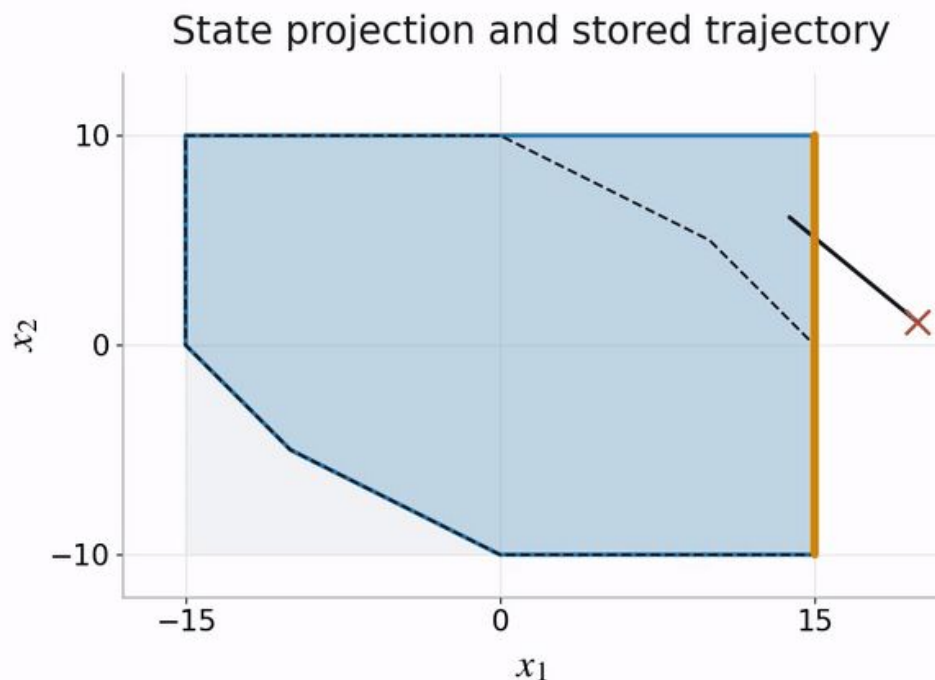
$$\mathcal{P}_{\ell+1} = \mathcal{P}_{\ell} \cap \{z \in \mathbb{R}^{n_x + n_u} : \hat{a}_j^\top z \leq (g_{\text{proj}}^\ell)_j\}$$

Multiple Updates per Failure

- One failure informs more than one update to the polytope

The final trajectory: halfspace 1 of 3

Dashed outline: MSCI reference

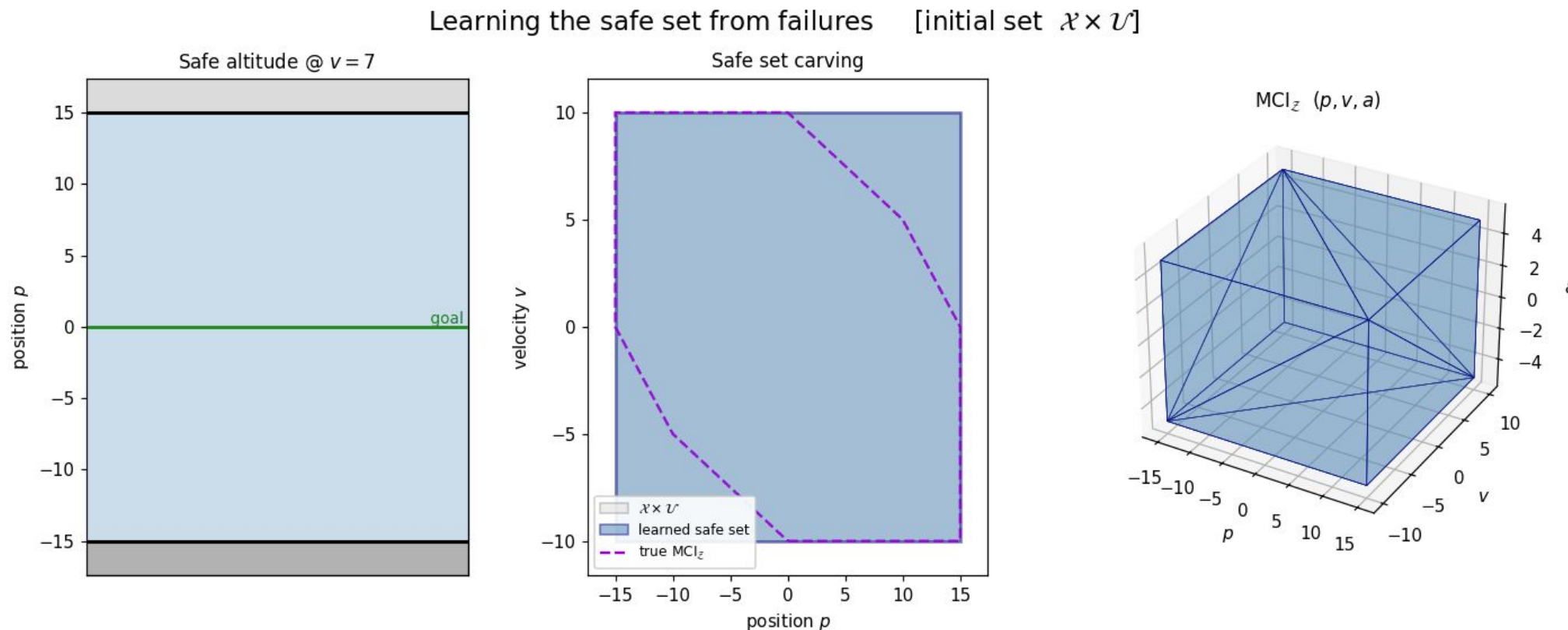


11 halfspaces in the current polytope

Recheck the same trajectory after each update; continue until it reveals no new failures.

Learning the Safe Set from Failures

- **FAIL** is a **model-free** algorithm that carves the initial set $X \times U$ down to the true MSCI from failing trajectories



FAIL Converges to the True Set

Monotone

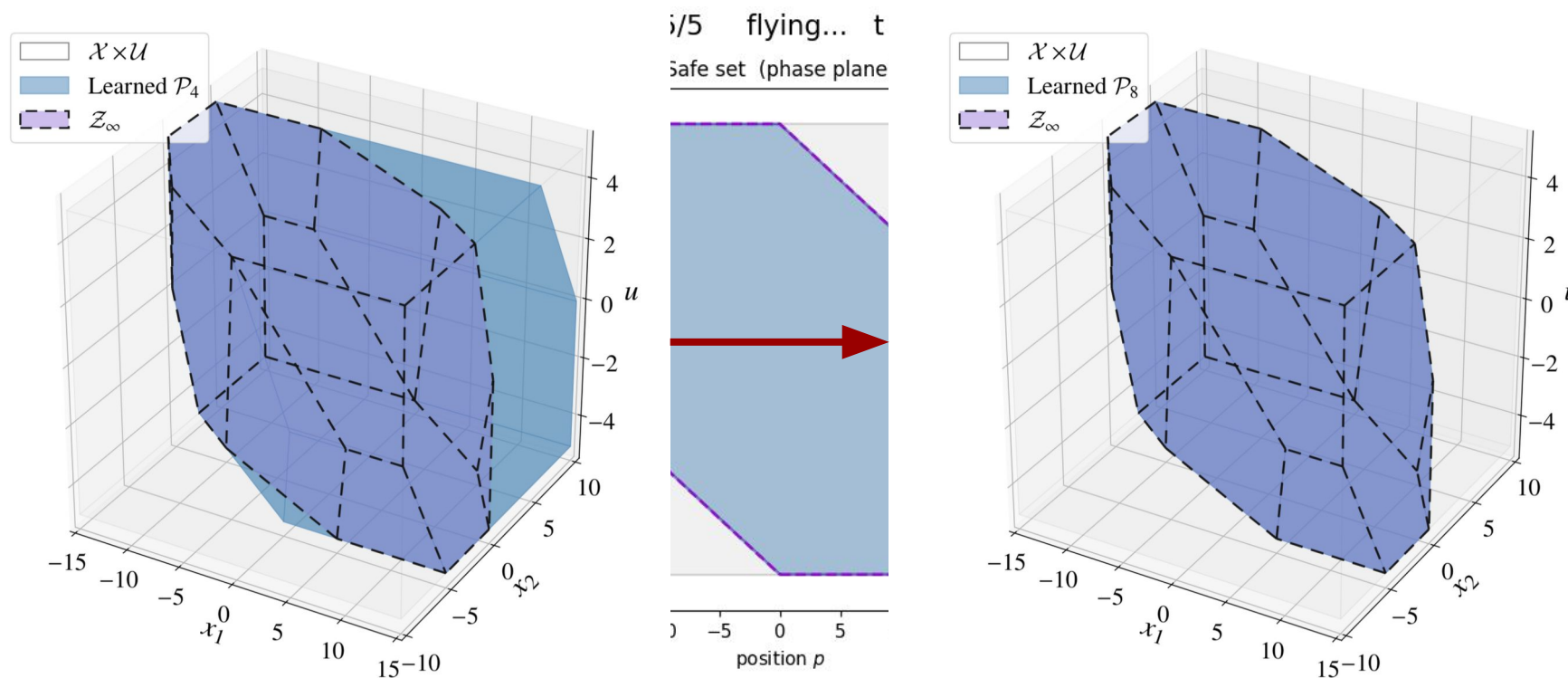
Sets only shrink; learned halfspaces never exclude a truly safe pair.

Model-free

Each halfspace is recovered from failing data alone, not from (A, B).

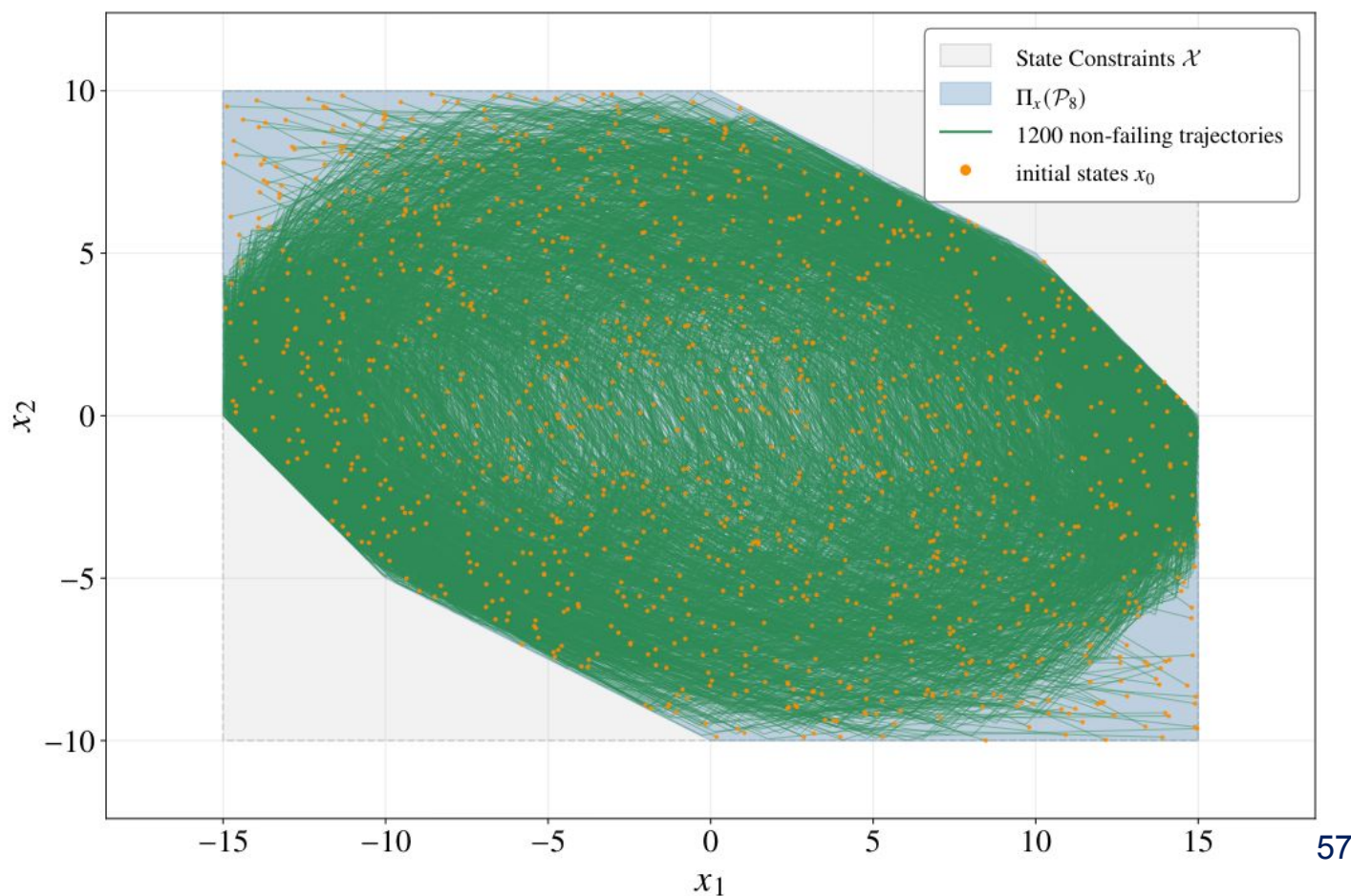
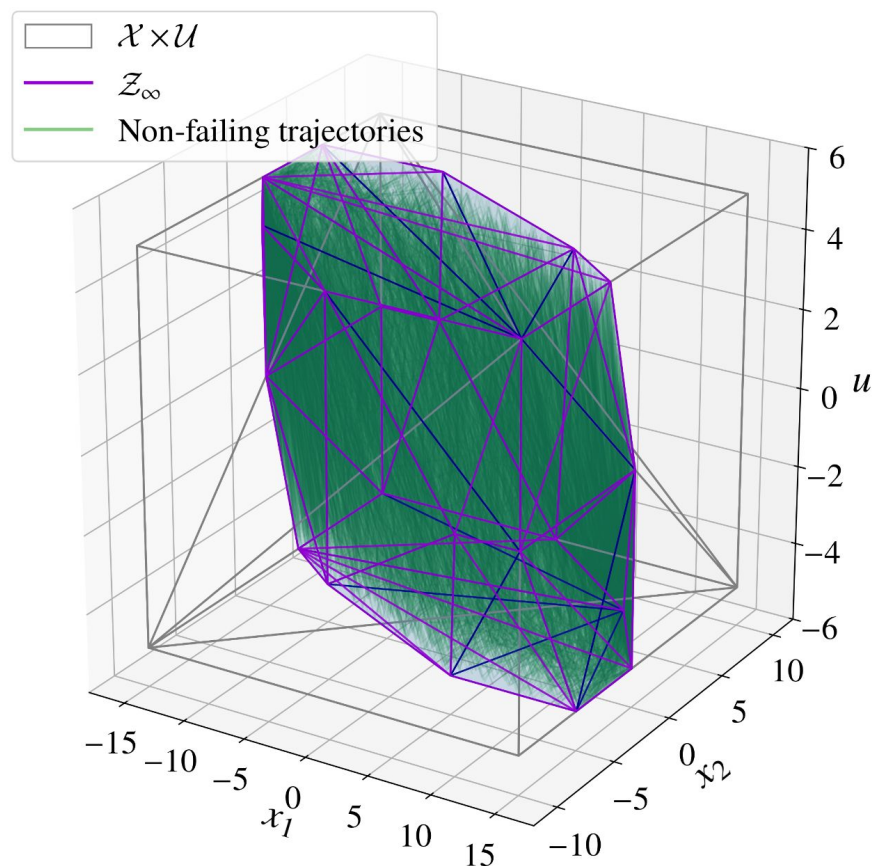
Finite

Terminates once no admissible trajectory exits the learned set.



Learning Captures Safe Controls

- Can sample controls from the learnt set and guarantee safety at any state in the learnt set - **no filtering needed!**

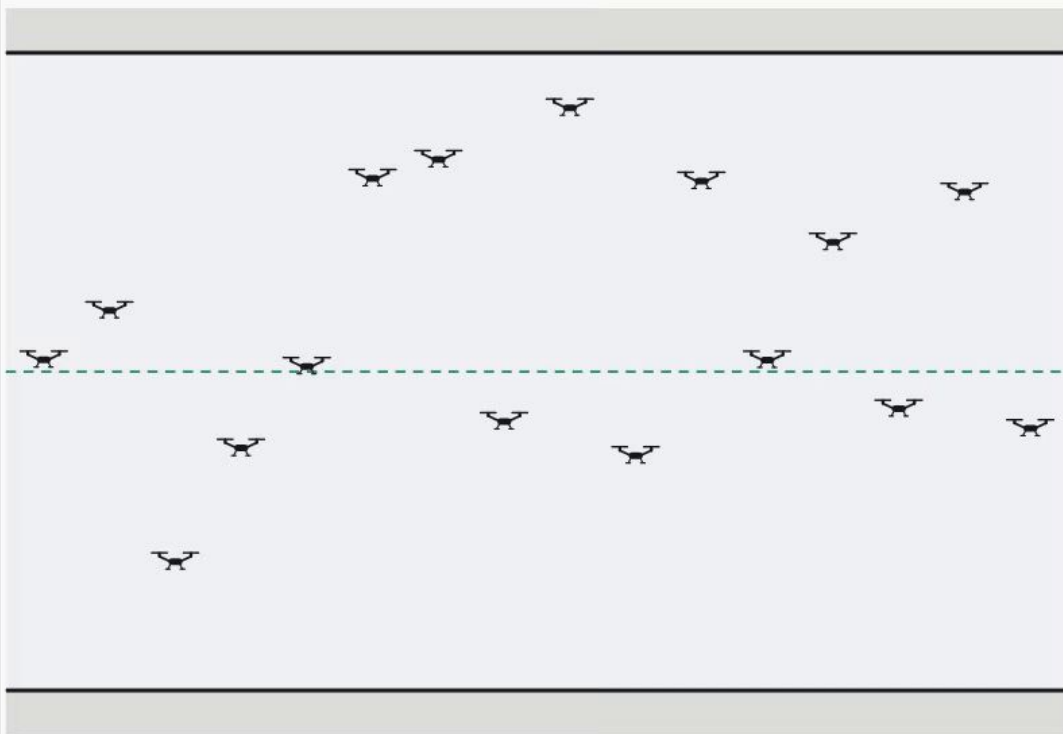


Learning Captures Safe Controls

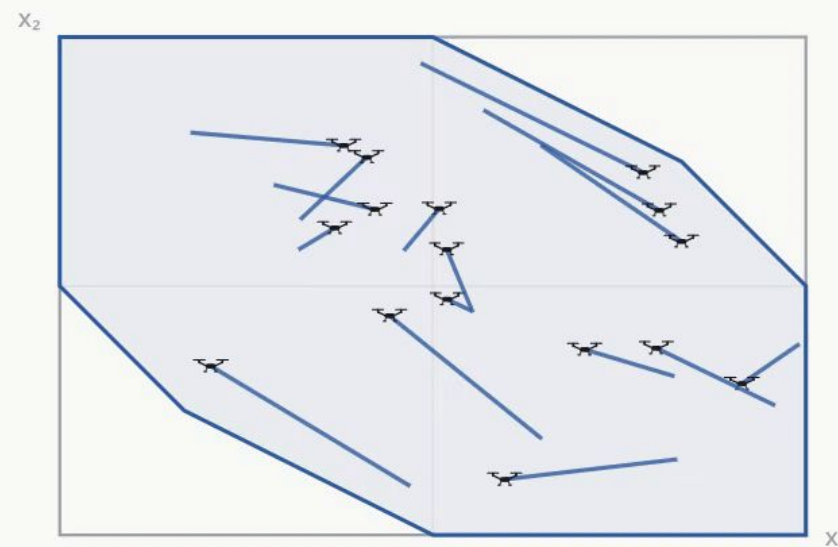
- Can sample controls from the learnt set and guarantee safety at any state in the learnt set - **no filtering needed!**

FIG. 6 · WHERE THE INPUTS COME FROM

16 vehicles · inputs drawn from $\mathcal{U}_L(x)$, the learned section · 0.2 s flown · left the set 0



corridor



— in flight
phase plane — left the set

Thank You!

Questions?



SIT-LMPC Website



SIT-LMPC Code



FAIL Website

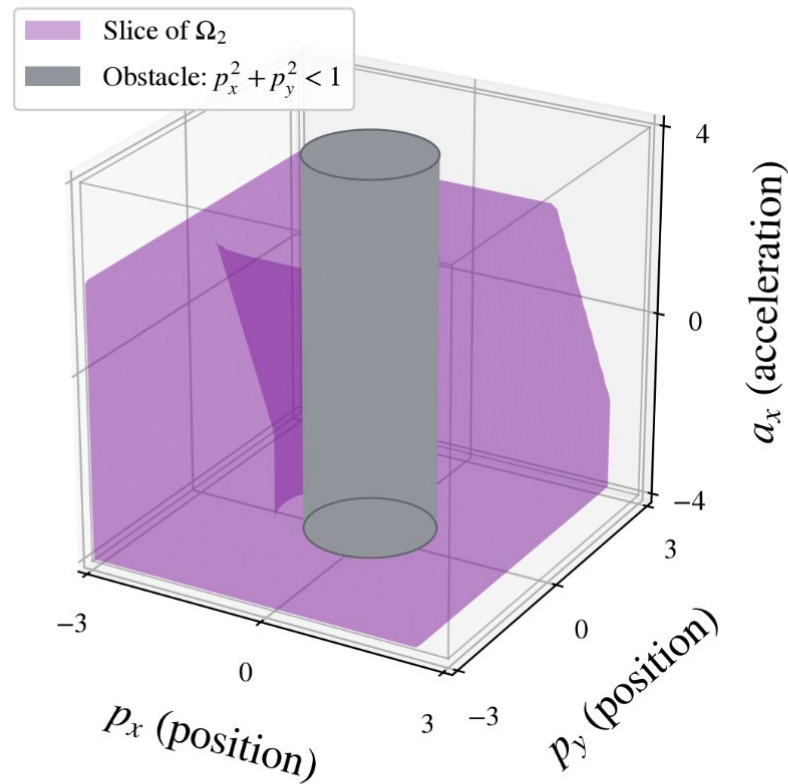


FAIL Paper

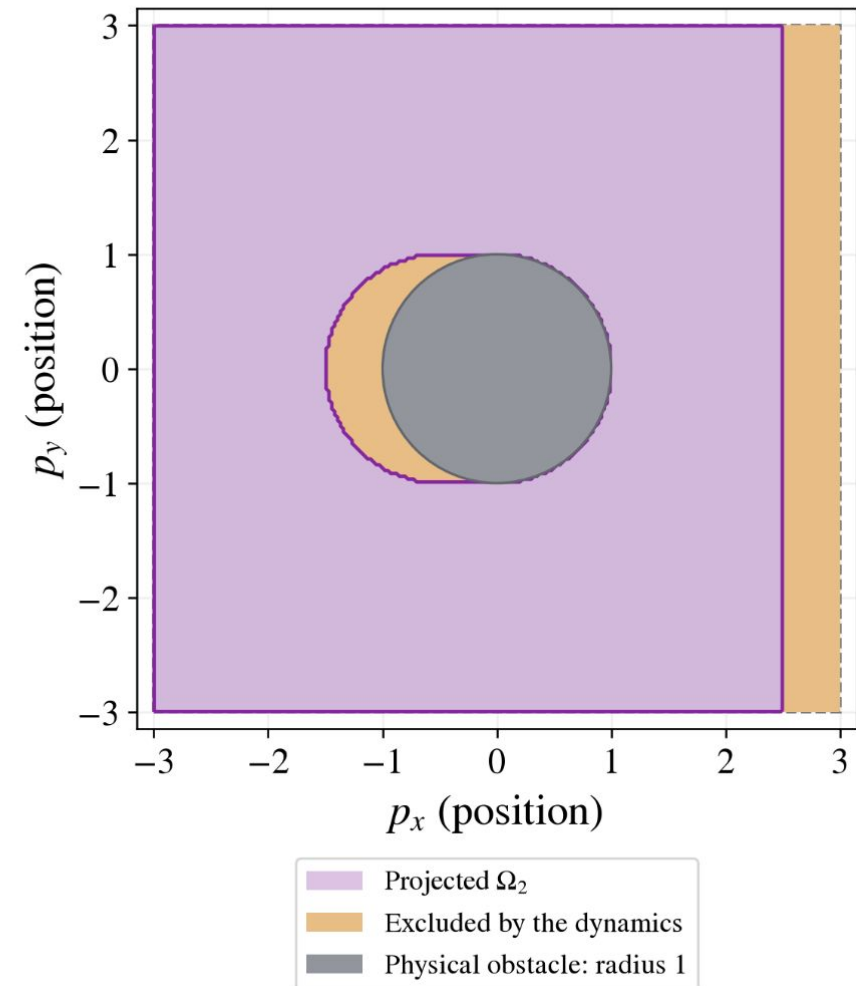
Non-convex Constraints

Planar point mass around a unit disk

Joint slice: $(v_x, v_y) = (1, 0)$, $a_y = 0$

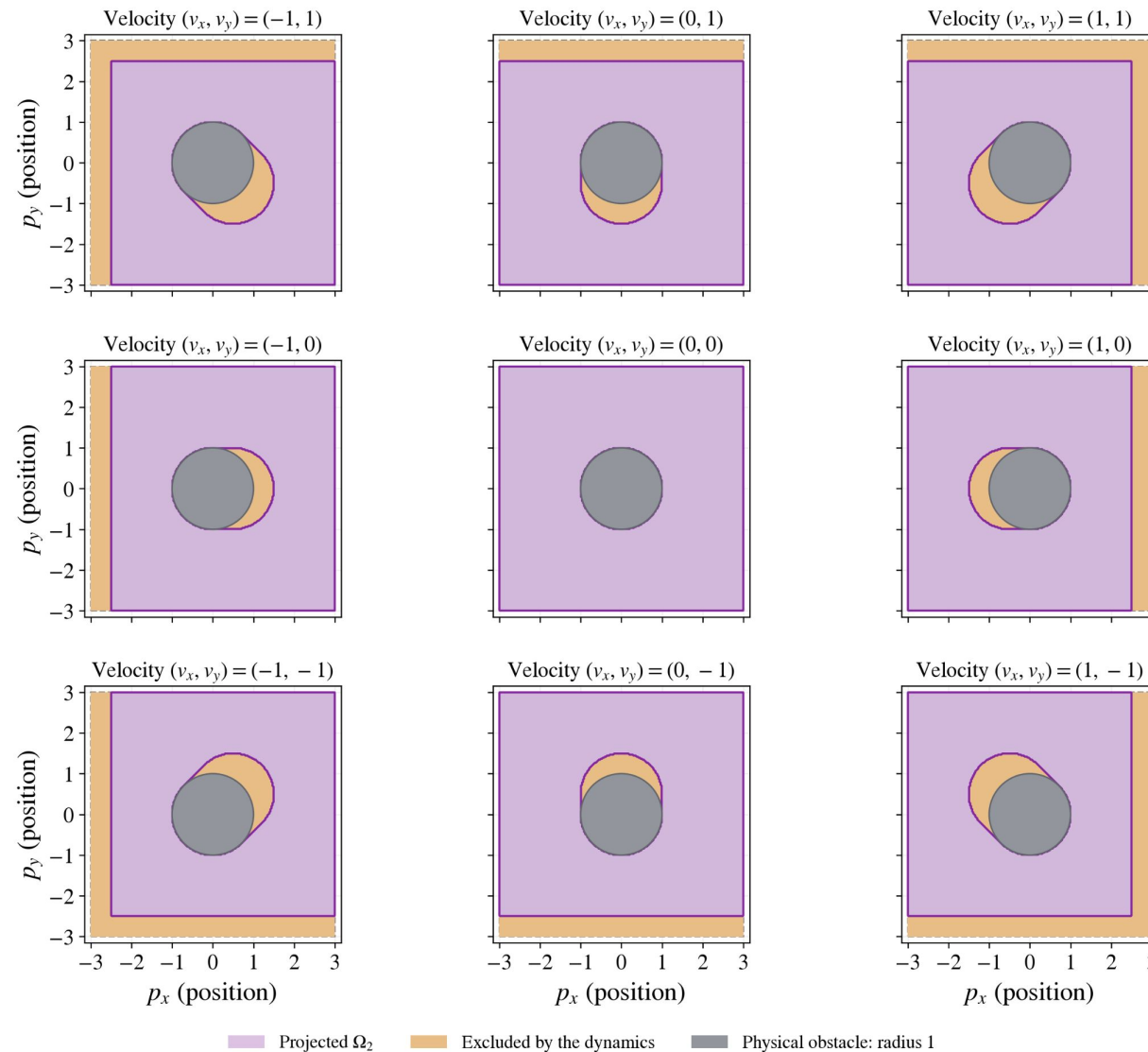


State projection: $(v_x, v_y) = (1, 0)$



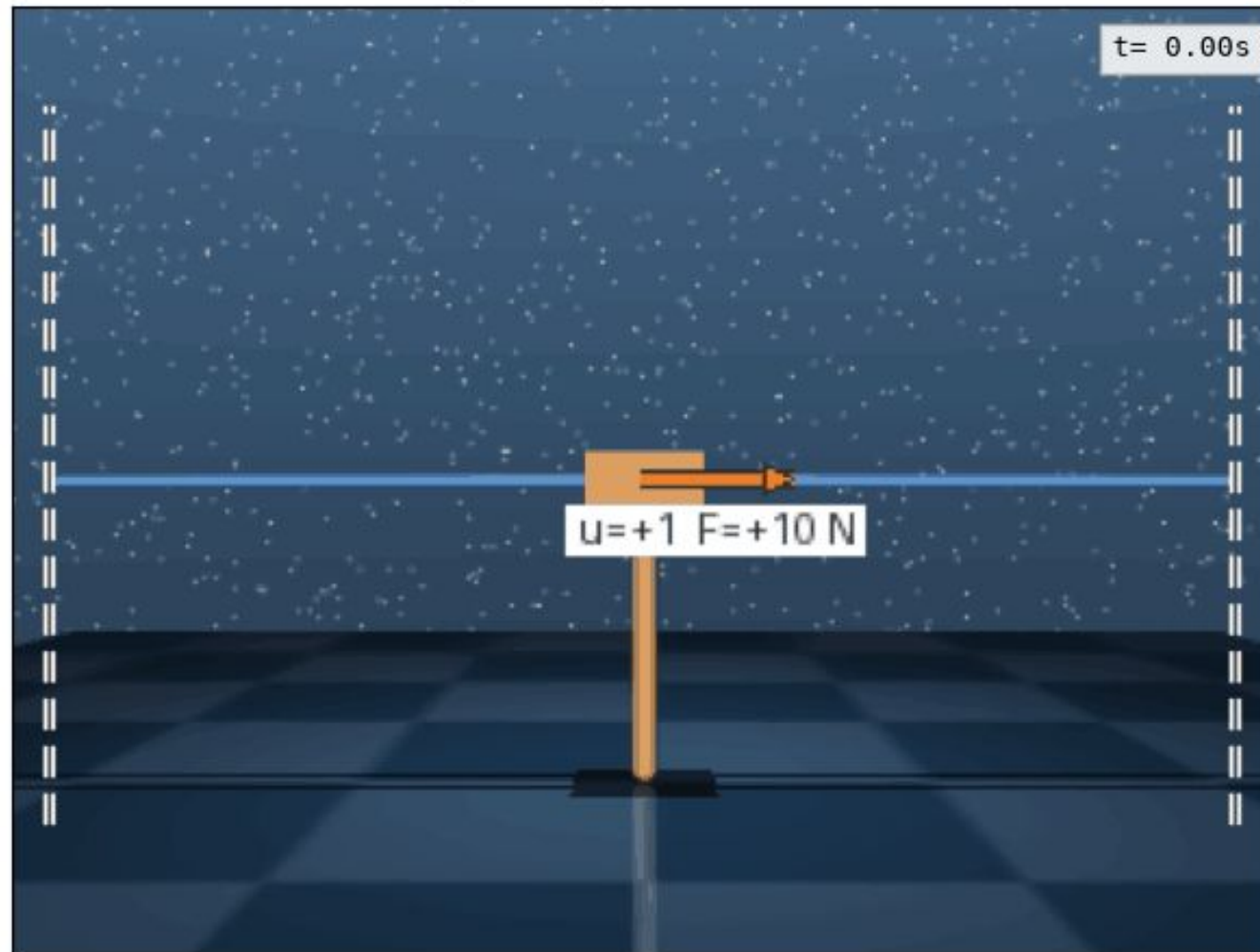
Non-convex Constraints

Position slices of the state projection



Nonlinear Dynamics - Short Horizon

DMC cart-pole swing-up · MuJoCo rendering · $H=5$
Dashed lines: cart body envelope (center limit ± 1.8 m); arrows: force



decision 0/125

References

- Koptev M, Figueroa N, Billard A. Reactive collision-free motion generation in joint space via dynamical systems and sampling-based MPC. The International Journal of Robotics Research. 2024;43(13):2049-2069. doi:10.1177/02783649241246557
- M. Minařík, R. Pěnička, V. Vonásek and M. Saska, "Model Predictive Path Integral Control for Agile Unmanned Aerial Vehicles," 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Abu Dhabi, United Arab Emirates, 2024, pp. 13144-13151, doi: 10.1109/IROS58592.2024.10802756.
- AD-MPCC: Adaptive Differentiable Model Predictive Contouring Control for Autonomous Racing
- S. Katayama and T. Ohtsuka, "Whole-body model predictive control with rigid contacts via online switching time optimization," 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 2022, pp. 8858-8865, doi: 10.1109/IROS47612.2022.9981790.
- https://erc-bpgc.github.io/blog/blog/model_predictive_control/
- U. Rosolia, X. Zhang, and F. Borrelli, "Robust Learning Model-Predictive Control for Linear Systems Performing Iterative Tasks," IEEE Transactions on Automatic Control, vol. 67, no. 2, pp. 856–869, 2022. doi: 10.1109/TAC.2021.3083559