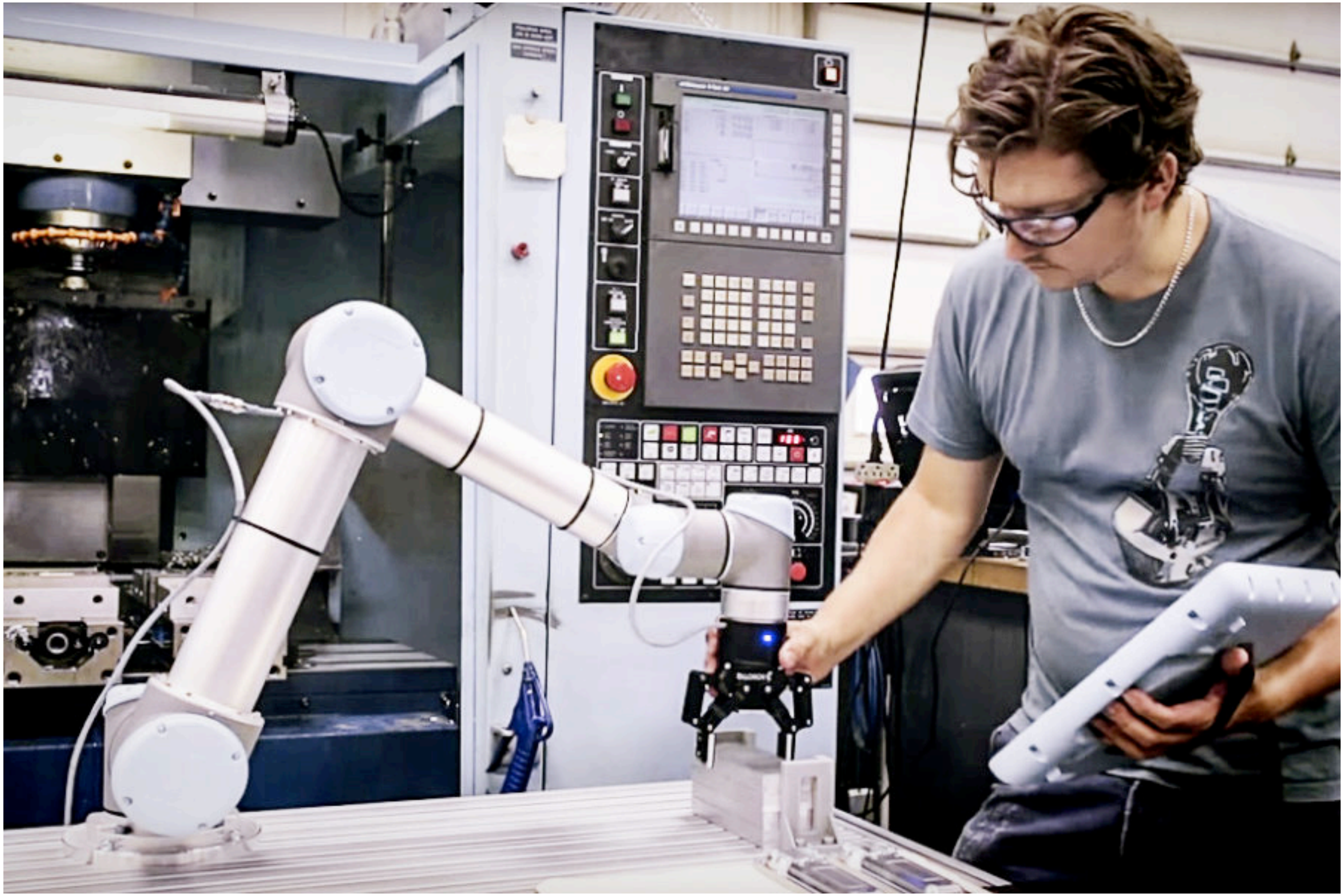
Safe Robot Control

Combining learning and model predictive control

Andrea Del Prete, University of Trento

Why Safety?

Today: Human-Robot Collaboration in Industry



https://www.therobotreport.com/manufacturing/ria-osha-robot-safety/

Why Safety?

Tomorrow: Black-box Data-Driven Control Policies

Zitkovich, Brianna, et al. "Rt-2: Vision-language-action models transfer web knowledge to robotic control." Conference on Robot Learning. PMLR, 2023.

What is Safety?

ISO/TS 15066 (2016, revised in 2022)

Emeric, C., Geoffroy, D., & Paul-Eric, D. (2020). Development of a new robotic programming support system for operators. Procedia Manufacturing, 51(2019), 73–80. https://doi.org/10.1016/j.promfg.2020.10.012

Safety Definition

What is safety?

- Joint angle, velocity, torque limits
- Collision avoidance
  - Self-collision
  - Static obstacles (e.g., table, wall)
  - Dynamic obstacles (e.g., humans, other robots)
- Collision management:
  - Contact shall not result in pain or injury

$$g(x, u) \leq 0$$

State of the art

Safety Guarantees

State of the art

- Main tools to ensure safety:
  - Control-Invariant Sets (CIS)
  - Control Barrier Functions (CBF)
  - Back-up Policies (BUP)
- Very similar tools
  - CBF and BUP implicitly define a CIS
- We focus on CIS in the rest of the presentation

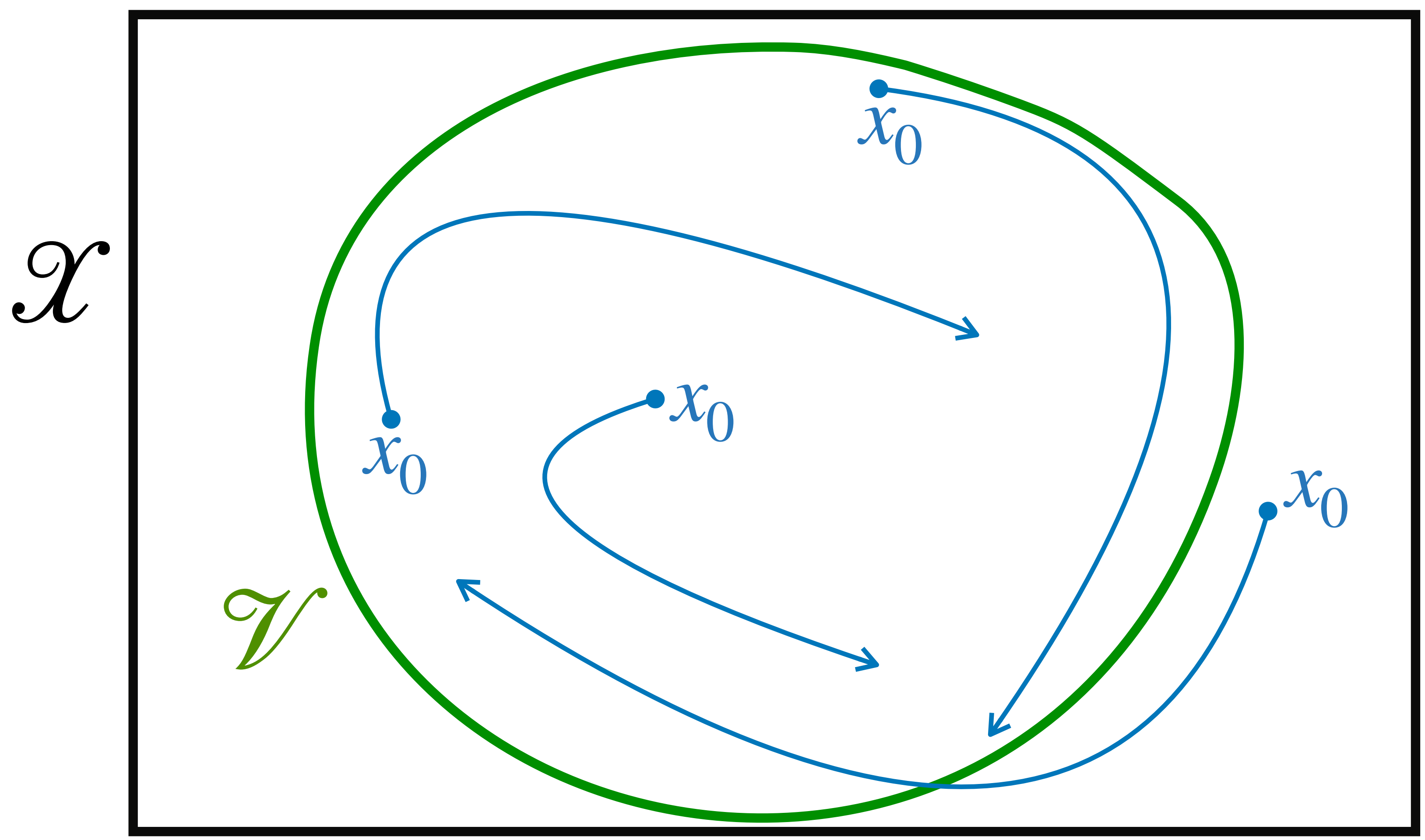
Control Invariant Sets

Definitions

- Constrained discrete-time dynamical system:

$$x_{i+1} = f(x_i, u_i) \qquad x \in \mathscr{X}, \quad u \in \mathscr{U}$$

$\mathscr{V}$ is a control invariant set ⟷ Once x is in $\mathscr{V}$, it can remain in $\mathscr{V}$.



Safety via Control Invariant Sets

How does it work?

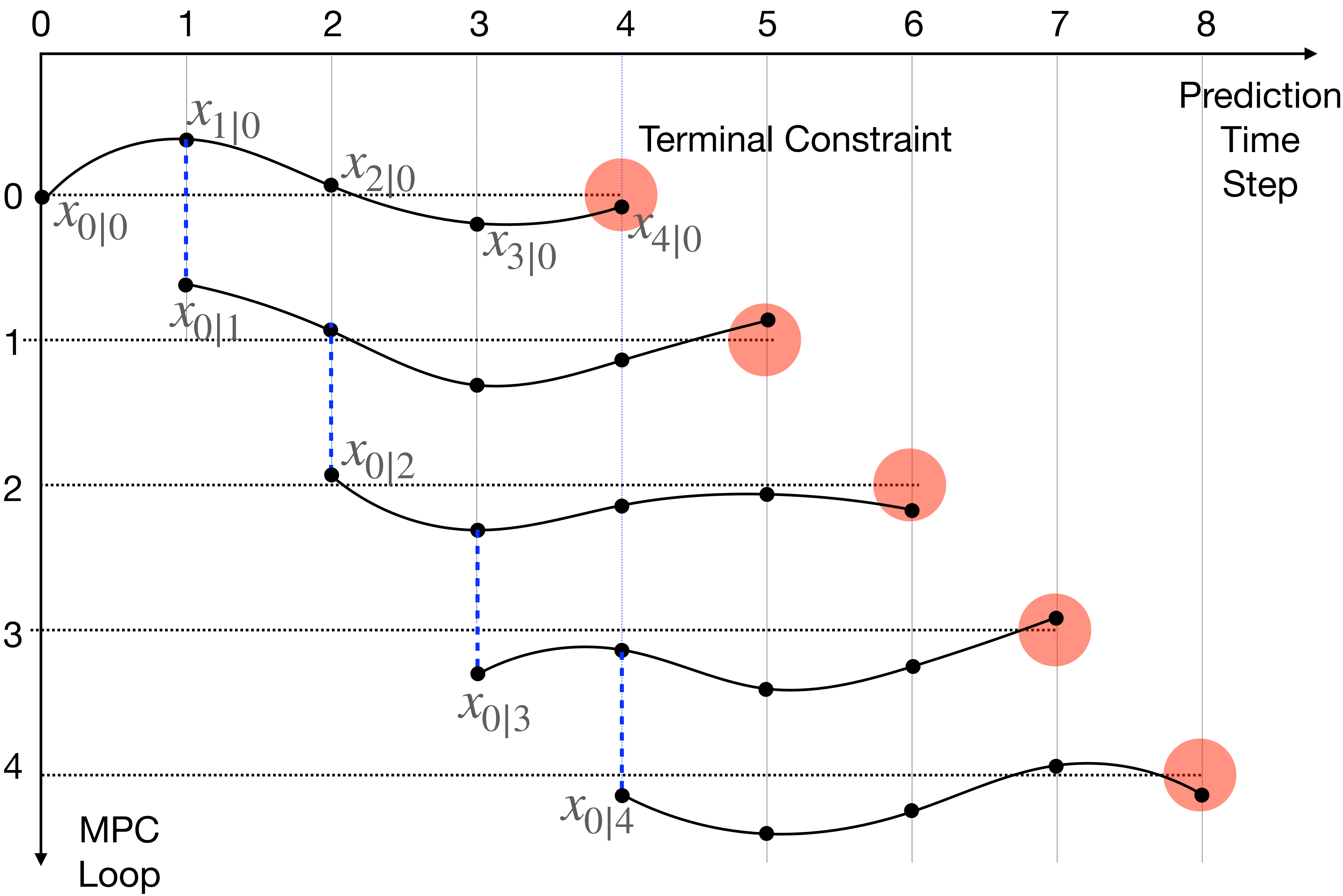
- Suppose we know a CIS $\mathscr{V}$.
- Suppose $\mathscr{V}$ is a subset of $\mathscr{X}$ (feasible state space).
- Suppose we start in $\mathscr{V}$.
- Then:
  - we can remain in $\mathscr{V}$ forever;
  - hence, we can remain in $\mathscr{X}$ forever;
  - hence, we ensure safety.

Recursive Feasibility

Model Predictive Control (MPC)

- Using a CIS $\mathscr{V}$ as terminal set ensures recursive feasibility in MPC

$$
\begin{aligned}
\underset{\{x_i\}_0^N, \{u_i\}_0^{N-1}}{\text{minimize}} \quad & \sum_{i=0}^{N-1} \ell_i(x_i, u_i) + \ell_N(x_N) \\
\text{subject to} \quad & x_0 = x_{init} \\
& x_{i+1} = f(x_i, u_i) \qquad i = 0 \ldots N-1 \\
& x_i \in \mathcal{X}, u_i \in \mathcal{U} \qquad i = 0 \ldots N-1 \\
& \boxed{x_N \in \mathcal{V}}
\end{aligned}
$$



0
1
2
3
4
5
6
7
8
Prediction Time Step
Terminal Constraint
$x_{1|0}$
$x_{2|0}$
$x_{0|0}$
$x_{3|0}$
$x_{4|0}$
$x_{0|1}$
$x_{0|2}$
$x_{0|3}$
$x_{0|4}$
0
1
2
3
4
MPC Loop

Limitations of State of the art

Control Invariant Sets

- CIS are in general unknown for nonlinear systems/constraints
- Numerical approximation techniques exist, however:
  - They are computationally demanding (curse of dimens.)
  - A numerical approximation of a CIS is **not** a CIS
    - → all safety guarantees are lost!
- Control Barrier Functions and Backup Policies suffer from similar issues.

Our Contributions

Learning Control Invariant Sets

Viability Boundary Optimal Control (VBOC)

- Method to numerically approximate CIS
- It generates data solving Trajectory Optimization problems
- It uses supervised learning to approximate set
- PROS:
  - Better accuracy/efficiency trade-off than other methods
- CONS:
  - Tailored to fully-actuated multi-body systems (e.g., manipulators)

La Rocca, Saveriano, Del Prete (2023). VBOC: Learning the Viability Boundary of a Robot Manipulator using Optimal Control. IEEE RAL

Safe Control with approximate CIS

Receding Constraint MPC

- Novel MPC formulation, featuring two constraints:
  - A soft terminal constraint
  - A hard receding constraint
- PROS
  - Recursive feasibility under weaker conditions (N-Step CIS)
  - Safe abort under even weaker conditions (inner approx. of CIS)
- CONS
  - Hard to prove N-Step CIS or inner approx. of CIS

Lunardi, La Rocca, Saveriano, Del Prete (2024). Receding-Constraint Model Predictive Control using a Learned Approximate Control-Invariant Set. IEEE ICRA.

Receding-Constraint MPC

Gianni Lunardi
Asia La Rocca
Matteo Saveriano
Andrea Del Prete

Lunardi, La Rocca, Saveriano, Del Prete (2024). Receding-Constraint Model Predictive Control using a Learned Approximate Control-Invariant Set. IEEE ICRA.

Recursive Feasibility

Model Predictive Control (MPC)

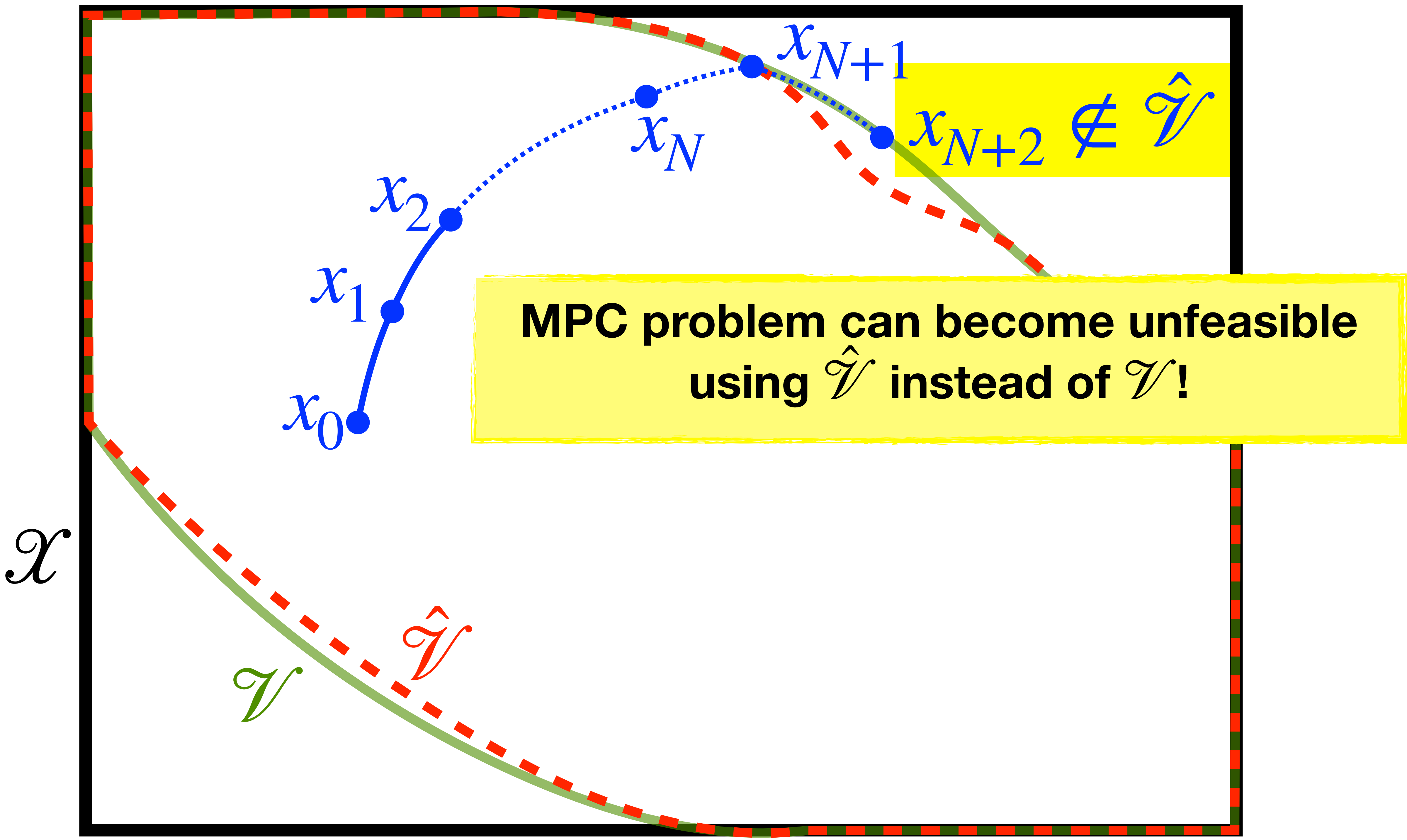
- Using a CIS $\mathscr{V}$ as terminal set ensures recursive feasibility in MPC

$$
\begin{array}{lll}
\underset{\{x_i\}_0^N, \{u_i\}_0^{N-1}}{\text{minimize}} & \sum_{i=0}^{N-1} \ell_i(x_i, u_i) + \ell_N(x_N) & \\
\text{subject to} & x_0 = x_{init} & \\
 & x_{i+1} = f(x_i, u_i) & i = 0 \ldots N-1 \\
 & x_i \in \mathcal{X}, u_i \in \mathcal{U} & i = 0 \ldots N-1 \\
 & \boxed{x_N \in \hat{\mathcal{V}}} &
\end{array}
$$

What if the terminal set is an approximation of a CIS $\hat{\mathscr{V}} \approx \mathscr{V}$?

Approximate Control Invariance

Graphical example



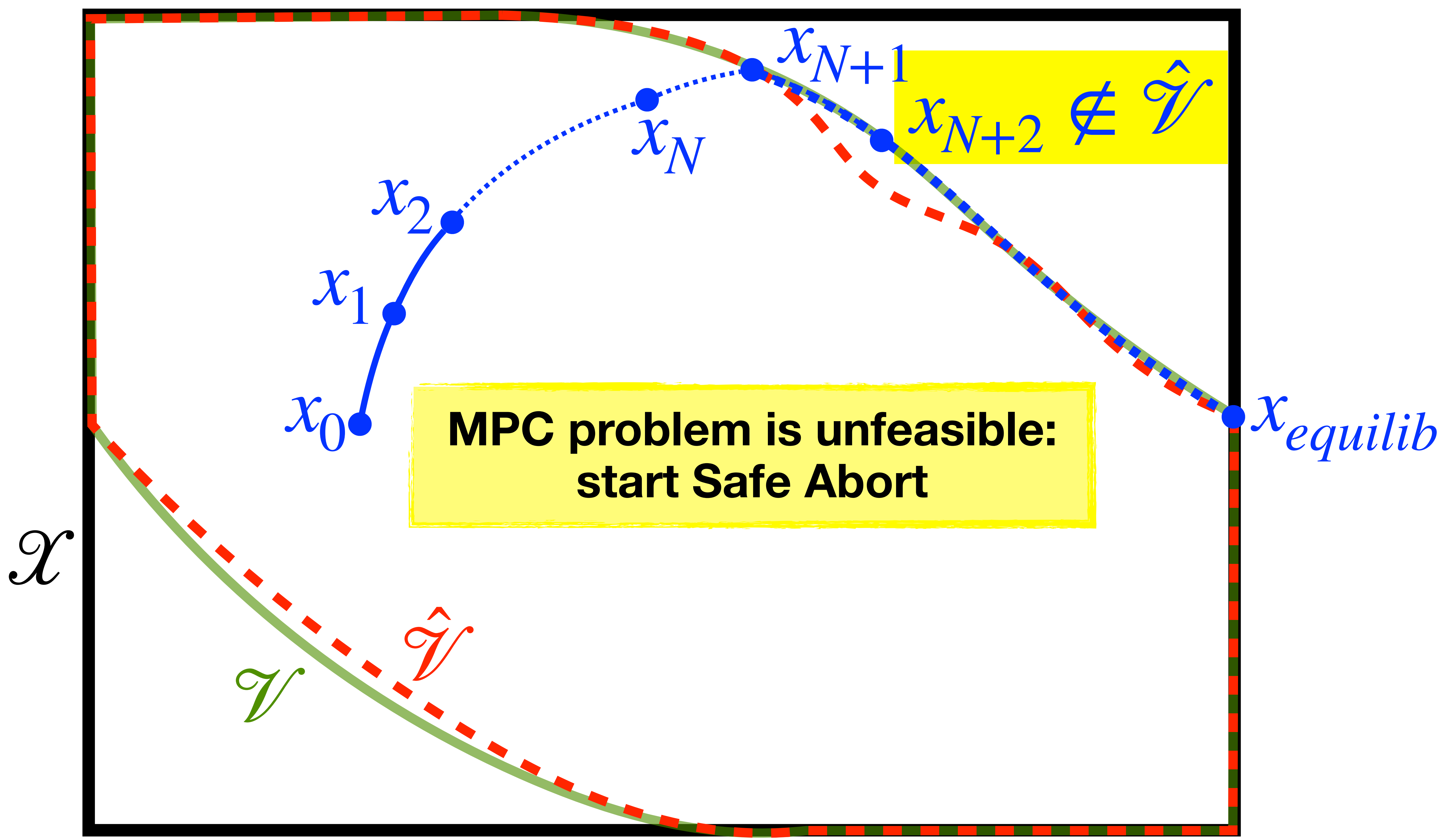
Idea #1: Safe Abort

Ensuring Safety

- Assume $\hat{\mathscr{V}} \subseteq \mathscr{V}$
    - => Even if $\hat{\mathscr{V}}$ is not a CIS, any state in $\hat{\mathscr{V}}$ is “safe”
- **Safe Abort:**
    - If MPC problem becomes unfeasible
    - Find (and follow) trajectory that:
        - starts from last predicted state in $\hat{\mathscr{V}}$
        - reaches an equilibrium state
- Such a trajectory is guaranteed to exist

Approximate Control Invariance

Safe Task Abortion



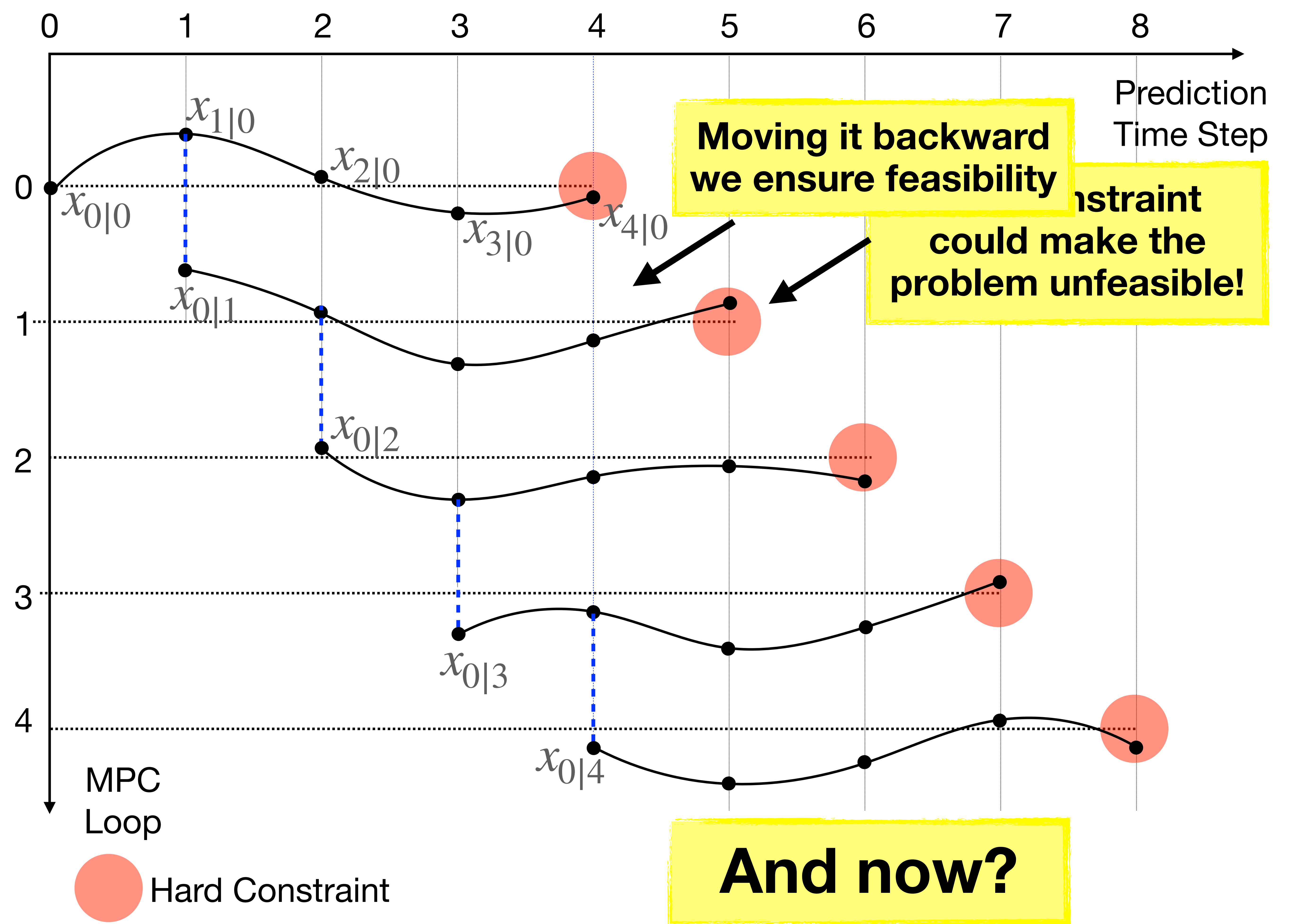
Nice! This ensures
SAFETY.

Can we also ensure
RECURSIVE
FEASIBILITY?

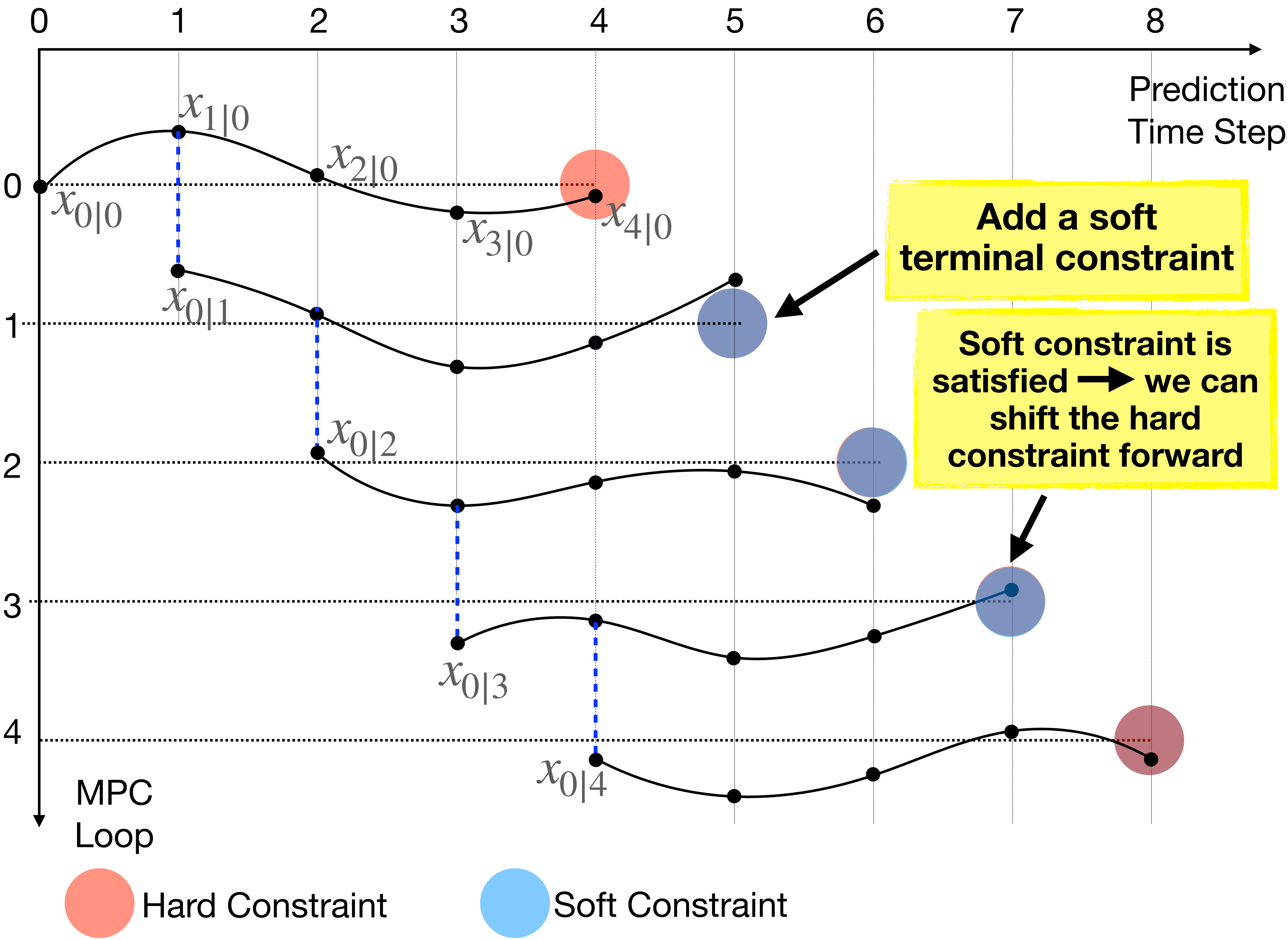
Idea #2: Receding Constraint

Ensuring Recursive Feasibility

- **Observation**
  - Having the terminal state in $\hat{\mathscr{V}}$ is not necessary to ensure safety
  - Having any future state in $\hat{\mathscr{V}}$ would be sufficient
- **Idea**
  - Adapt online the time step for which we constrain the state in $\hat{\mathscr{V}}$



0
1
2
3
4
5
6
7
8
Prediction Time Step
MPC Loop
$x_{0|0}$
$x_{1|0}$
$x_{2|0}$
$x_{3|0}$
$x_{4|0}$
$x_{0|1}$
$x_{0|2}$
$x_{0|3}$
$x_{0|4}$
Moving it backward we ensure feasibility
could make the problem unfeasible!
And now?
Hard Constraint



0
1
2
3
4
5
6
7
8
Prediction Time Step
0
1
2
3
4
MPC Loop
$x_{0|0}$
$x_{1|0}$
$x_{2|0}$
$x_{3|0}$
$x_{4|0}$
$x_{0|1}$
$x_{0|2}$
$x_{0|3}$
$x_{0|4}$
Add a soft terminal constraint
Soft constraint is satisfied → we can shift the hard constraint forward
Hard Constraint
Soft Constraint

Receding Constraint MPC

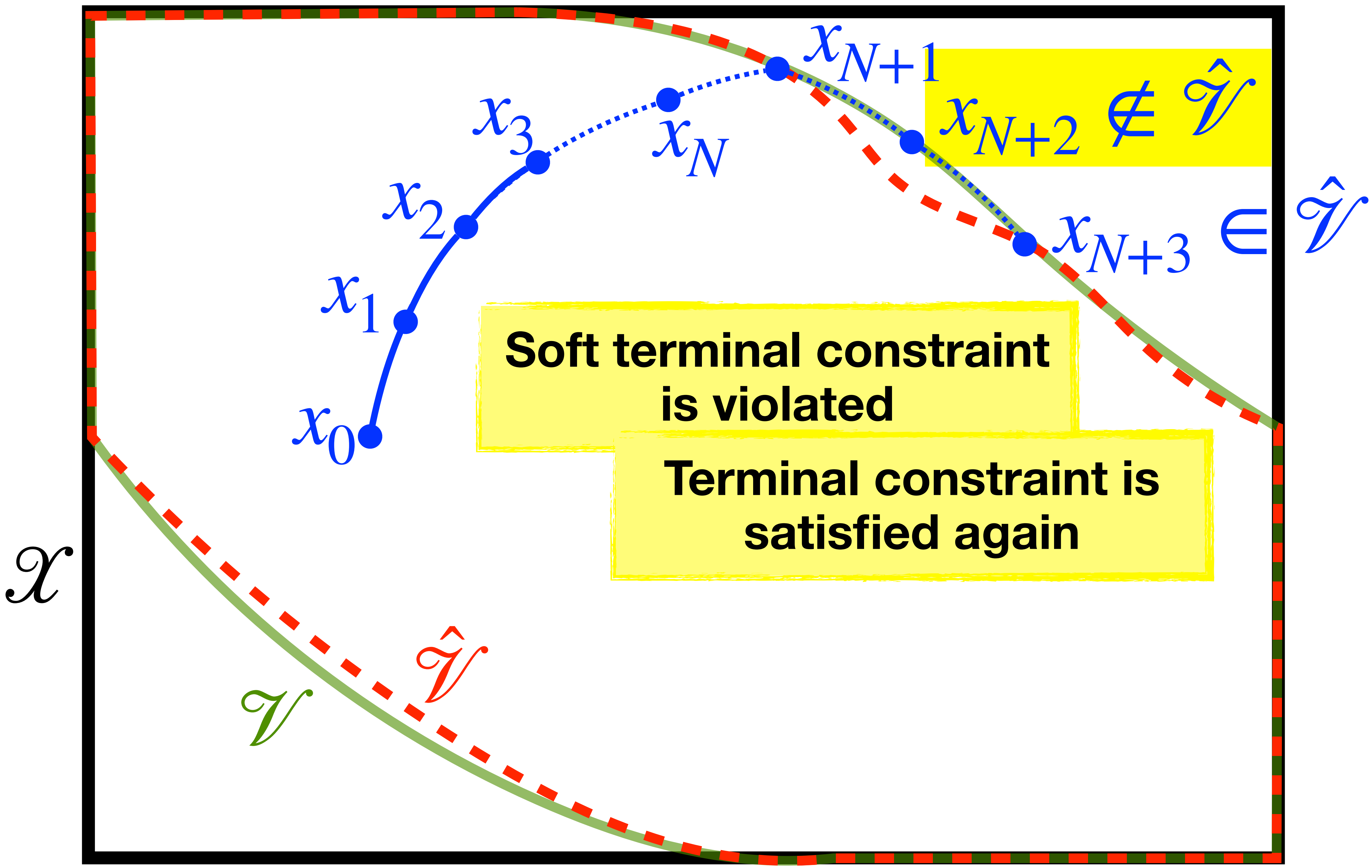
N-Step Control Invariant Set

- Assume $\hat{\mathscr{V}} \subseteq \mathscr{V}$ is an N-Step CIS, defined as follows
  - If $x_0 \in \hat{\mathscr{V}}$ then it is possible to have $x_k \in \hat{\mathscr{V}}$ for some $k \in [1,N]$
- Make hard constraint on $\hat{\mathscr{V}}$ recede in time
- Add soft terminal constraint on $\hat{\mathscr{V}}$

- **Recursive feasibility** is guaranteed
  - Note: N-Step CIS is a weaker requirement than CIS

N-Step Control Invariance

Graphical example



Simulation Results

Setup

- Comparing 5 MPC formulations
- 3 DoF robot manipulator
- Acados software library
- Setpoint regulation: $x^{\text{ref}} = (q_0^{max} - 0.05,\, q_1^{mid},\, q_2^{mid},\, 0,\, 0,\, 0)$
- 100 simulations from random initial joint configurations
- Different horizons N (34-36) to ensure computation time < dt (5 ms)
- https://github.com/idra-lab/safe-mpc

Results

Safety Margin 2%

MPC Formulation	# Tasks Completed	# Tasks Safely Aborted	# Tasks Failed
Naive	69	-	31
Soft Terminal	69	-	31
Soft Terminal with Abort	70	11	19
Hard Terminal with Abort	70	8	22
Receding Constraint	**77**	**18**	**5**

Can we do better?

Results

Safety Margin 10%

MPC Formulation	# Tasks Completed	# Tasks Safely Aborted	# Tasks Failed
Naive	69	-	31
Soft Terminal	69	-	31
Soft Terminal with Abort	70	22	8
Hard Terminal with Abort	70	21	9
Receding Constraint	**77**	**20**	**3**

Cost & Computation Time

Safety Margin 10%

		99-Percentile	
MPC Formulation	Cost Increase	MPC Computation Time [ms]	Safe Abort Computation Time [ms]
Naive	0%	3.75	-
Soft Terminal	0.05%	5.50	-
Soft Terminal with Abort	0.042%	3.73	130
Hard Terminal with Abort	0.042%	3.88	100
Receding Constraint	**0.023%**	**3.95**	**80**

Future Work

- Learn safe-abort policy to warm-start safe-abort OCP solver
- Use robust optimization to handle dynamics uncertainties
- Application to black-box policies (e.g., from RL)
- Computation/certification of:
  - N-Step Control-Invariant Set
  - Inner approximation of CIS

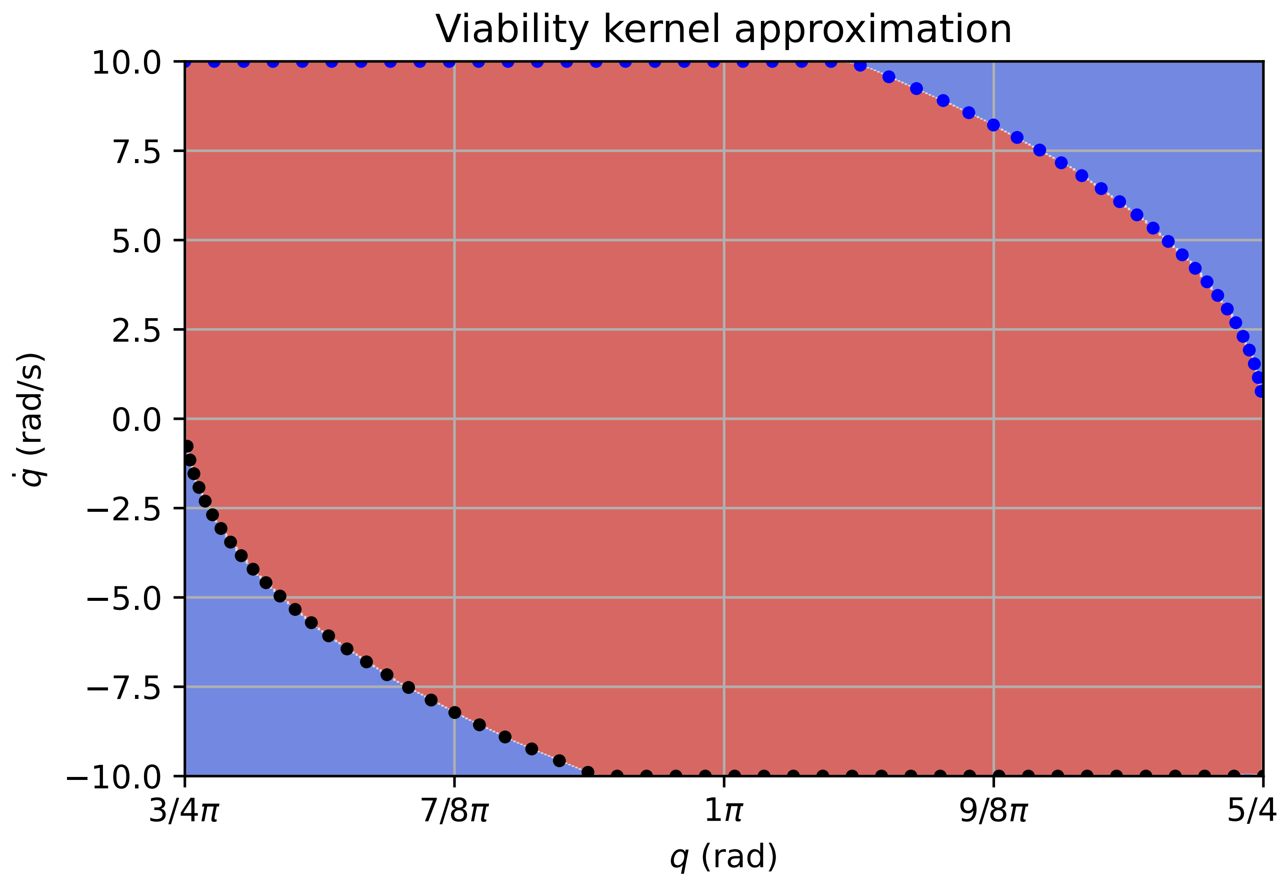
VBOC: Learning the Viability Kernel of a Robot Manipulator

Asia La Rocca
Matteo Saveriano
Andrea Del Prete

La Rocca, Saveriano, Del Prete (2023). VBOC: Learning the Viability Boundary of a Robot Manipulator using Optimal Control. IEEE RAL

Problem Definition

- Compute viability kernel for robot manipulator
  - Set of states starting from which it is possible to avoid constraint violation
  - Largest CIS
- Nonlinear differentiable dynamics
- Nonlinear constraints
- No analytical solution



Backward Reachability

State of the art

- Given a state x, use Trajectory Optimization to determine if it is safe
- Compute trajectory starting from x and reaching an equilibrium state
- ∞-Step Backward Reachability ≈ Viability

TO problem formulation

State of the art

$$
\begin{aligned}
\underset{\{x_i\}_0^N, \{u_i\}_0^{N-1}}{\text{maximize}} \quad & 1 \\
\text{subject to} \quad & x_{i+1} = f(x_i, u_i) \quad \forall\, i = 0, \ldots, N-1 \\
& x_i \in \mathcal{X},\ u_i \in \mathcal{U} \quad \forall\, i = 0, \ldots, N-1 \\
& x_0 = x^{sample} \\
& x_N = x_{N-1}
\end{aligned}
$$

Learning the Viability Kernel

State of the art

- Sample random states x_i
- For each x_i, use TO to compute a label SAFE / UNSAFE
- Train a classifier using supervised learning

Our Idea

- Compute states on the boundary of $\mathscr{V}$
- Learn directly the boundary of $\mathscr{V}$
- Better accuracy and smaller exploration space

$$
\begin{aligned}
\underset{\{x_i\}_0^N, \{u_i\}_0^{N-1}}{\text{maximize}} \quad & \cancel{1} \;\; a^\top x_0 \\
\text{subject to} \quad & x_{i+1} = f(x_i, u_i) \quad \forall i = 0, \ldots, N-1 \\
& x_i \in \mathcal{X},\; u_i \in \mathcal{U} \quad \forall i = 0, \ldots, N-1 \\
& \cancel{x_0 = x^{sample}} \\
& x_N = x_{N-1}
\end{aligned}
$$

Problem formulation

General form

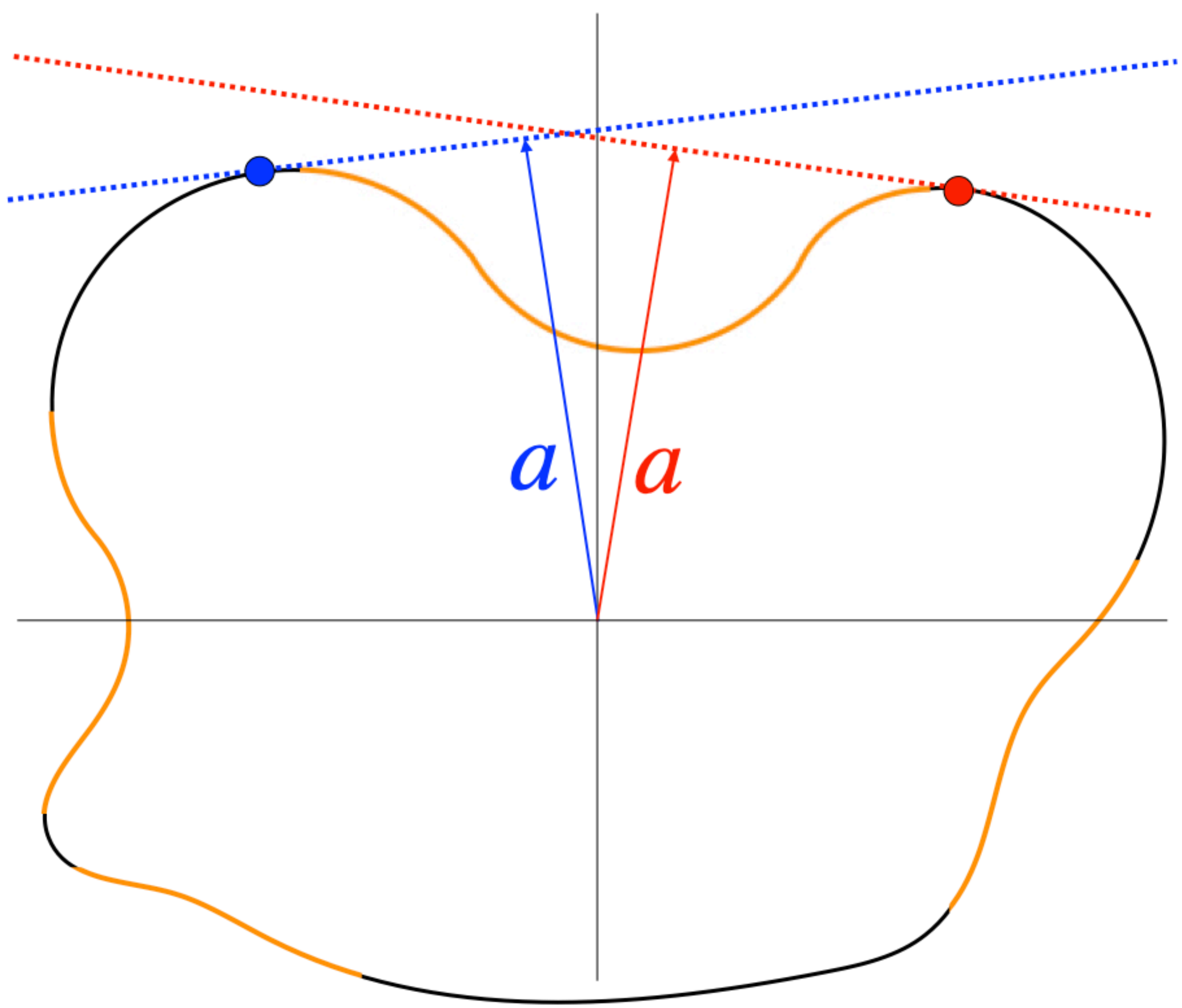
$$
\begin{aligned}
\underset{\{x_i\}_0^N, \{u_i\}_0^{N-1}}{\text{maximize}} \quad & a^\top x_0 \\
\text{subject to} \quad & x_{i+1} = f(x_i, u_i) \quad \forall\, i = 0, \ldots, N-1 \\
& x_i \in \mathcal{X},\; u_i \in \mathcal{U} \quad \forall\, i = 0, \ldots, N-1 \\
& S x_0 = s \\
& x_N = x_{N-1},
\end{aligned}
$$

Lemma

If N is sufficiently long $\longrightarrow$ $x_0^* \in \partial \mathscr{V}$

Complete Coverage?

- Can we completely cover the boundary of $\mathscr{V}$?
  - In general: **NO**!



Start-Convexity

- Assume:
  - the robot can compensate for gravity in any configuration;
  - the set $\mathscr{U}$ is convex.
- Then:
  - $\mathscr{V}$ is **star-convex** w.r.t. $\dot{q}$

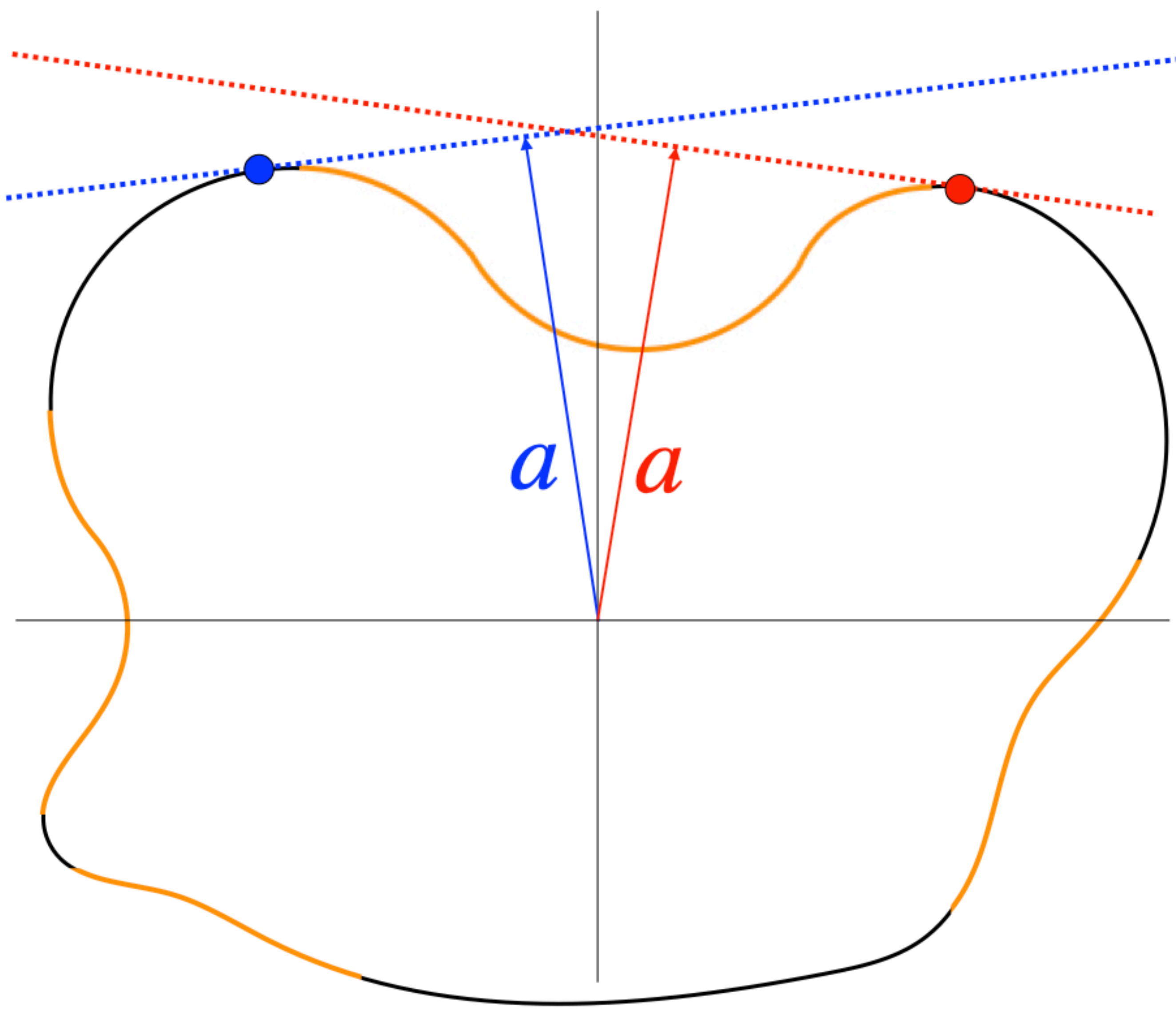
If $(q, \dot{q}) \in \mathscr{V}$

$$\Downarrow$$

$$(q, \alpha\dot{q}) \in \mathscr{V} \quad \forall \alpha \in [0,1]$$

Complete Coverage?

- Can we completely cover the boundary of $\mathscr{V}$?
  - In general: **NO**!
  - If $\mathscr{V}$ is star-convex: **YES**!



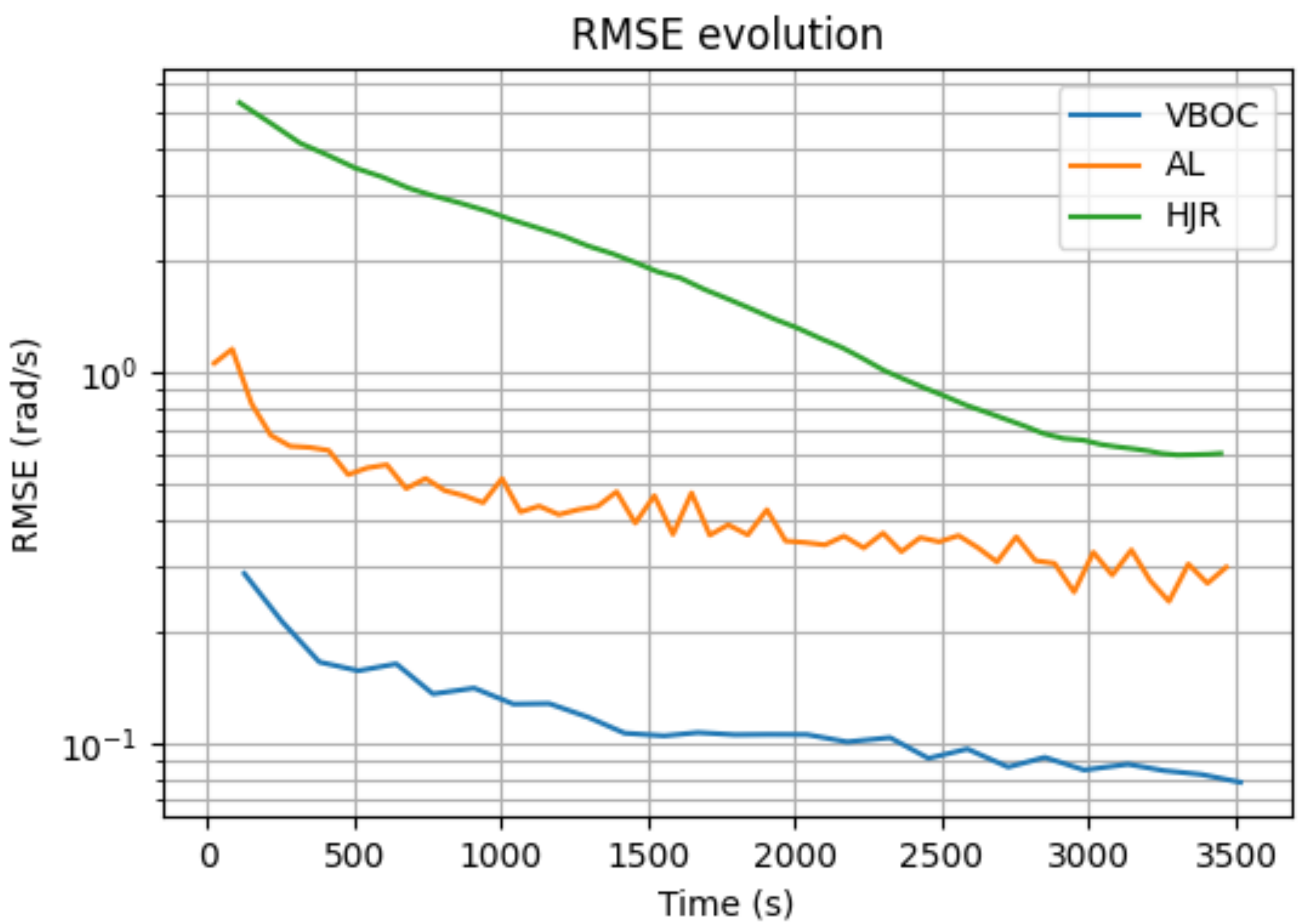
Application to robot manipulators

$$a = \begin{bmatrix} 0 \\ d \end{bmatrix}, \quad S = \begin{bmatrix} I & 0 \\ 0 & I - dd^\top \end{bmatrix}, \quad s = \begin{bmatrix} q^{init} \\ 0 \end{bmatrix}$$

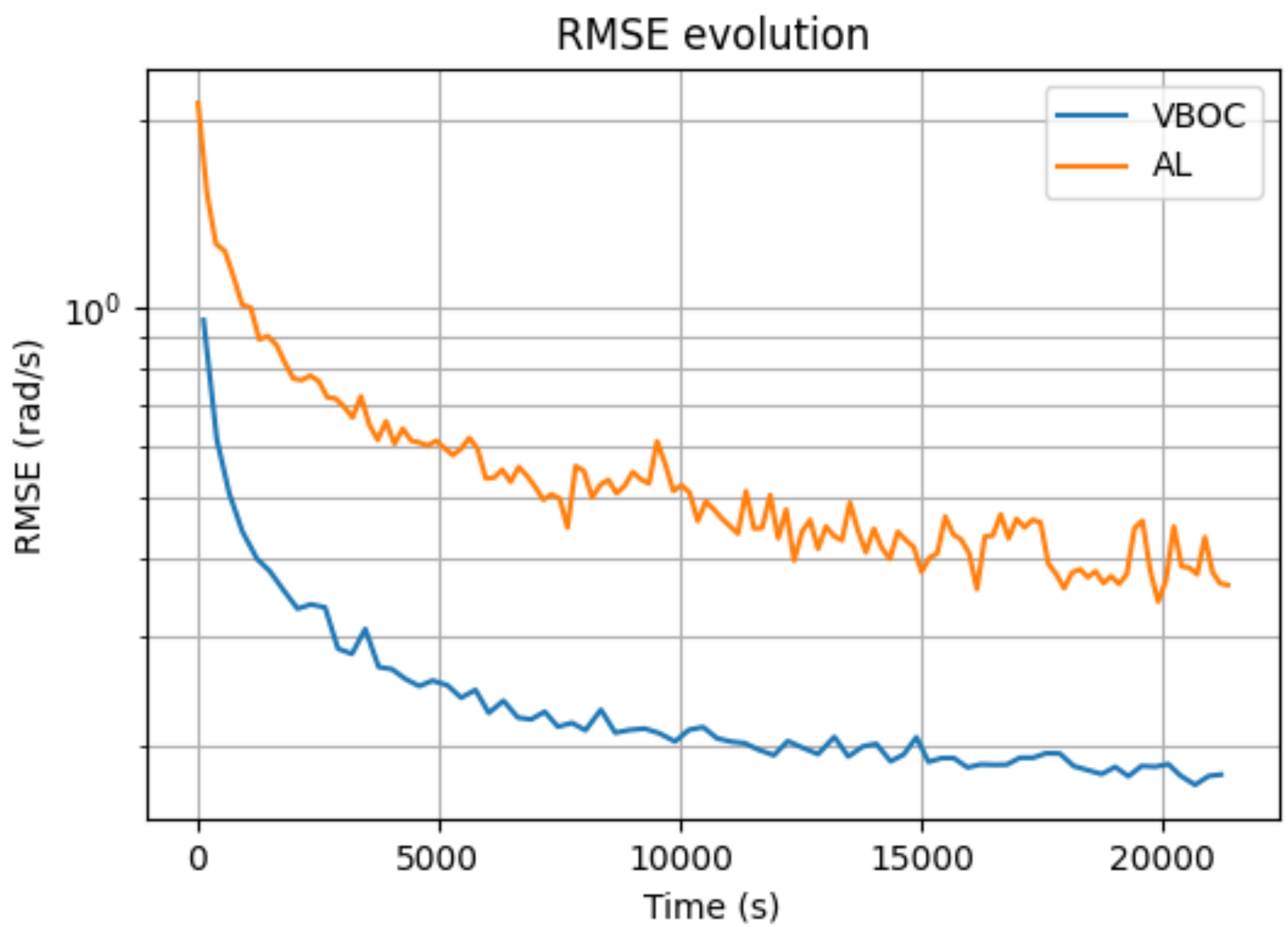
Learning the Viability Kernel

- Sample random "states" (q_i, d_i)
- d = velocity direction
- For each (q_i, d_i), use TO to compute max joint velocity norm v_i
- Use supervised learning to solve regression

2-DoF Manipulator - 1 Hour



3-DoF Manipulator - 6 Hour



Future Work

- Extension to under-actuated robots (no star-convexity)
- Scale to higher dimensions (e.g., exploit GPU)
- Provide guarantees (e.g., inner approximation)
- Account for uncertainties (e.g., dynamics, state)
  - Extension to dynamic obstacles
- More comparison with state of the art (e.g., CBF)

Conclusions

- Complete framework for safe control:
  - Learning approximate safe set (for robot manipulators)
  - Safe control using approximate safe set
- Main limitations:
  - algorithms to compute $\hat{\mathscr{V}}$ do not scale
  - cannot certify set properties (e.g. N-Step Control Invariance)
- Hope: connection with RL

Thank you!

Safe Robot Control

Combining learning and model predictive control

Andrea Del Prete, University of Trento