Research Overview

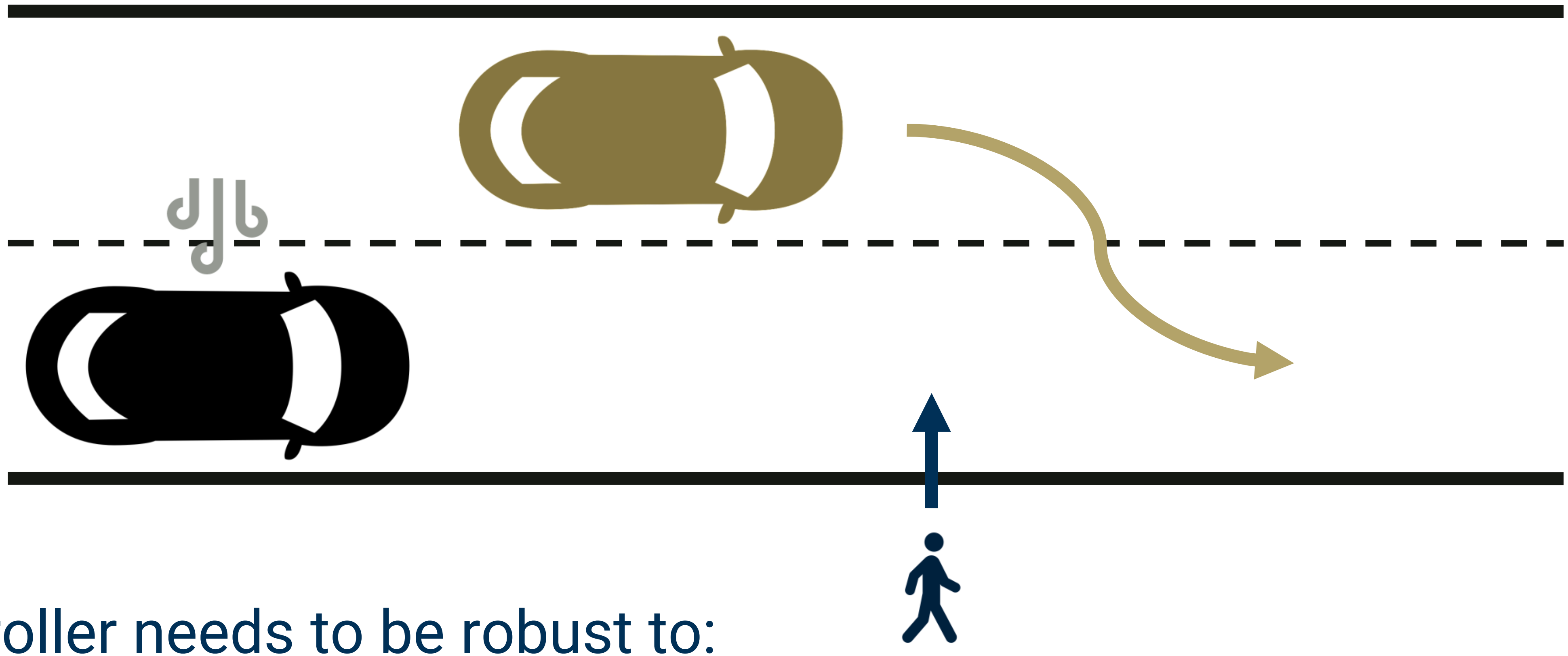
Autonomous Control and Decision Systems (ACDS) Lab
Evangelos Theodorou

Georgia Tech

Outline

- Differentiable Optimization for Robust Model Predictive Control Architectures.
- Safety Embedded Optimal Decision Making and Control via (Tolerant) Barrier States
- Robustifying Perception Against Adversaries.

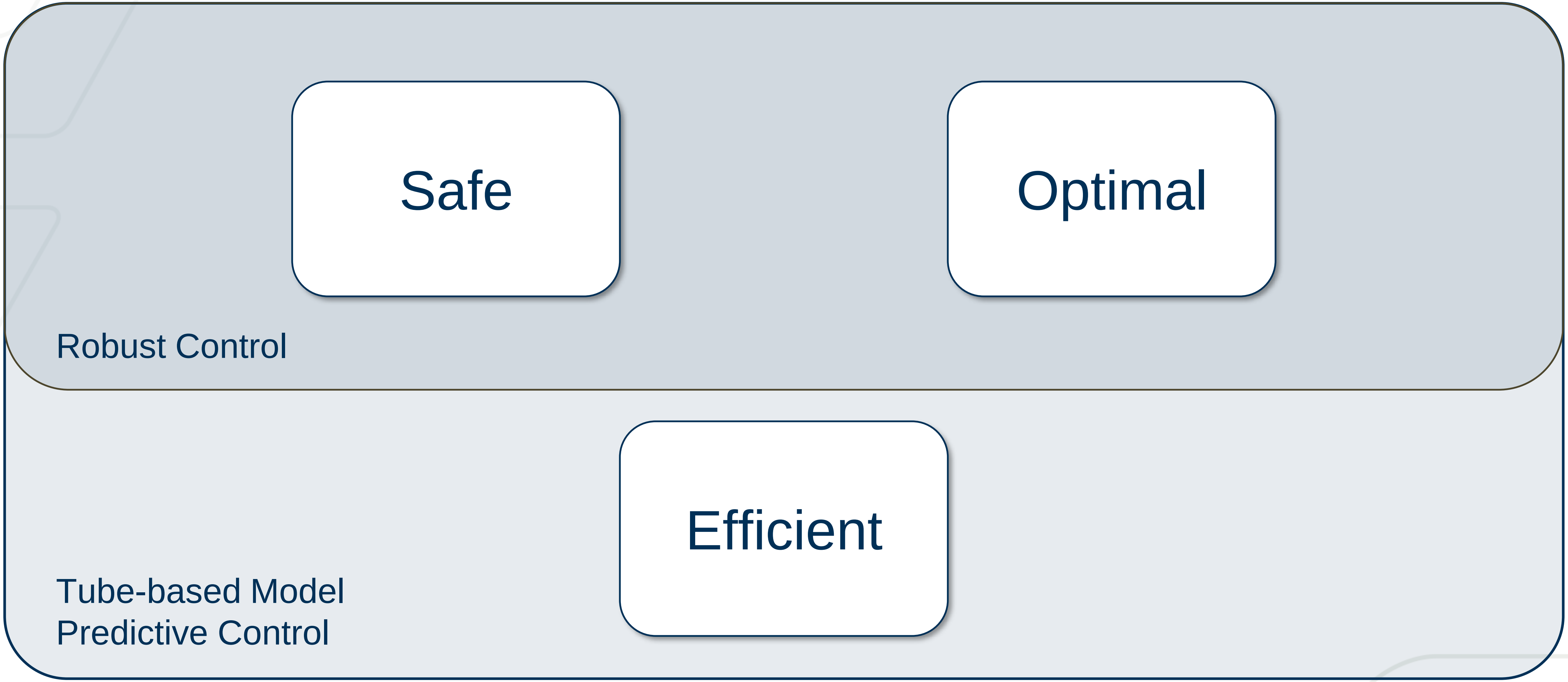
Motivation



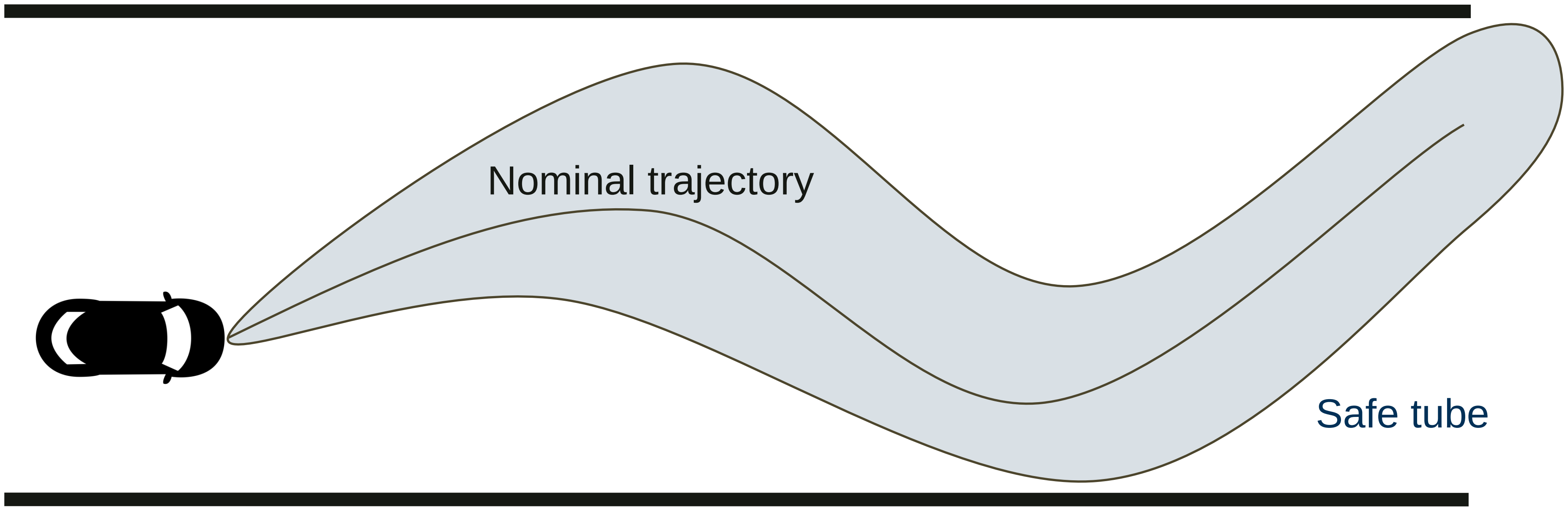
Controller needs to be robust to:

- Uncertainty, disturbances
- Dynamic obstacles in the environment

Controller Criteria



Tube-based Model Predictive Control (Tube-based MPC)



- Nominal controller generates nominal trajectory
- Feedback controller keeps true state safe in the presence of uncertainty
- Note that tube is for theoretical bounds
- Computing the tube is **intractable!**

Georgia Tech

Tube-based Model Predictive Control (Tube-based MPC)

- Downsides of traditional tube-based MPC:
  - Need to know uncertainty *a priori*
  - Does not respond to the environment
  - Difficult to tune – parameters have nonlinear effect on tube shape and size
- Our work: **Differentiable** Tube-based MPC (DT-MPC) [1]
  - Tune controller parameters online and in real-time, but in a principled (optimal) manner
  - Responds to the environment, without knowing disturbances a priori
  - Enforces safety at all timesteps using efficient state constraint satisfaction methodology (discrete barrier states [2])

[1] Oshin, A., & Theodorou, E. A. (2023). Differentiable Robust Model Predictive Control. *arXiv preprint arXiv:2308.08426*.
[2] Almubarak, H., Stachowicz, K., Sadegh, N., & Theodorou, E. A. (2022). Safety Embedded Differential Dynamic Programming using Discrete Barrier States. *IEEE Robotics and Automation Letters*, *7*(2), 2755-2762.

Theoretical Foundations of Our Work

- Optimal control algorithm is a function of the problem parameters:

$$z^*(\theta) = \mathrm{OC}(\theta) \longleftarrow \text{Solver (iLQR, DDP, etc.)}$$

- How to choose parameters “optimally”?
- Define task-based loss function → bilevel optimization problem

Upper-level $$\min_{\theta} L(z^*(\theta))$$

Lower-level $$z^*(\theta) = \mathrm{OC}(\theta)$$

- How to compute gradient $\nabla_\theta L(z^*(\theta))$ efficiently?

Georgia Tech

Theoretical Foundations of Our Work

- Key insight: gradients can be computed by solving a control problem!

$$\nabla_\theta L(z^*(\theta)) = \left(\frac{\mathrm{d}z^*(\theta)}{\mathrm{d}\theta}\right)^\top \nabla_z L(z^*(\theta))$$

Theorem 2.1 (Implicit function theorem (IFT) [29]). *Let $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ be a continuously differentiable function. Fix a point (z_0, θ_0) such that $F(z_0, \theta_0) = 0$. If the Jacobian matrix of partial derivatives $\frac{\partial F}{\partial z}(z_0, \theta_0)$ is invertible, then there exists a function $z^*(\cdot)$ defined in a neighborhood of θ_0 such that $z^*(\theta_0) = z_0$ and*

$$\frac{\mathrm{d}}{\mathrm{d}\theta} z^*(\theta) = -\left(\frac{\partial}{\partial z} F(z^*(\theta), \theta)\right)^{-1} \frac{\partial}{\partial \theta} F(z^*(\theta), \theta).$$

- Choice of F?

Georgia Tech

Theoretical Foundations of Our Work

Claim 2.2 (Implicit derivative of Problem 1). *Let τ^* be a solution to Problem 1 with parameters θ. Then, there exists Lagrange multipliers λ^* which together with τ^* satisfy $\nabla_z \mathcal{L}(z^*, \theta) = 0$ and Theorem 2.1 holds with $F = \nabla_z \mathcal{L}$. Furthermore, the Jacobian is given as*

$$\frac{\mathrm{d}}{\mathrm{d}\theta} z^*(\theta) = -\mathcal{L}_{zz}^{-1} \mathcal{L}_{z\theta}. \tag{6}$$

- Generalize previous work [3-5] showing a second-order approximation is necessary to compute accurate derivatives

$$\nabla_\theta L(z^*(\theta)) = \left(\frac{\mathrm{d}z^*(\theta)}{\mathrm{d}\theta} \right)^\top \nabla_z L(z^*(\theta)) = -\mathcal{L}_{\theta z} \underbrace{\mathcal{L}_{zz}^{-1} \nabla_z L(z^*(\theta))}_{\text{Solving this system is equivalent to an iteration of DDP!}}$$

[3] Amos, B., Jimenez, I., Sacks, J., Boots, B., & Kolter, J. Z. (2018). Differentiable MPC for End-to-End Planning and Control. *Advances in Neural Information Processing Systems*, *31*.
[4] Dinev, T., Mastalli, C., Ivan, V., Tonneau, S., & Vijayakumar, S. (2022). Differentiable Optimal Control via Differential Dynamic Programming. *arXiv preprint arXiv:2209.01117*.
[5] Jin, W., Wang, Z., Yang, Z., & Mou, S. (2020). Pontryagin Differentiable Programming: An End-to-End Learning and Control Framework. *Advances in Neural Information Processing Systems*, 33, 7979-7992.

Differentiable Optimal Control

- Gradients taken with respect to problem parameters “for free”
  - Only requires one additional iteration of the optimizer (e.g., DDP)
  - Reuses matrix factorization from final iteration
- Contrast to automatic differentiation, which requires full unrolling of the optimizer iterations ($O(K)$ where K is the number of solver iterations)
- Learnable parameters:
  - Nominal controller cost function weights
  - Sensitivity to obstacles $\rightarrow$ determines tube size
  - Model parameters (unknown coefficients, parameters of a NN model, etc.)

Tube-based MPC

Nominal control problem

$$\bar{\tau}(\bar{\theta}) = \arg\min_{\tau} \bar{J}(\tau, \bar{\theta}) = \arg\min_{\tau} \sum_{t=0}^{T-1} \bar{\ell}(x_t, u_t, \bar{\theta}) + \bar{\phi}(x_T, \bar{\theta}),$$

$$\text{subject to} \;\; x_{t+1} = f(x_t, u_t), \;\; \forall t = 0, \ldots, T-1, \quad x_0 = \bar{\xi},$$

$$x_t \in \mathbb{Z}(\bar{\theta}) \subset \mathbb{X}, \;\; \forall t = 0, \ldots, T,$$

Ancillary control problem

$$\tau^*(\theta) = \arg\min_{\tau} J(\tau, \bar{\tau}, \theta) = \arg\min_{\tau} \sum_{t=0}^{T-1} \ell(x_t - \bar{x}_t, u_t - \bar{u}_t, \theta) + \phi(x_T - \bar{x}_T, \theta),$$

$$\text{subject to} \;\; x_{t+1} = f(x_t, u_t), \;\; \forall t = 0, \ldots, T-1, \quad x_0 = \xi,$$

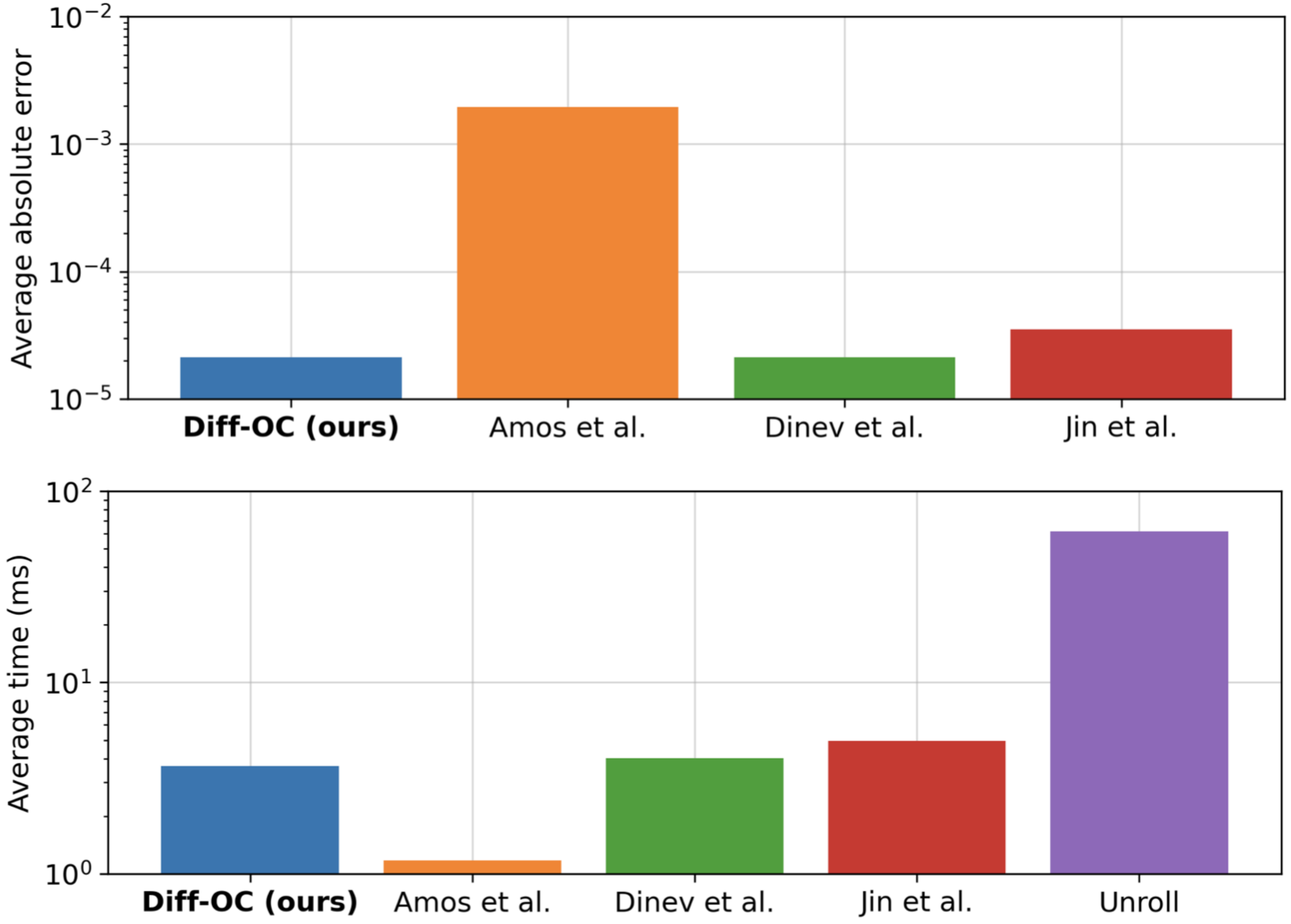
$$x_t \in \mathbb{X}, \;\; \forall t = 0, \ldots, T.$$

Connections with Prior Works

- Differentiable MPC – Amos et al. (2018) [3]
  - Only iLQR approximation -> gradients are incorrect
  - Requires matrix calculus
- Dinev et al. (2022) [4]
  - DDP approximation
  - Requires matrix calculus
- Pontryagin Differentiable Programming – Jin et al. (2020) [5]
  - Requires solving matrix control system -> slow
- Our work:
  - Agnostic of control solver
  - Quadratic approximation is *necessary* to compute accurate gradients (IFT)

[3] Amos, B., Jimenez, I., Sacks, J., Boots, B., & Kolter, J. Z. (2018). Differentiable MPC for End-to-end Planning and Control. *Advances in Neural Information Processing Systems*, *31*.
[4] Dinev, T., Mastalli, C., Ivan, V., Tonneau, S., & Vijayakumar, S. (2022). Differentiable Optimal Control via Differential Dynamic Programming. *arXiv preprint arXiv:2209.01117*.
[5] Jin, W., Wang, Z., Yang, Z., & Mou, S. (2020). Pontryagin Differentiable Programming: An End-to-end Learning and Control Framework. *Advances in Neural Information Processing Systems*, *33*, 7979-7992.

Comparison with Prior Works



- Inverse optimal control objective – learn cost function weights that generate expert behavior (17 parameters)
- Quadrotor dynamics (12 states, 4 controls)
- Note log scale on y-axis

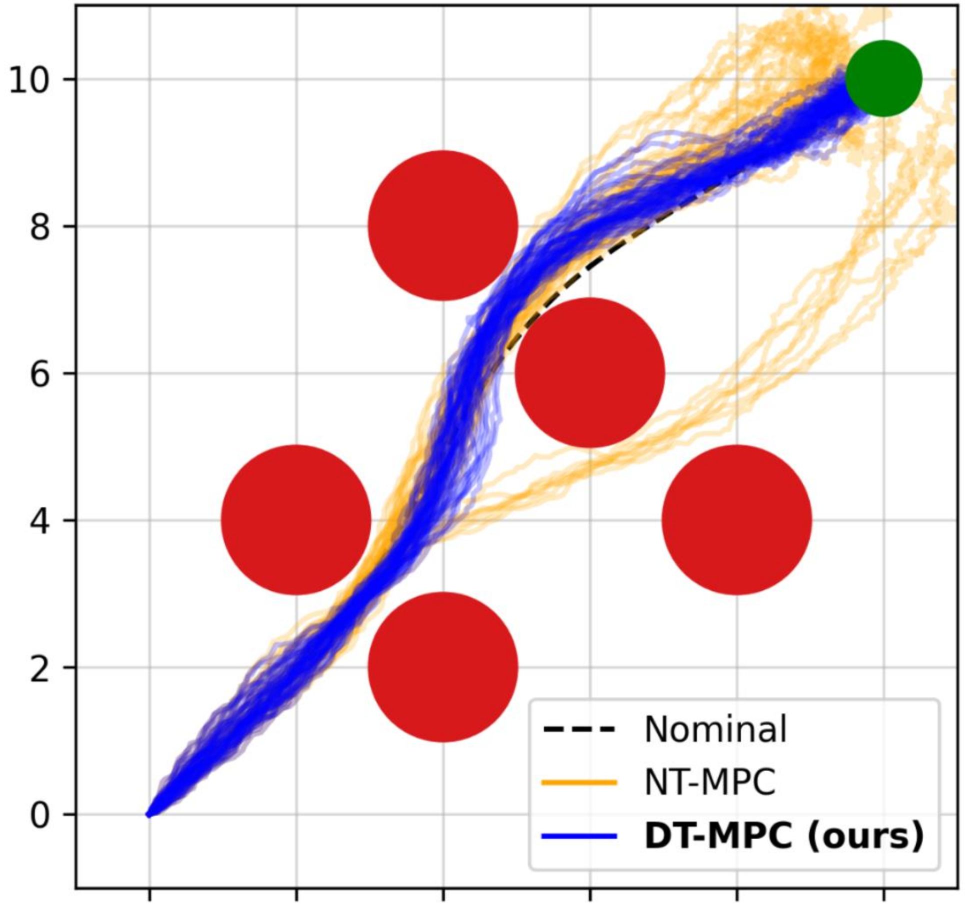
Amos, B., Jimenez, I., Sacks, J., Boots, B., & Kolter, J. Z. (2018). Differentiable MPC for End-to-End Planning and Control. *Advances in Neural Information Processing Systems*, *31*.
Dinev, T., Mastalli, C., Ivan, V., Tonneau, S., & Vijayakumar, S. (2022). Differentiable Optimal Control via Differential Dynamic Programming. *arXiv preprint arXiv:2209.01117*.
Jin, W., Wang, Z., Yang, Z., & Mou, S. (2020). Pontryagin Differentiable Programming: An End-to-End Learning and Control Framework. *Advances in Neural Information Processing Systems*, *33*, 7979-7992.

Experimental Results – Summary

	Dubins Vehicle		Quadrotor		Robot Arm	
	Successes	Collisions	Successes	Collisions	Successes	Collisions
NT-MPC	14%	0%	14%	20%	0%	56%
DT-MPC (ours)	**100%**	0%	**76%**	**4%**	**78%**	**10%**

- Success: reach target
- Collision: hit an obstacle, leave environment bounds
- Disturbances:
  - **~1/2 the max control magnitude** (discrete-time) for Dubins/Quadrotor
  - **5x the max control magnitude for the robot arm!**
  - Sampled uniformly

Experimental Results



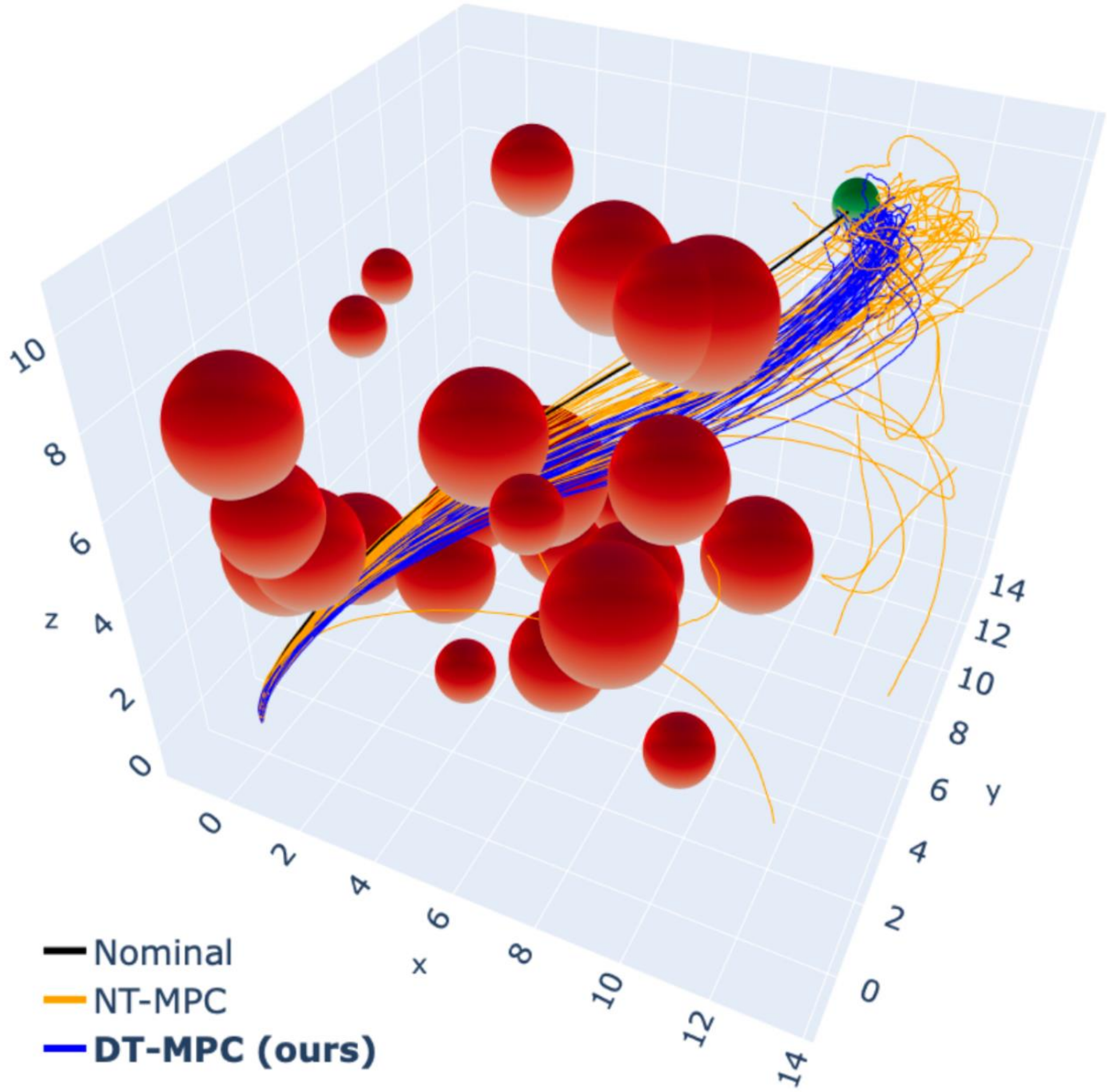
Dubins Vehicle Task

DT-MPC:

- Safer while completing the task with higher probability
- Emergent behavior – tube size and shape is adapted based on disturbances encountered
- Max velocity control: 0.1 m s^{-1}
- Max turning rate: ~0.03 rad s^{-1}
- Max disturbance magnitude: 0.05 s^{-1}

Georgia Tech

Experimental Results

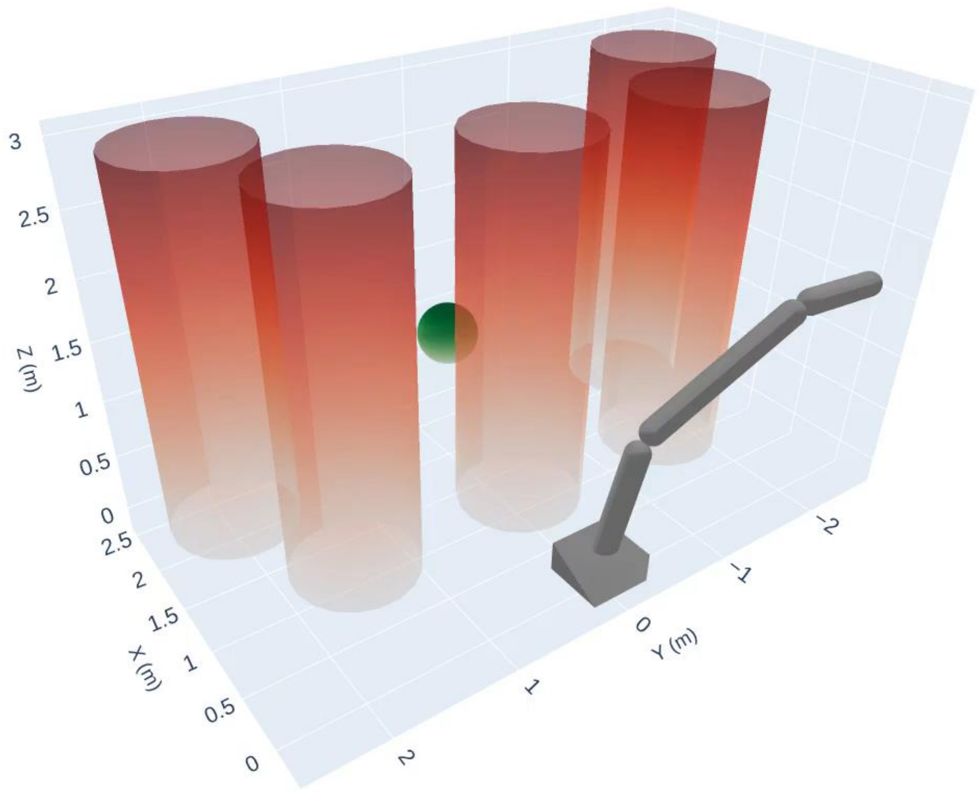


Quadrotor Task

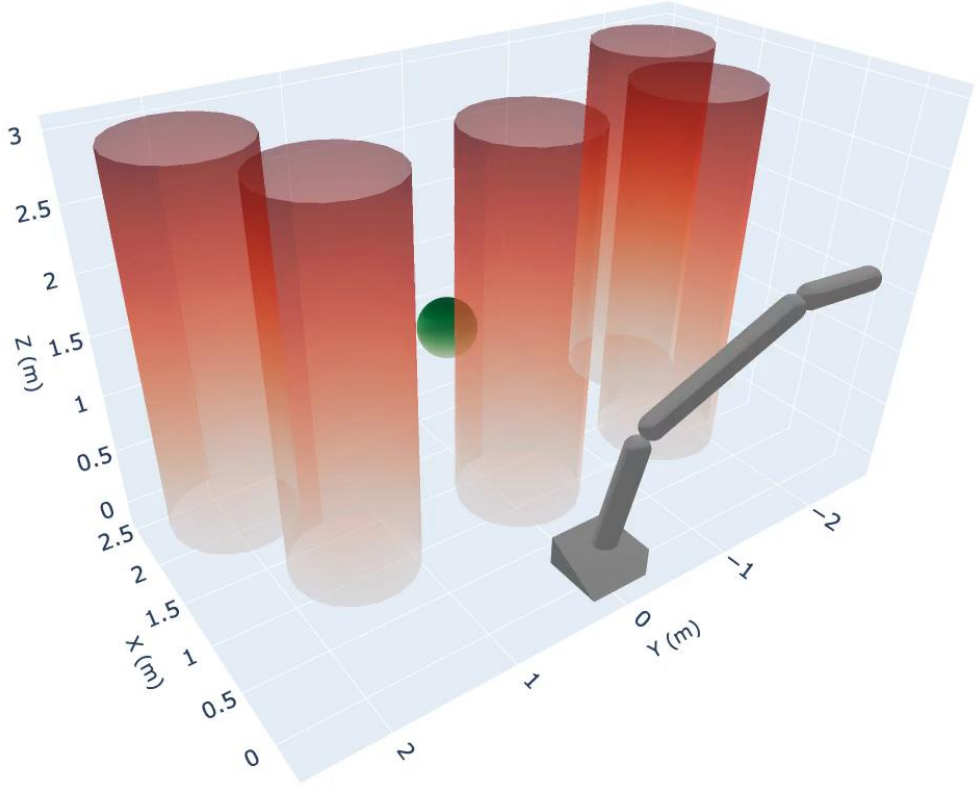
DT-MPC:

- Robust to very large disturbances
- Max controls (roll-pitch-yaw): 0.2 Nm
- Max disturbance magnitude: 0.1 Nm

Experimental Results



NT-MPC



DT-MPC

- Max control: ~0.02 Nm
- **Max disturbance: 0.1 Nm**

Outline

- Differentiable Optimization for Robust Model Predictive Control Architectures.

- Safety Embedded Optimal Decision Making and Control via (Tolerant) Barrier States

- Robustifying Perception Against Adversaries.

Safety Embedded Optimal Decision Making and Control via (Tolerant) Barrier States

Hassan Almubarak
Autonomous and Control Decision Systems Lab
Georgia Institute of Technology

V(t,x) Autonomous Control & Decision Systems Laboratory

Georgia Tech

Safety-critical Control Problem

We consider the safety-critical dynamical system

$$\dot{x} = f(t, x, u)$$

where $x \in \mathcal{D} \subset \mathbb{R}^n, u \in \mathcal{U} \subset \mathbb{R}^m, f \in C^1(\mathcal{D} \times \mathcal{U}, \mathcal{D}), f(0) = 0$ (w.l.o.g).

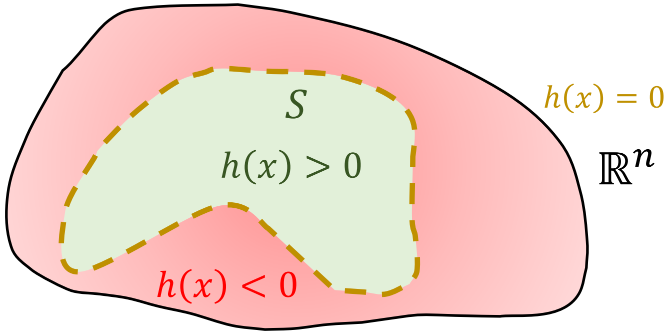
subject to

$$h(t, x) > 0\ \forall t; \forall x_0 \in S \subset \mathbb{R}^n$$

where $S = \{x \in \mathbb{R}^n : h(t, x) > 0\}$.

Safety Condition

safe set

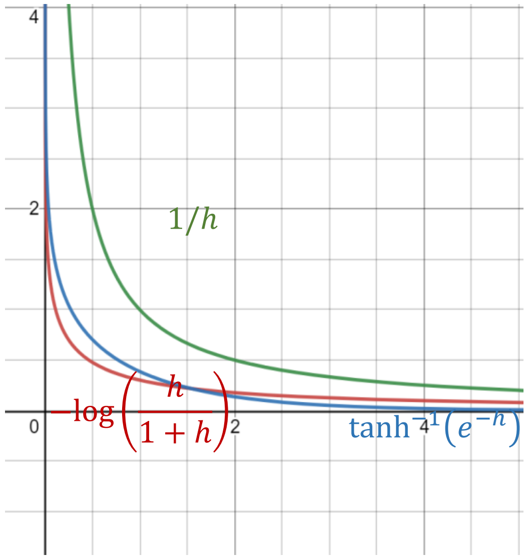


Goal: Compute a feedback control policy $U^*(x)$ that achieves performance objectives while rendering the nonempty set S controlled invariant over the whole horizon.

Georgia Tech

Barrier States (BaS)

Barrier function $B \in C^{\infty}(S, \mathbb{R})$:
$$\lim_{a \to 0} B(a) = \infty, \ \lim_{a \to \infty} B(a) = 0, \ \lim_{a \in \mathbb{R}^+} B(a) \geq 0.$$



Barrier over the state $\beta(x) := B \circ h(x)$.

$\Rightarrow \beta(x) \to \infty$ if and only if $h(x) \to 0$.

Barrier function ovulation over time:

$$\dot{\beta}(x) = B'\big(h(x)\big)\left(L_{f(x,u)} h(x)\right)$$

Idea: augment the state equation of the barrier to the model of system:

$$\boxed{\begin{aligned} \dot{x} &= f(x,u) \\ \dot{\beta} &= f_\beta(x, \beta, u) \end{aligned}} \rightarrow \text{Safety Embedded System}$$

New model:

$$\dot{\bar{x}} = \bar{f}\left(\bar{x}, u\right)$$

where $\bar{x} = \begin{bmatrix} x \\ \beta \end{bmatrix}, \bar{f} = \begin{bmatrix} f \\ f_\beta \end{bmatrix}$.

Georgia Tech

Safety Embedded Regulation

Define $z := \beta(x) - \beta(0) \Rightarrow z(0) = 0$.

Stabilizable Barrier state:

$$\dot{z} = f_z := B'(B^{-1}(z + \beta_0))\left(L_{f(x,u)}h(x)\right) - \gamma\left(z + \beta_0 - \beta(x)\right)$$

Hence, the safety embedded system:

$$\dot{\bar{x}} = \bar{f}\left(\bar{x}, u\right)$$

where $\bar{x} = \begin{bmatrix} x \\ z \end{bmatrix}, \bar{f} = \begin{bmatrix} f \\ f_z \end{bmatrix}$.

Results:

- Assume: stabilizing continuous feedback controller $u = K\left(\bar{x}\right)$ for $\dot{\bar{x}} = \bar{f}(\bar{x}, u)$.
- Then, $u = K(\bar{x})$ is safe with respect to the safety region $S = \{x \in \mathcal{D}: h(x) > 0\}$ (safely stabilizes the origin of the original safety-critical system).

Illustrative Example

Consider the open loop unstable linear system given by

$$\dot{x} = \begin{bmatrix} 1 & -5 \\ 0 & -1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

subject to $\mathcal{C} = \{x : (x_1 - 2)^2 + (x_2 - 2)^2 - 0.5^2 > 0\}\ \forall t > 0$ given that $x(0) \in \mathcal{C}$ with desired closed-loop system's poles at -3 and -5.

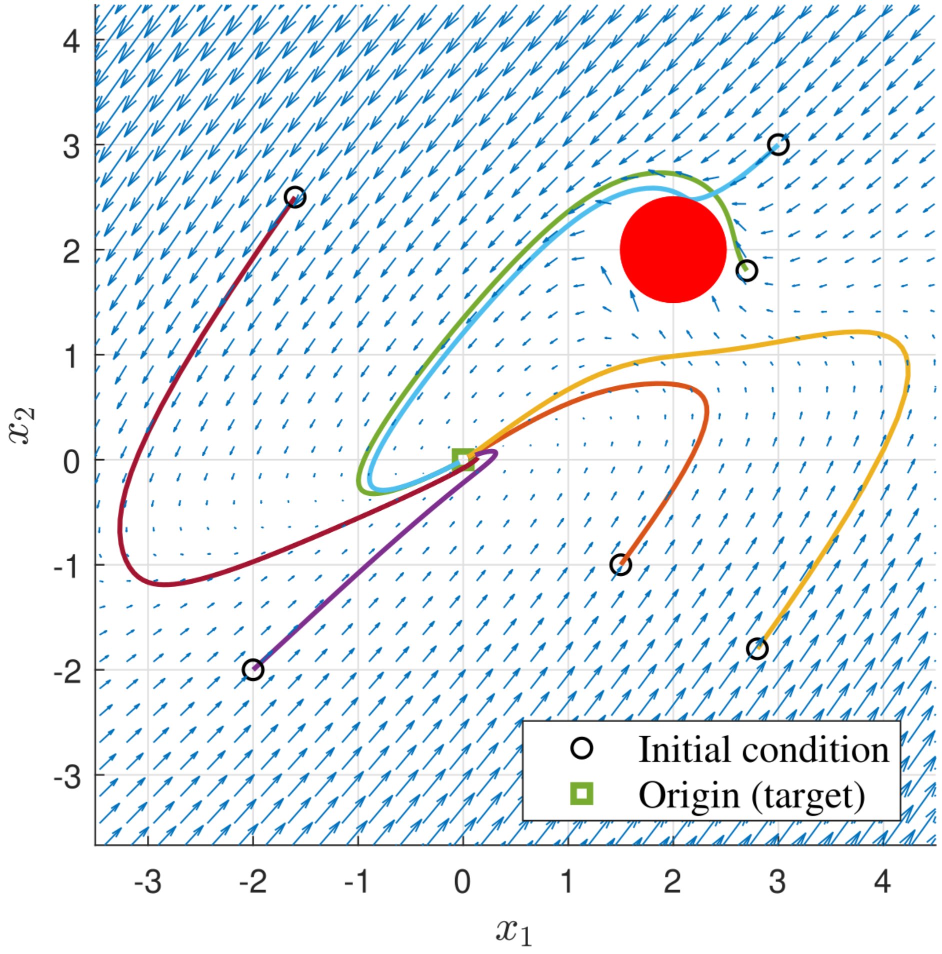
Using the inverse barrier function, we define the BaS and linearizing the system yields

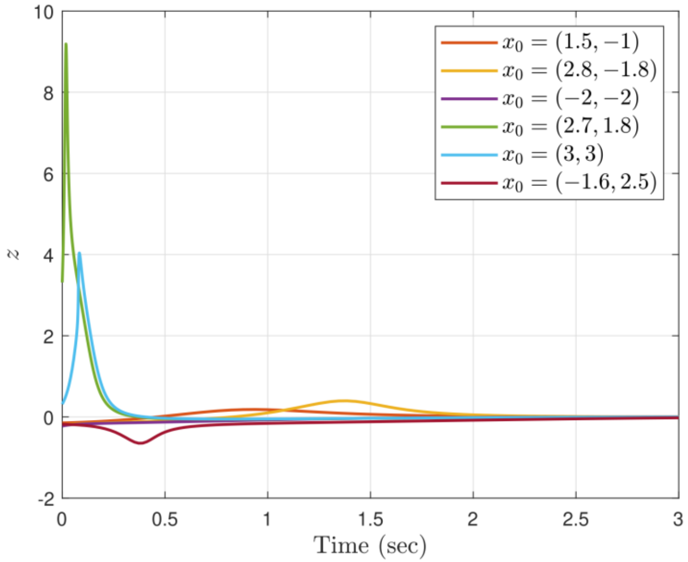
$$\dot{\bar{x}} = \begin{bmatrix} 1 & -5 & 0 \\ 0 & -1 & 0 \\ \frac{4\gamma + 4}{7.75^2} & \frac{4\gamma - 24}{7.75^2} & -\gamma \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \\ \frac{4}{7.75^2} \end{bmatrix} u$$

Using pole-placement, the safe stabilizing controller is

Nonlinear controller! $u = -4.43 x_1 + 8.38\, x_2 - 5.63\, z$

Constrained Linear Control Example





Safety Embedded Optimal Control

Consider the optimal control problem

$$V\big(x(0)\big) = \min_u \frac{1}{2}\int_0^\infty Q(x) + u^\intercal R u \, dt$$

subject to $\dot{x} = f(x) + g(x)u$ and $\mathcal{C} = \{x \in \mathcal{D}: h(x) > 0\}\ \forall t \geq 0.$

Embed BaS :

$$V\left(\bar{x}(0)\right) = \min_u \frac{1}{2}\int_0^\infty Q\left(\bar{x}\right) + u^\intercal R u \, dt \text{ subject to } \dot{\bar{x}} = \bar{f}\left(\bar{x}\right) + \bar{g}\left(\bar{x}\right)u$$

Hamilton-Jacobi-Bellman (HJB) equation is

$$\min_u V_{\bar{x}}^{*}\left(\bar{f}\left(\bar{x}\right) + \bar{g}\left(\bar{x}\right)u\right) + \frac{1}{2}u^\intercal R u + \frac{1}{2}Q\left(\bar{x}\right) = 0$$

Results:

- Assume there exists a unique analytic value function $V^*\left(\bar{x}\right)$ satisfying the HJB equation
- Then
  - The optimal safe feedback control is $u^*_{safe}\left(\bar{x}\right) = -R^{-1}\bar{g}\left(\bar{x}\right)^\intercal V_{\bar{x}}^{*}\left(\bar{x}\right)$
  - $V^*\left(\bar{x}\right)$ is a Lyapunov function and $u^*_{safe}\left(\bar{x}\right)$ renders the embedded system's origin asymptotically stable.
  - The barrier state Z is bounded guaranteeing the generation of safe trajectories.

Expanding the dynamic programming principle about a nominal trajectory $(\tilde{\bar{x}}, \tilde{u})$

$$V_k\left(\bar{x}_k\right) = \min_{u_k}\left[l\left(k, \bar{x}_k, u_k\right) + V_{k+1}\left(\bar{f}\left(k, \bar{x}_k, u_k\right)\right)\right]$$

We get a variation function H. Recursively compute the local second order model of V and the control gains in the backward pass:

$$V_k = V_{k+1} - \frac{1}{2} H_{u_k} H_{uu_k}^{-1} H_{u_k}^{\intercal},\ V_{\bar{x}_k} = H_{\bar{x}_k} - H_{\bar{x}u_k} H_{uu_k}^{-1} H_{u_k},\ V_{\bar{x}\bar{x}_k} = \frac{1}{2}\left(H_{\bar{x}\bar{x}_k} - H_{\bar{x}u_k} H_{uu_k}^{-1} H_{u\bar{x}_k}\right)$$

where

$$H_{\bar{x}_k} = l_{\bar{x}_k} + V_{\bar{x}_{k+1}}^{\intercal} \bar{f}_{\bar{x}_k},\ H_{u_k} = l_{u_k} + V_{\bar{x}_{k+1}}^{\intercal} \bar{f}_{u_k},\ H_{\bar{x}\bar{x}_k} = l_{\bar{x}\bar{x}_k} + \bar{f}_{\bar{x}_k}^{\intercal} V_{\bar{x}\bar{x}_{k+1}} \bar{f}_{\bar{x}_k} + V_{\bar{x}_{k+1}} \bar{f}_{\bar{x}\bar{x}_k},$$

$$H_{uu_k} = l_{uu_k} + \bar{f}_{u_k}^{\intercal} V_{\bar{x}\bar{x}_{k+1}} \bar{f}_{u_k} + V_{\bar{x}_{k+1}} \bar{f}_{uu_k},\ H_{\bar{x}u_k} = l_{\bar{x}u_k} + \bar{f}_{\bar{x}_k}^{\intercal} V_{\bar{x}\bar{x}_{k+1}} \bar{f}_{u_k} + V_{\bar{x}_{k+1}} \bar{f}_{\bar{x}u_k}$$

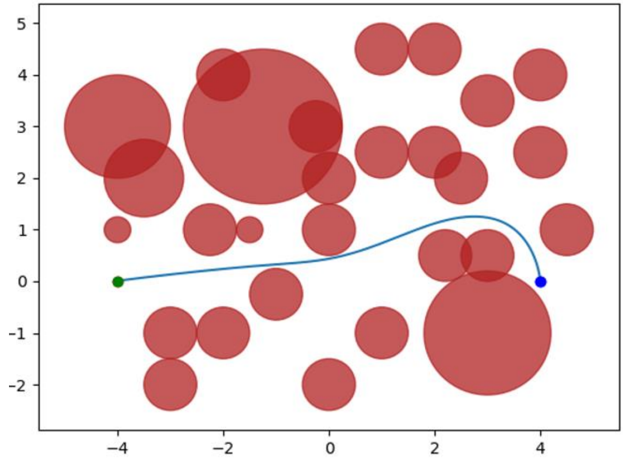
The feedforward and feedback control gains $\mathbf{k}_k = H_{uu_k}^{-1} H_{u_k}$ and $\mathbf{K}_k = H_{uu_k}^{-1} H_{u\bar{x}_k}$.

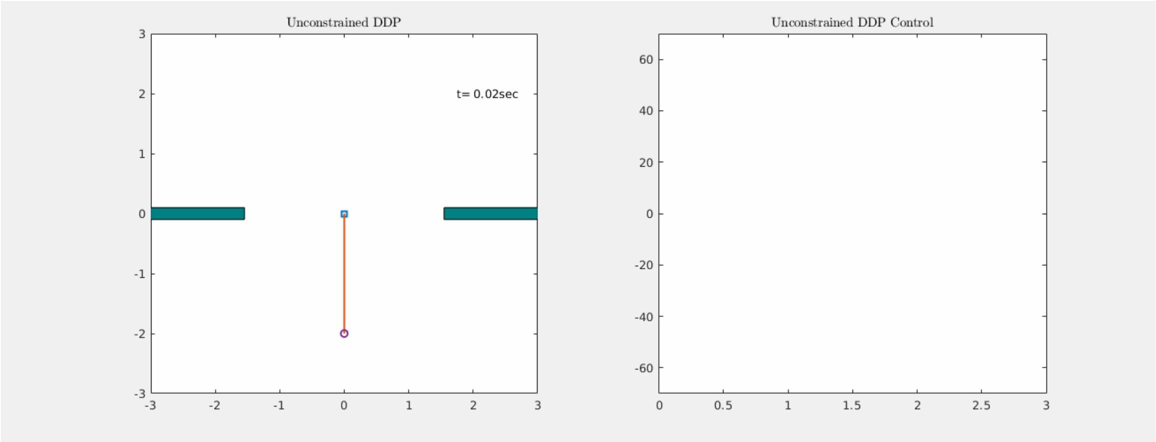
Then the forward pass consists of $\quad \delta u_k^* = \mathbf{k}_k + \mathbf{K}_k \delta\tilde{\bar{x}}_k$

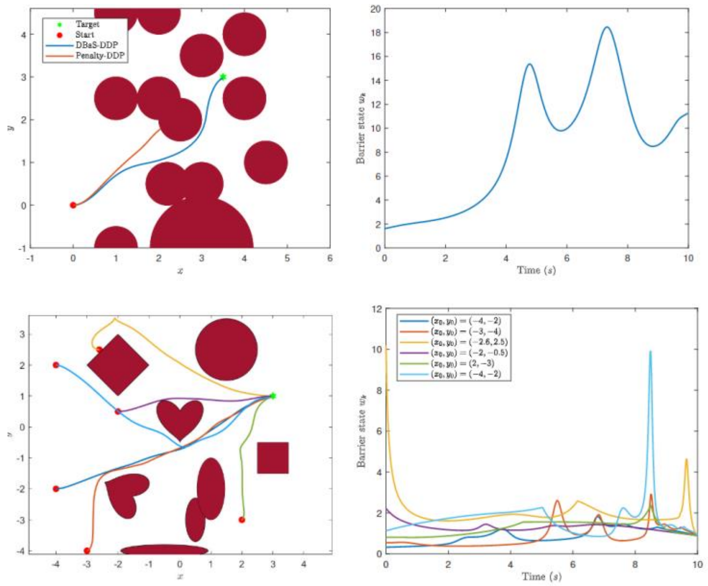
$$u_k = \tilde{u} + \delta u_k^*$$

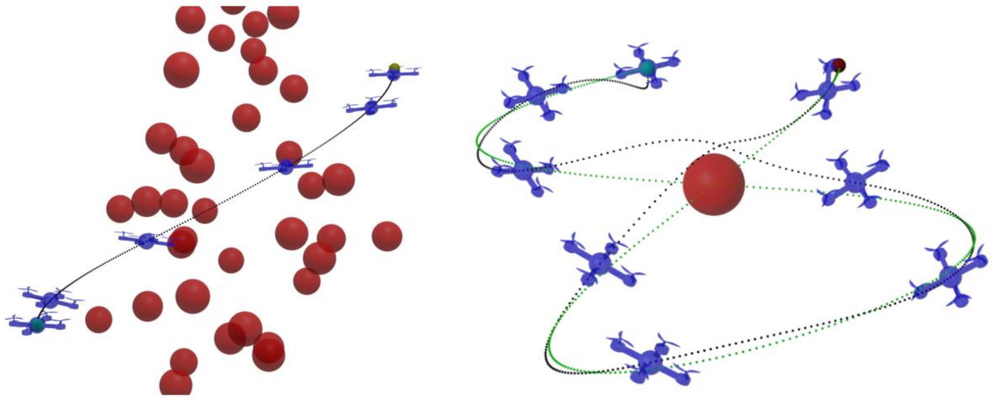
$$\bar{x}_{k+1} = \hat{\bar{f}}\left(k, \bar{x}_k, u_k\right)$$

The backward passes and forward passes are iterated until convergence.

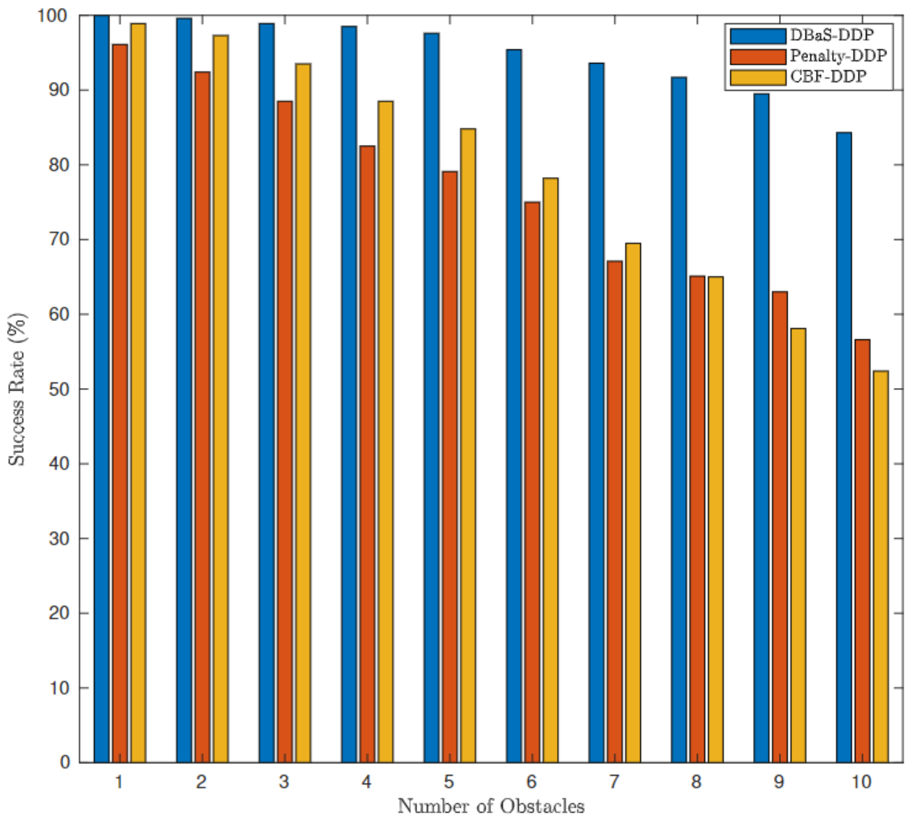



Unconstrained DDP
t= 0.02sec
Unconstrained DDP Control

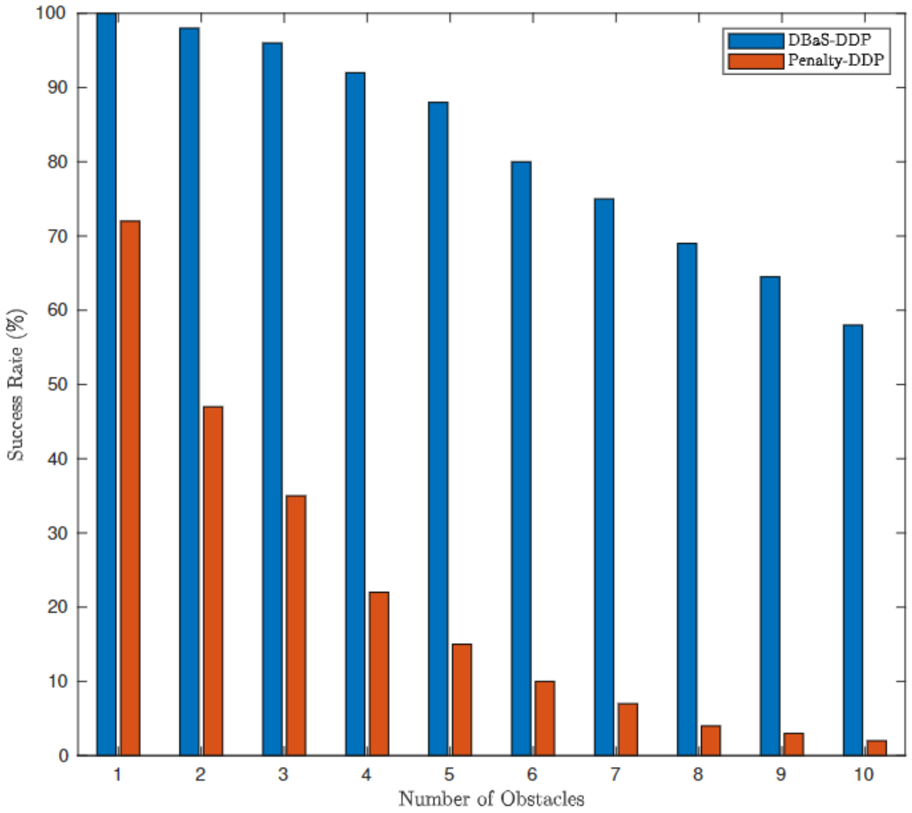

Target
Start
DBaS-DDP
Penalty-DDP
Time (s)
Time (s)



Double Integrator (point robot)



Differential wheeled robot

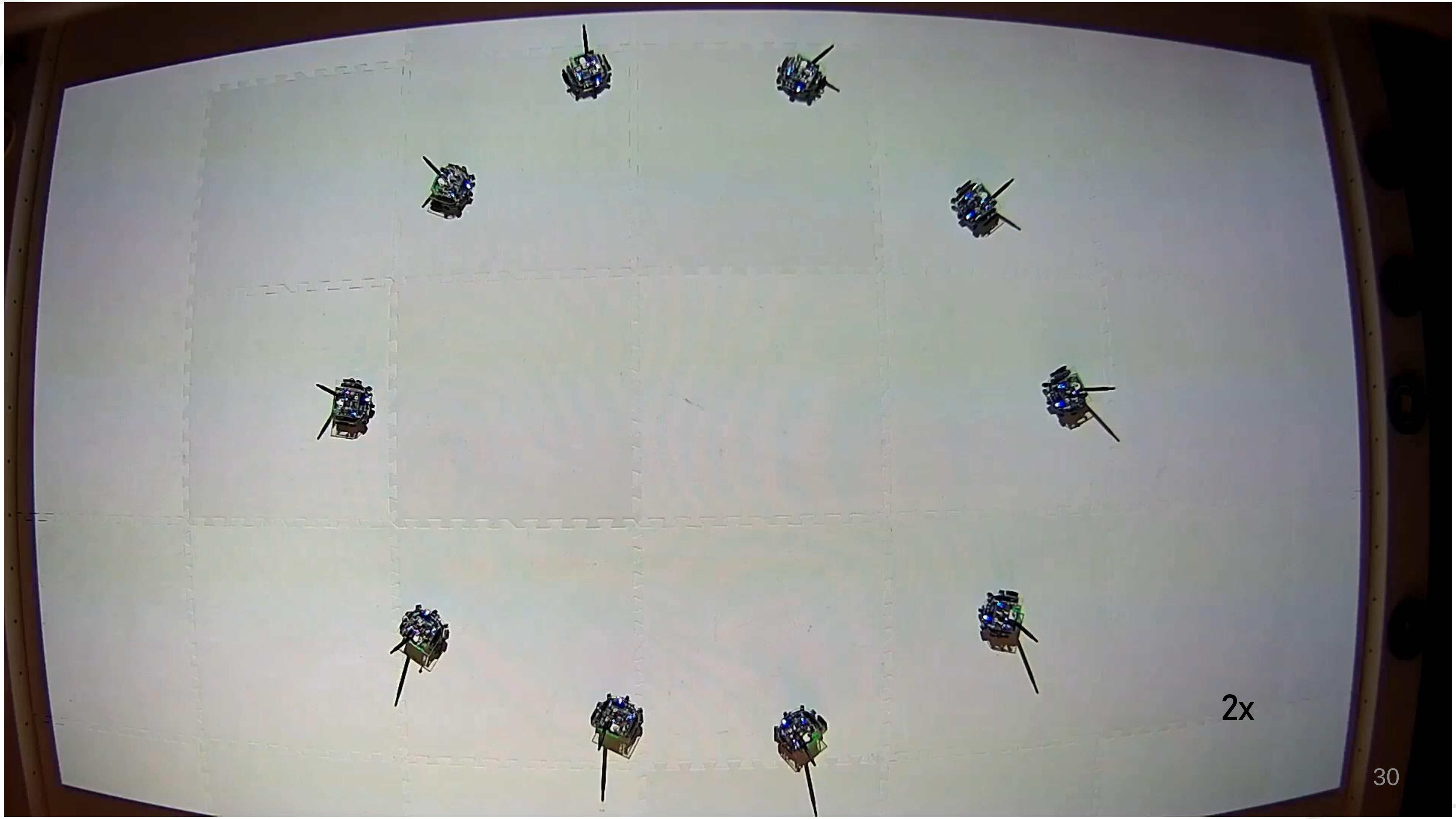


Penalty refers to adding the barrier directly to the cost function.

Success rate is the number of percentage of trajectories that *reach* the target *safely*.

Georgia Tech

a. Switch positions while avoiding collision - Robotarium



What are the issues of DBaS?
(in trajectory optimization)

Embedded Barrier States

- Provide safety guarantees and convergence
- Relative-degree requirement of CBF is avoided
- Provide feedback policies of the barrier that enhance safety and robustness
- Easy to tune
- Prone to local minima around the unsafe regions/obstacles
- Limited exploration (only feasible solutions are allowed)
  → may limit local iterative algorithms to converge to a meaningful minima

Augmented Lagrangian

- Rely on treating the constraints as soft ones by incorporating them into the cost function
- Allow for unsafe trajectory initialization
- Allow for intermediate solutions to be partially unsafe, which enhances their ability to arrive at nontrivial solutions.
- Final solution might also be unsafe
- Require additional parameters that need considerable tuning

Georgia Tech

Solution: Tolerant-BaS

Embedded Tolerant Barrier States

$$\sigma(h) = \frac{1}{1+e^{c_1 h}} \qquad \sigma^+(h) = \frac{1}{c_2}\log(1+e^{-c_2 h})$$

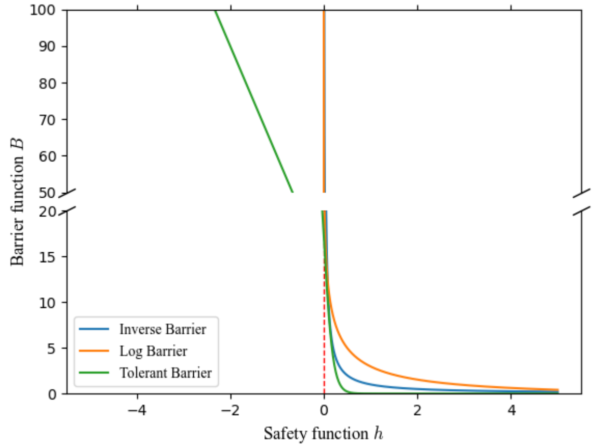
$$\tilde{B} = p\sigma(h) + m\sigma^+(h)$$

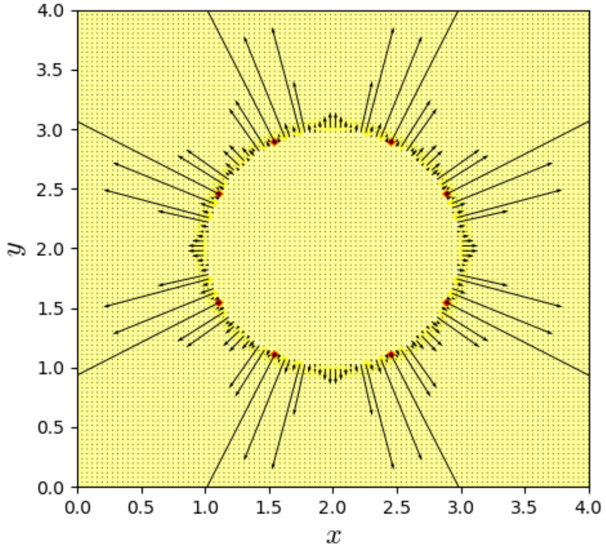
$$\tilde{B}_x = \left(p\frac{\partial\sigma(h)}{\partial h} + m\frac{\partial\sigma^+(h)}{\partial h}\right)\frac{\partial h}{\partial x}$$

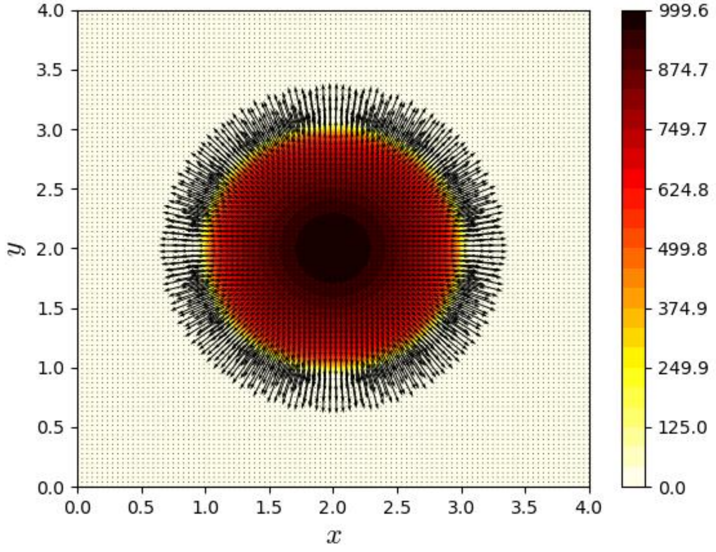
- Merge the safety capabilities of DBaS-DDP and the exploration efficiency of tolerant approaches into a single methodology
- Allow for temporary constraint violation while iteratively improving the solution
- Can approximate BaS safety guarantees
- Have access to the gradient information within the unsafe set, avoiding local minima
- Provide feedback policies of the barrier that enhance safety and robustness

Georgia Tech

Solution: Tolerant-BaS

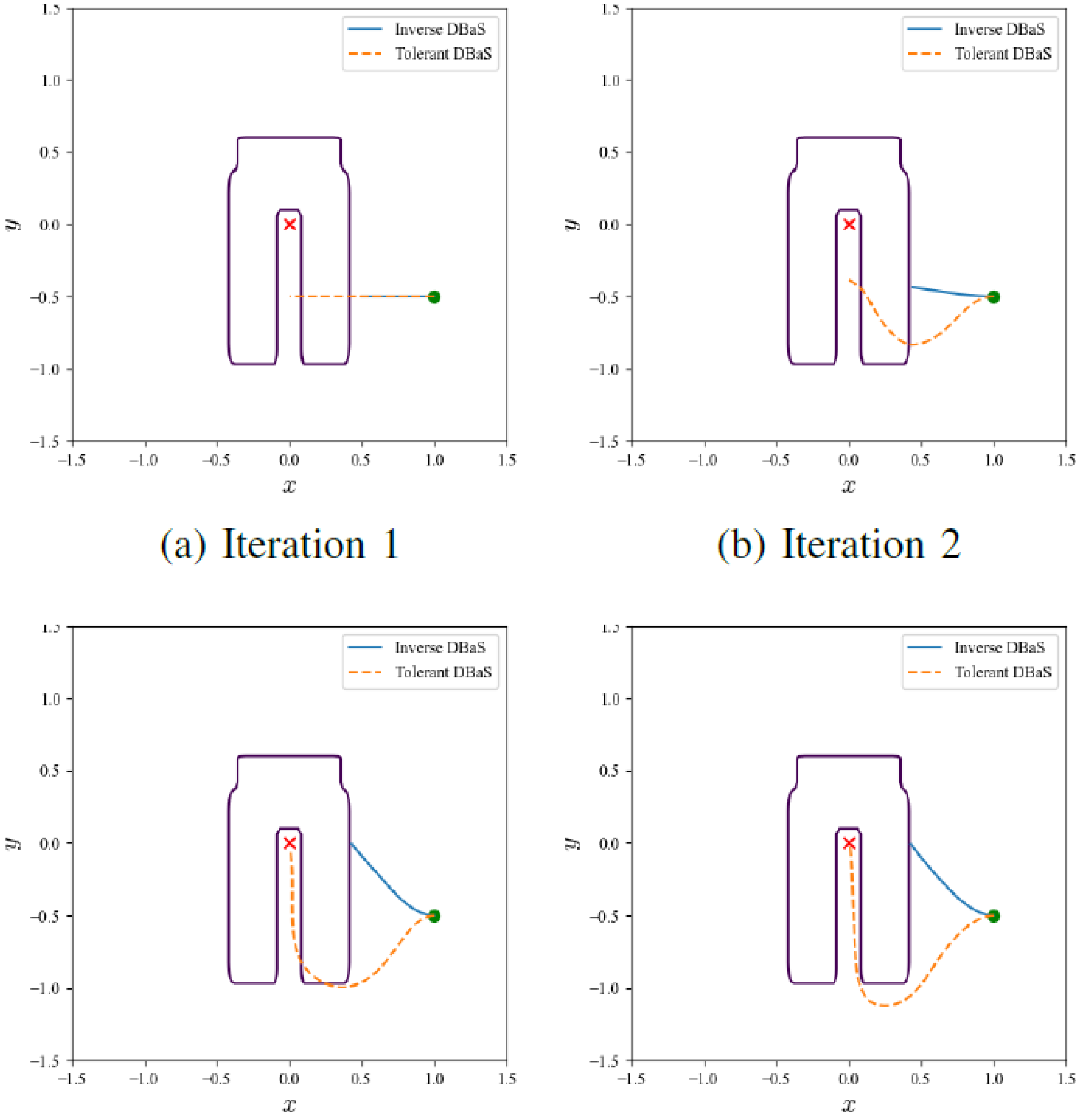






Georgia Tech

Solution: Tolerant-BaS



(a) Iteration 1

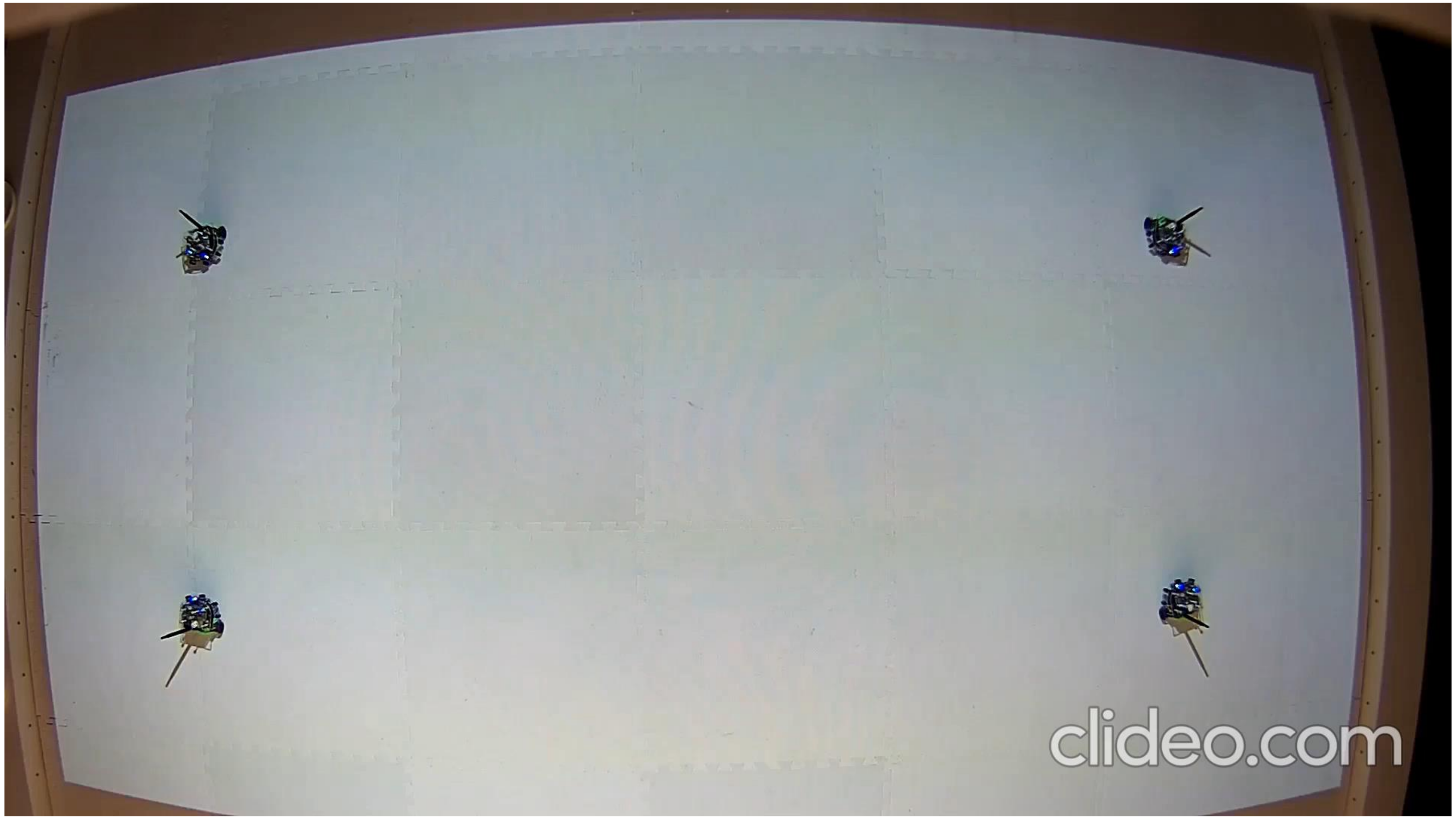
(b) Iteration 2

(c) Iteration 6

(d) Iteration 9

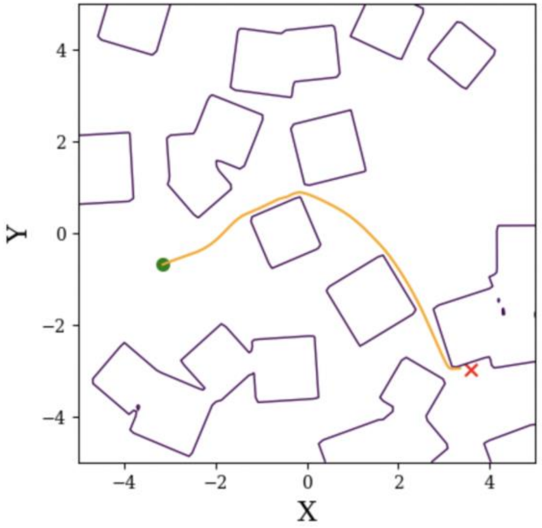
Georgia Tech

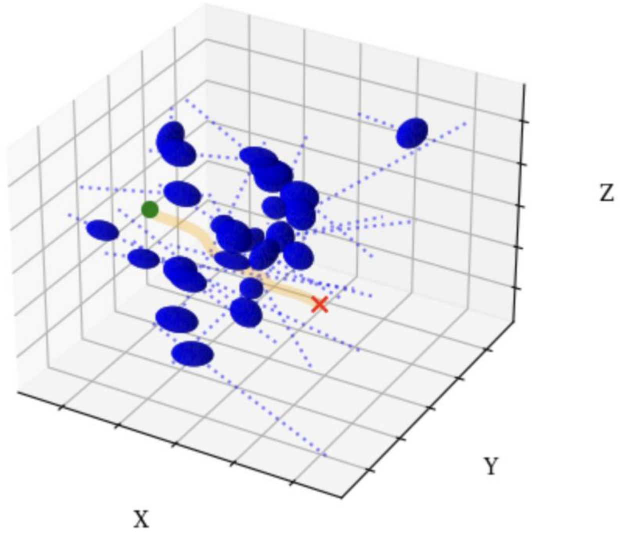
Multi-agent Example

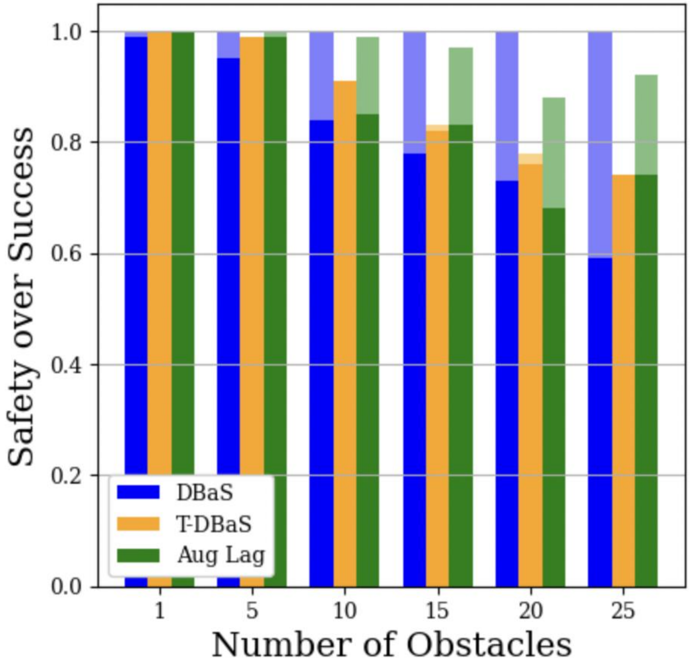


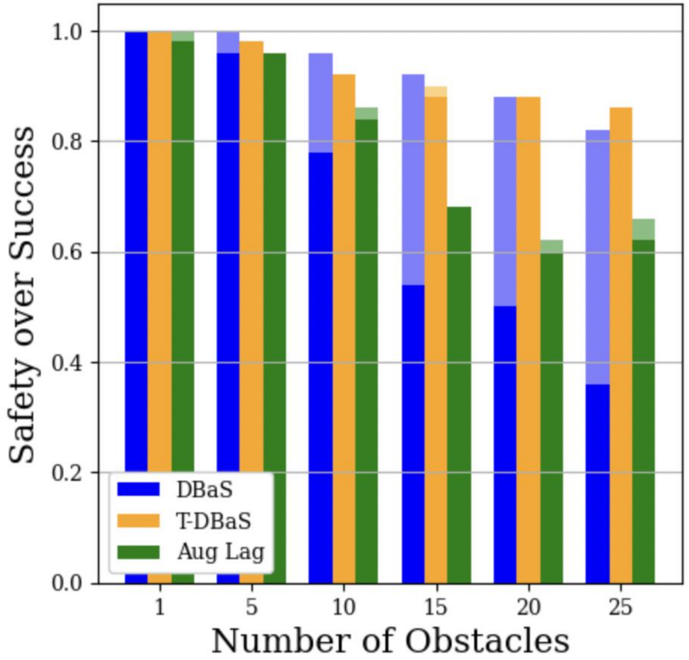
Georgia Tech

Implementations and Comparison



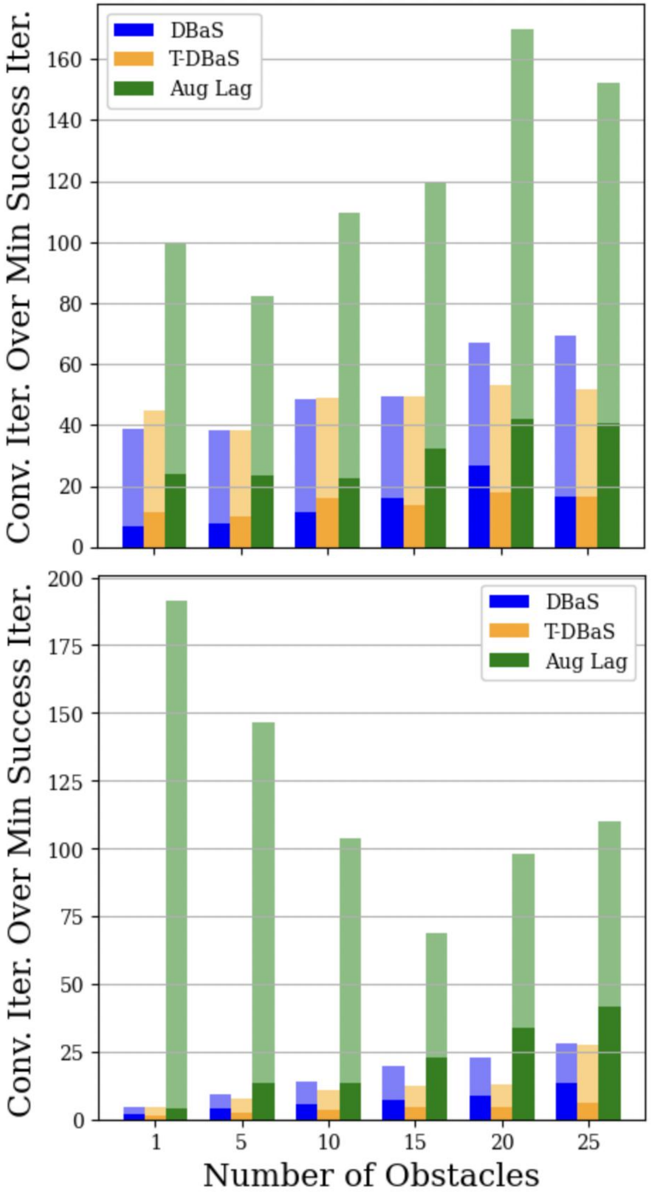


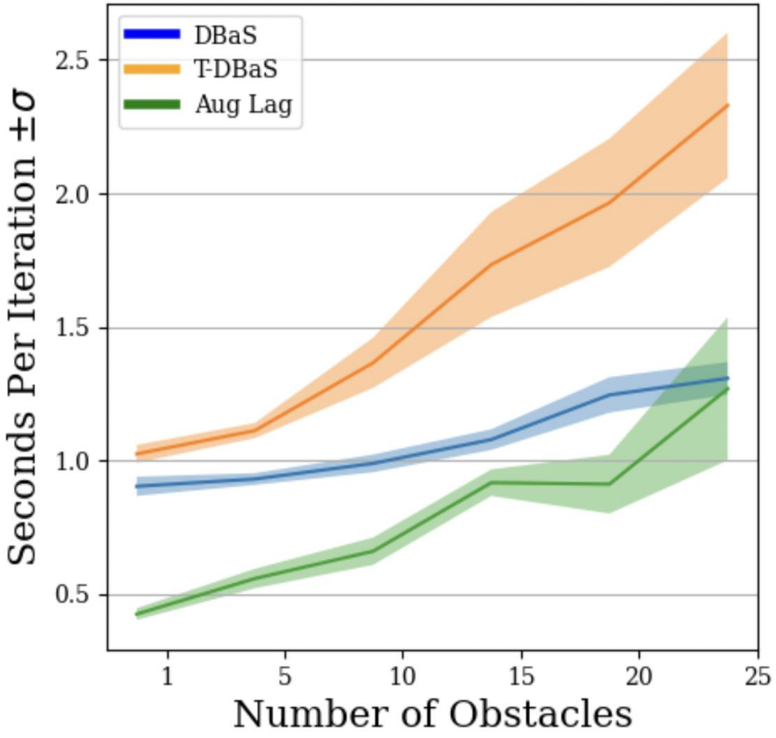




Georgia Tech

Implementations and Comparison





References of Published Related Work

[1] Almubarak H., Sadegh, N., and Theodorou, E. A. “Safety Embedded Control of Nonlinear Systems via Barrier States”. In IEEE Control Systems Letters, vol. 6, pp. 1328-1333, 2021, and In 60th IEEE Conference on Decision and Control (CDC), 2021.

[2] Almubarak H., Stachowicz, K., Sadegh, N., & Theodorou, E. A. “Safety Embedded Differential Dynamic Programming Using Discrete Barrier States”. In IEEE Robotics and Automation Letters, vol. 7, no. 2, pp. 2755-2762, April 2022, and in ICRA2022.

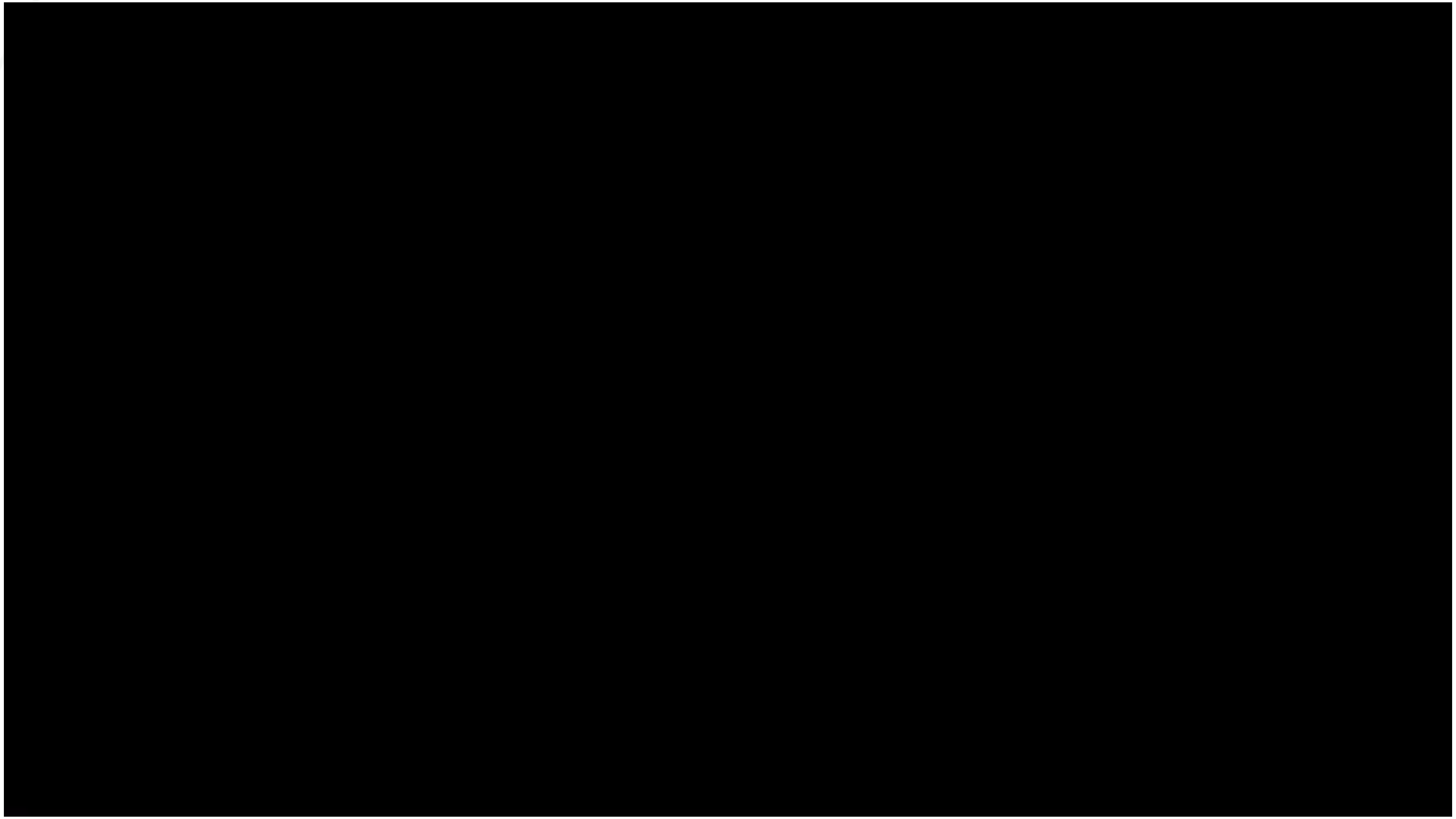
[3] Almubarak H., Theodorou, E. A., & Sadegh, N. Barrier States Embedded Iterative Dynamic Game for Robust and Safe Trajectory Optimization. In American Control Conference (ACC) (pp. 5166-5172), 2022.

[4] Kuperman, J. E., Almubarak, H., Saravanos, A. D., & Theodorou, E. A.. Improved Exploration for Safety-Embedded Differential Dynamic Programming Using Tolerant Barrier States. To be presented at ICAR 2023 (accepted).

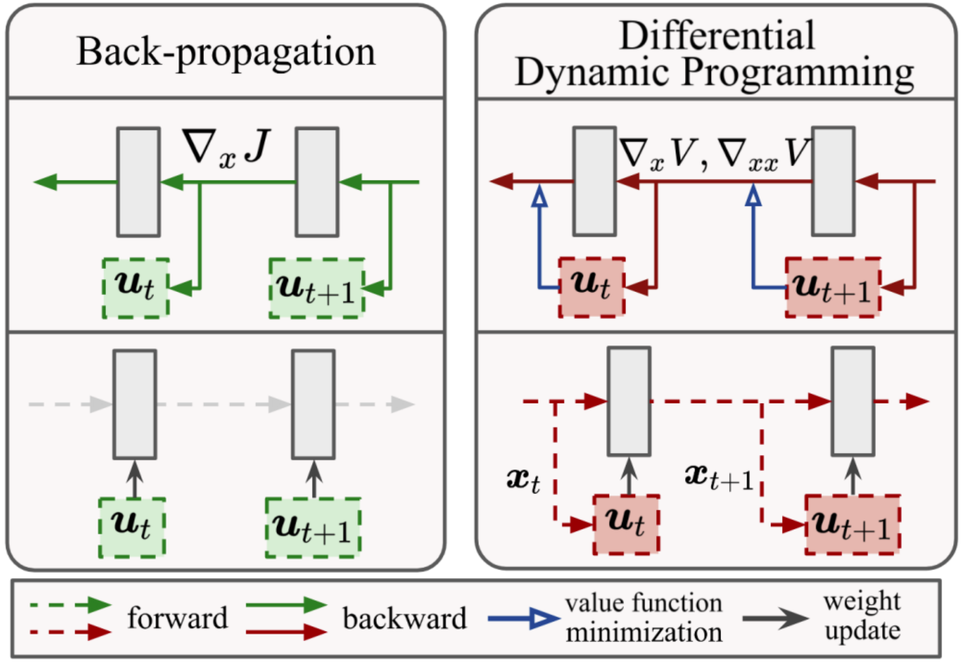
Outline

- Differentiable Optimization for Robust Model Predictive Control Architectures.
- Safety Embedded Optimal Decision Making and Control via (Tolerant) Barrier States
- Robustifying Perception Against Adversaries.

Motivation



Parallels Between Optimal Control and Deep Learning

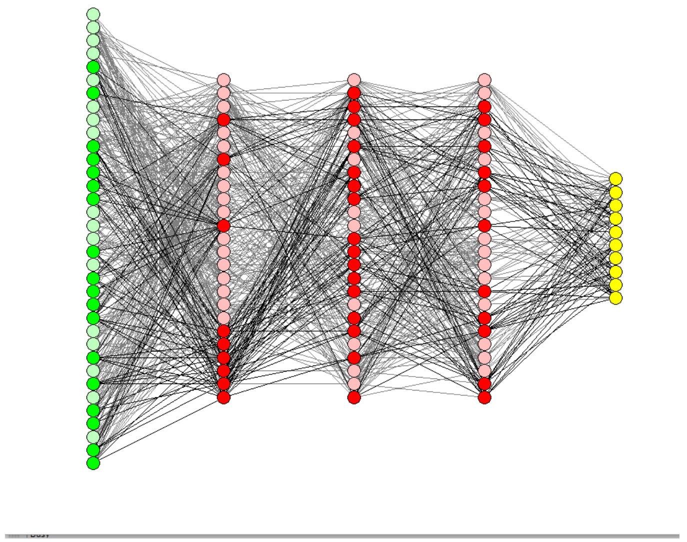


Cost Function

$$\min_{\mathbf{u}} J(\bar{\mathbf{u}}; \mathbf{x}_0) = \min_{\mathbf{u}} \left[\phi(\mathbf{x}_T) + \sum_{t=0}^{T-1} \ell_t(\mathbf{x}_t, \mathbf{u}_t) \right]$$

Dynamics

$$\mathbf{x}_{t+1} = f_t(\mathbf{x}_t, \mathbf{u}_t)$$



Optimal Control → **Deep Learning**

- **State** → **Output Activation**
- **Controls** → **Weights**
- **Time Horizon** → **Number of Layers**
- **Terminal Cost** → **Loss Functions**

Introduction

- Robustness in Neural Networks
  - Deep learning models potent but fragile under adversarial attacks
  - Adversarial Training: optimal weights for worst case perturbation

$$\min_{\theta} \mathbb{E}\left[\max_{\delta \in S} \mathcal{L}(\mathbf{x}+\delta, \mathbf{y}; \theta)\right]$$

- Robustness in Optimal Control

$$\min_{\mathbf{u}} \max_{\mathbf{v}} \left\{ \phi(t_f, \mathbf{x}_{t_f}) + \int_{t_0}^{t_f} \ell(\mathbf{x}, \mathbf{u}, \mathbf{v}, t) d\tau \right\}$$

$$\frac{d\mathbf{x}(t)}{dt} = F(\mathbf{x}(t), \mathbf{u}(t), \mathbf{v}(t), t), \quad x(t_0) = x_0$$

Methodology

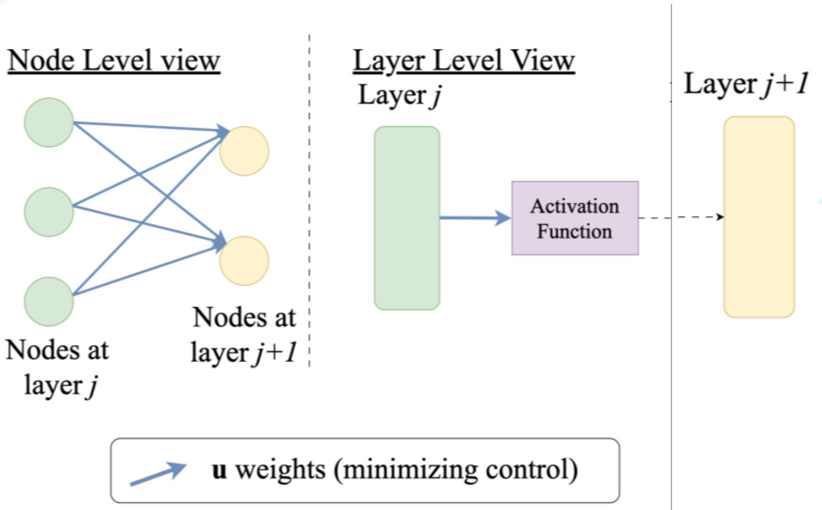
Neural ODEs

For system with dynamics:

$$\frac{\mathrm{dx}}{\mathrm{dt}} = F(t, \mathbf{x}(t), \theta), \quad \mathbf{x}(t_0) = \mathbf{x}_0$$

Minimization of loss function

$$\min_{\theta} \mathcal{L}(\mathbf{x}(t_f))$$



Game Theoretic OC

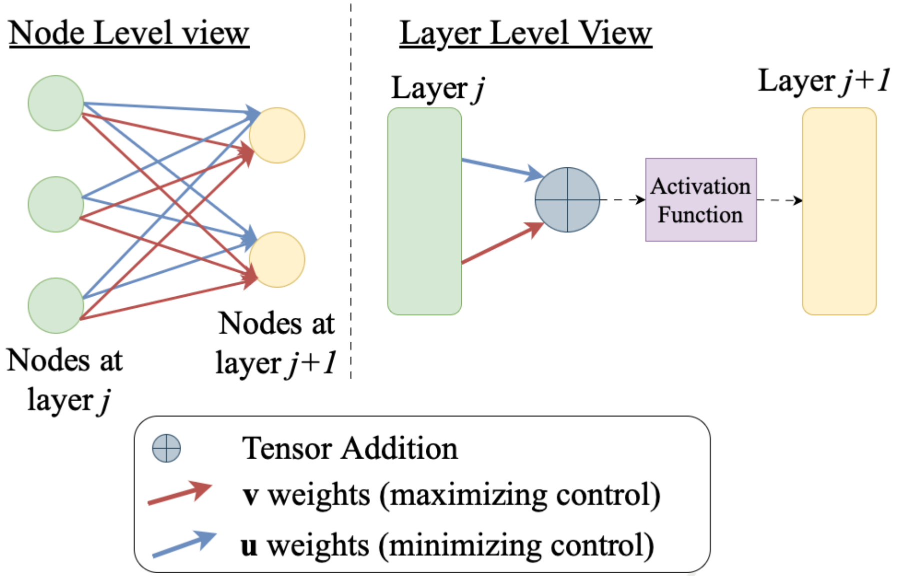
GTSONO

Dynamics with Disturbances

$$\begin{cases} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = F(t, \mathbf{x}, \mathbf{u}, \mathbf{v}), & \mathbf{x}(t_0) = \mathbf{x}_0 \\ \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = 0, & \mathbf{u}(t_0) = \theta \\ \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = 0, & \mathbf{v}(t_0) = \eta \end{cases}$$

Find saddle point

$$\min_{\mathbf{u}} \max_{\mathbf{v}} \left[\Phi(x_{t_f}) + \int_{t_0}^{t_f} \ell(t, \mathbf{x}, \mathbf{u}, \mathbf{v}) \mathrm{d}t \right]$$



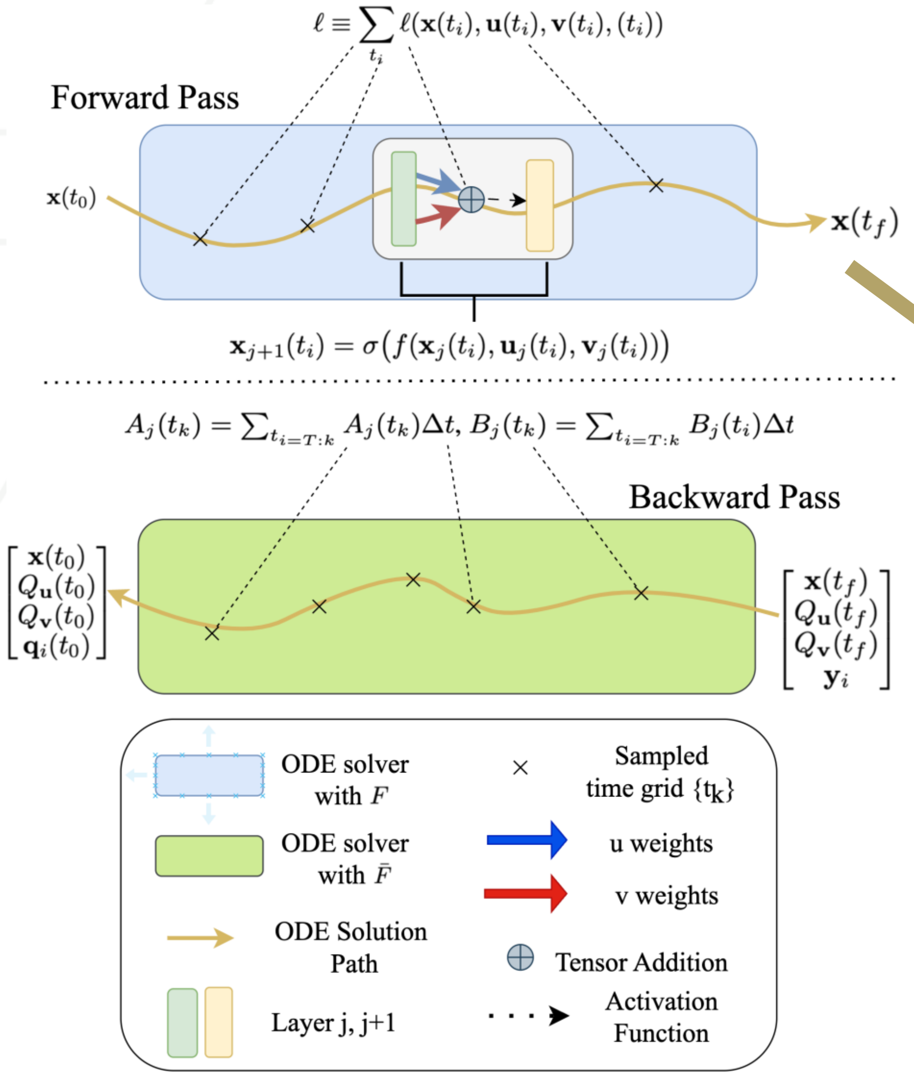
Georgia Tech

Methodology

Update Law of closed loop minimax DDP: $\begin{bmatrix}\delta u_t \\ \delta v_t\end{bmatrix} = \begin{bmatrix}\ell_u \\ \ell_v\end{bmatrix} + \begin{bmatrix}K_u \\ K_v\end{bmatrix}\delta x_t$

Feedback gains

Feedforward gains



Forward Pass

Backward Pass

GTSONO -- Algorithm

Algorithm 1 GTSONO

Input: dataset $\mathcal{D}$, parameterized vector field $F(\cdot,\cdot,\mathbf{u},\mathbf{v})$, integration time $[t_0, t_f]$, ODESolver: '*ODESolve*', learning rate η, time step Δt, Tikhonov regularization constants $R_\mathbf{u}, R_\mathbf{v}$

repeat

  $\mathbf{x}(t_f) = \mathit{ODESolve}(\mathbf{x}(t_0), t_0, t_f, F)$, where $x(t_0) \sim \mathcal{D}$

  $Q(t,\mathbf{x},\mathbf{u},\mathbf{v}) = \Phi(\mathbf{x}(t_f)) + \int_t^{t_f} \ell(\tau,\mathbf{x},\mathbf{u},\mathbf{v})d\tau$

  for t_i in $\{t_f, t_f+\Delta t, \ldots, t_0+\Delta t, t_0\}$

    $[\mathbf{x}(t_{i-1}, Q_\mathbf{u}(t_{i-1}), Q_\mathbf{v}(t_{i-1}), \mathbf{q}_i(t_{i-1})] = \mathit{ODESolve}([\mathbf{x}(t_i), Q_\mathbf{u}(t_i), Q_\mathbf{v}(t_i), \mathbf{q}_i(t_i)], t_{i-1}, t_i, \bar{F})$

  end for

  Compute $\ell_\mathbf{u}, \ell_\mathbf{v}$

  Update controls: $\mathbf{u} \leftarrow \mathbf{u} + \eta\ell_\mathbf{u}$, $\mathbf{v} \leftarrow \mathbf{v} + \eta\ell_\mathbf{v}$

until converges

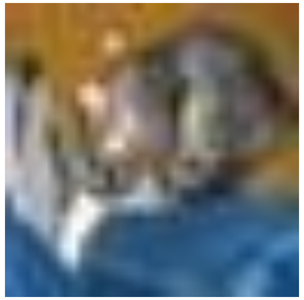
Experiments

1. Optimizer Comparison

2. Adapt GTSONO to adversarial training methods

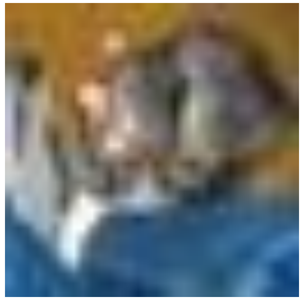
3. Minimax DDP vs GDA with Hessian Precondition

- Attacks:
  a. Projected Gradient Descent,
  b. Fast Gradient Sign Method,
  c. Carlini-Wanger



Natural Image

FGSM attacked



PGD attacked

Results

1. Optimizer Comparison

- Compare GTSONO against state-of-the-art neural ODE optimizers, under natural training

- Outperform benchmark optimizers, providing on average more robust and more confident evaluations

Table 1: Average ± standard deviation of test set accuracy (%) on the CIFAR10 for each optimizer. A_{nat} denotes the natural accuracy. PGD_s^ϵ denotes the accuracy under PGD attack, taking s steps in the direction of the gradient with a perturbation distance ϵ. $FGSM_\alpha$ describes the accuracy under FGSM attack where the single gradient step is multiplied with constant α. CW_∞ denotes the accuracy under the CW attack.

Optimizer	A_{nat}	$FGSM_{0.03}$	$FGSM_{0.05}$	$PGD_{0.03}^{20}$	$PGD_{0.05}^{20}$	CW_∞
Adam	78.7 ± 1.1	48.2 ± 0.7	30.8 ± 0.5	45.1 ± 1.2	29.1 ± 0.3	15.4 ± 0.6
SGD	77.5 ± 0.6	47.3 ± 1.3	33.8 ± 1.3	45.8 ± 1.5	29.3 ± 1.4	18.3 ± 1.6
SNOpt	**79.1** ± **0.4**	48.7 ± 1.0	35.7 ± 1.0	46.8 ± 1.4	32.1 ± 1.4	7.5± 1.0
GTSONO	74.7 ± 0.6	51.7 ± **0.3**	37.9 ± 0.4	49.9 ± 0.5	34.9 + 1.1	18.0 ± 1.5
C-GTSONO	74.7 ± 0.7	**51.8** ± 0.4	**38.0** ± **0.2**	**50.6** ± **0.3**	**35.0 + 0.2**	**36.3** ± **2.2**

Table 2: Average ± standard deviation of test set accuracy (%) on the SVHN for each optimizer.

Optimizer	A_{nat}	$FGSM_{0.03}$	$FGSM_{0.05}$	$PGD_{0.03}^{20}$	$PGD_{0.05}^{20}$	CW_∞
Adam	98.9 ± 0.3	73.8 ± 0.4	55.9 ± 0.8	71.8 ± **0.1**	48.4 ± 1.0	20.3 ± 0.2
SGD	98.4 ± 0.0	74.4 ± 0.4	56.1 ± 0.5	72.4 ± 0.7	50.4 ± 1.1	23.4 ± 0.9
SNOpt	99.1 ± 0.1	73.5 ± 2.2	54.4 ± 2.3	71.9 ± 2.7	48.7 ± 3.3	22.4 ± 0.9
GTSONO	**99.6** ± **0.0**	78.0 ± 0.4	58.9 ± 0.4	76.7 ± 0.8	54.3 ± 0.8	31.6 ± 2.2
C-GTSONO	97.3 ± 0.2	**80.8** ± **0.2**	**65.2** ± **0.3**	**80.3** ± 0.4	**62.3** ± **0.3**	**50.5** ± **0.8**

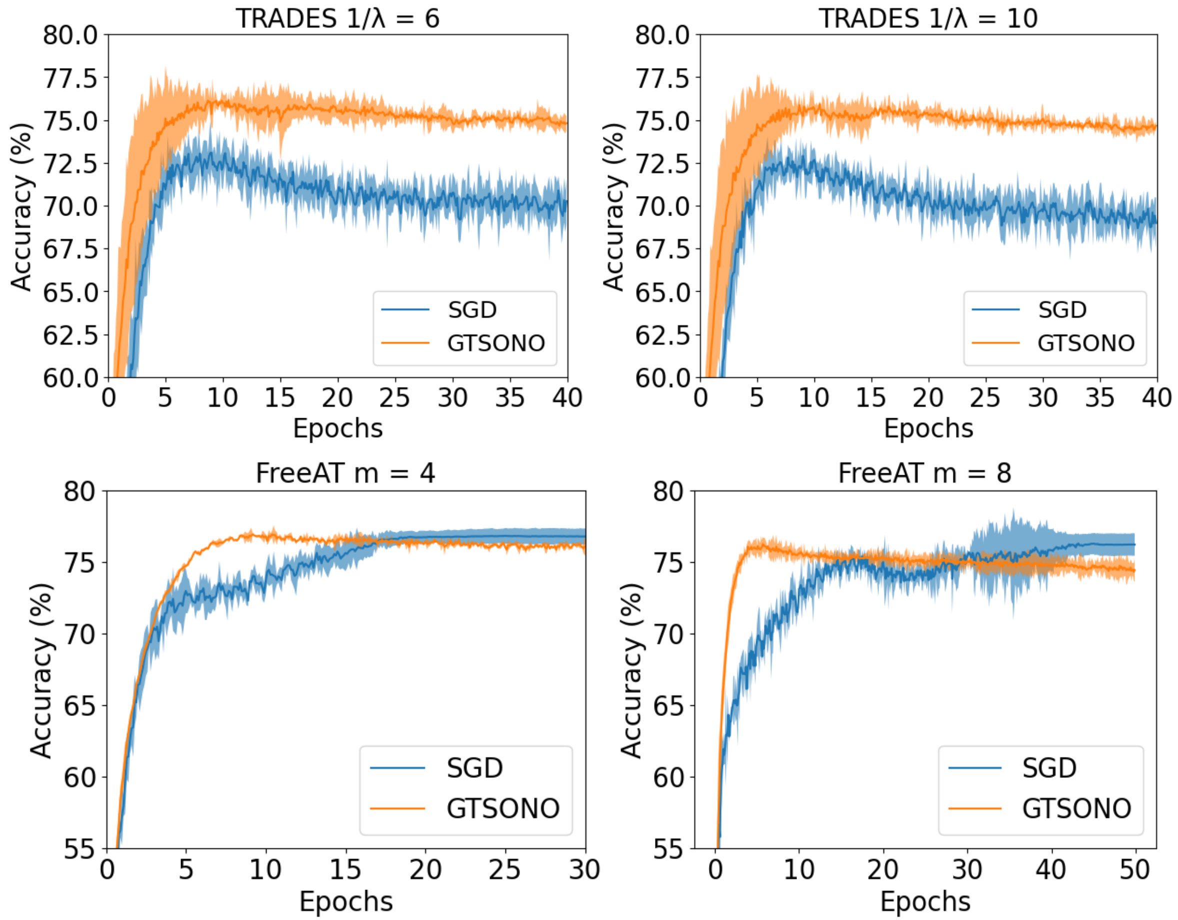
Results

2. Adapt GTSONO to adversarial training methods

Ablation Study on TRADES and Free Adversarial Training (FreeAT)

Method	A_{nat}	$PGD_{20}^{0.03}$	$PGD_{40}^{0.03}$	CW_{∞}
TRADES(λ^{-1}=6)	73.2	54.5	54.3	**22.8**
C-GTSONO(λ^{-1}=6)	**75.8**	**58.9**	**58.8**	20.4
TRADES(λ^{-1}=10)	69.4	57.2	56.9	**25.2**
C-GTSONO(λ^{-1}=10)	**72.6**	**61.2**	**61.0**	20.7

Optimizer	A_{nat}	$PGD_{20}^{0.03}$	$PGD_{40}^{0.03}$	CW_{∞}
SGD (m=4)	**76.8**	48.8	48.1	6.5
C-GTSONO (m=4)	75.7	**50.5**	**50.2**	**17.4**
SGD (m=8)	**76.2**	54.8	54.2	9.5
C-GTSONO (m=8)	74.4	**56.5**	**56.4**	**18.9**



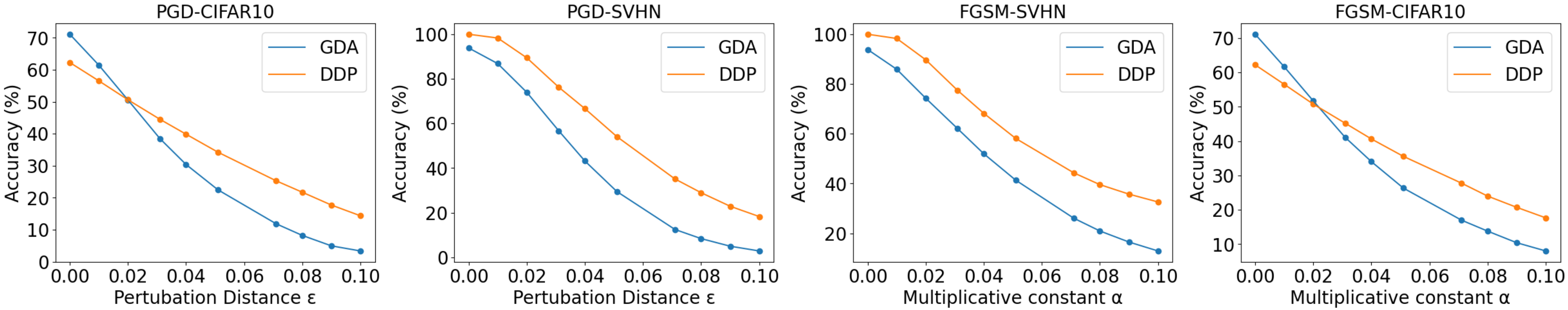
Adapting GTSONO on TRADES and
Free Adversarial Training was found to provide:

1. Faster Convergence – Shorter training
2. Superior robust performance

Results

3. Minimax DDP vs GDA with Hessian Precondition

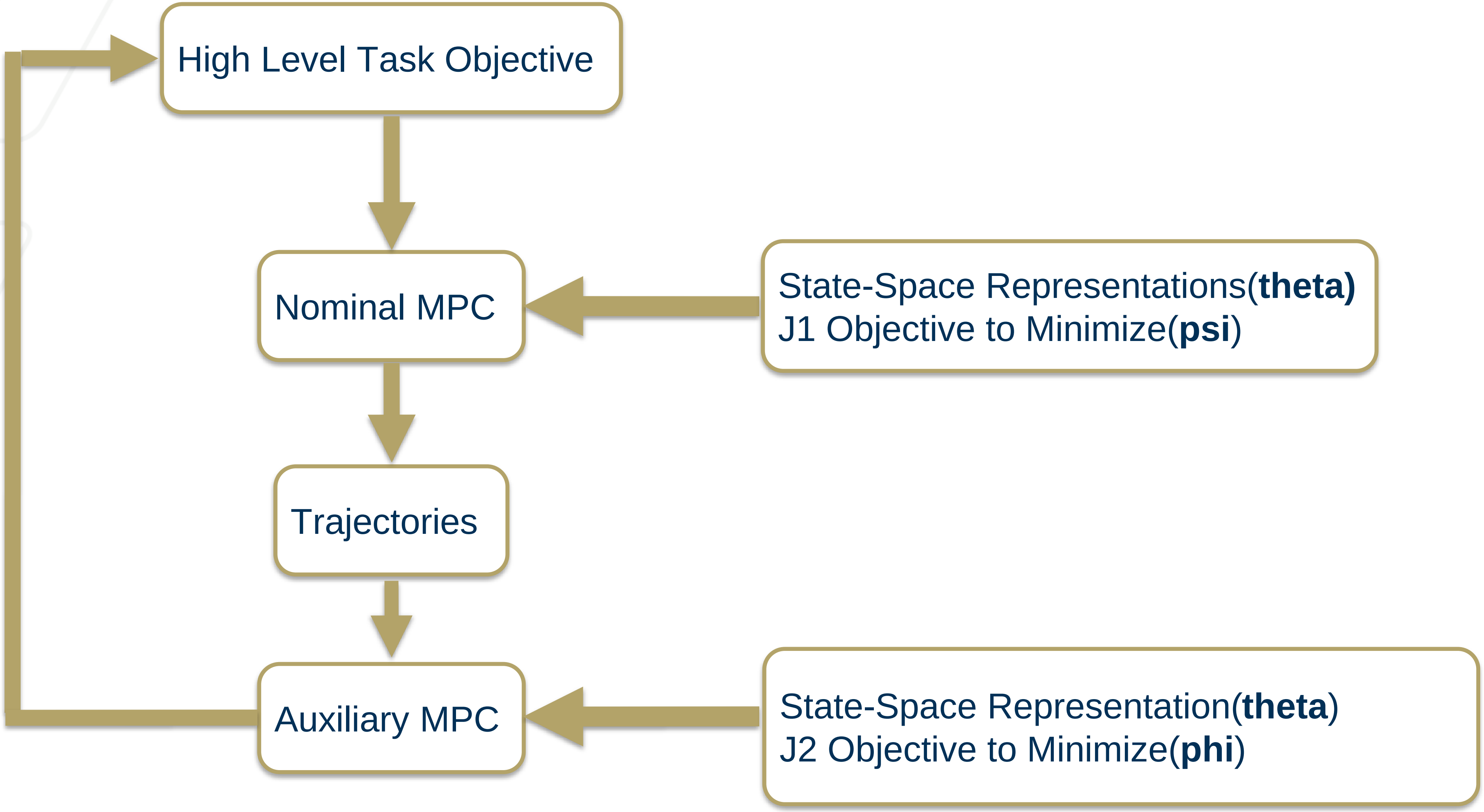
Recall open loop minimax DDP → Generalization of Preconditioned GDA



Open loop DDP update rule in GTSONO was more robust, especially in large disturbances

Where do we go next?

Differentiable Robust MPC Architecture



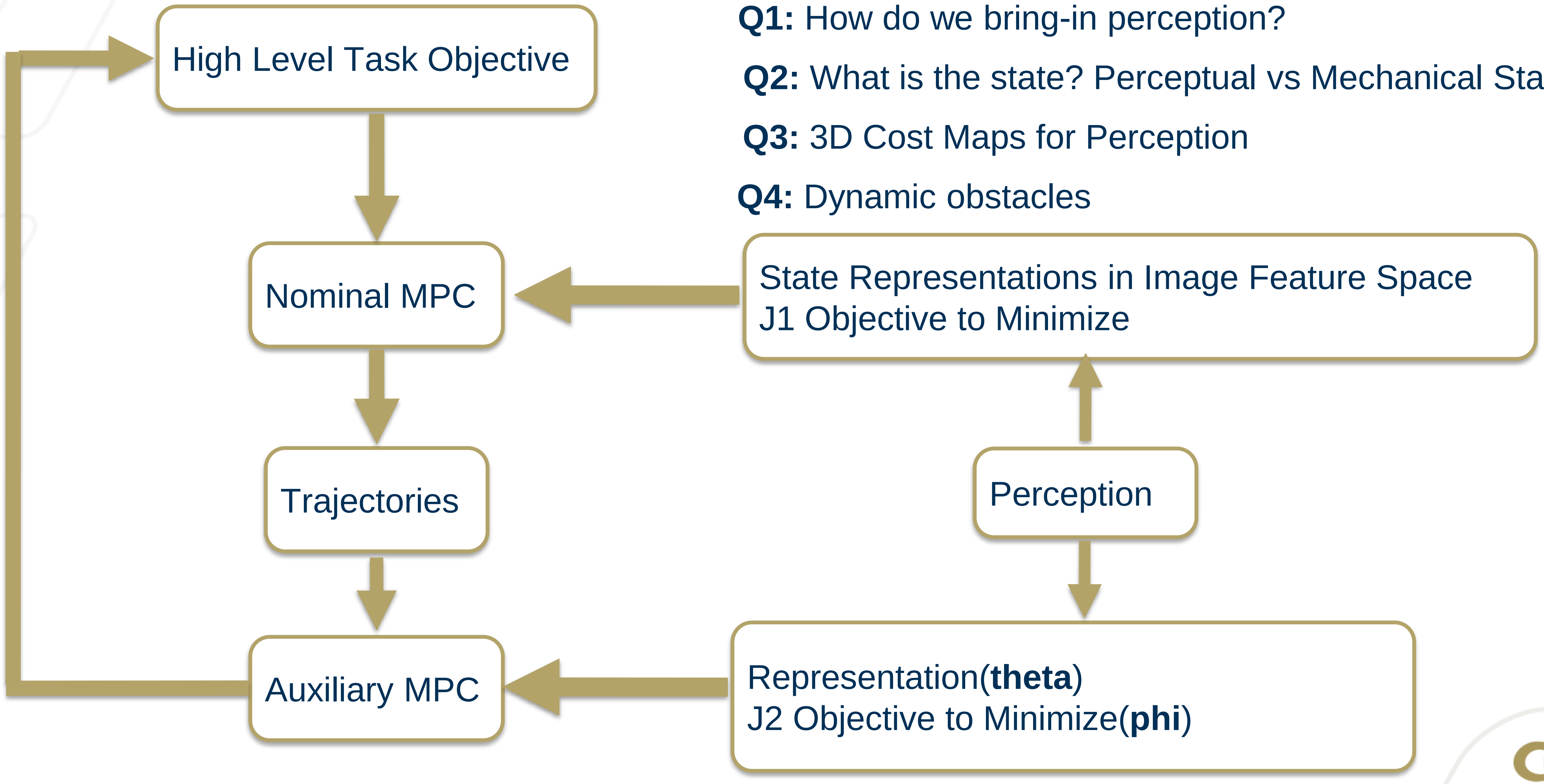
Differentiable Robust MPC with Perception

Q1: How do we bring-in perception?

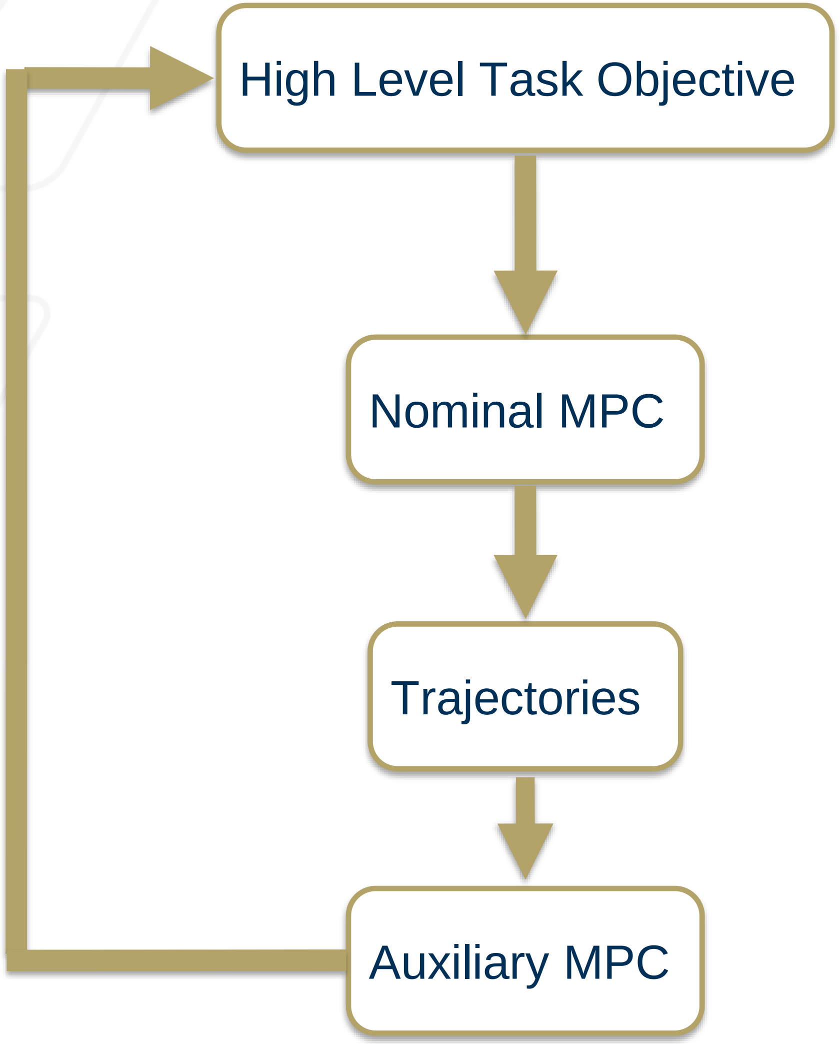
Q2: What is the state? Perceptual vs Mechanical States

Q3: 3D Cost Maps for Perception

Q4: Dynamic obstacles



Differentiable Robust MPC with Perception



Georgia Tech