Autonomous Robotics

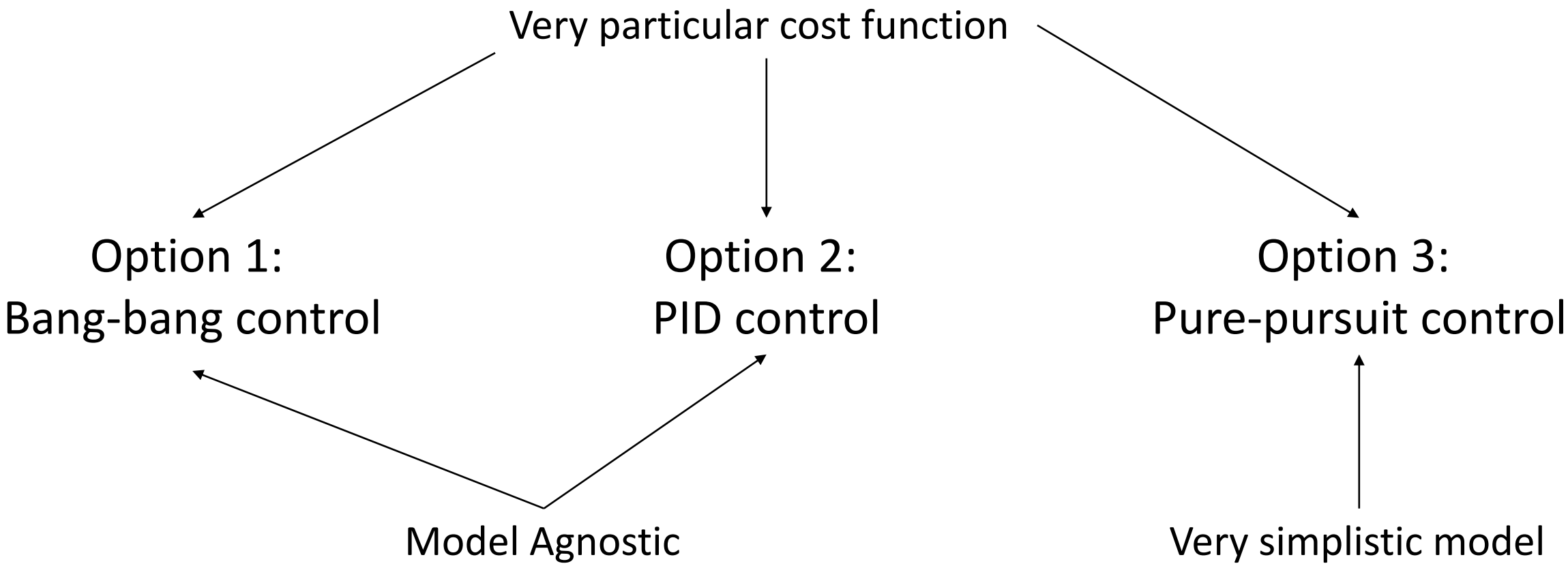
Spring 2026

Abhishek Gupta, Siddhartha Srinivasa

TAs: Helen Wang, Rohan Baijal, Sidharth Talia, Christopher Tan

Recap

Controller Design Decisions



Generalized Problem: Optimal Control

- Minimize sum of costs, subject to dynamics and other constraints

$$\min_{u_{1:T}} \sum_{t=1}^{T} c(x_t, u_t)$$

$$\text{s.t. } x_{t+1} = f(x_t, u_t)$$

Can be costs like smoothness, preferences, speed

Can be constraints like velocity/acceleration bounds

Linear System

- **Linear** system (model)
- **Quadratic** cost function to minimize

$$x_{t+1} = Ax_t + Bu_t$$

$$\sum_t x_t^\top Q x_t + u_t^\top R u_t$$

$$\underset{(N \times 1)}{x_{t+1}} = \underset{(N \times N)}{A}\ \underset{(N \times 1)}{x_t} + \underset{(N \times M)}{B}\ \underset{(M \times 1)}{u_t}$$

STATE → NEXT STATE

CONTROL → NEXT STATE

Linear Quadratic Regulator

- Linear system (model)
- Quadratic cost function to minimize

$$x_{t+1} = Ax_t + Bu_t$$

$$\sum_t x_t^\top Q x_t + u_t^\top R u_t$$

Linear System

- Linear system (model)
- Quadratic cost function to minimize

$$x_{t+1} = Ax_t + Bu_t$$

$$\sum_t x_t^\top Q x_t + u_t^\top R u_t$$

$$\underset{(N \times 1)}{x_{t+1}} = \underset{(N \times N)}{A}\ \underset{(N \times 1)}{x_t} + \underset{(N \times M)}{B}\ \underset{(M \times 1)}{u_t}$$

STATE → NEXT STATE

CONTROL → NEXT STATE

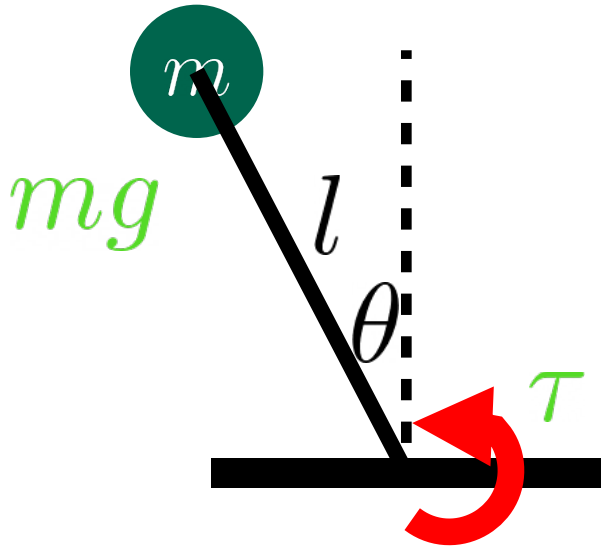
Lecture Outline

Examples of LQR Systems

LQR Solutions

Sampling-based MPC

Example: Inverted Pendulum (Linear System)



$$mgl\sin\theta + \tau = ml^2\ddot{\theta}$$

$$\ddot{\theta} = \frac{g}{l}\sin\theta + \frac{\tau}{ml^2} \approx \frac{g}{l}\theta + \frac{\tau}{ml^2}$$

$$\begin{bmatrix}\theta_{t+1}\\ \dot{\theta}_{t+1}\end{bmatrix} = \begin{bmatrix}1 & \Delta t\\ \frac{g}{l}\Delta t & 1\end{bmatrix}\begin{bmatrix}\theta_t\\ \dot{\theta}_t\end{bmatrix} + \begin{bmatrix}0\\ \Delta t\end{bmatrix}\frac{\tau}{ml^2}$$

x_{t+1} A x_t B u_t

Quadratic Cost Function

- **Linear** system (model)
- **Quadratic** cost function to minimize

$$x_{t+1} = Ax_t + Bu_t$$

$$\sum_t x_t^\top Q x_t + u_t^\top R u_t$$

$$x_t^\top Q x_t$$

(1 x N)(N x N)(N x 1)

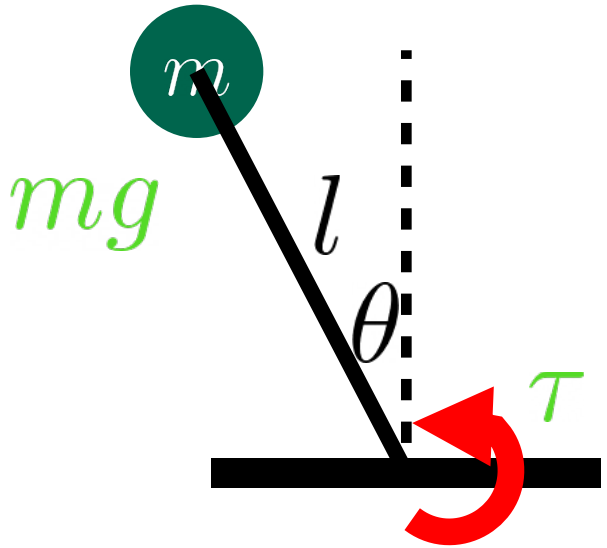
STATE COST

$$u_t^\top R u_t$$

(1 x M)(M x M)(M x 1)

CONTROL COST

Example: Inverted Pendulum (State Cost)



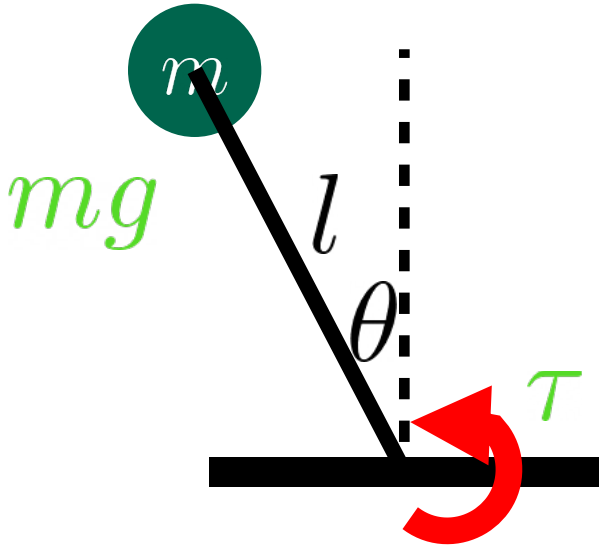
$$x_t^\top Q x_t \qquad \text{(QUADRATIC FORM)}$$

$$= \begin{bmatrix} \theta_t \\ \dot{\theta}_t \end{bmatrix}^\top \begin{bmatrix} Q_{\theta\theta} & Q_{\theta\dot{\theta}} \\ Q_{\dot{\theta}\theta} & Q_{\dot{\theta}\dot{\theta}} \end{bmatrix} \begin{bmatrix} \theta_t \\ \dot{\theta}_t \end{bmatrix}$$

$$= Q_{\theta\theta}\theta_t^2 + 2Q_{\theta\dot{\theta}}\theta_t\dot{\theta}_t + Q_{\dot{\theta}\dot{\theta}}\dot{\theta}_t^2$$

$$Q \succ 0 \leftrightarrow z^\top Q z > 0, \ \forall z \neq 0$$

Cost)



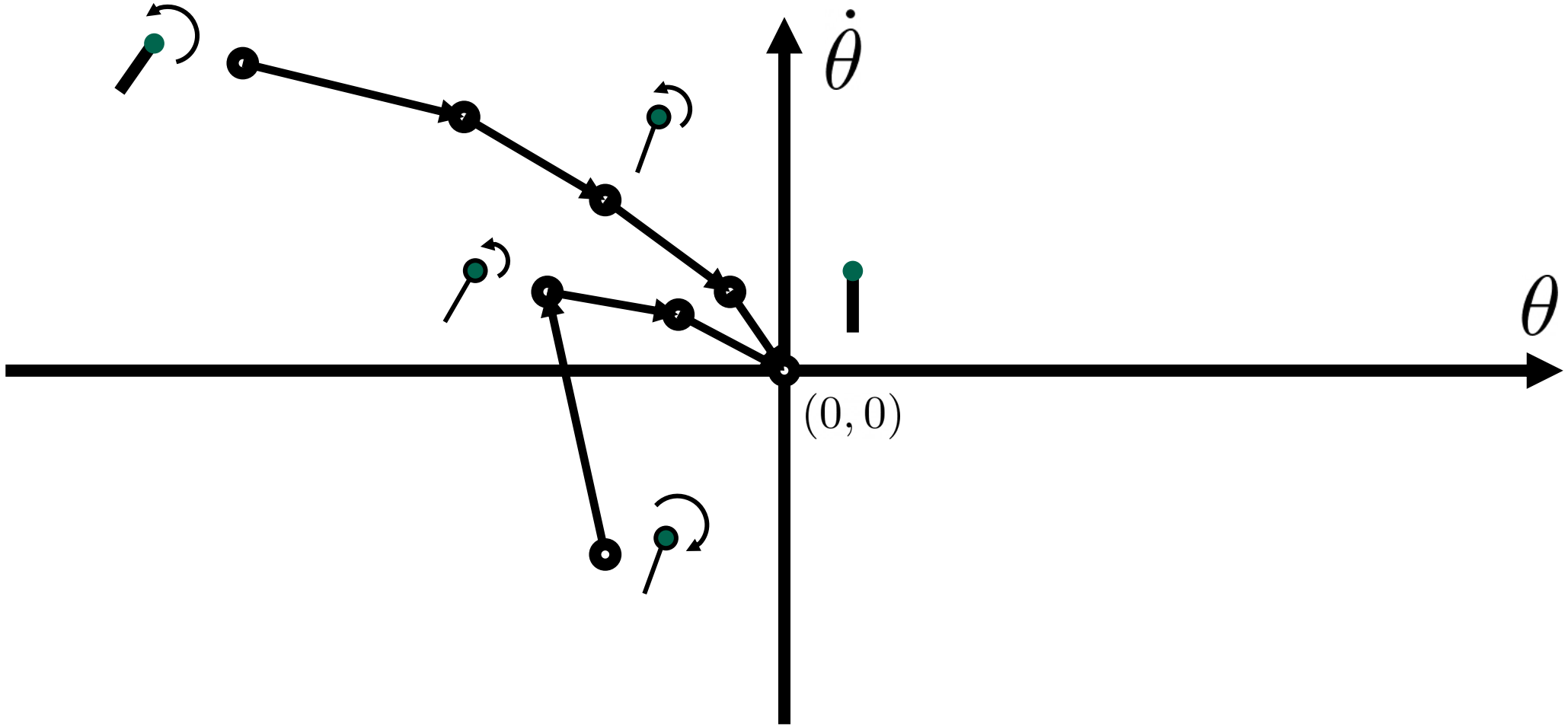
$$u_t^\top R u_t \qquad \text{(QUADRATIC FORM)}$$

$$= \frac{\tau_t}{ml^2} \left[R_{\tau\tau} \right] \frac{\tau_t}{ml^2}$$

$$= R_{\tau\tau} \left(\frac{\tau_t}{ml^2} \right)^2$$

$$R \succ 0 \leftrightarrow z^\top R z > 0, \ \forall z \neq 0$$

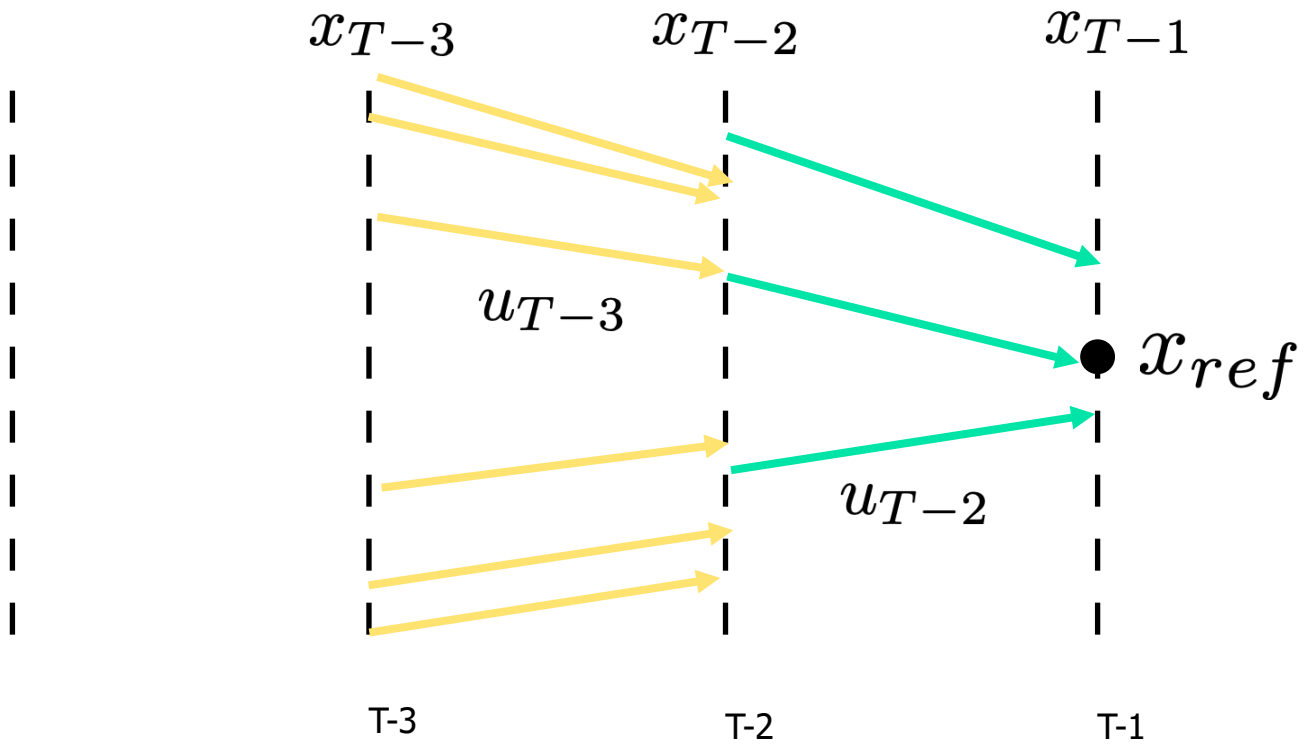
Example: Inverted Pendulum



How do we solve for controls?

Dynamic programming to the rescue!

Start from timestep T-1 and solve backwards



Lecture Outline

Examples of LQR Systems

LQR Solutions

Sampling-based MPC

Bellman Equation for Dynamic Programming

- **Linear** system (model)
- **Quadratic** cost function to minimize

$$x_{t+1} = Ax_t + Bu_t$$

$$\sum_t x_t^\top Q x_t + u_t^\top R u_t$$

$$J^*(x_t) = \min_{u_t} x_t^\top Q x_t + u_t^\top R u_t + J^*(x_{t+1})$$

MINIMUM COST, STARTING FROM x_t

IMMEDIATE COST

MINIMUM FUTURE COST, STARTING FROM x_{t+1}

Start from the back: Time-to-go = 0

$$J_0(x) = \min_u x^\top Q x + u^\top R u$$

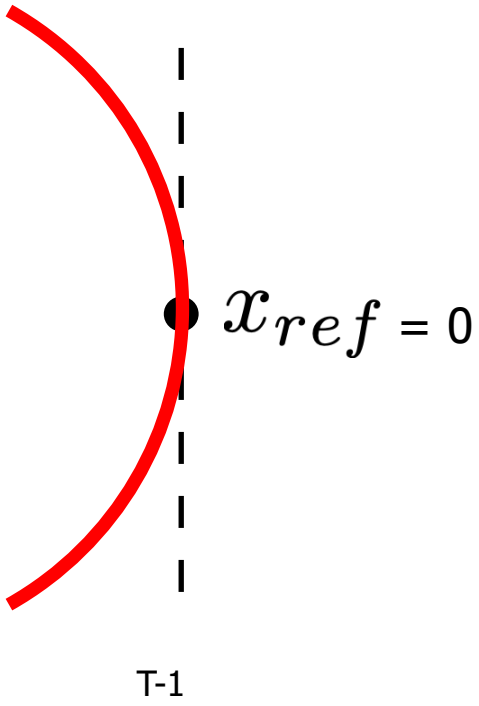
(whiteboard)

Start from the back: Time-to-go = 0

$$J_0(x) = \min_u x^\top Q x + u^\top R u = x^\top Q x = x^\top P_0 x$$

Minimized with u = 0

$$P_0 = Q$$

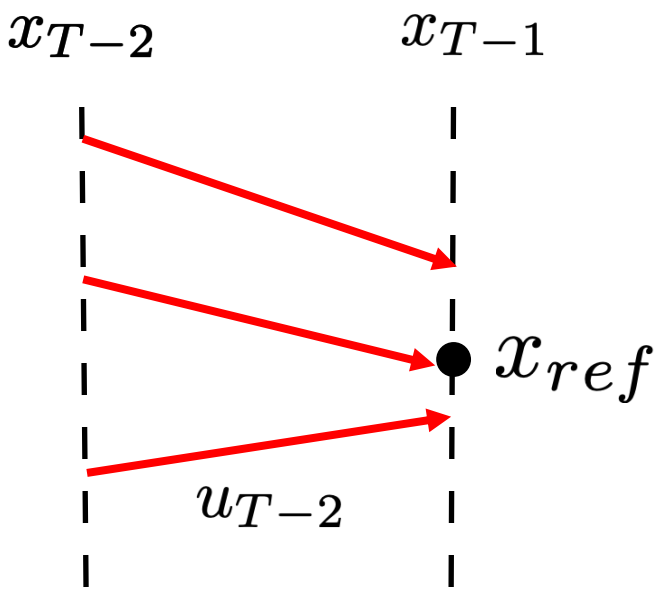


Note that the cost is quadratic in x

Take one step towards the start: Time-to-go = 1

$$J_0(x) = \min_u x^\top Q x + u^\top R u = x^\top Q x = x^\top P_0 x$$

$$J_1(x) = \min_u x^\top Q x + u^\top R u + J_0(Ax + Bu)$$



Solve for control at timestep T-1, accounting for impact on the future, through dynamics

Take one step towards the start: Time-to-go = 1

$$J_1(x) = \min_u x^\top Q x + u^\top R u + J_0(Ax + Bu)$$

(Move to whiteboard)

Value Iteration (Horizon = 1)

$$J_1(x) = \min_u \left[x^\top Q x + u^\top R u + (Ax + Bu)^\top P_0 (Ax + Bu) \right]$$

$$\nabla_u [\cdot] = 2Ru + 2B^\top P_0 (Ax + Bu) = 0$$

$$u = -(R + B^\top P_0 B)^{-1} B^\top P_0 A x$$

$$J_1(x) = x^\top P_1 x$$

$$P_1 = Q + K_1^\top R K_1 + (A + BK_1)^\top P_0 (A + BK_1)$$

$$K_1 = -(R + B^\top P_0 B)^{-1} B^\top P_0 A$$

Turns into a recursion at time-to-go = i

$$K_i = -(R + B^\top P_{i-1} B)^{-1} B^\top P_{i-1} A$$

$$P_i = Q + K_i^\top R K_i + (A + B K_i)^\top P_{i-1} (A + B K_i)$$

$$u = K_i x, \; J_i(x) = x^\top P_i x$$

Optimal controller is linear in x

Optimal cost is quadratic in x

RUNTIME: $O(H(n^3 + m^3))$

The LQR algorithm

Algorithm OptimalValueControl(A, B, Q, R, time-to-go):

if time-to-go == 0:
return 0, Q

else:
P_{i-1} = OptimalValueControl(A, B, Q, R, time-to-go - 1)

$$K_i = -(R + B^\top P_{i-1} B)^{-1} B^\top P_{i-1} A$$

$$P_i = Q + K_i^\top R K_i + (A + B K_i)^\top P_{i-1} (A + B K_i)$$

return K_i, P_i

Optimal controller is linear in x

Optimal cost is quadratic in x

Unpacking LQR intuitively

$$K_i = -(R + B^\top P_{i-1} B)^{-1} B^\top P_{i-1} A$$

$$P_i = Q + K_i^\top R K_i + (A + BK_i)^\top P_{i-1} (A + BK_i)$$

$$u = K_i x, \; J_i(x) = x^\top P_i x$$

Unpacking LQR intuitively

$$K_i = -(R + B^\top P_{i-1} B)^{-1} B^\top P_{i-1} A$$

Recall Kalman Filtering

$$\frac{B^T P_{i-1} A}{R + B^T P_{i-1} B}$$

Set A, B = I

$$\frac{P_{i-1}}{R + P_{i-1}}$$

Tradeoff between future cost P_{i-1} and current cost R

Unpacking LQR intuitively

$$x^\top \left[P_i = Q + K_i^\top R K_i + (A + BK_i)^\top P_{i-1} (A + BK_i) \right] x$$

Current state cost

Current action cost

Optimal cost in the future based on dynamics

Linear Quadratic Regulator

- For **linear** systems with **quadratic** costs, we can write down very efficient algorithms that return the optimal sequence of actions!
  - Special case where dynamic programming can be applied to continuous states and actions (typically only discrete states and actions)
- Many LQR extensions: non-linear systems, linear time-varying systems, trajectory following for non-linear systems, arbitrary costs, etc.

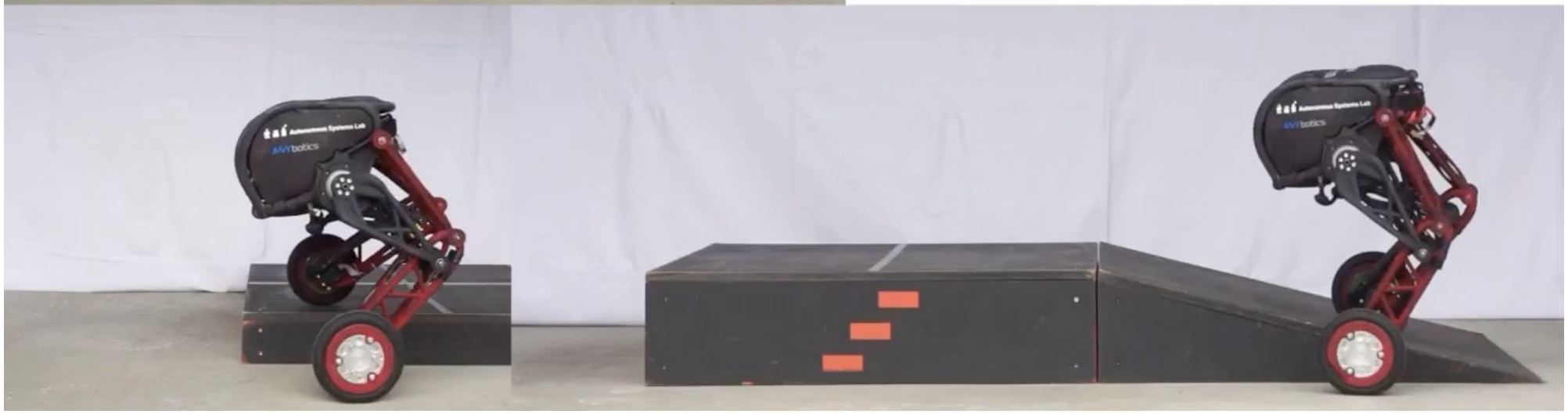
LQR in Action: Stanford Helicopter

ABBEEL ET AL., 2006

HTTPS://YOUTU.BE/0JL04JJJOCC

LQR in Action

Overcoming challenging indoor environements.



Klemm et al 2020

Lecture Outline

Examples of LQR Systems

LQR Solutions

Sampling-based MPC

Why might this not be enough?

$$\min_{u_{1:T}} \sum_{t=1}^{T} c(x_t, u_t)$$

Non-linear

$$\text{s.t.} \quad x_{t+1} = f(x_t, u_t)$$

Non-quadratic

$$\sum_t x_t^\top Q x_t + u_t^\top R u_t$$

$$x_{t+1} = A x_t + B u_t$$

Idea 1: Shift the burden to brute-force computation

Idea 2: Convert into a locally linear system

Let’s revisit ideas from Bayesian filtering

	Linear Gaussian assumption	Sampling-based approximation
Filtering	Kalman Filtering	Particle Filtering
Control	LQR	Sampling based MPC

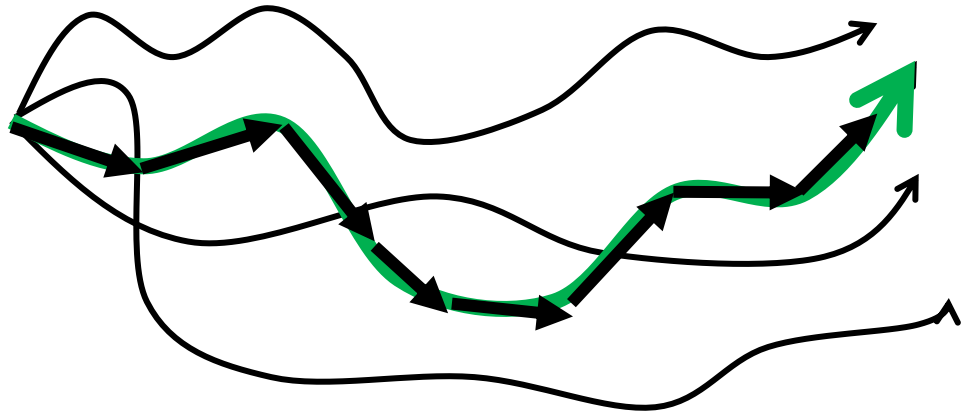
Solving Optimal Control with Sampling

$$\min_{u_{1:T}} \sum_{t=1}^{T} c(x_t, u_t)$$

$$\text{s.t.} \quad x_{t+1} = f(x_t, u_t)$$

1. Sample a set of K action trajectories of T steps from start state
2. Evaluate each K step action sequence through the model and get per trajectory cost
3. Choose minimum trajectory cost trajectory
4. Execute lowest cost actions

Random Sampling



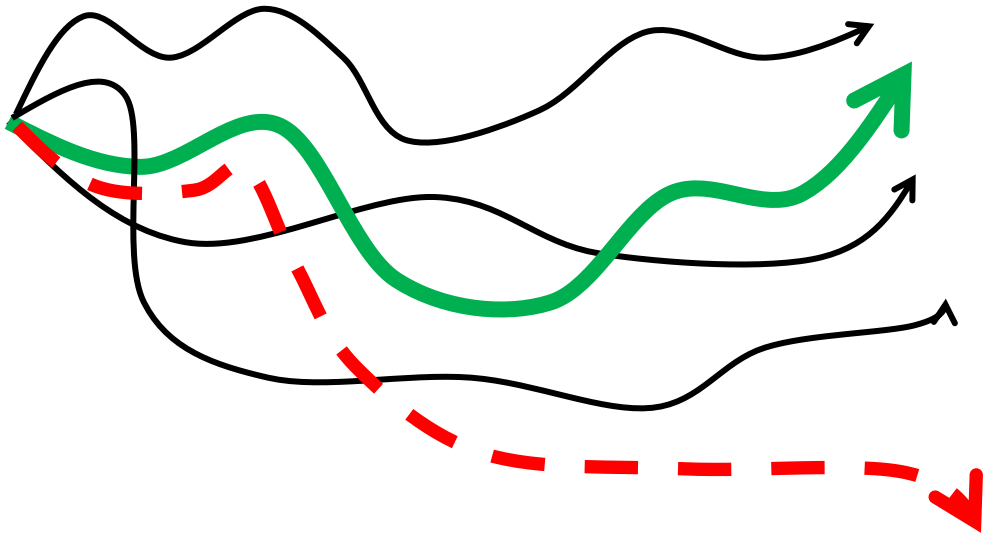
Can soften by taking softmin rather than argmin

Solving Optimal Control with Sampling – issues?

$$\min_{u_{1:T}} \sum_{t=1}^{T} c(x_t, u_t)$$

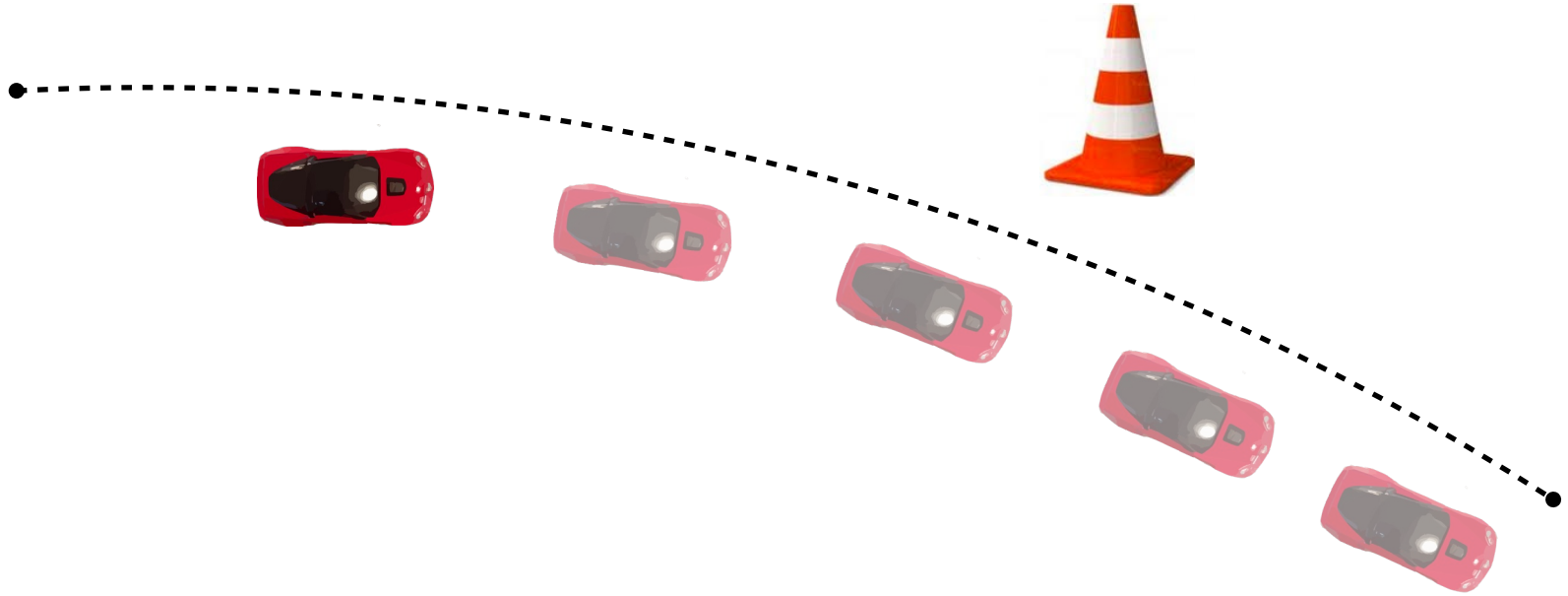
$$\text{s.t.} \quad x_{t+1} = f(x_t, u_t)$$

1. Sample a set of K action trajectories of T steps from start state
2. Evaluate each K step action sequence through the model and get per trajectory cost
3. Choose minimum trajectory cost trajectory
4. Execute lowest cost actions



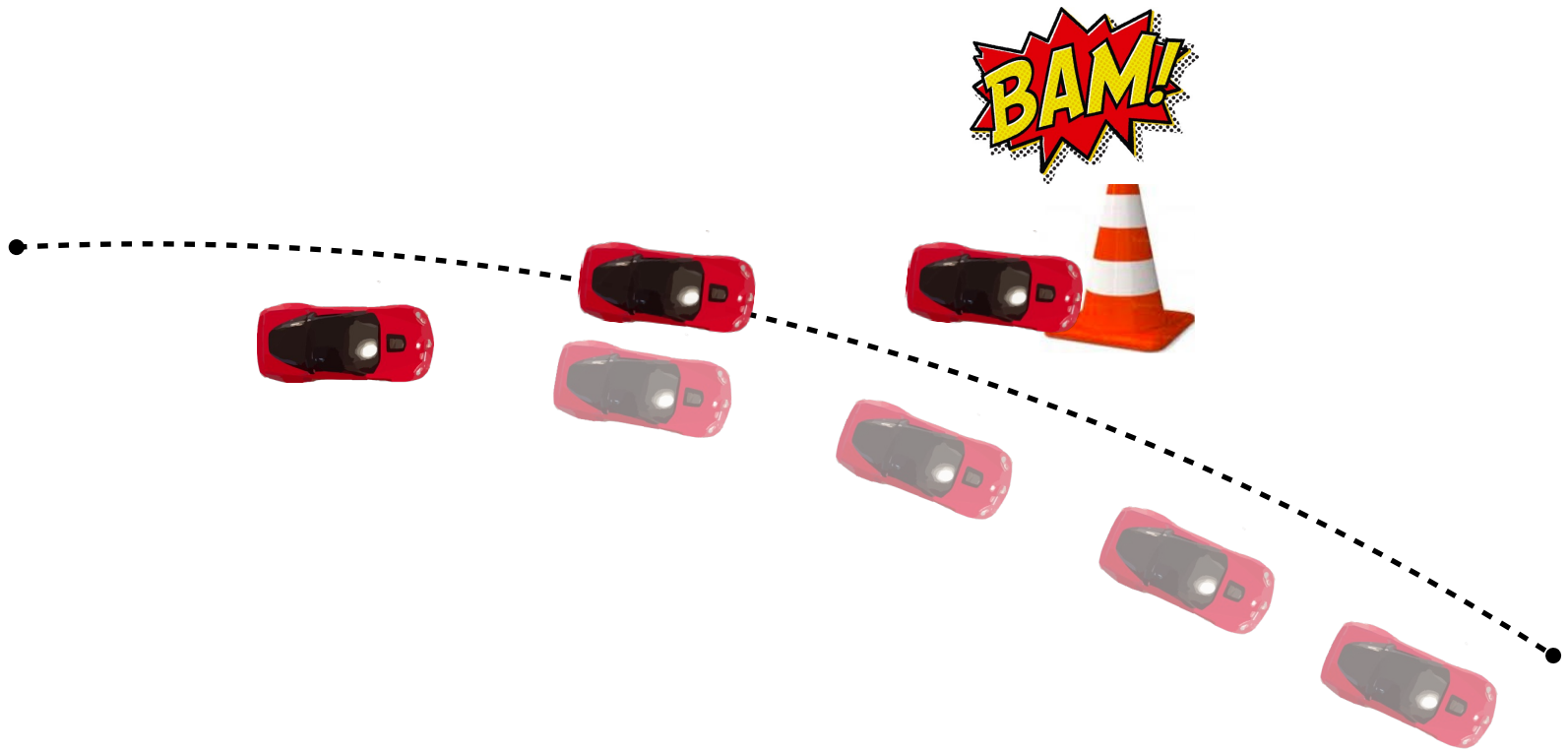
1. Open-loop controller may not be able to deal with unexpected events/divergences
2. Computation of full controller can be expensive:
   → Do it on the fly!
3. Model might be wrong, errors may accumulate
4. …

Why do we need to replan?



What happens if the controls are planned once and executed?

Why do we need to replan?

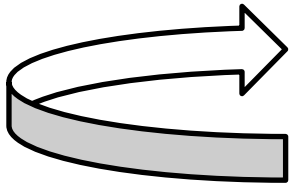


What happens if the controls are planned once and executed?

Solving Optimal Control with Sampling – issues?

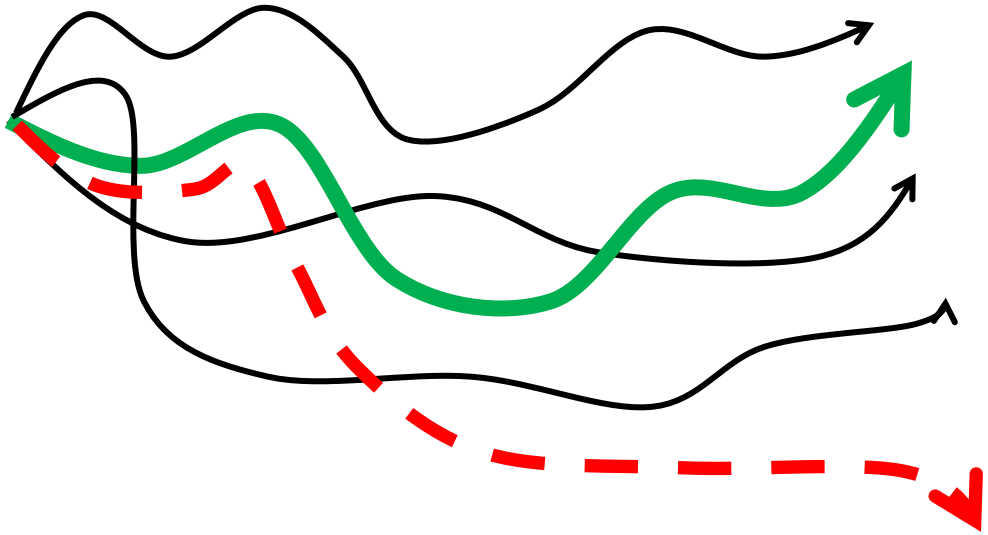
$$\min_{u_{1:T}} \sum_{t=1}^{T} c(x_t, u_t)$$

$$\text{s.t.} \quad x_{t+1} = f(x_t, u_t)$$

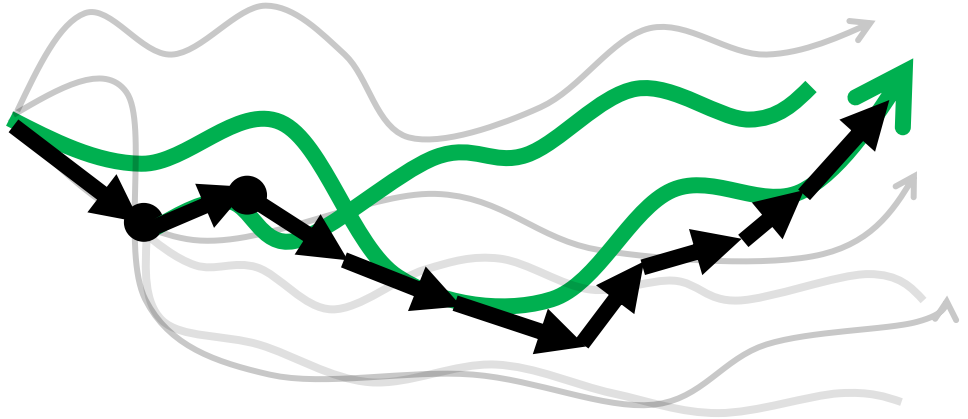


1. Plan with random shooting from s_t
2. Execute the first action a_0 and reach s_{t+1}

A stationary feedback controller may not be able to deal with unexpected events

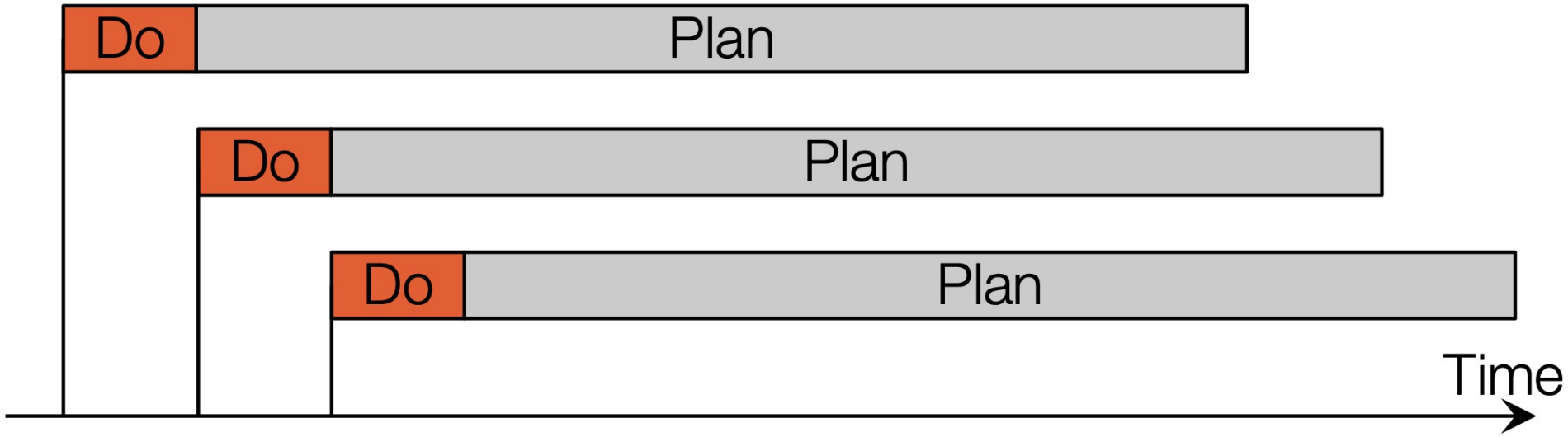


Replanning can help with divergence



Model-Predictive/Receding Horizon Control

General Replanning Framework - MPC

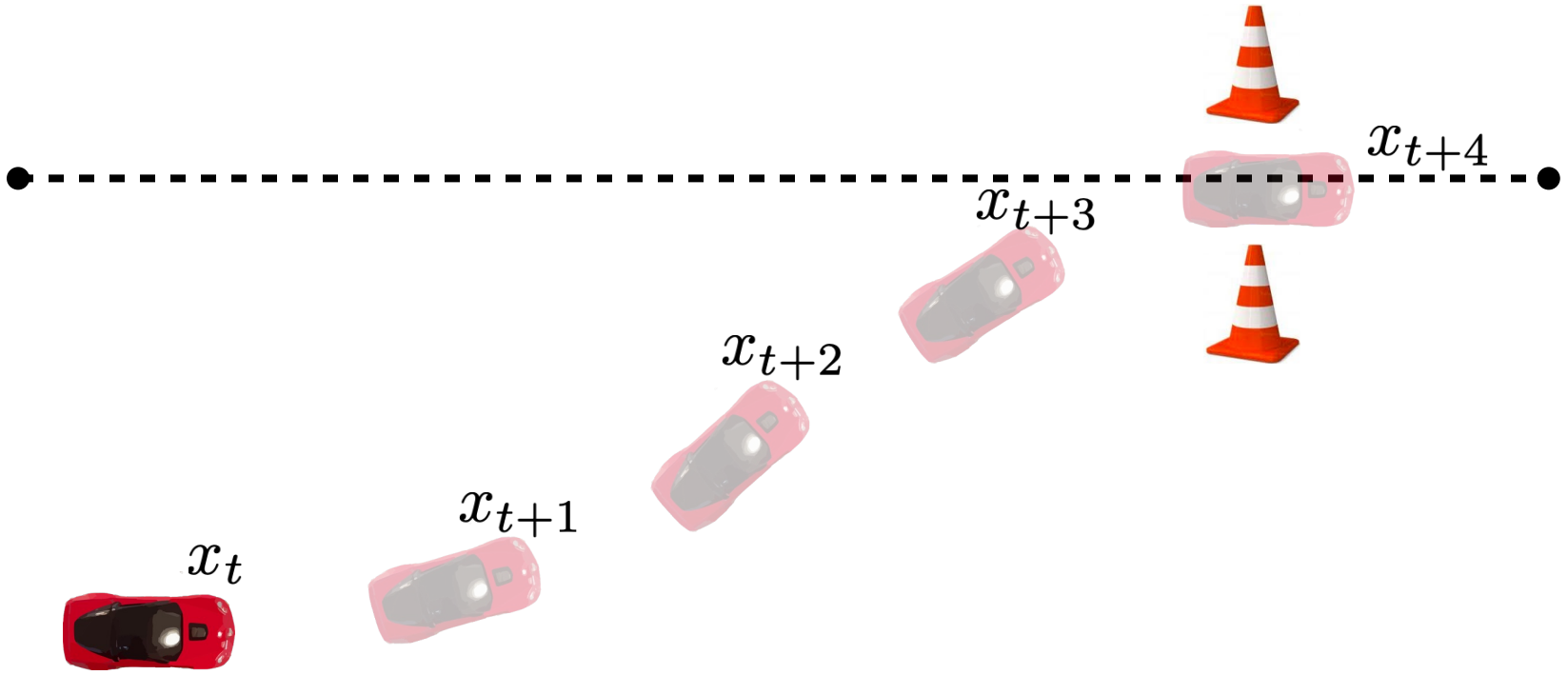


Step 1: Solve optimization problem to a horizon

Step 2: Execute the first control

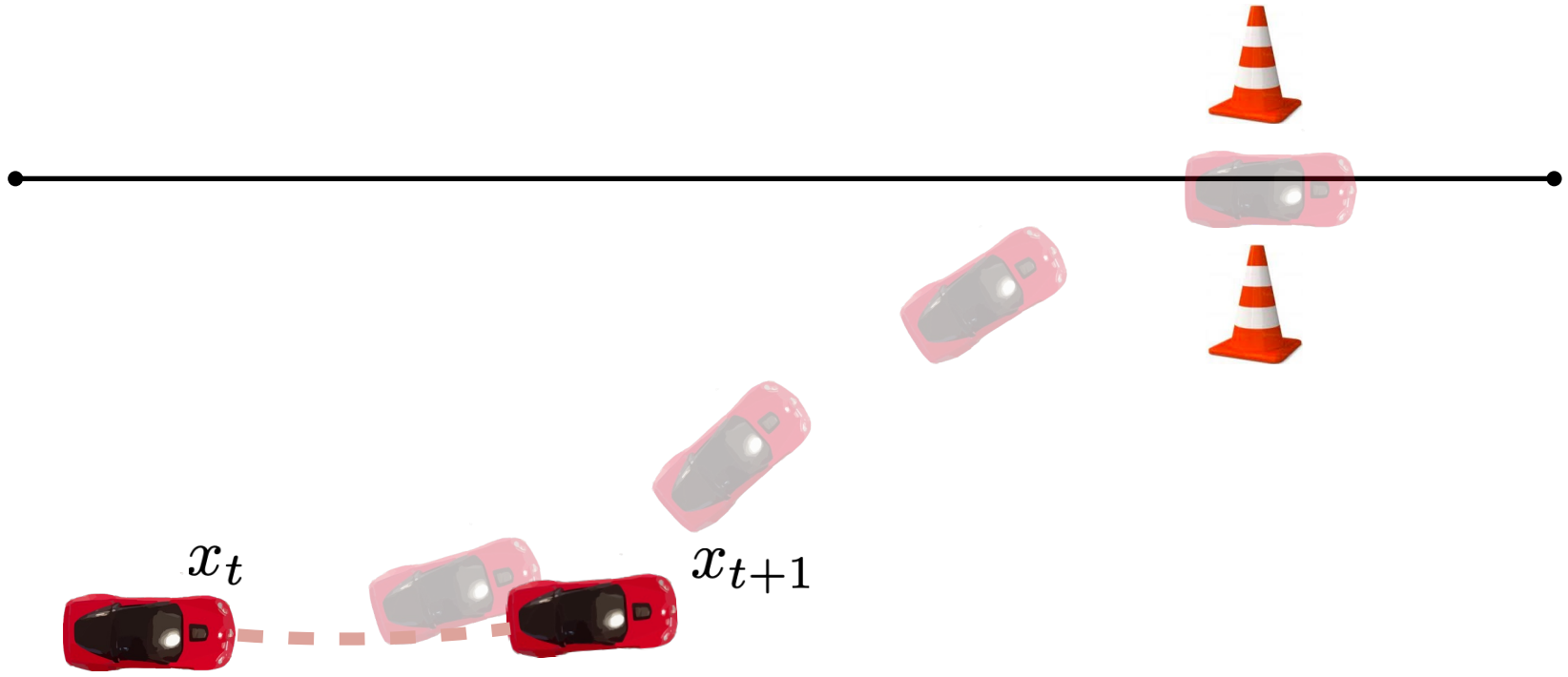
Step 3: Repeat!

How are the controls executed?



Step 1: Solve optimization problem to a horizon

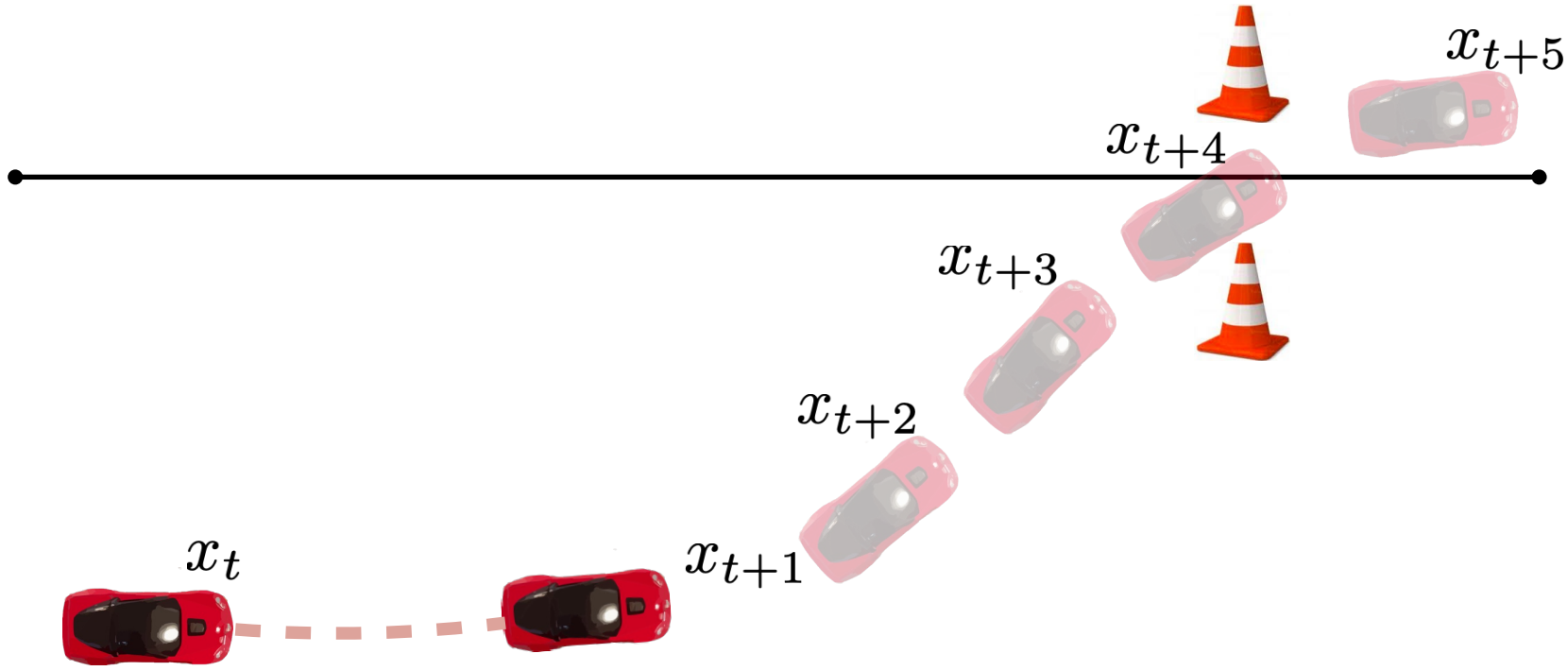
How are the controls executed?



Step 1: Solve optimization problem to a horizon

Step 2: Execute the first control

How are the controls executed?

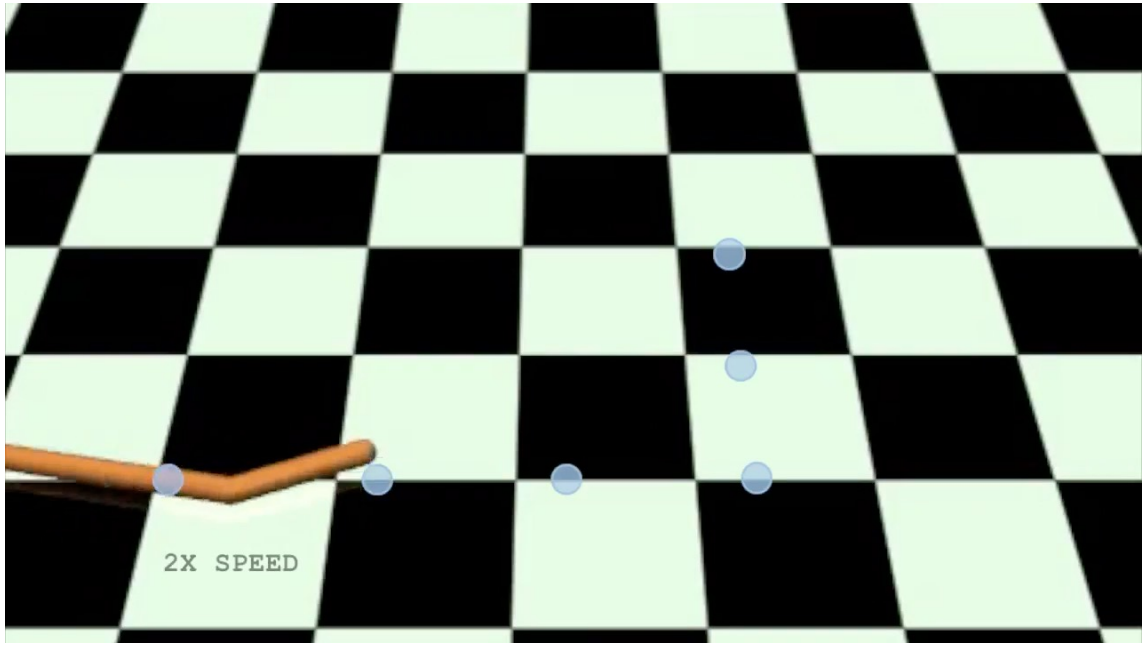


Step 1: Solve optimization problem to a horizon

Step 2: Execute the first control

Step 3: Repeat!

Does it work?



Nagabandi et al

Why might this not work?

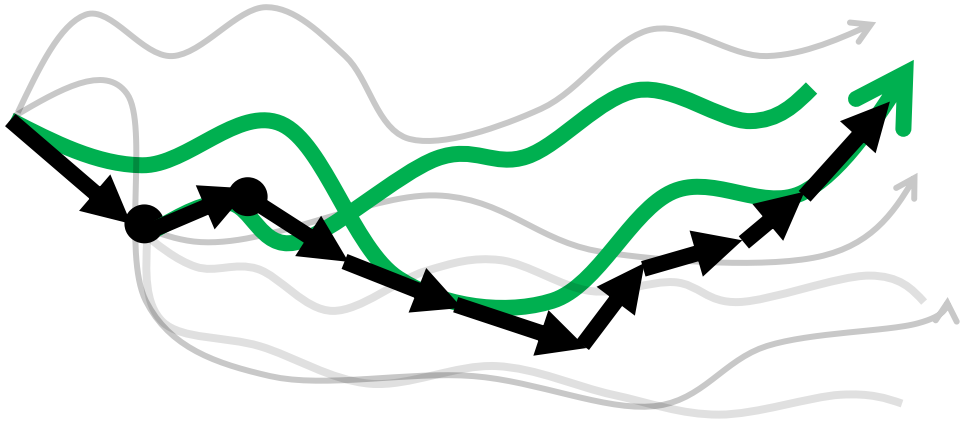
$$\min_{u_{1:T}} \sum_{t=1}^{T} c(x_t, u_t)$$

$$\text{s.t.} \quad x_{t+1} = f(x_t, u_t)$$

1. Sample a set of K action trajectories of T steps from start state
2. Evaluate each K step action sequence through the model and get per trajectory cost
3. Choose minimum trajectory cost trajectory
4. Execute lowest cost actions

Searching for a needle in a haystack by random shooting, high variance!

Planning with Shooting + MPC

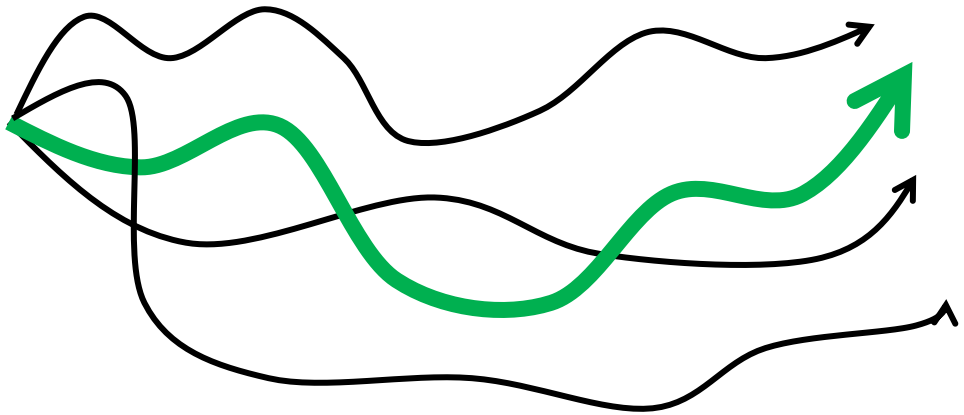


Better Sampling Techniques for MPC

Sampled from stationary uniform/gaussian distribution

$$\arg \min_{u_0, u_1, \ldots, u_T} \sum_{t=1}^{T} c(x_t, u_t)$$

$$x_{t+1} = f(x_t, u_t)$$

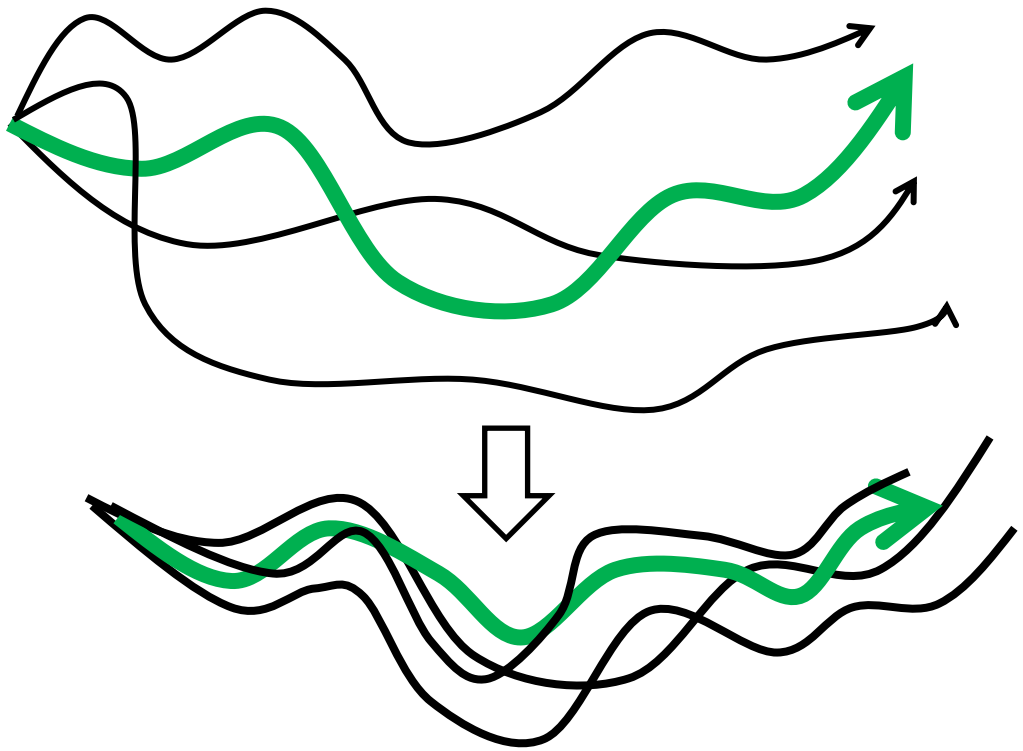


Can we inform the sampling function with the cost function?

Idea: Iteratively upweight sampling distribution around the things that are lower cost

Better Sampling Techniques for Shooting - MPPI

Idea: Iteratively upweight sampling distribution around the things that are lower costs



Referred to as **MPPI**, lower variance!

Sample N action sequences

$$(u_0^j, u_1^j, \ldots, u_T^j)_{j=1}^N \sim p(u)$$

Sample trajectories using these action sequences with the model

$$x_{t+1}^j = f(x_t^j, u_t^j)$$

Update action sampler by upweighting low cost actions

$$p(u) \leftarrow p(u) \frac{\exp \sum_{t=1}^T -c(x_t, u_t)}{Z}$$

Does it work?

Does it work?

Lecture Outline

Examples of LQR Systems

LQR Solutions

Sampling-based MPC