Autonomous Robotics

Winter 2025

Abhishek Gupta

TAs: Carolina Higuera, Entong Su, Bernie Zhu

Class Outline

State Estimation

- Robotic System Design
- Filtering
- Localization
- SLAM

Control

- Feedback Control
- PID Control
- MPC
- LQR

Planning

- Search
- Heuristic Search
- Motion Planning
- Lazy Search

Learning

- Imitation Learning
- Policy Gradient
- Actor-Critic
- Model-Based RL

Logistics

- Seeded Paper Discussion 2 - Wed Feb 19
- HW3 due Feb 20

- Post questions, discuss any issues you are having on Ed.
- Students with **no** access to 002, e-mail us with your student ID.
- Students that have not been added to the class, email abhgupta@cs.washington.edu with the subject-line “Waitlisted for CSE478”

Recap

Controller Design Decisions

Very particular cost function

Option 1:
Bang-bang control

Option 2:
PID control

Option 3:
Pure-pursuit control

Model Agnostic

Very simplistic model

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

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

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

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

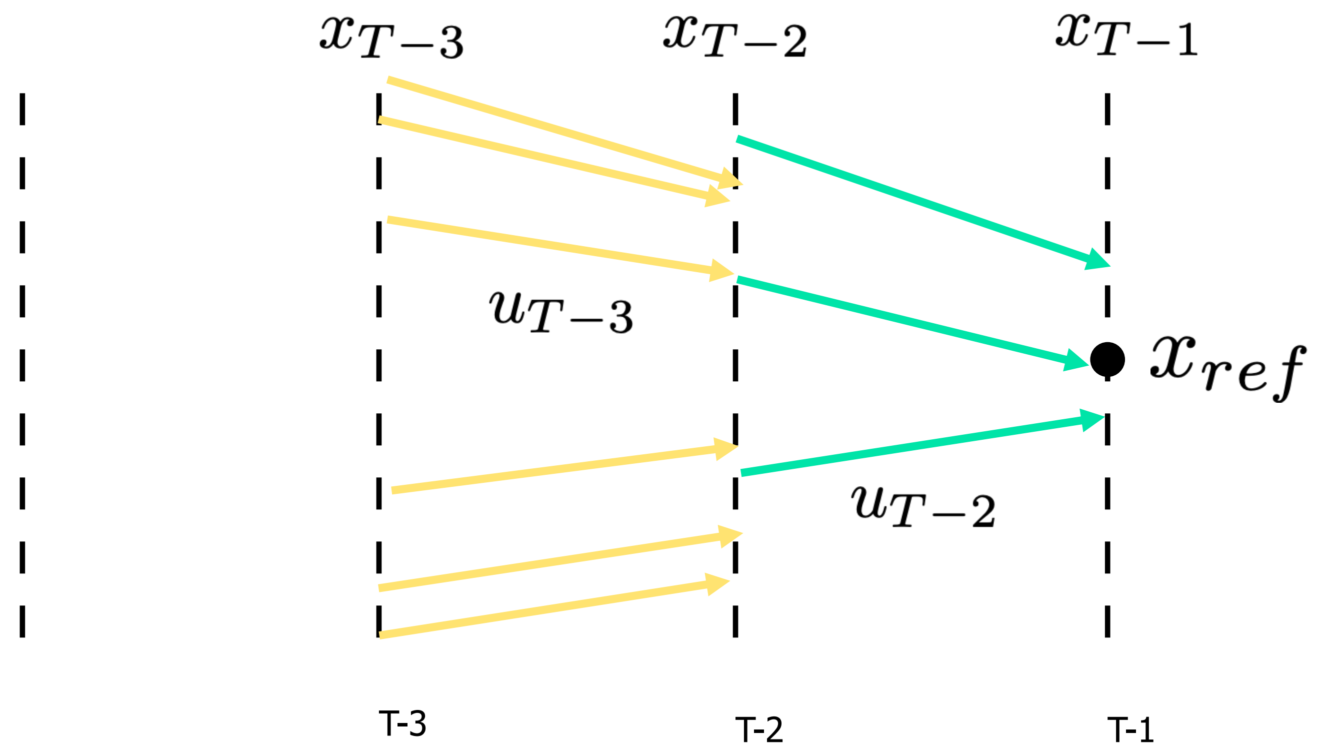
(N x 1) (N x N)(N x 1) (N x M)(M x 1)

STATE → NEXT STATE **CONTROL → NEXT STATE**

How do we solve for controls?

Dynamic programming to the rescue!

Start from timestep T-1 and solve backwards



Lecture Outline

Solving LQR Problems

$\downarrow$

LQR Variants

$\downarrow$

From LQR to 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

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

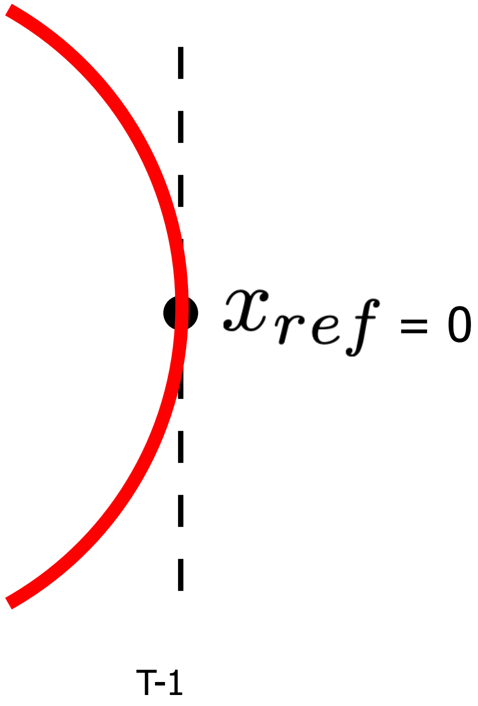
(whiteboard)

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

$$P_0 = Q$$

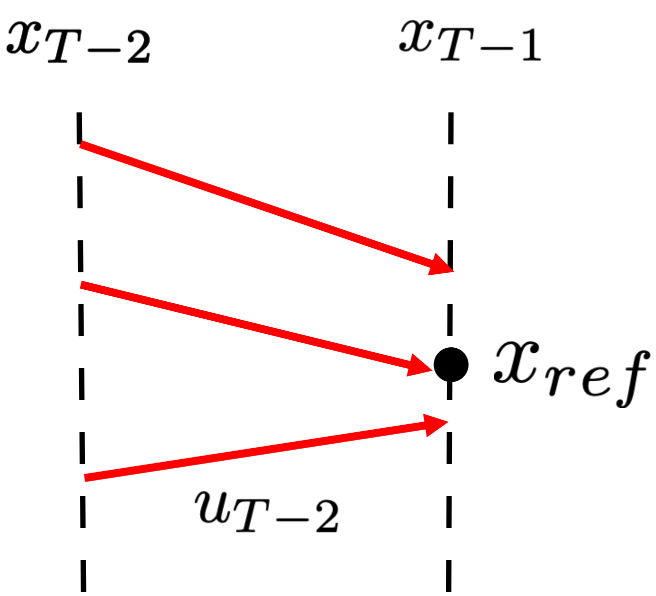


Note that the cost is quadratic in x

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

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

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

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

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)
```

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

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

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

Current state cost

Current action cost

Optimal cost in the future based on dynamics

Linear Quadratic Regulator

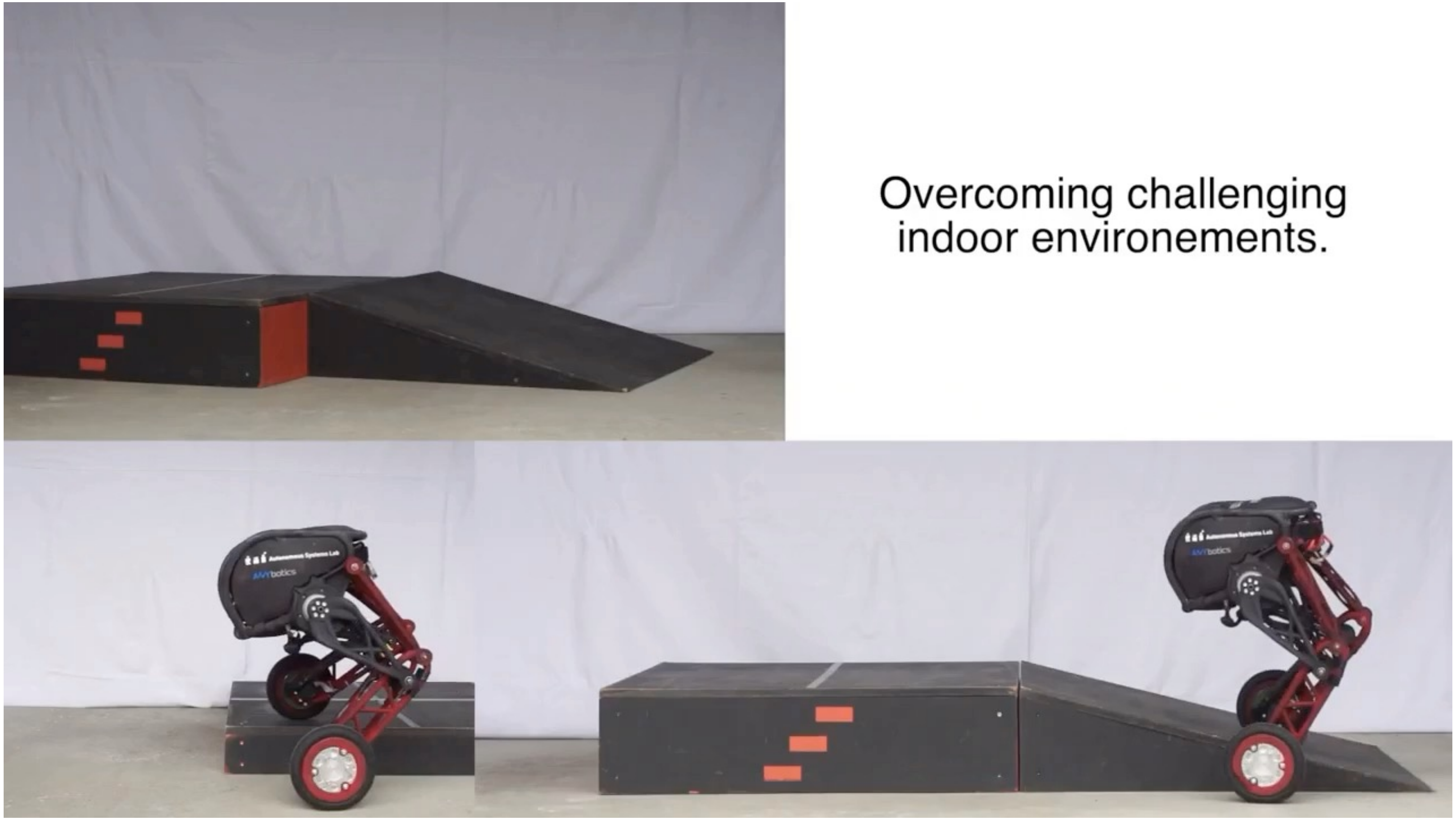
- For **linear** systems with **quadratic** costs, we can write down very efficient algorithms that return the optimal sequence of actions!
  - Special case where dynamic programming can be applied to continuous states and actions (typically only discrete states and actions)
- Many LQR extensions: non-linear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



Klemm et al 2020

Lecture Outline

Solving LQR Problems

↓

LQR Variants

↓

From LQR to MPC

LQR assumptions revisited

$$
\begin{aligned}
x_{t+1} &= Ax_t + Bu_t \\
g(x_t, u_t) &= x_t^\top Q x_t + u_t^\top R u_t
\end{aligned}
$$

= for keeping a linear system at the all-zeros state while preferring to keep the control input small.

- Extensions make it more generally applicable:
  - Affine systems
  - Systems with stochasticity
  - Non-linear systems
  - Linear time varying (LTV) systems
  - Trajectory following for non-linear systems

LQR assumptions revisited

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ g(x_t, u_t) &= x_t^\top Q x_t + u_t^\top R u_t \end{aligned}$$

= for keeping a linear system at the all-zeros state while preferring to keep the control input small.

- Extensions make it more generally applicable:
  - Affine systems
  - Systems with stochasticity
  - **Non-linear systems**
  - **Linear time varying (LTV) systems**
  - Trajectory following for non-linear systems

LQR Ext1: non-linear systems

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

We can keep the system at the state x^* iff

$$\exists u^* \text{s.t.} \quad x^* = f(x^*, u^*)$$

Linearizing the dynamics around x^* gives:

$$x_{t+1} \approx f(x^*, u^*) + \underbrace{\frac{\partial f}{\partial x}(x^*, u^*)}_{A}(x_t - x^*) + \underbrace{\frac{\partial f}{\partial u}(x^*, u^*)}_{B}(u_t - u^*)$$

Equivalently:

$$x_{t+1} - x^* \approx A(x_t - x^*) + B(u_t - u^*)$$

Let $z_t = x_t - x^*$, let $v_t = u_t - u^*$, then:

$$z_{t+1} = Az_t + Bv_t, \qquad \text{cost} = z_t^\top Q z_t + v_t^\top R v_t$$ [=standard LQR]

$$v_t = Kz_t \Rightarrow u_t - u^* = K(x_t - x^*) \Rightarrow u_t = u^* + K(x_t - x^*)$$

LQR Ext2: Linear Time Varying (LTV) Systems

$$
\begin{aligned}
x_{t+1} &= A_t x_t + B_t u_t \\
g(x_t, u_t) &= x_t^\top Q_t x_t + u_t^\top R_t u_t
\end{aligned}
$$

LQR Ext2: Linear Time Varying (LTV) Systems

Set $P_0 = 0$.
for $i = 1, 2, 3, \ldots$

$$
\begin{aligned}
K_i &= -(R_{H-i} + B_{H-i}^\top P_{i-1} B_{H-i})^{-1} B_{H-i}^\top P_{i-1} A_{H-i} \\
P_i &= Q_{H-i} + K_i^\top R_{H-i} K_i + (A_{H-i} + B_{H-i} K_i)^\top P_{i-1} (A_{H-i} + B_{H-i} K_i)
\end{aligned}
$$

The optimal policy for a i-step horizon is given by:

$$\pi(x) = K_i x$$

The cost-to-go function for a i-step horizon is given by:

$$J_i(x) = x^\top P_i x.$$

LQR Ext3: Trajectory Following for Non-Linear Systems

- A state sequence $x_0^*, x_1^*, \ldots, x_H^*$ is a feasible target trajectory if and only if
- Problem statement: $\exists u_0^*, u_1^*, \ldots, u_{H-1}^* \ : \ \forall t \in \{0, 1, \ldots, H-1\} \ : \ x_{t+1}^* = f(x_t^*, u_t^*)$

$$\begin{aligned} &\min_{u_0, u_1, \ldots, u_{H-1}} \sum_{t=0}^{H-1} (x_t - x_t^*)^\top Q (x_t - x_t^*) + (u_t - u_t^*)^\top R (u_t - u_t^*) \\ &\text{s.t.} \quad x_{t+1} = f(x_t, u_t) \end{aligned}$$

- Transform into linear time varying case (LTV):

$$x_{t+1} \approx f(x_t^*, u_t^*) + \underbrace{\frac{\partial f}{\partial x}(x_t^*, u_t^*)}_{A_t}(x_t - x_t^*) + \underbrace{\frac{\partial f}{\partial u}(x_t^*, u_t^*)}_{B_t}(u_t - u_t^*)$$

$$x_{t+1} - x_{t+1}^* \approx A_t (x_t - x_t^*) + B_t (u_t - u_t^*)$$

LQR Ext3: Trajectory Following for Non-Linear Systems

- Transformed into linear time varying case (LTV):

$$\min_{u_0,u_1,\ldots,u_{H-1}} \sum_{t=0}^{H-1} (x_t - x_t^*)^\top Q (x_t - x_t^*) + (u_t - u_t^*)^\top R (u_t - u_t^*)$$

$$\text{s.t. } x_{t+1} - x_{t+1}^* = A_t(x_t - x_t^*) + B_t(u_t - u_t^*)$$

- Now we can run the standard LQR back-up iterations.
- Resulting policy at i time-steps from the end:

$$u_{H-i} - u_{H-i}^* = K_i(x_{H-i} - x_{H-i}^*)$$

- The target trajectory need not be feasible to apply this technique, however, if it is infeasible then there will an offset term in the dynamics:

$$x_{t+1} - x_{t+1}^* = f(x_t, u_t) - x_{t+1}^* + A_t(x_t - x_t^*) + B_t(u_t - u_t^*)$$

Iteratively Apply LQR

Initialize the algorithm by picking either (a) A control policy $\pi^{(0)}$ or (b) A sequence of states $x_0^{(0)}, x_1^{(0)}, \ldots, x_H^{(0)}$ and control inputs $u_0^{(0)}, u_1^{(0)}, \ldots, u_H^{(0)}$. With initialization (a), start in Step (1). With initialization (b), start in Step (2).

Iterate the following:

(1) Execute the current policy $\pi^{(i)}$ and record the resulting state-input trajectory $x_0^{(i)}, u_0^{(i)}, x_1^{(i)}, u_1^{(i)}, \ldots, x_H^{(i)}, u_H^{(i)}$.

(2) Compute the LQ approximation of the optimal control problem around the obtained state-input trajectory by computing a first-order Taylor expansion of the dynamics model, and a second-order Taylor expansion of the cost function.

(3) Use the LQR back-ups to solve for the optimal control policy $\pi^{(i+1)}$ for the LQ approximation obtained in Step (2).

(4) Set $i = i + 1$ and go to Step (1).

Lecture Outline

Solving LQR Problems

↓

LQR Variants

↓

From LQR to MPC

Why might this not be enough?

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

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

Non-linear

Non-quadratic

Use linear/quadratic Taylor expansion about current nominal states/actions

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

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

Might be a poor, local approximation!

May not be able to incorporate constraints

Let's revisit ideas from Bayesian filtering

	Linear Gaussian assumption	Sampling-based approximation
<u>Filtering</u>	Kalman Filtering	Particle Filtering
<u>Control</u>	LQR	Sampling based MPC

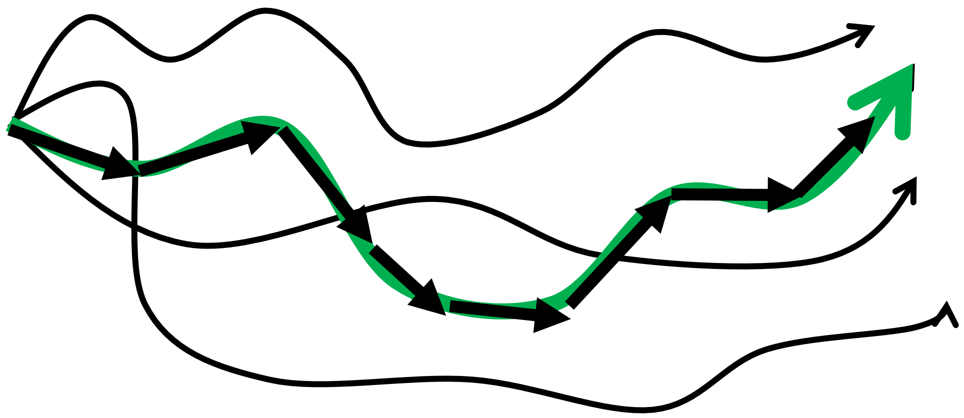
Solving Optimal Control with Sampling

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

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

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



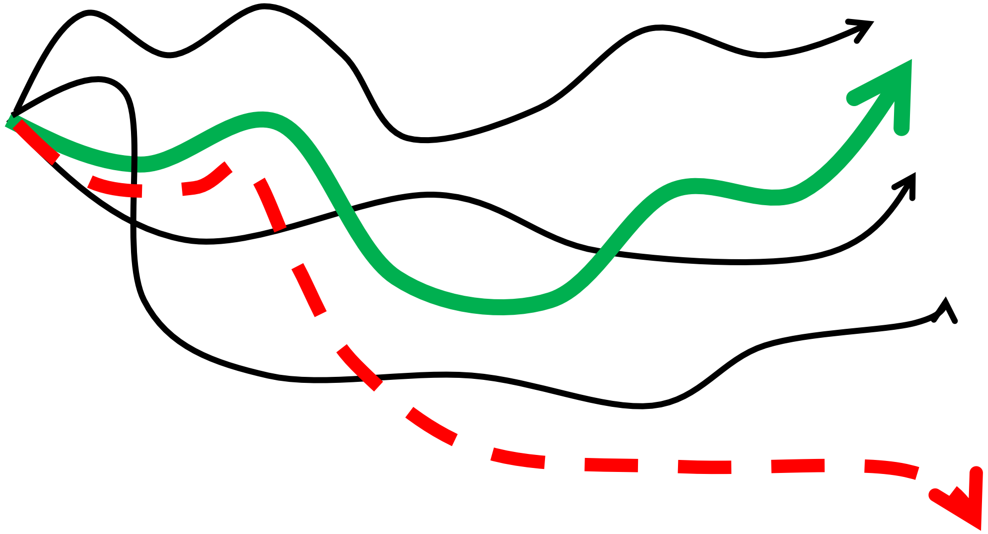
Can soften by taking softmin rather than argmin

Solving Optimal Control with Sampling – issues?

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

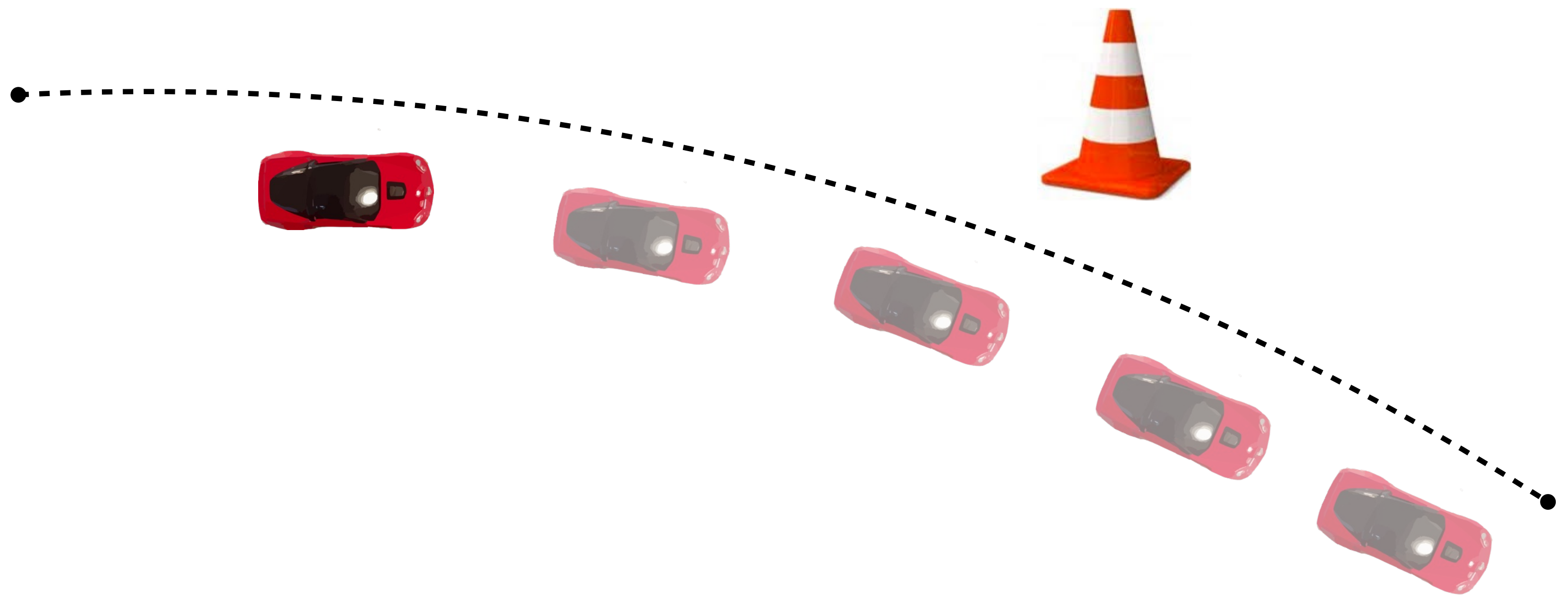
$$\text{s.t. } \; x_{t+1} = f(x_t, u_t)$$

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



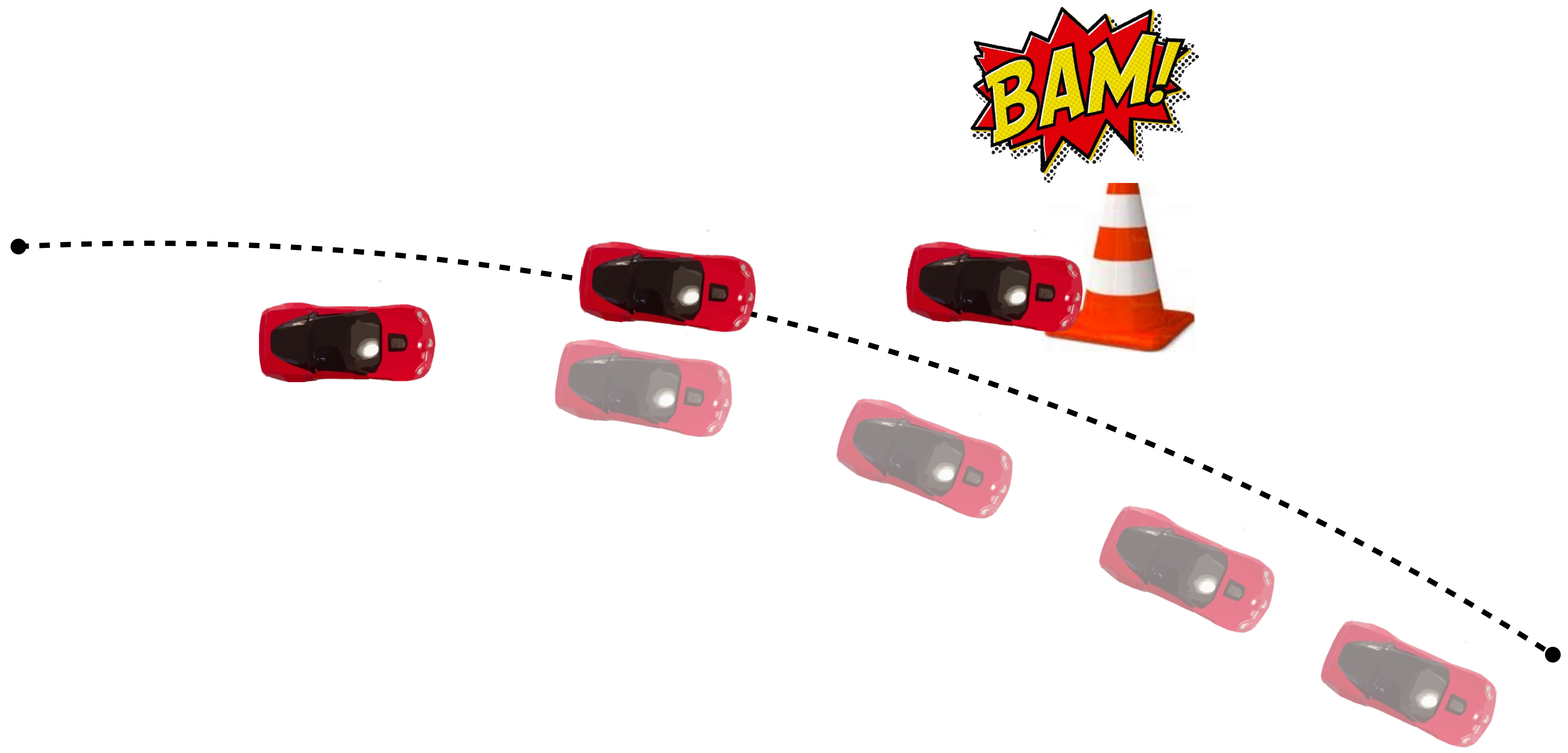
1. Open-loop controller may not be able to deal with unexpected events/divergences
2. Computation of full controller can be expensive: → Do it on the fly!
3. Model might be wrong, errors may accumulate
4. …

Why do we need to replan?



What happens if the controls are planned once and executed?

Why do we need to replan?

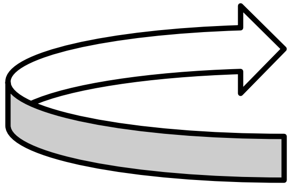


What happens if the controls are planned once and executed?

Solving Optimal Control with Sampling – issues?

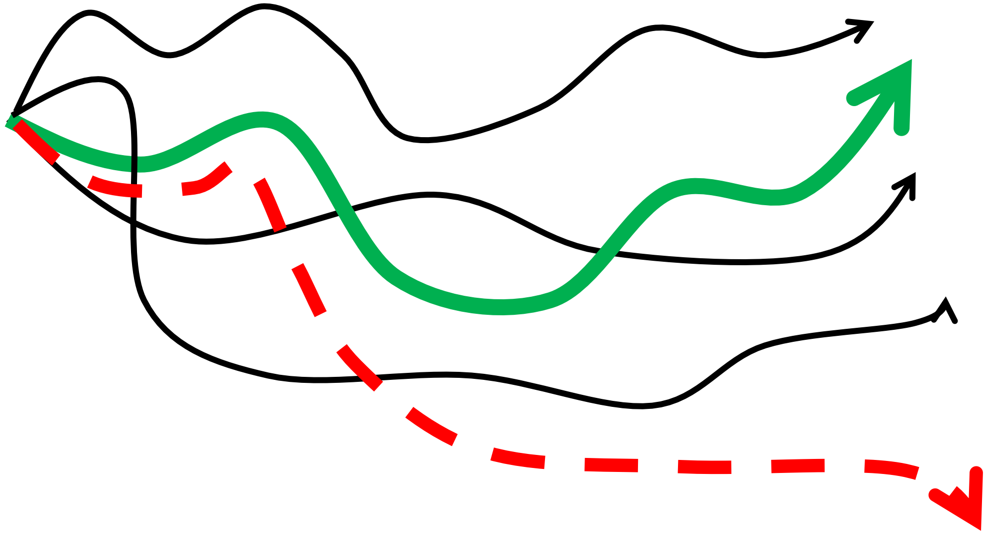
$$\min_{u_{1:T}} \sum_{t=1}^{T} c(x_t, u_t)$$

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

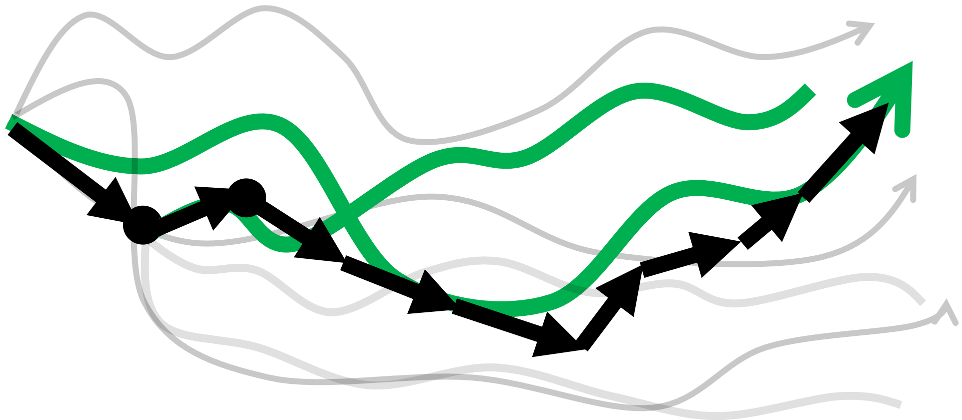


1. Plan with random shooting from s_t
2. Execute the first action a_0 and reach s_{t+1}

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

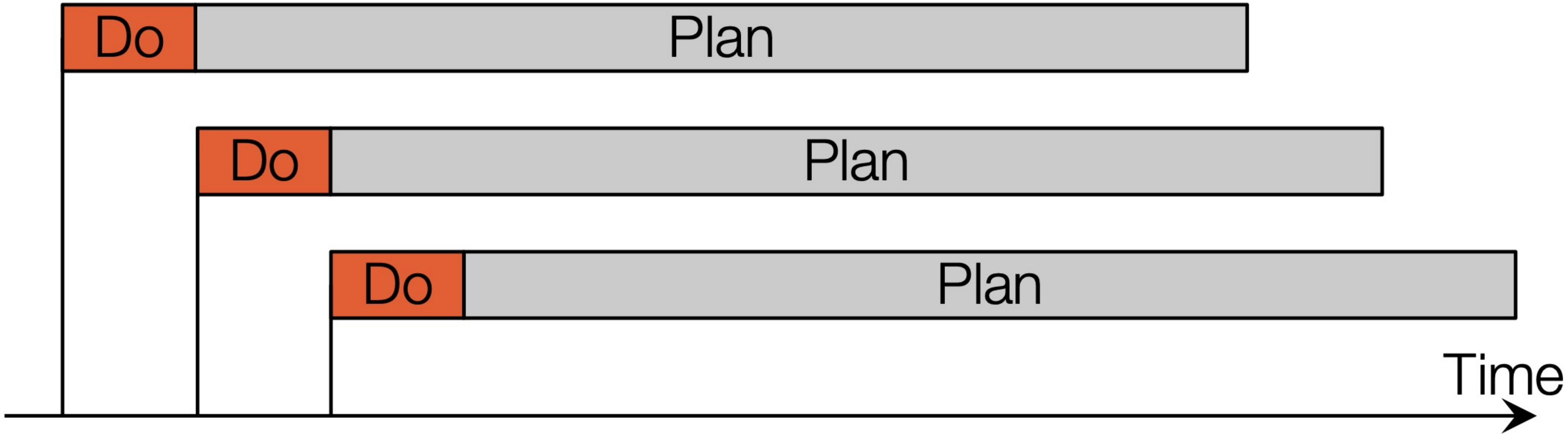


Replanning can help with divergence



Model-Predictive/Receding Horizon Control

General Replanning Framework - MPC

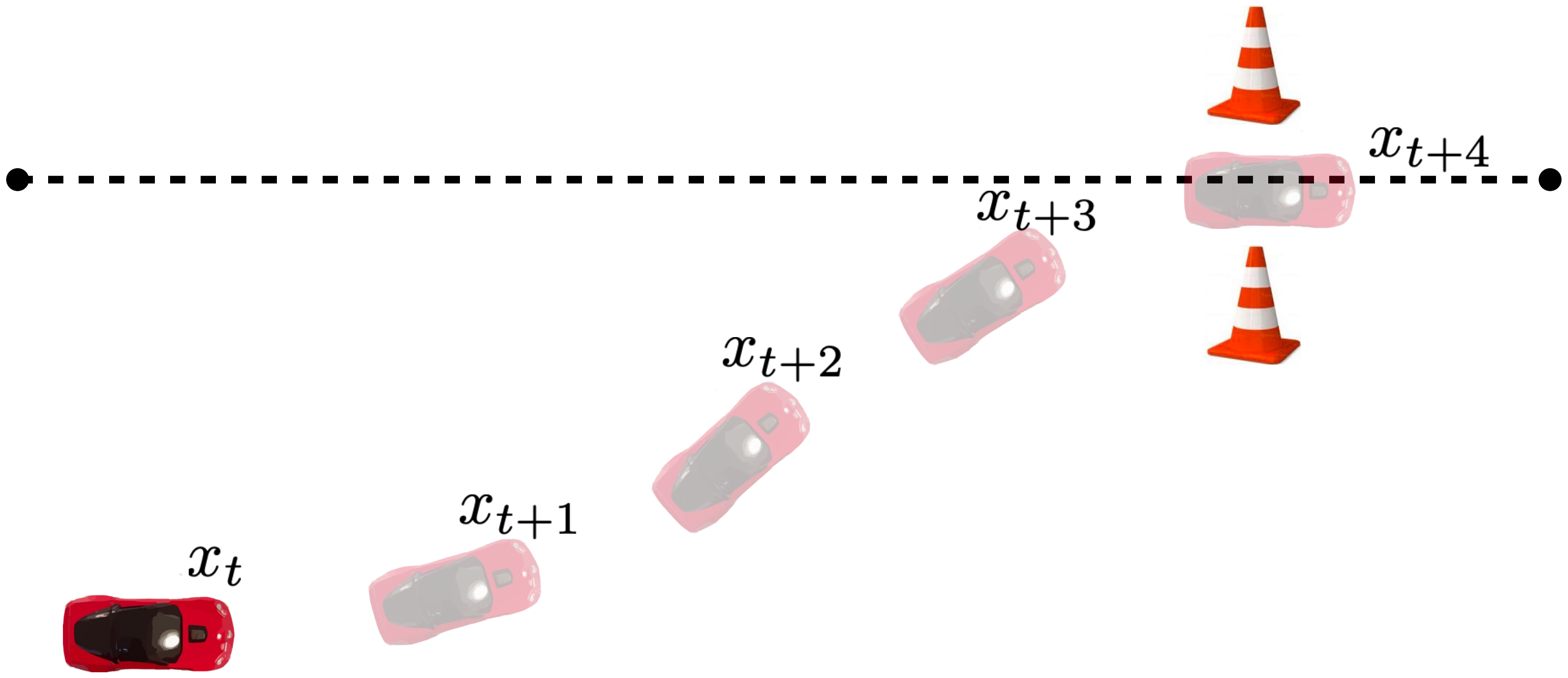


Step 1: Solve optimization problem to a horizon

Step 2: Execute the first control

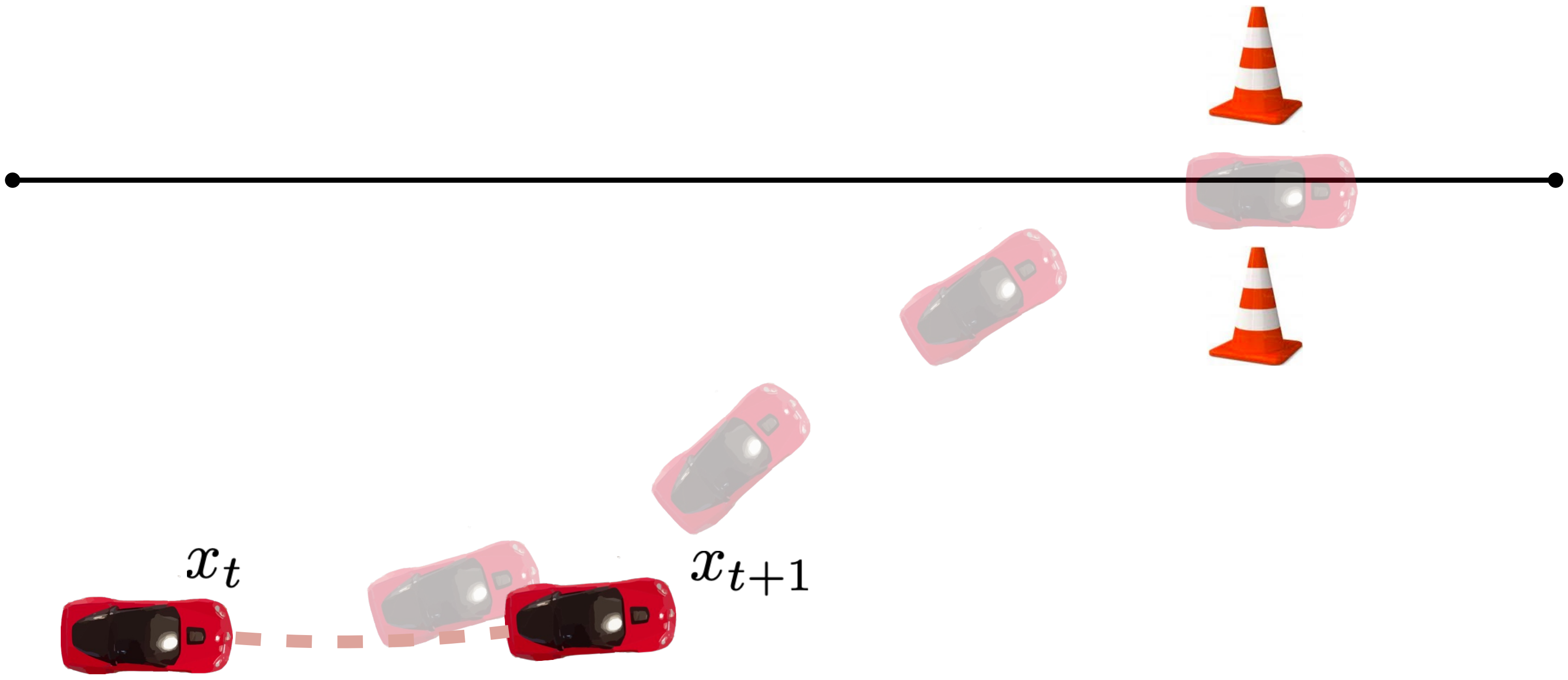
Step 3: Repeat!

How are the controls executed?



Step 1: Solve optimization problem to a horizon

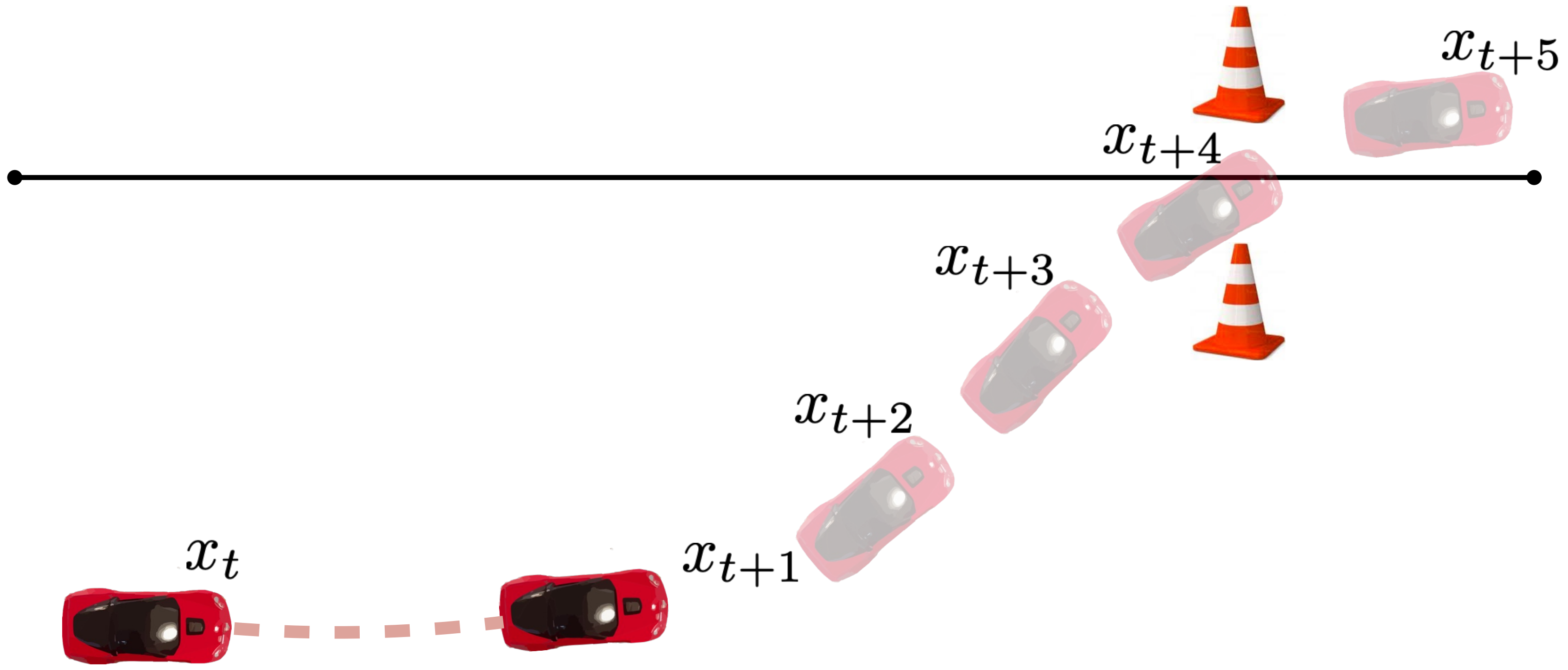
How are the controls executed?



Step 1: Solve optimization problem to a horizon

Step 2: Execute the first control

How are the controls executed?

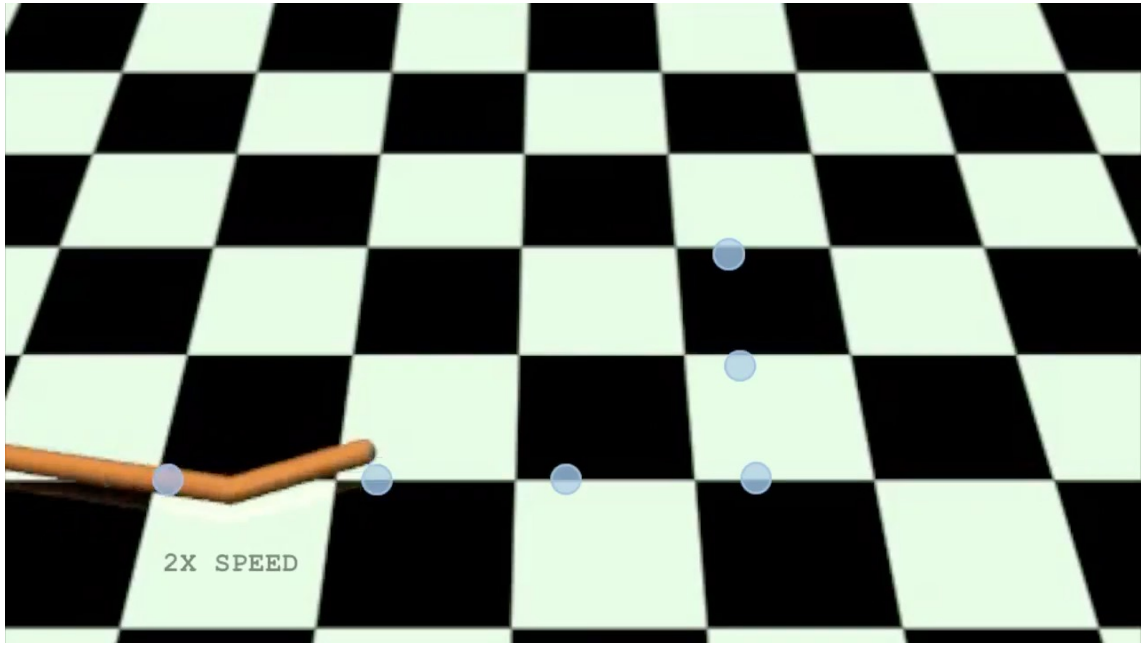


Step 1: Solve optimization problem to a horizon

Step 2: Execute the first control

Step 3: Repeat!

Does it work?



Nagabandi et al

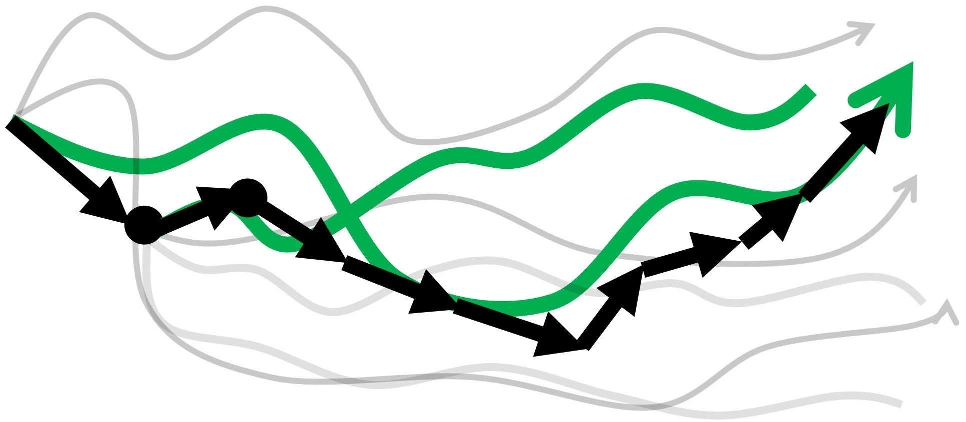
Why might this not work?

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

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

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

Planning with Shooting + MPC



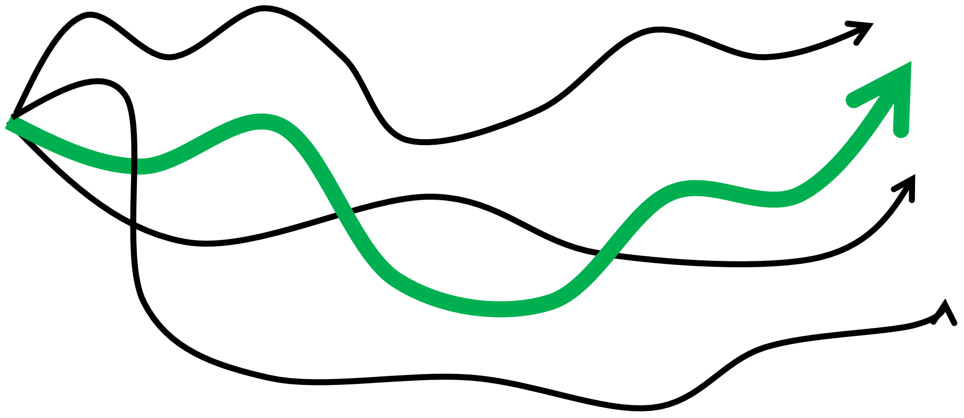
Searching for a needle in a haystack by random shooting, high variance!

Better Sampling Techniques for MPC

Sampled from stationary uniform/gaussian distribution

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

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

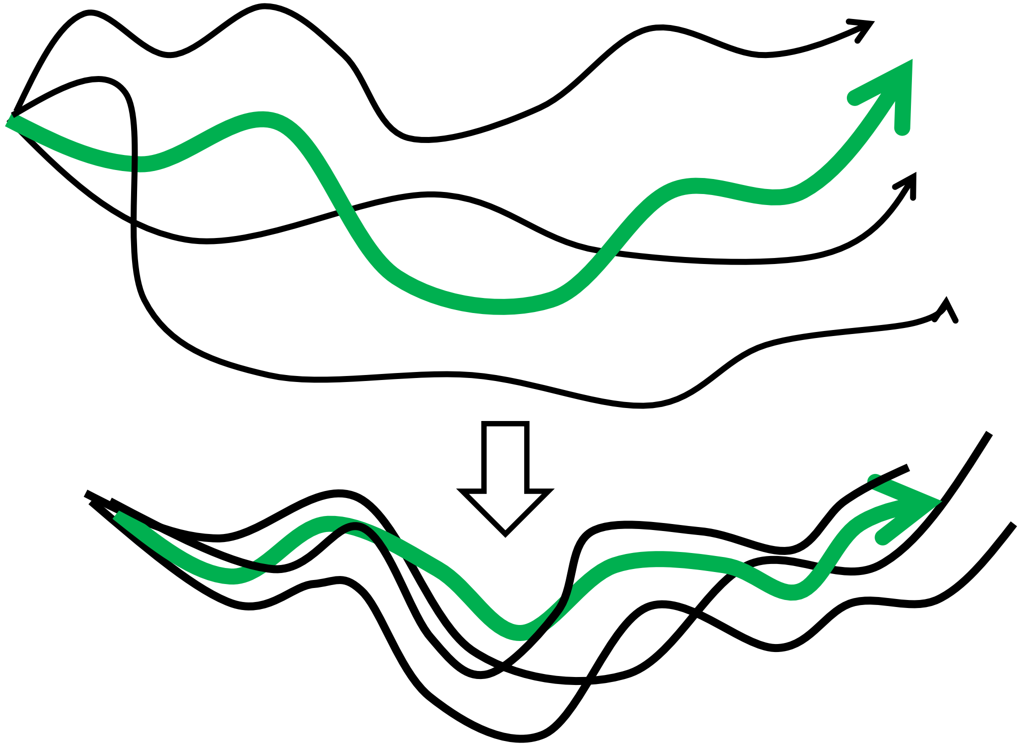


Can we inform the sampling function with the cost function?

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

Solving LQR Problems

↓

LQR Variants

↓

From LQR to MPC

Class Outline

State Estimation

- Robotic System Design
- Filtering
- Localization
- SLAM

Control

- Feedback Control
- PID Control
- MPC
- LQR

Planning

- Search
- Heuristic Search
- Motion Planning
- Lazy Search

Learning

- Imitation Learning
- Policy Gradient
- Actor-Critic
- Model-Based RL