Autonomous Robotics
Winter 2024

Abhishek Gupta

TAs: Karthikeya Vemuri, Arnav Thareja

Marius Memmel, Yunchu Zhang

Slides borrowed from many sources – Sidd Srinivasa, Sanjiban Choudhury

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

- HW 2 due tomorrow
- HW3 out on Feb 3 (Saturday)

Lecture Outline

Linear Quadratic Regulator Problems

Linear Quadratic Regulator

Beyond the Linear Quadratic Regulator

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

Control as an Optimization Problem

- For a sequence of H control actions
  1. Use model to predict consequence of actions (i.e., H future states)
  2. Evaluate the cost function
- Compute optimal sequence of H control actions (minimizes cost)

Generalized Problem: Optimal Control

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

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

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

Can be costs like smoothness, preferences, speed

Can be constraints like velocity/acceleration bounds

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

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

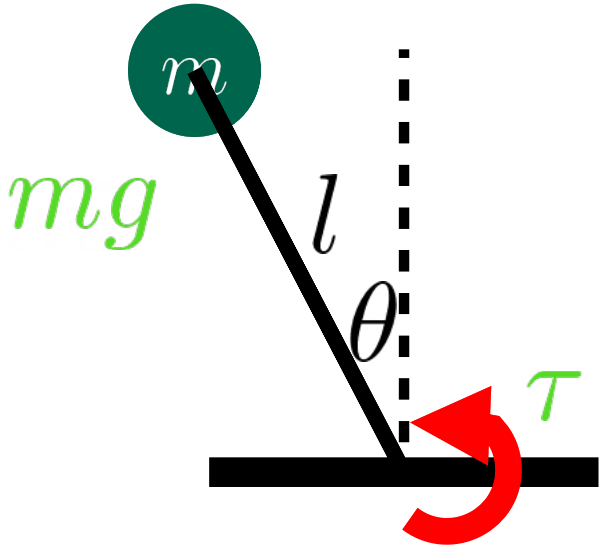
$$\sum_t x_t^\top Q x_t + u_t^\top R u_t$$

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

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

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

Example: Inverted Pendulum (Linear System)



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

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

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

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)

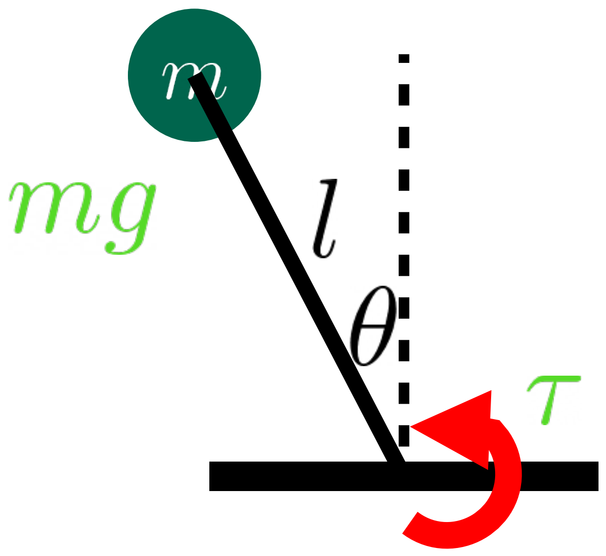
STATE COST

$$u_t^\top R u_t$$

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

CONTROL COST

Example: Inverted Pendulum (State Cost)



$$x_t^\top Q x_t$$

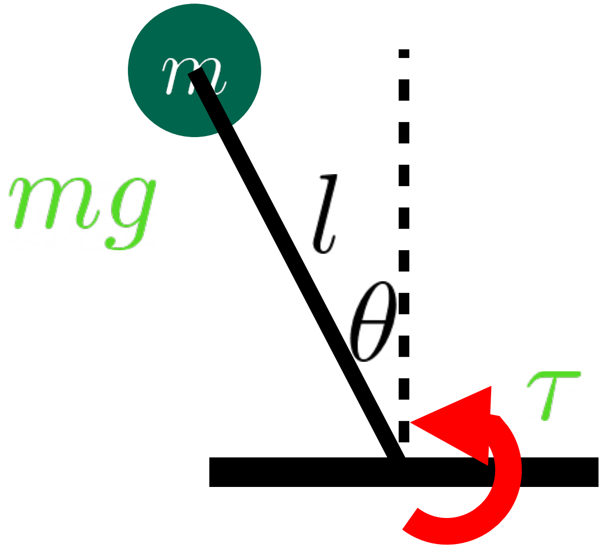
(QUADRATIC FORM)

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

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

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

Example: Inverted Pendulum (Control Cost)



$$u_t^\top R u_t$$

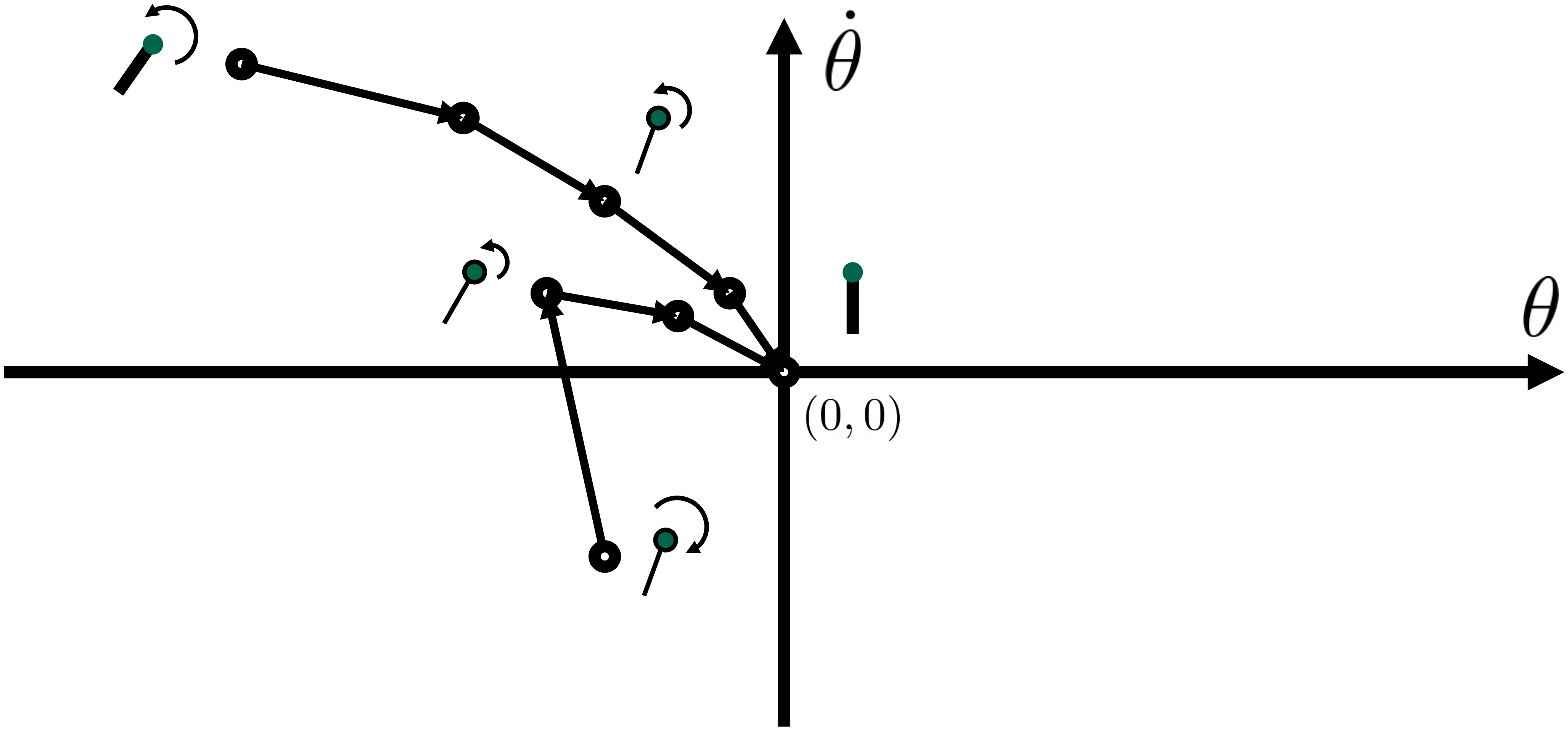
(QUADRATIC FORM)

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

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

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

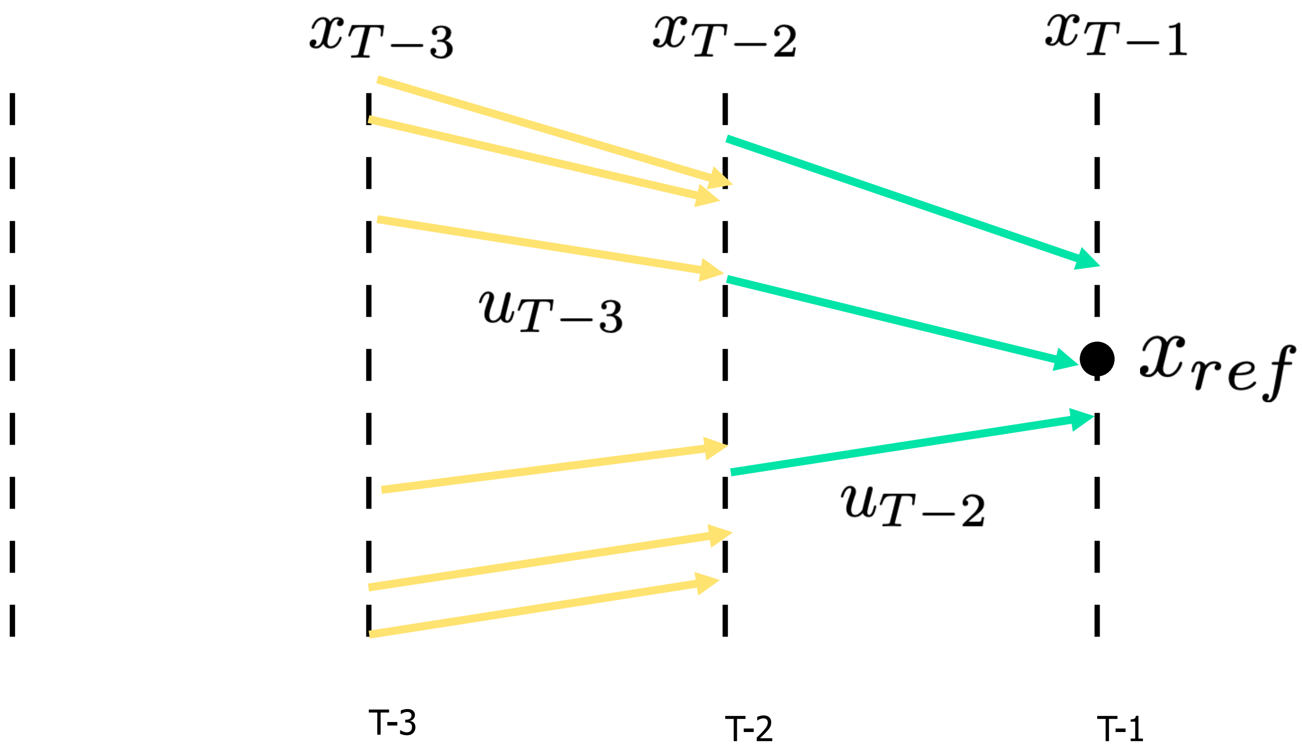
Example: Inverted Pendulum



How do we solve for controls?

Dynamic programming to the rescue!

Start from timestep T-1 and solve backwards



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

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

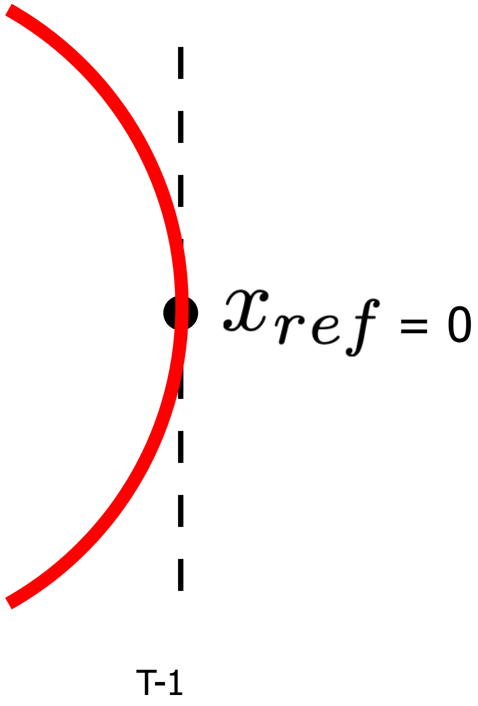
(whiteboard)

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

$$J_0(x) = \min_u x^\top Qx + u^\top Ru = x^\top Qx = x^\top P_0 x$$

Minimized with u = 0

$$P_0 = Q$$

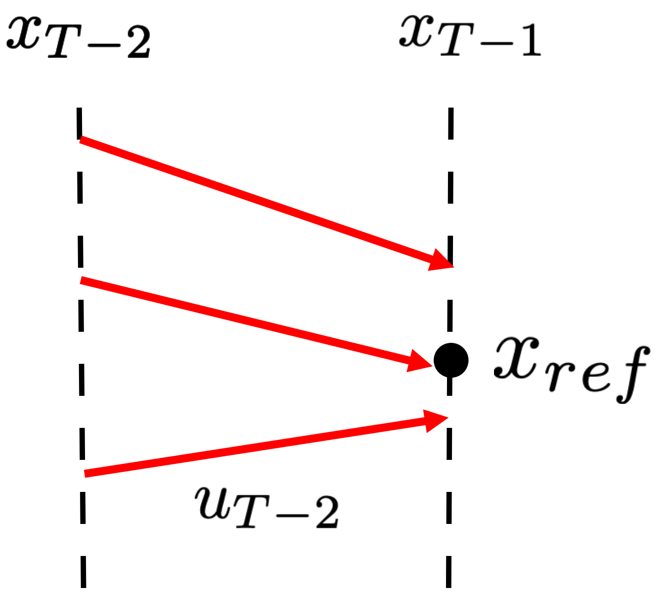


Note that the cost is quadratic in x

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

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

Current state cost

Current action cost

Optimal cost in the future based on dynamics

Linear Quadratic Regulator

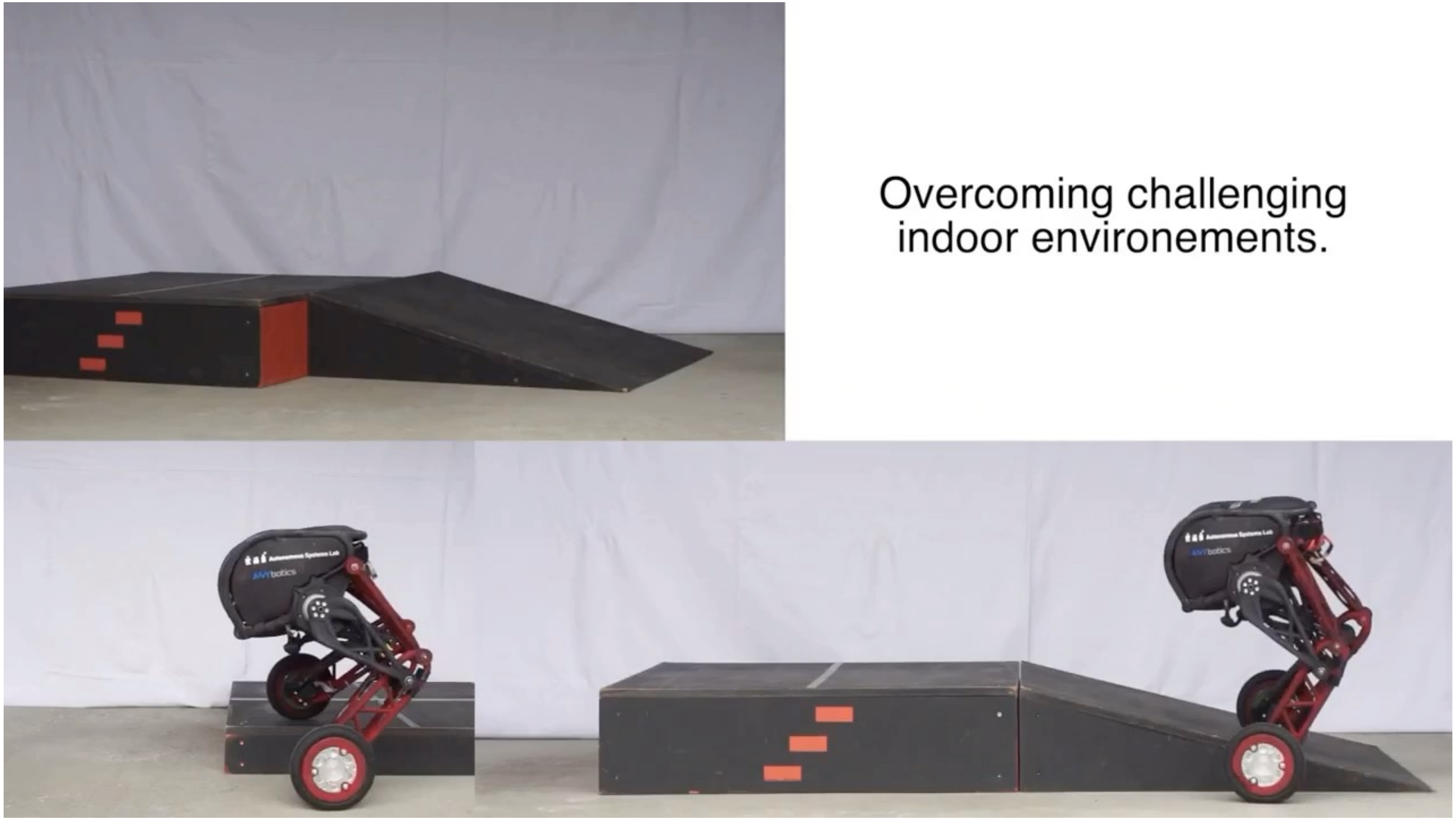
- For **linear** systems with **quadratic** costs, we can write down very efficient algorithms that return the optimal sequence of actions!
  - Special case where dynamic programming can be applied to continuous states and actions (typically only discrete states and actions)
- Many LQR extensions: non-linear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



Klemm et al 2020

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
  - Penalization for change in control inputs
  - 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

$$\exists u_0^*, u_1^*, \ldots, u_{H-1}^* \ : \ \forall t \in \{0, 1, \ldots, H-1\} \ : \ x_{t+1}^* = f(x_t^*, u_t^*)$$

- Problem statement:

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

- 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).

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