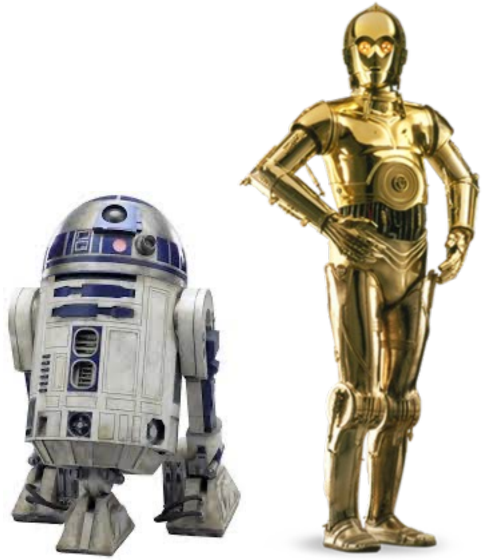
Autonomous Robotics
Winter 2024

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 on Feb 2
- HW3 out on Feb 3 (Saturday) barring no hiccups in testing ☺

Lecture Outline

Recap of Pure Pursuit

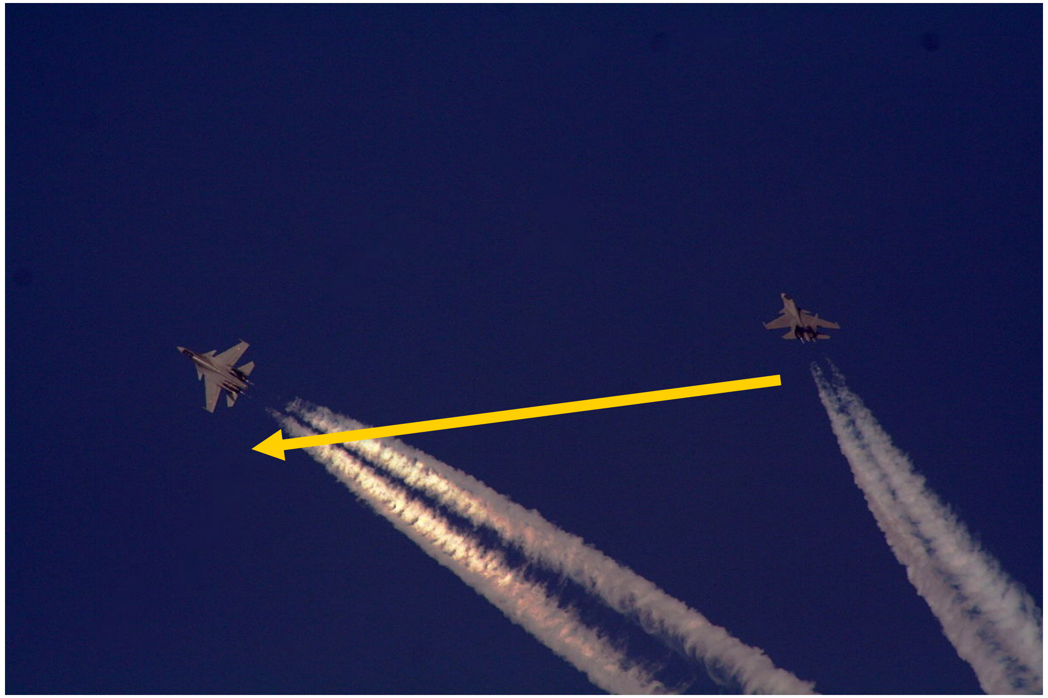
↓

From tracking to optimal control

↓

Linear Quadratic Regulator

Pure Pursuit Control



Aerial combat in which aircraft pursues another aircraft by pointing its nose directly towards it

Similar to carrot on a stick!

Rationale: Controller should leverage model!

$$\dot{x} = v\cos\theta$$

$$\dot{y} = v\sin\theta$$

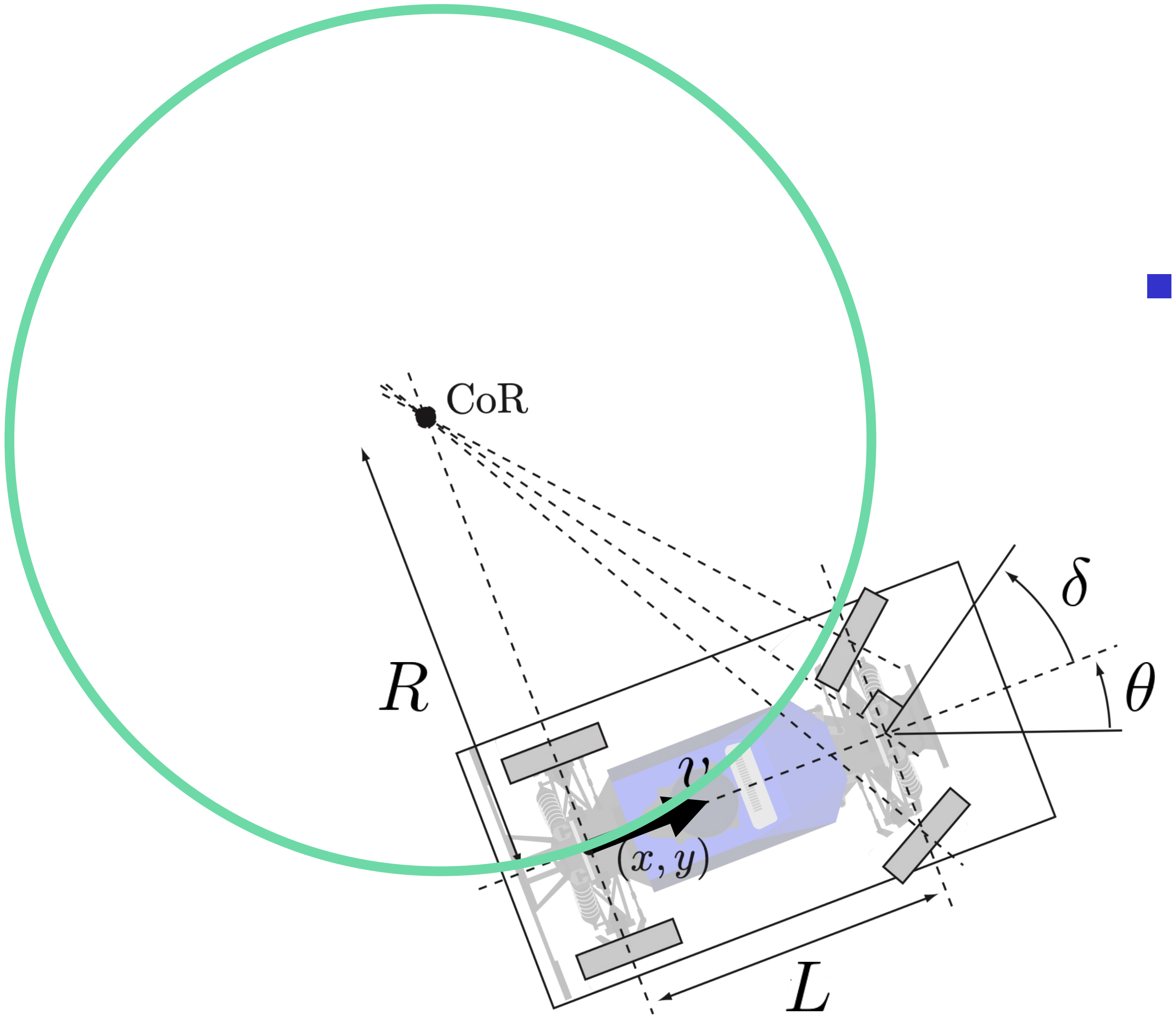
$$\dot{\theta} = \omega = \frac{v}{R} = \frac{v\tan\delta}{L}$$

PID control doesn't directly utilize the fact that we know the kinematic car model

Key Idea:

The car is always moving in a circular arc

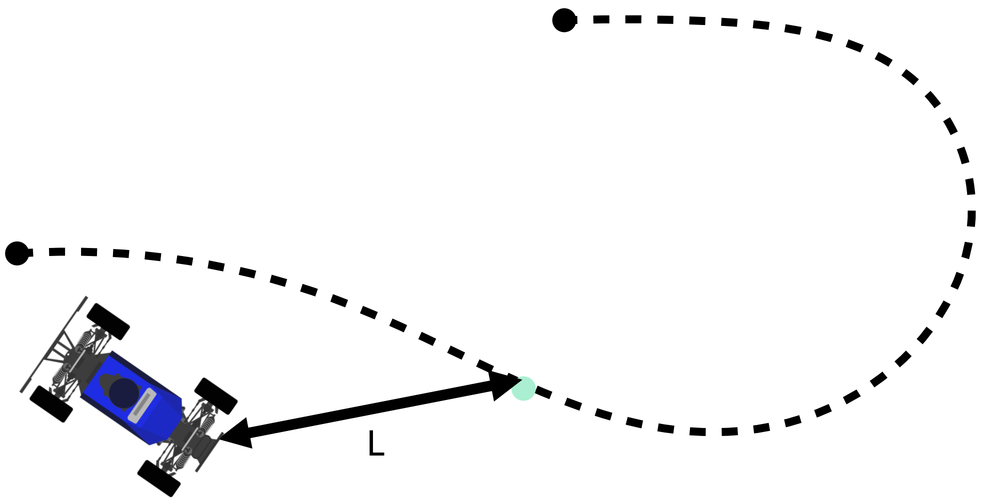
Pure Pursuit Controller



- Assume the car is moving with fixed steering angle

COULTER, 1992

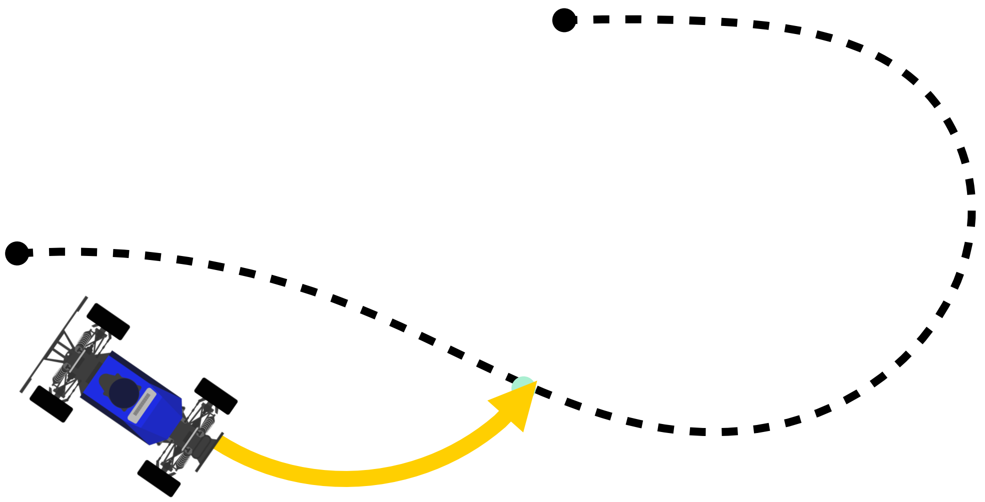
Consider a reference at a lookahead distance



$$\left\| \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_{ref} \\ y_{ref} \end{bmatrix} \right\| = L$$

Problem: Can we solve for a steering angle that guarantees that the car will pass through the reference?

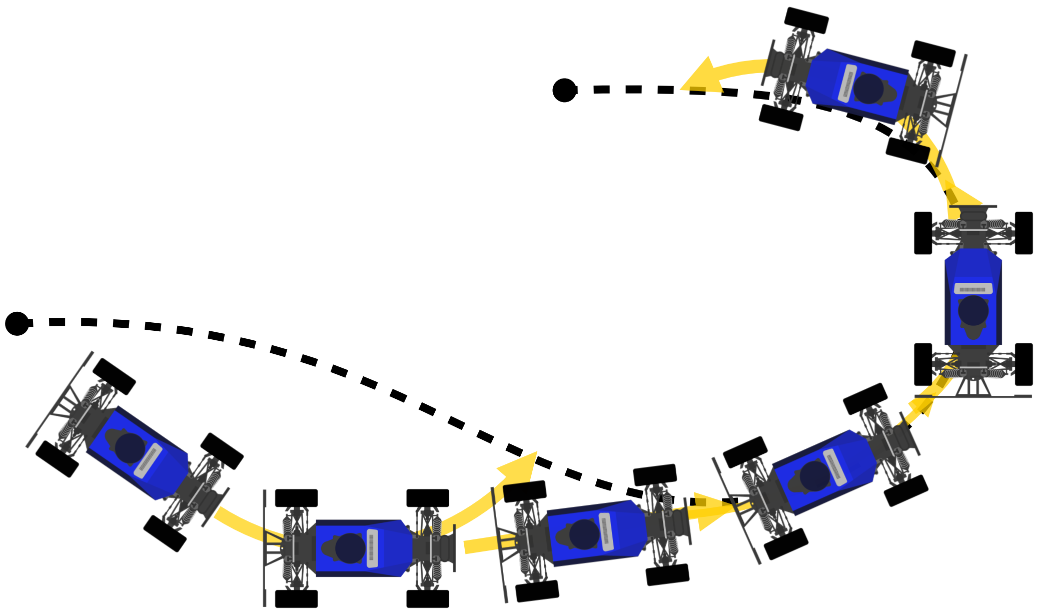
Solution: Compute a circular arc



We can always solve for a arc that passes through a lookahead point

Note: As the car moves forward, the point keeps moving

Pure pursuit: Keep chasing looakahead



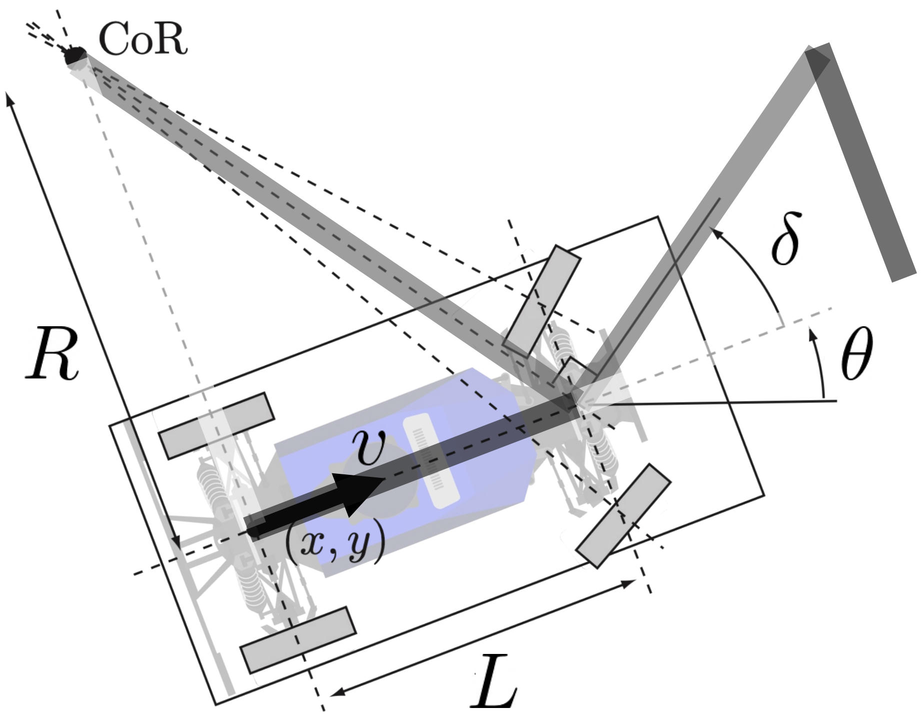
1. Find a lookahead and compute arc

2. Move along the arc

3. Go to step 1

Equations of Motion

RECALL



$$\dot{x} = v\cos\theta$$

$$\dot{y} = v\sin\theta$$

$$\dot{\theta} = \omega = \frac{v}{R} = \frac{v\tan\delta}{L}$$

$$\tan\delta = \frac{L}{R} \rightarrow R = \frac{L}{\tan\delta}$$

Kinematic Car Model

RECALL

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

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

X-COORDINATE

Y-COORDINATE

HEADING

$$\begin{bmatrix} v \\ \delta \end{bmatrix}$$

SPEED

STEERING ANGLE

$$\begin{bmatrix} 1 \\ u \end{bmatrix}$$

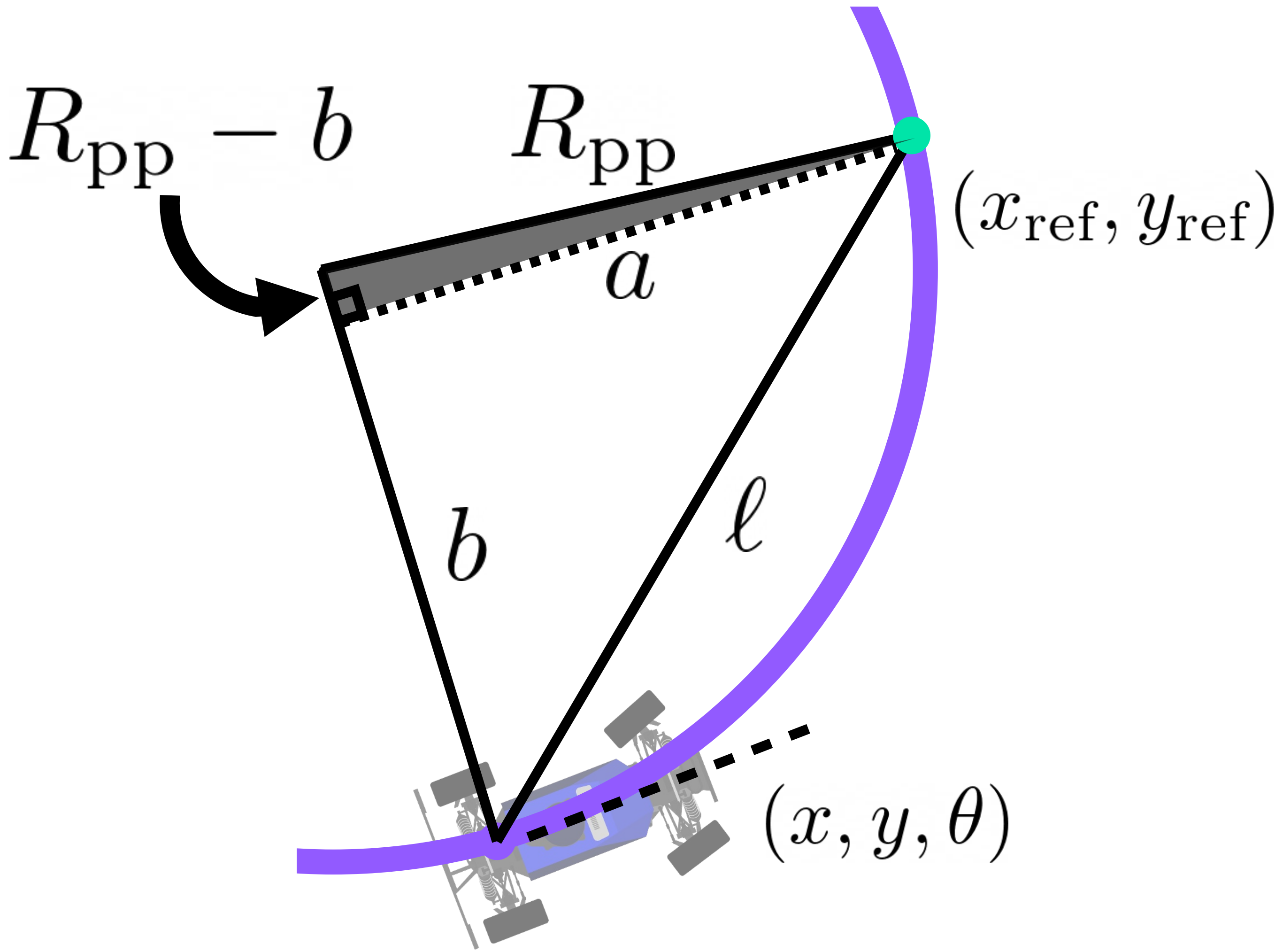
Pure pursuit: Control law derivation

Whiteboard

Computing the Arc Radius

$$(R_{\text{pp}} - b)^2 + a^2 = R_{\text{pp}}^2$$

$$R_{\text{pp}} = \frac{a^2 + b^2}{2b}$$

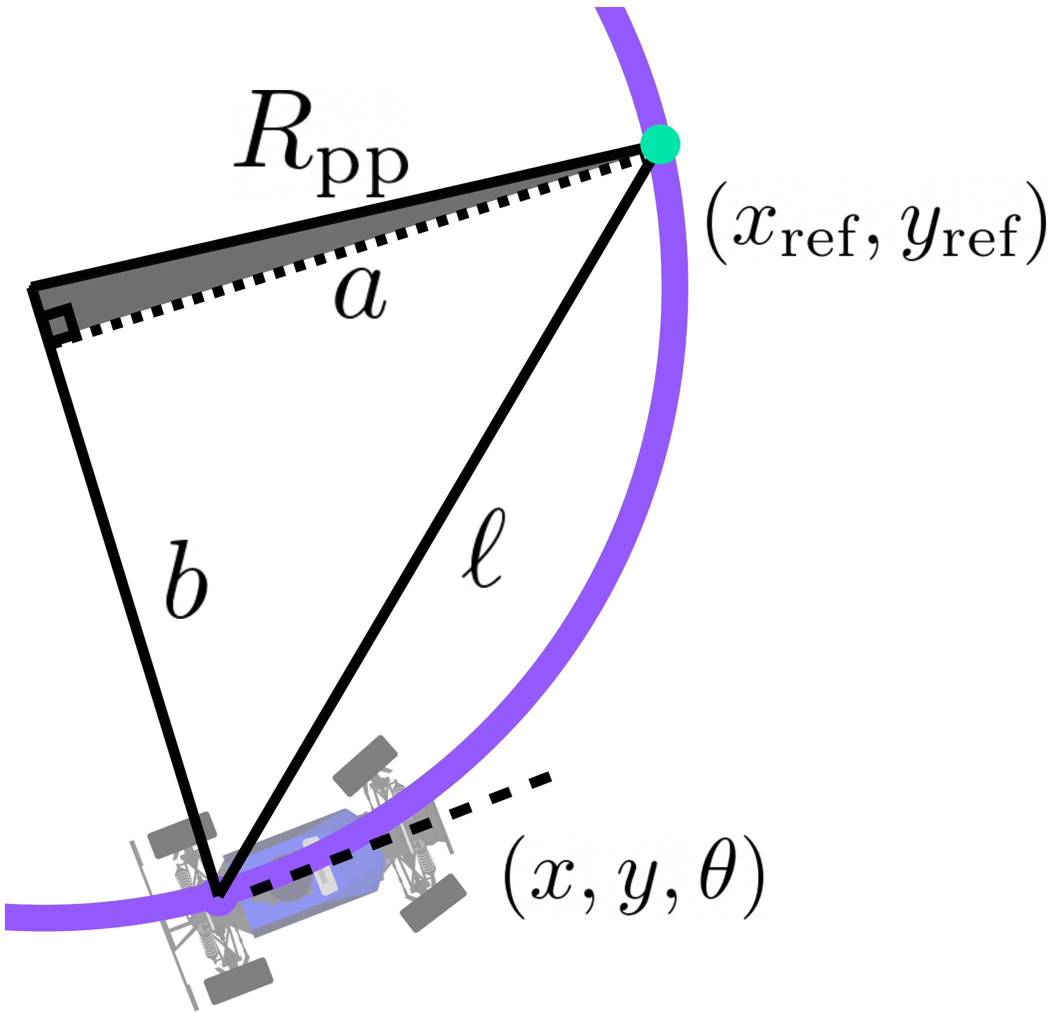


Computing the Arc Radius

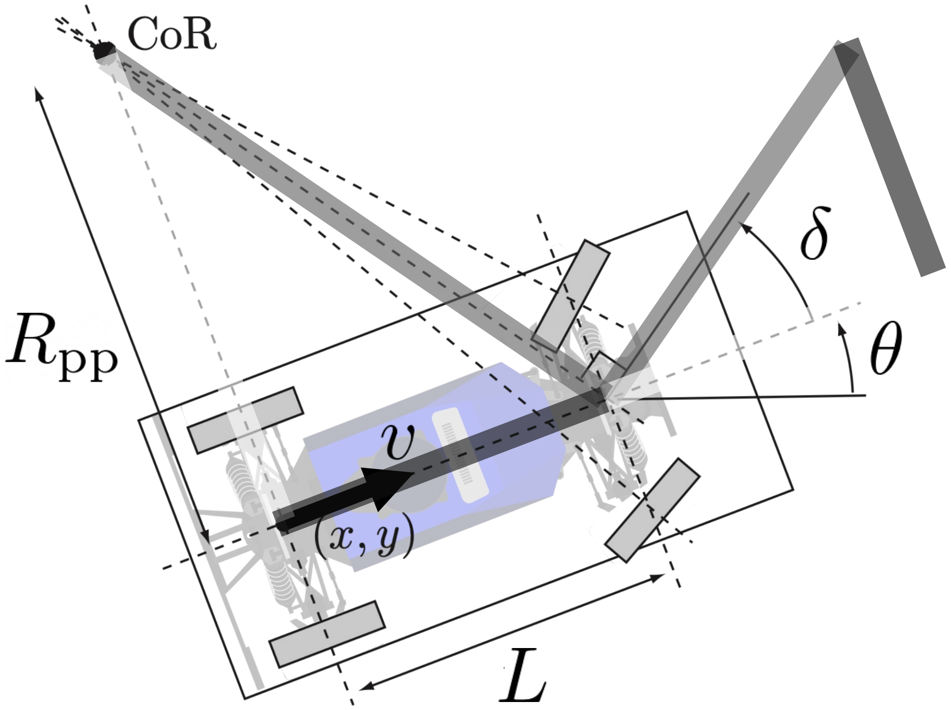
$$R_{\text{pp}} = \frac{a^2 + b^2}{2b}$$

$$\begin{bmatrix} a \\ b \end{bmatrix} = R(-\theta)\left(\begin{bmatrix} x_{\text{ref}} \\ y_{\text{ref}} \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix}\right)$$

Different than cross-track error (this is ref. position in robot frame;
vice versa for cross-track error)



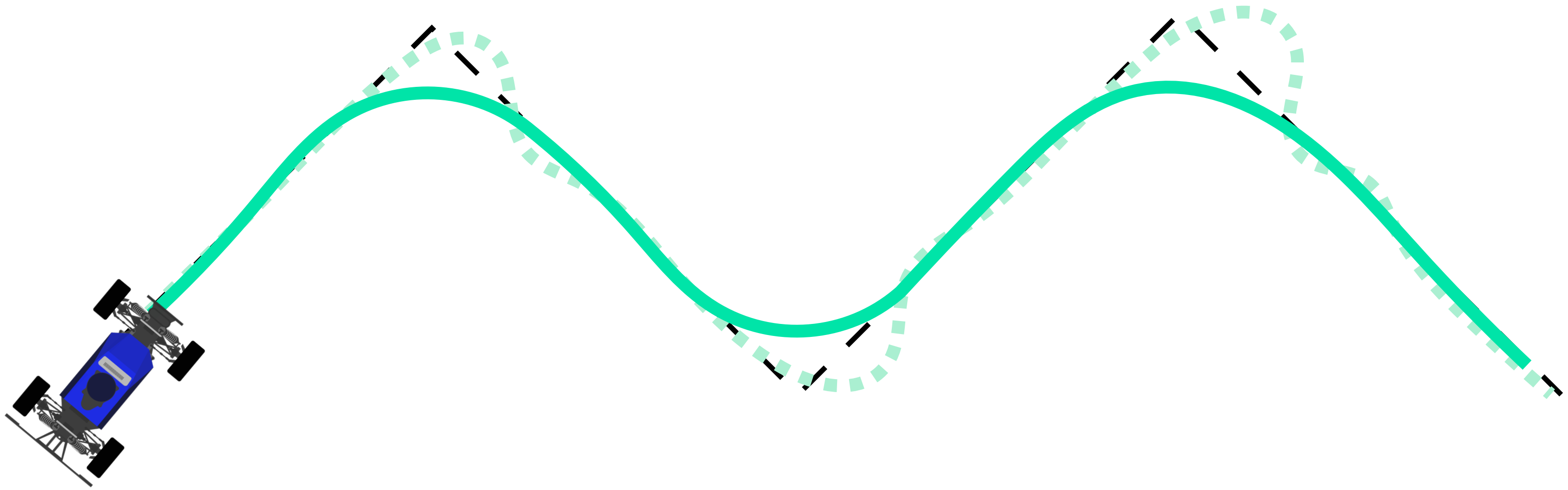
Computing the Steering Angle



$$R_{\mathrm{pp}} = \frac{a^2 + b^2}{2b}$$

$$\tan\delta = \frac{L}{R_{\mathrm{pp}}}$$

Question: How do I choose L?



Controller Design Decisions

1. Get a reference path/trajectory to track
2. Pick a reference state from the reference path/trajectory
3. Compute error to reference state
4. Compute control law to minimize error

Option 1:
Bang-bang control

Option 2:
PID control

Option 3:
Pure-pursuit control

Are we done?

Lecture Outline

Recap of Pure Pursuit

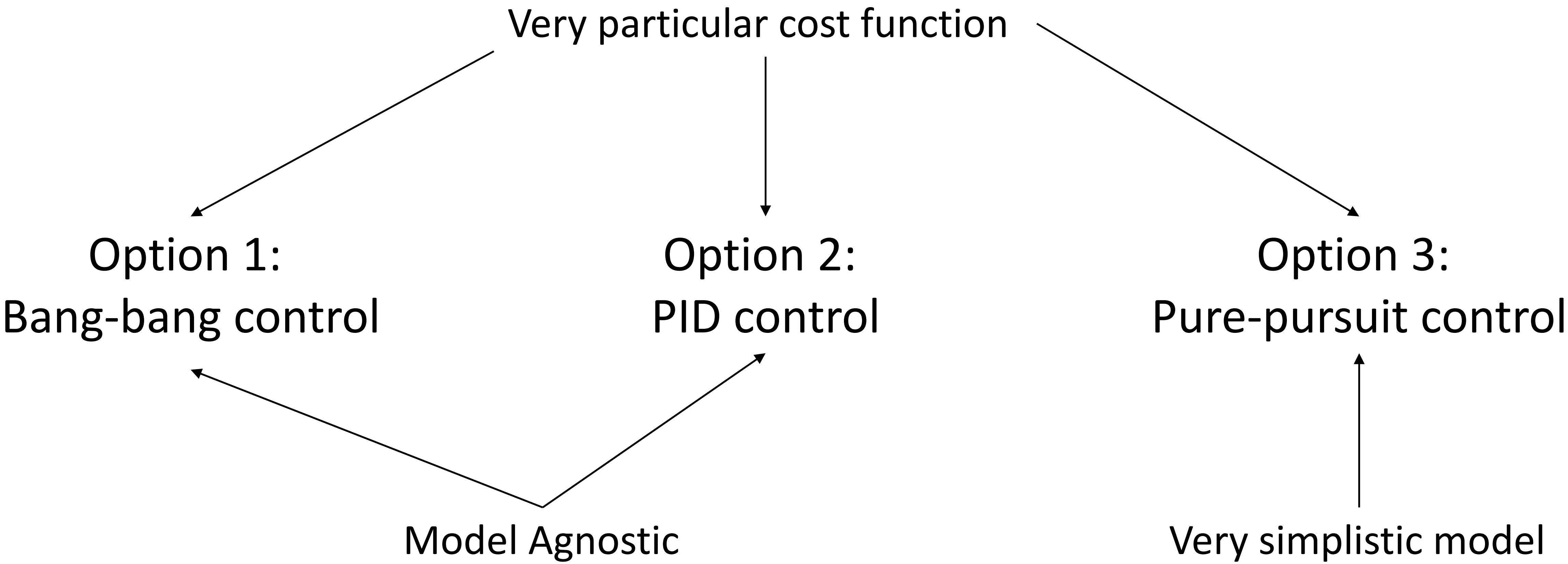
↓

From tracking to optimal control

↓

Linear Quadratic Regulator

Controller Design Decisions



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

$$\min_{u_{1:T}} \sum_{t=1}^{T} g(x_t, u_t) + G(x_T, u_T)$$

$$\text{s.t.} \quad x_{t+1} = Ax_t + Bu_t$$

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

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

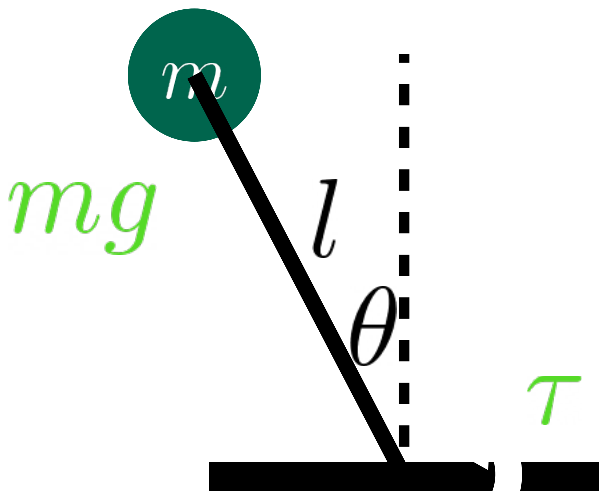
$$\sum_t x_t^\top Q x_t + u_t^\top R u_t$$

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

(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) $$x_{t+1} = Ax_t + Bu_t$$
- **Quadratic** cost function to minimize $$\sum_t x_t^\top Q x_t + u_t^\top R u_t$$

$$x_t^\top Q x_t$$

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

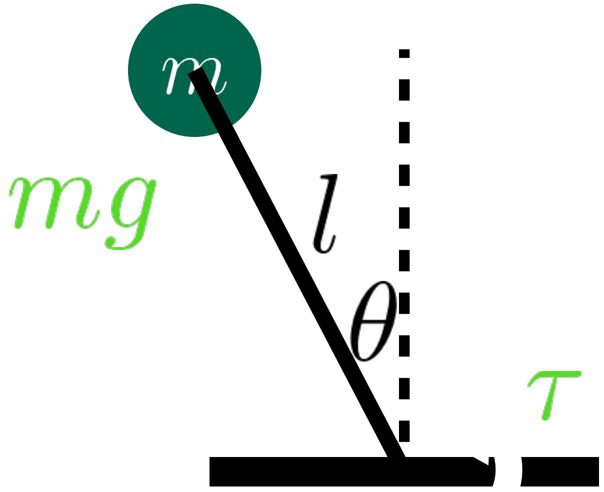
STATE COST

$$u_t^\top R u_t$$

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

CONTROL COST

Example: Inverted Pendulum (State Cost)



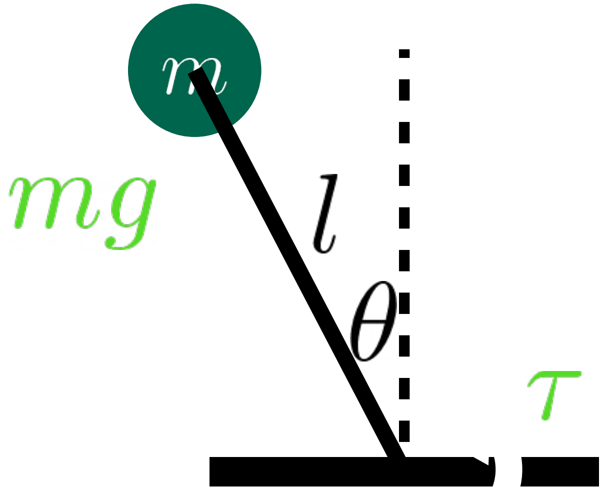
$$x_t^\top Q x_t$$ (QUADRATIC FORM)

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

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

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

Example: Inverted Pendulum (Control Cost)



$$u_t^\top R u_t$$

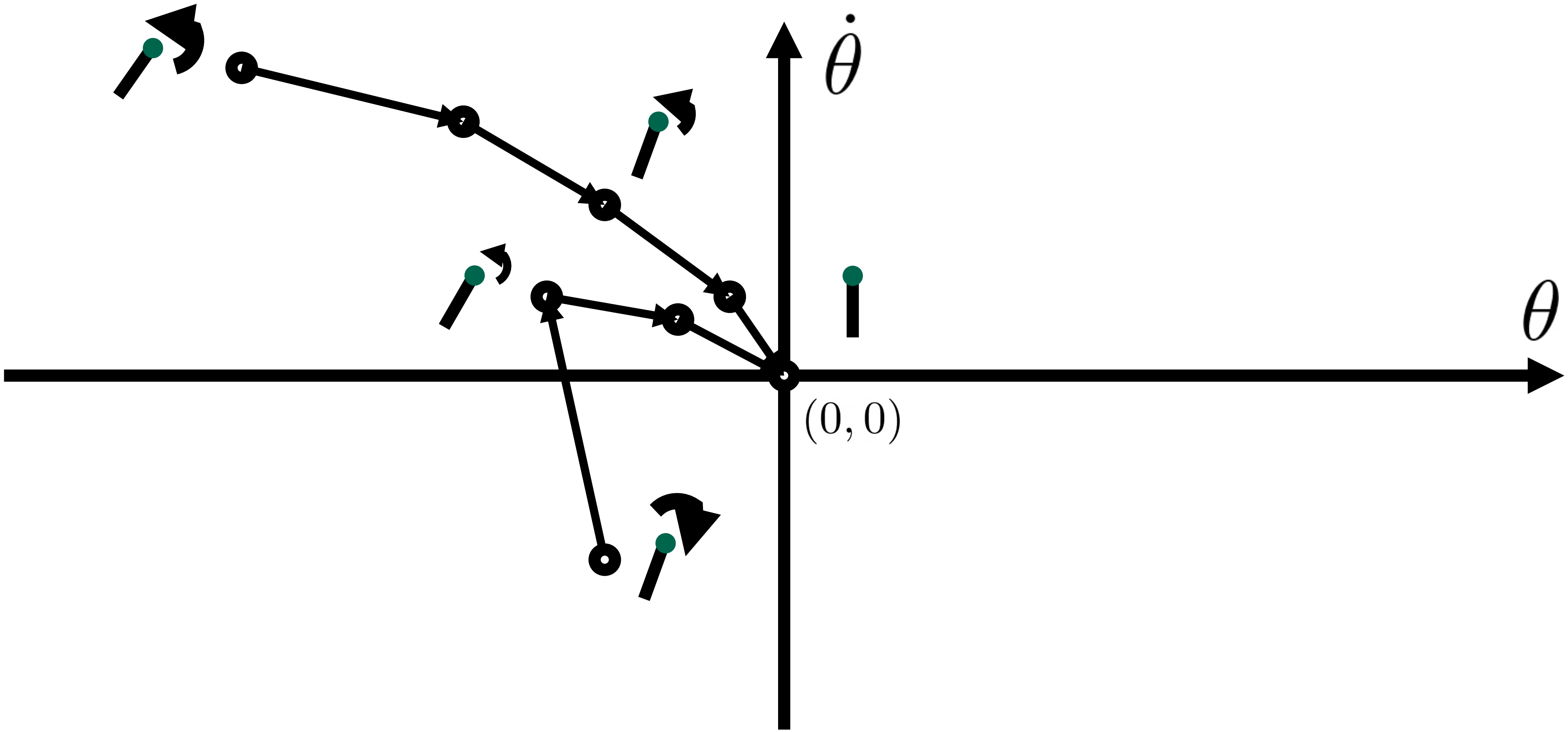
(QUADRATIC FORM)

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

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

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

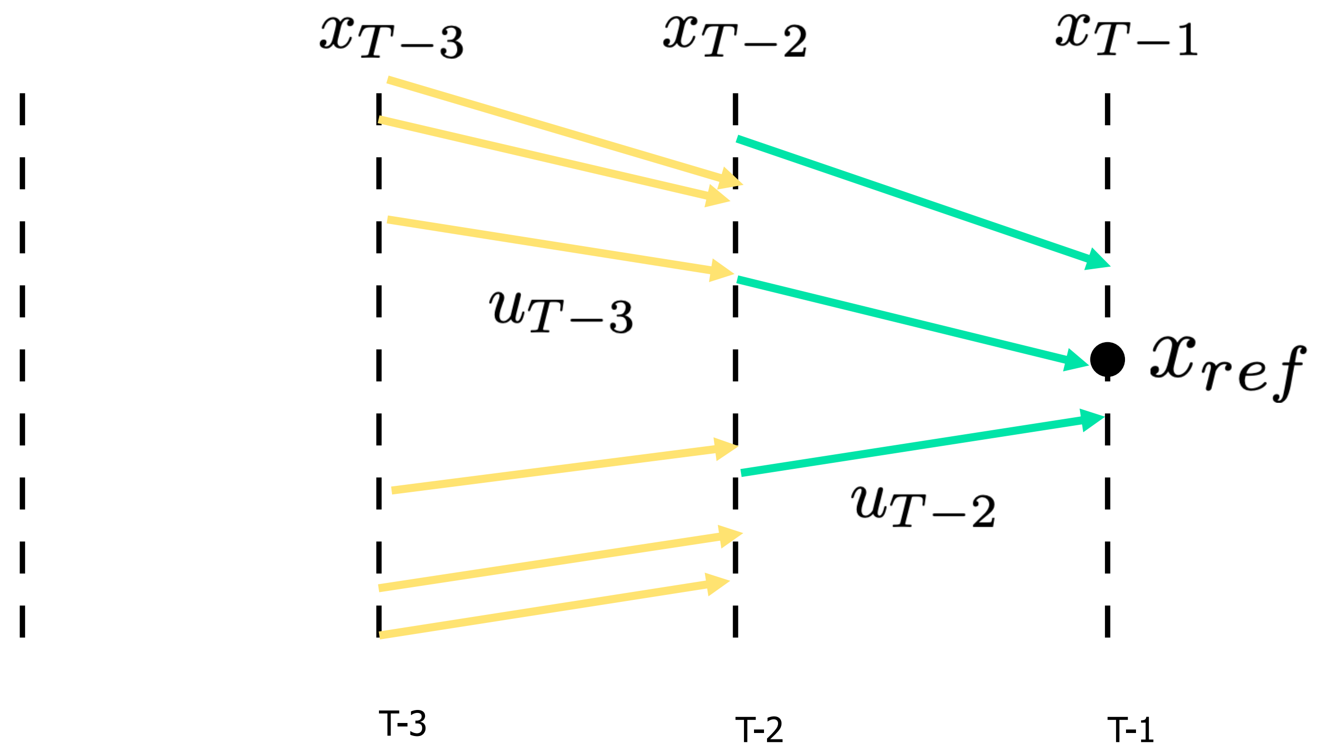
Example: Inverted Pendulum



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

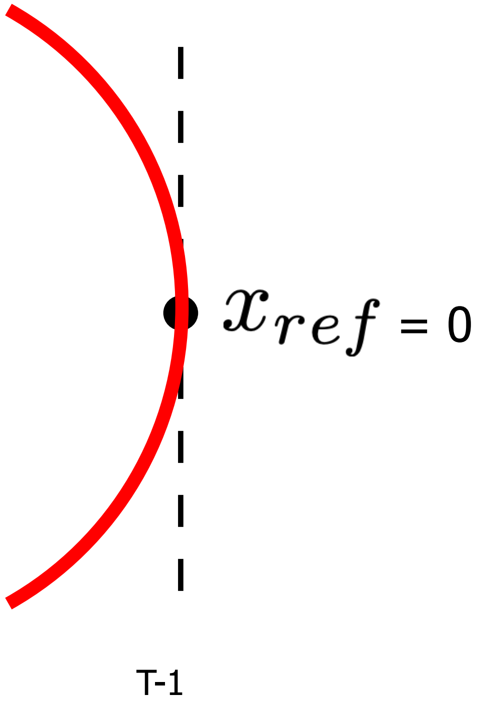
MINIMUM FUTURE COST, STARTING FROM x_{t+1}

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

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

Minimized with u = 0

$$P_0 = Q$$

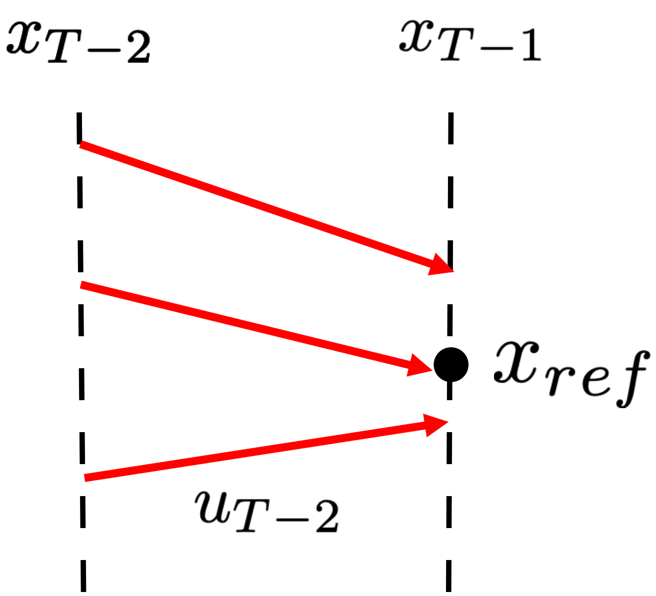


Note that the cost is quadratic in x

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 Q, 0

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

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

- Q → Current state cost
- $K_i^\top R K_i$ → Current action cost
- $(A + BK_i)^\top P_{i-1} (A + BK_i)$ → Optimal cost in the future based on dynamics

Linear Quadratic Regulator

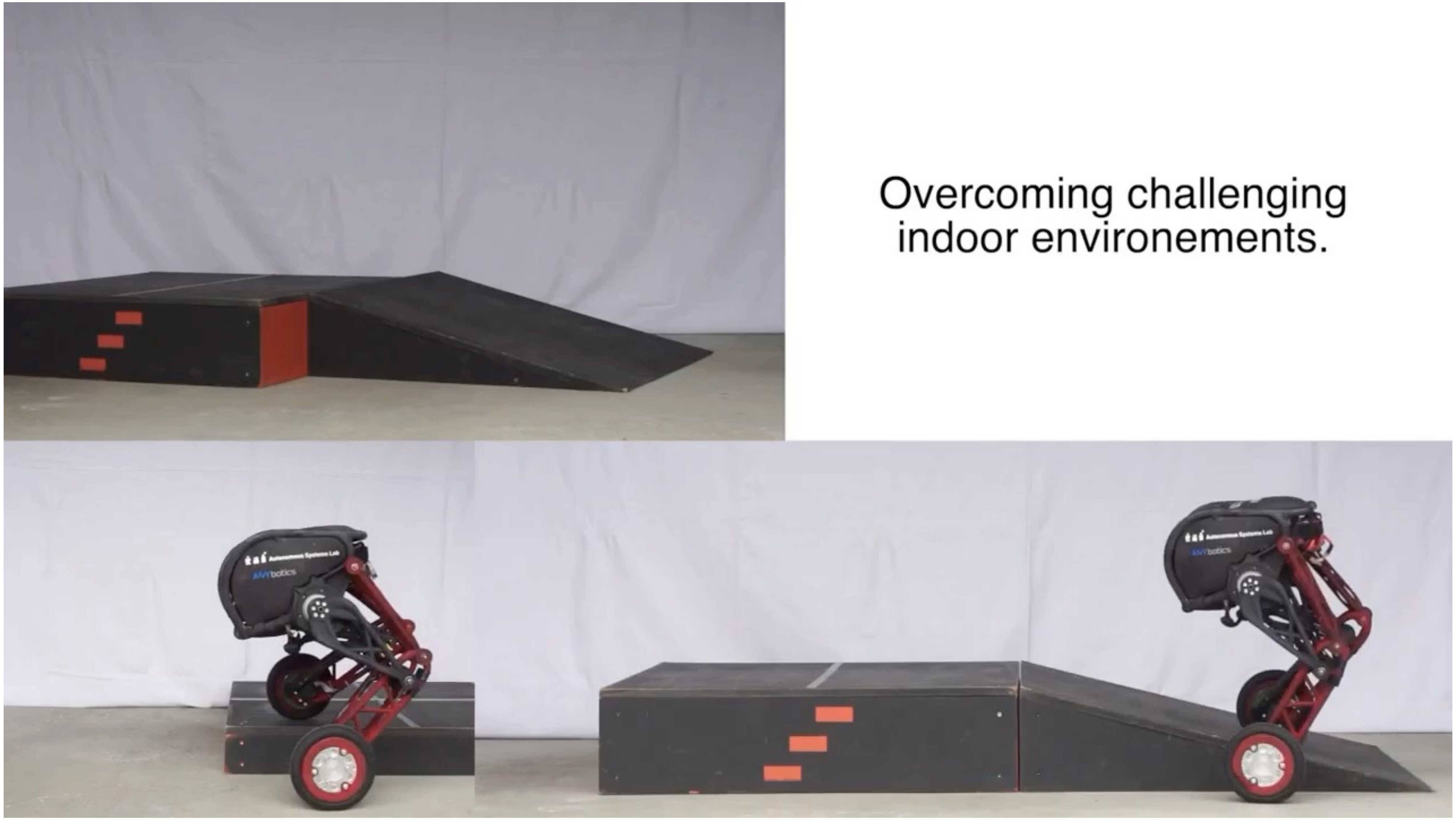
- For **linear** systems with **quadratic** costs, we can write down very efficient algorithms that return the optimal sequence of actions!
  - Special case where dynamic programming can be applied to continuous states and actions (typically only discrete states and actions)
- Many LQR extensions: non-linear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

Recap: Course Overview

Filtering/Smoothing
Localization
Mapping
SLAM

Search
Motion Planning
TrajOpt
Stability/Certification

MDPs and RL
Imitation Learning
Solving POMDPs

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