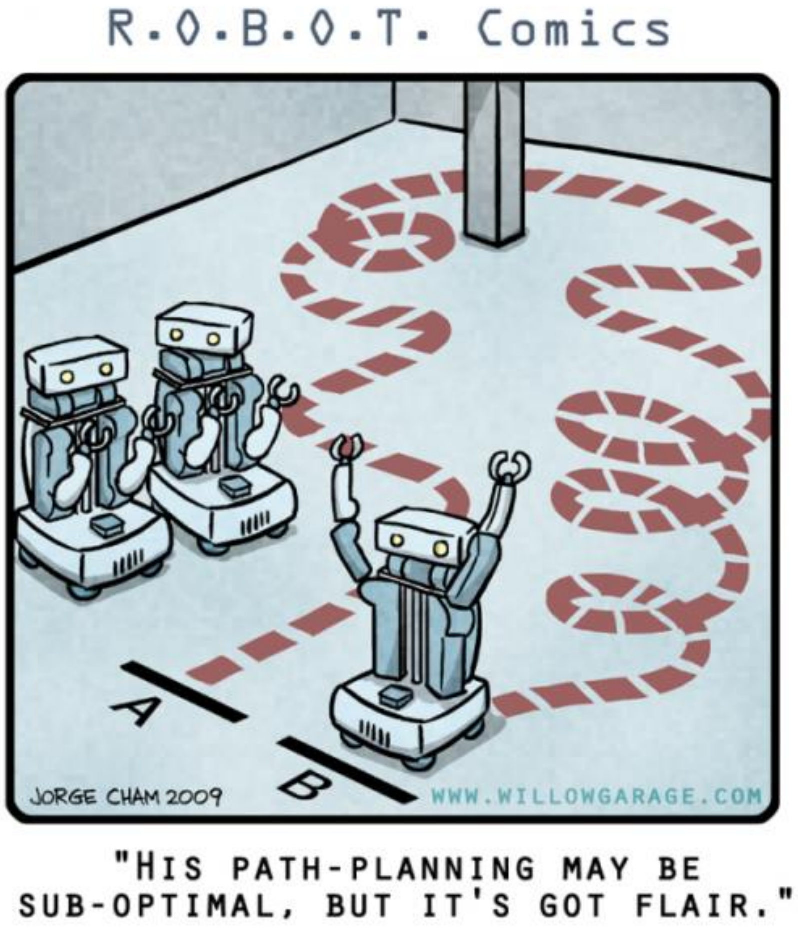
CSC2621
Imitation Learning for Robotics

Florian Shkurti

Week 2: Introduction to Optimal Control & Model-Based RL

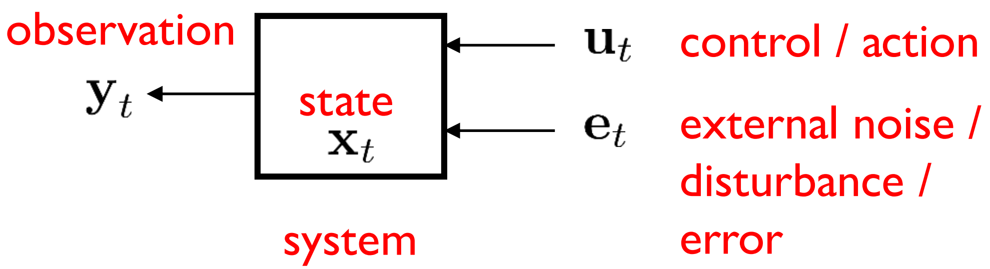
Today's agenda

- Intro to Control & Reinforcement Learning
- Linear Quadratic Regulator (LQR)
- Iterative LQR
- Model Predictive Control
- Learning the dynamics and model-based RL

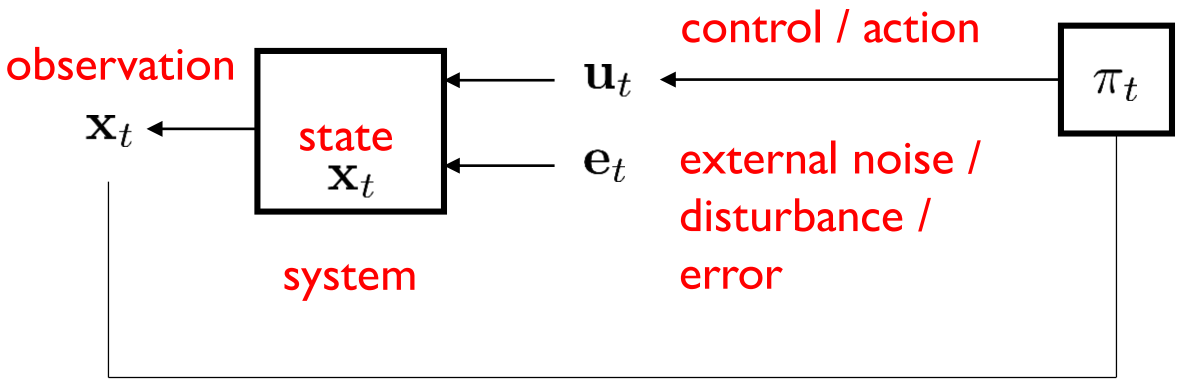


Acknowledgments
Today's slides have been influenced by: Pieter Abbeel (ECE287), Sergey Levine (DeepRL), Ben Recht (ICML'18), Emo Todorov, Zico Kolter

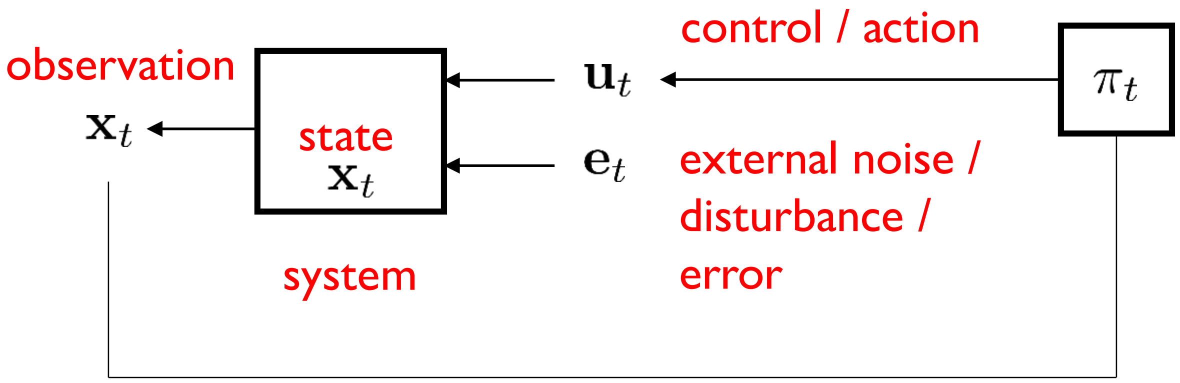
Optimal Control



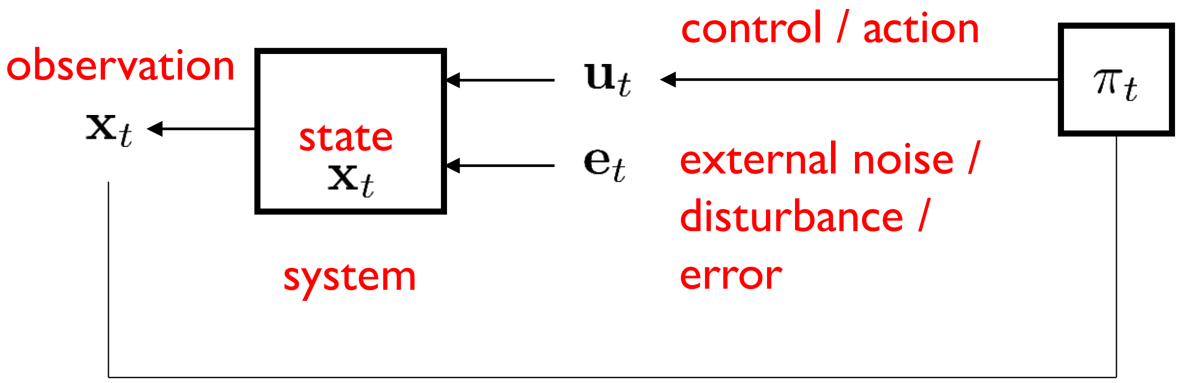
Optimal Control



Optimal Control



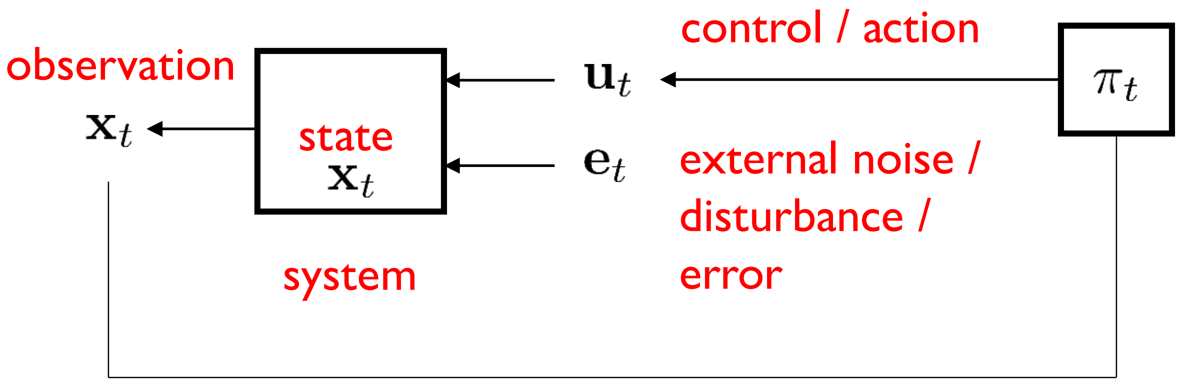
Optimal Control



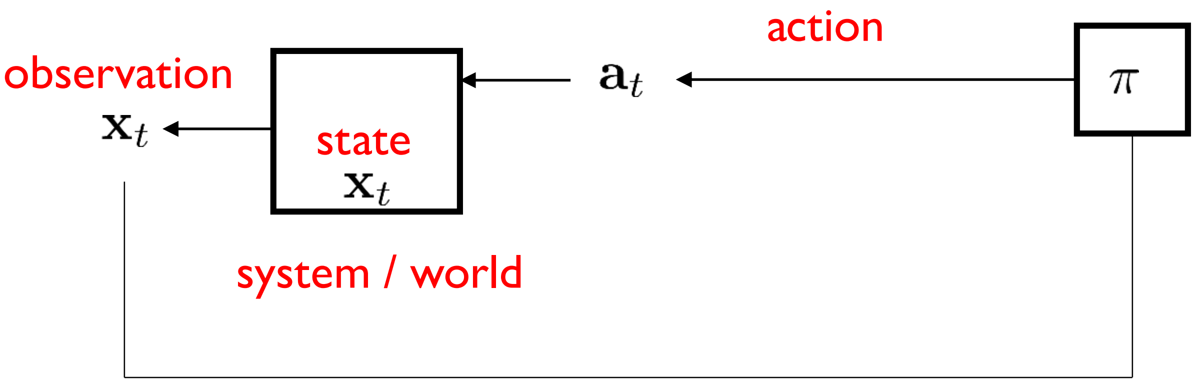
$$\begin{aligned}
\underset{\pi_0,\ldots,\pi_{T-1}}{\text{minimize}} \quad & \mathbb{E}_{\mathbf{e}_t}\left[\sum_{t=0}^{T} c(\mathbf{x}_t, \mathbf{u}_t)\right] \\
\text{subject to} \quad & \mathbf{x}_{t+1} = f_t(\mathbf{x}_t, \mathbf{u}_t, \mathbf{e}_t) \quad \text{known dynamics} \\
& \mathbf{u}_t = \pi_t(\mathbf{x}_{0:t}, \mathbf{u}_{0:t-1})
\end{aligned}$$

control law / policy

Optimal Control



Reinforcement Learning

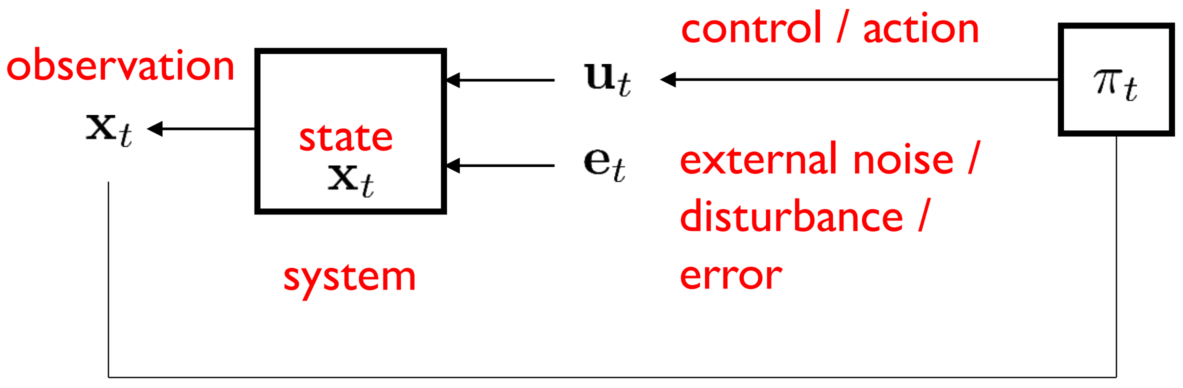


$$
\begin{aligned}
\underset{\pi_0,\ldots,\pi_{T-1}}{\text{minimize}} \quad & \mathbb{E}_{\mathbf{e}_t}\left[\sum_{t=0}^{T} c(\mathbf{x}_t, \mathbf{u}_t)\right] \\
\text{subject to} \quad & \mathbf{x}_{t+1} = f_t(\mathbf{x}_t, \mathbf{u}_t, \mathbf{e}_t) \\
& \mathbf{u}_t = \pi_t(\mathbf{x}_{0:t}, \mathbf{u}_{0:t-1})
\end{aligned}
$$

known dynamics

control law / policy

Optimal Control



$$\underset{\pi_0,\ldots,\pi_{T-1}}{\text{minimize}} \quad \mathbb{E}_{\mathbf{e}_t}\left[\sum_{t=0}^{T} c(\mathbf{x}_t, \mathbf{u}_t)\right]$$

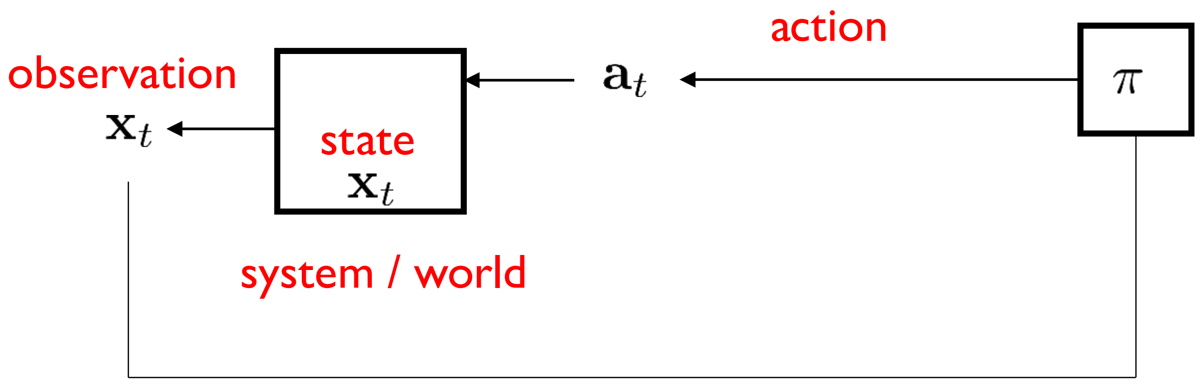
$$\text{subject to} \quad \mathbf{x}_{t+1} = f_t(\mathbf{x}_t, \mathbf{u}_t, \mathbf{e}_t)$$

known dynamics

$$\mathbf{u}_t = \pi_t(\mathbf{x}_{0:t}, \mathbf{u}_{0:t-1})$$

control law / policy

Reinforcement Learning



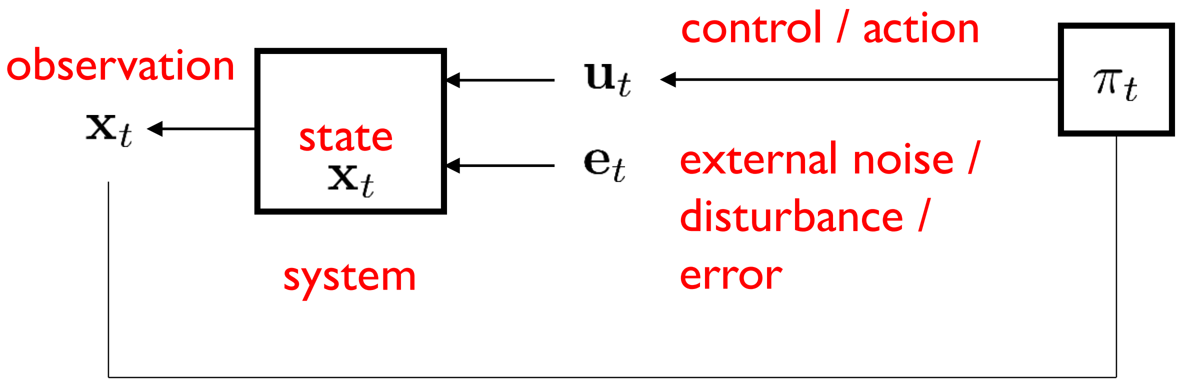
$$\underset{\theta}{\text{maximize}} \quad \mathbb{E}_{\tau \sim p_\theta(\tau)}\left[\sum_{t=0}^{T} r(\mathbf{x}_t, \mathbf{a}_t)\right]$$

$$\begin{aligned} p_\theta(\tau) &= p_\theta(\mathbf{x}_{0:T}, \mathbf{a}_{0:T-1}) \\ &= p(\mathbf{x}_0) \prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t \mid \mathbf{x}_t)\, p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t) \end{aligned}$$

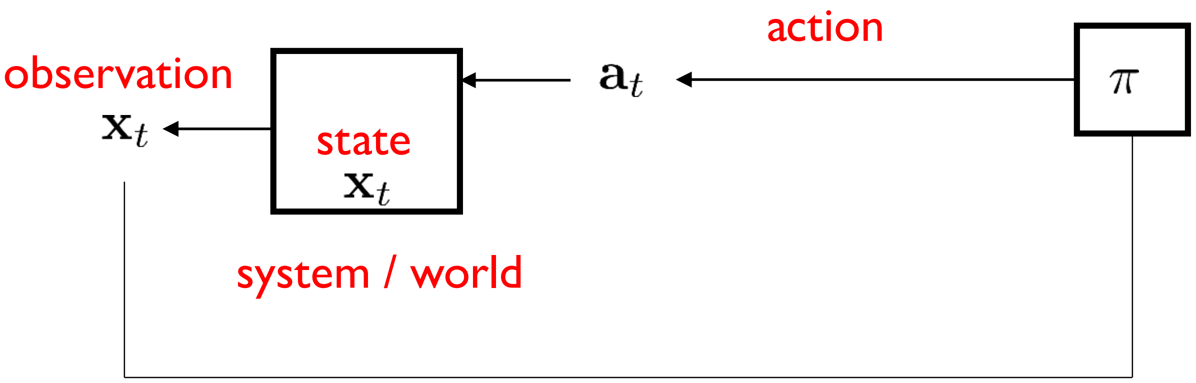
policy dynamics

cost = - reward

Optimal Control



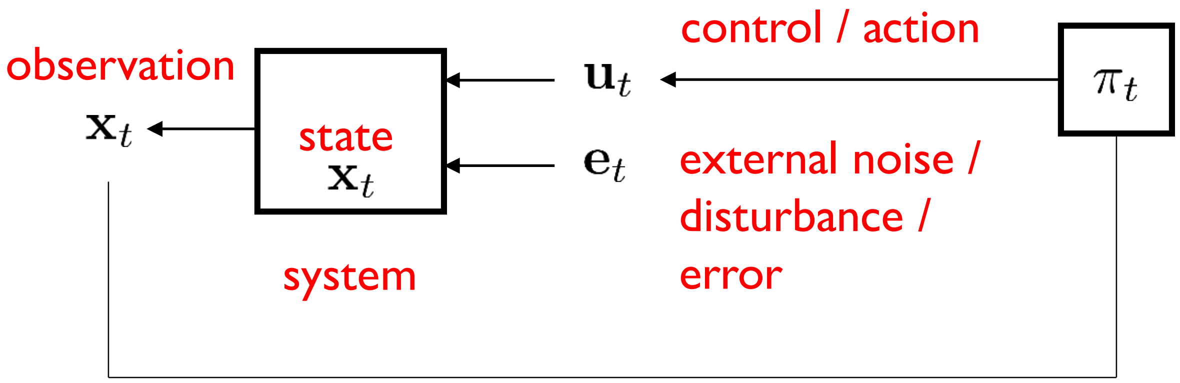
Reinforcement Learning



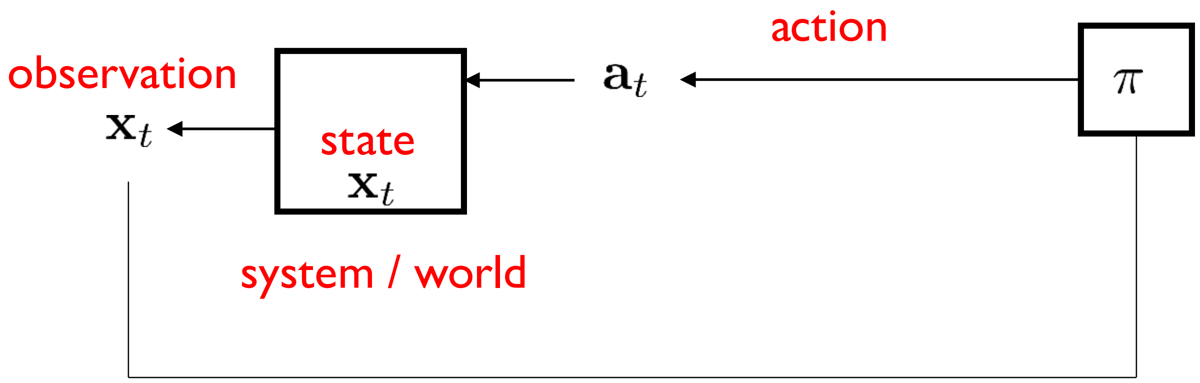
$$J(\mathbf{x}_t) = \min_{\mathbf{u}_t} \left[c(\mathbf{x}_t, \mathbf{u}_t) + \mathbb{E}_{\mathbf{e}_t} \left[J(f_t(\mathbf{x}_t, \mathbf{u}_t, \mathbf{e}_t)) \right] \right]$$

Optimal cost-to-go:
“if you land at state x and you follow the optimal actions
what is the expected cost you will pay?

Optimal Control



Reinforcement Learning



For finite time horizon

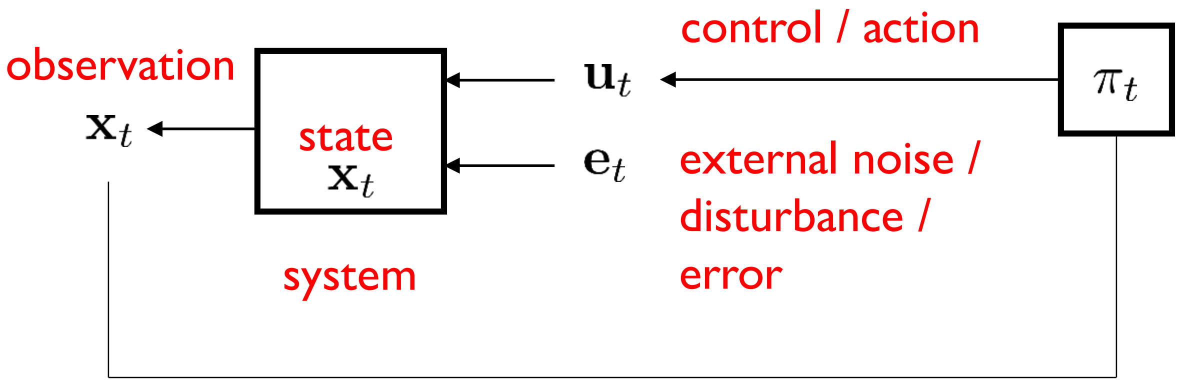
$$J(\mathbf{x}_t) = \min_{\mathbf{u}_t} \left[c(\mathbf{x}_t, \mathbf{u}_t) + \mathbb{E}_{\mathbf{e}_t} \left[J(f_t(\mathbf{x}_t, \mathbf{u}_t, \mathbf{e}_t)) \right] \right]$$

Optimal cost-to-go

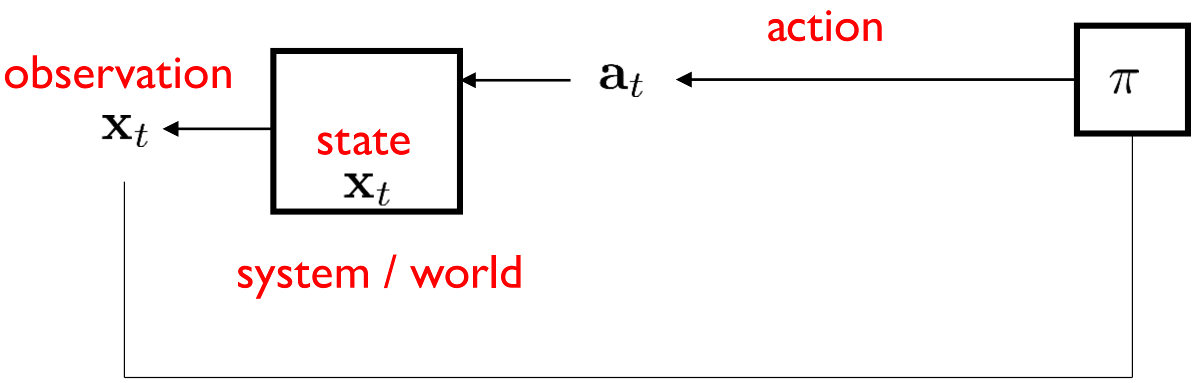
$$V^*(\mathbf{x}_t) = \max_{\mathbf{a}_t} \left[r(\mathbf{x}_t, \mathbf{a}_t) + \mathbb{E}_{\mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1} | \mathbf{x}_t, \mathbf{a}_t)} \left[V^*(\mathbf{x}_{t+1}) \right] \right]$$

Optimal value function:
"if you land at state x and you follow the optimal policy
what is the expected reward you will accumulate?"

Optimal Control



Reinforcement Learning



For finite time horizon

$$J(\mathbf{x}_t) = \min_{\mathbf{u}_t} \left[c(\mathbf{x}_t, \mathbf{u}_t) + \mathbb{E}_{\mathbf{e}_t}\left[J(f_t(\mathbf{x}_t, \mathbf{u}_t, \mathbf{e}_t))\right]\right]$$

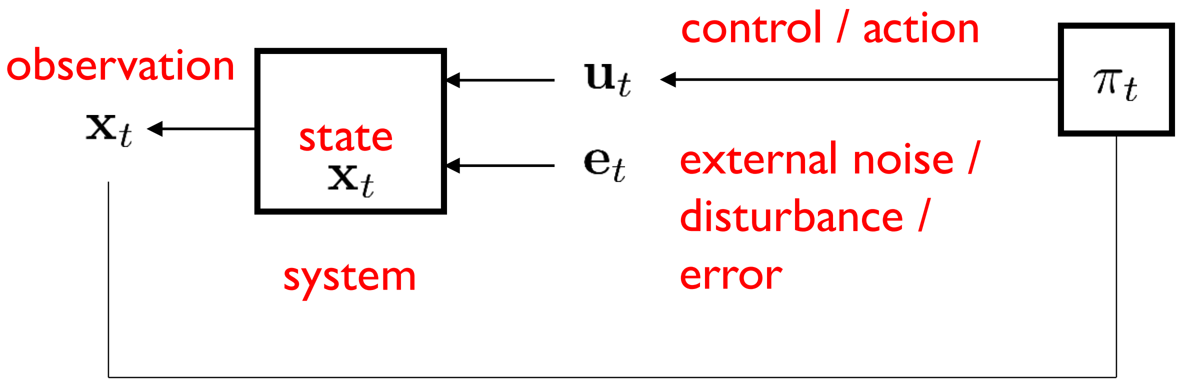
Optimal cost-to-go

$$V^*(\mathbf{x}_t) = \max_{\mathbf{a}_t} \underbrace{\left[r(\mathbf{x}_t, \mathbf{a}_t) + \mathbb{E}_{\mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1} | \mathbf{x}_t, \mathbf{a}_t)}\left[V^*(\mathbf{x}_{t+1})\right]\right]}_{Q^*(\mathbf{x}_t, \mathbf{a}_t)}$$

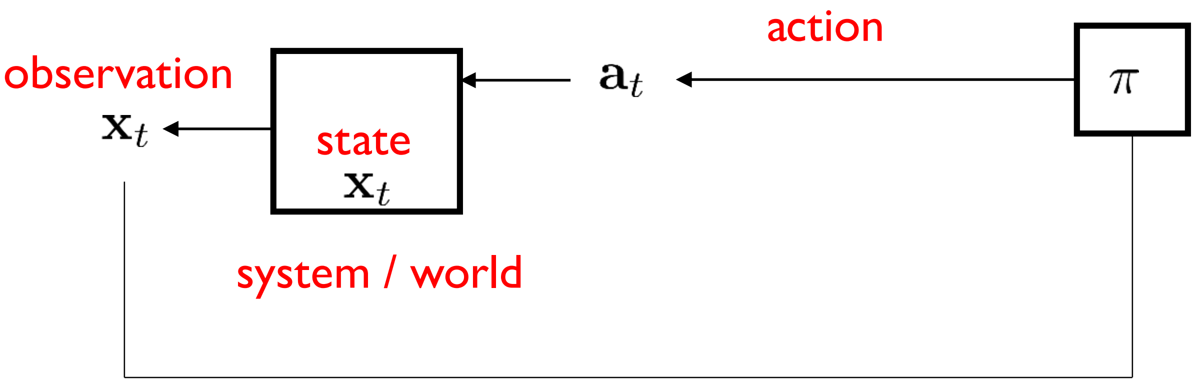
Optimal value function

Optimal state-action value function:
"if you land at state x, and you commit to first execute action a, and then follow the optimal policy how much reward will you accumulate?"

Optimal Control



Reinforcement Learning



For finite time horizon

$$J(\mathbf{x}_t) = \min_{\mathbf{u}_t} \left[c(\mathbf{x}_t, \mathbf{u}_t) + \mathbb{E}_{\mathbf{e}_t} \left[J(f_t(\mathbf{x}_t, \mathbf{u}_t, \mathbf{e}_t)) \right] \right]$$

Optimal cost-to-go

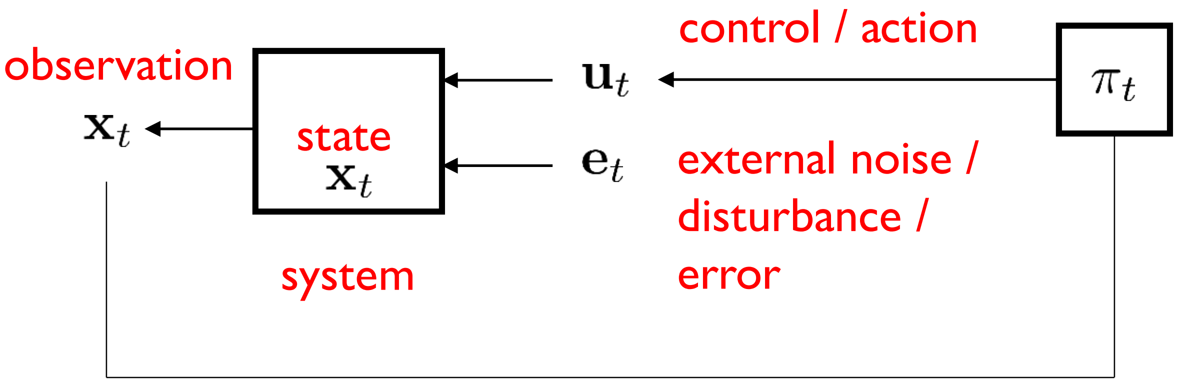
$$V^*(\mathbf{x}_t) = \max_{\mathbf{a}_t} \left[r(\mathbf{x}_t, \mathbf{a}_t) + \mathbb{E}_{\mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1} | \mathbf{x}_t, \mathbf{a}_t)} \left[V^*(\mathbf{x}_{t+1}) \right] \right]$$

Optimal value function

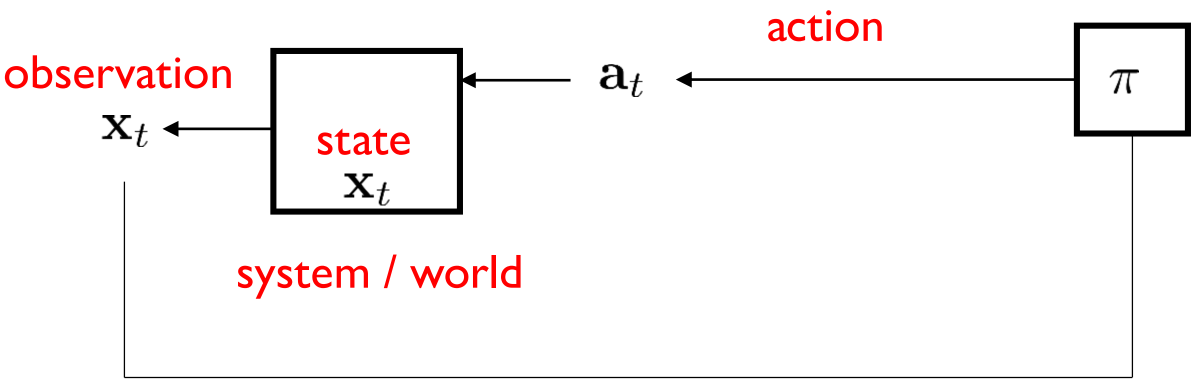
Value function of policy pi:
“if you land at state x and you follow policy pi
what is the expected reward you will accumulate?”

$$V^\pi(\mathbf{x}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi(\mathbf{a} | \mathbf{x}_t)} \left[r(\mathbf{x}_t, \mathbf{a}_t) + \mathbb{E}_{\mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1} | \mathbf{x}_t, \mathbf{a}_t)} \left[V^\pi(\mathbf{x}_{t+1}) \right] \right]$$

Optimal Control



Reinforcement Learning



For finite time horizon

$$J(\mathbf{x}_t) = \min_{\mathbf{u}_t} \left[c(\mathbf{x}_t, \mathbf{u}_t) + \mathbb{E}_{\mathbf{e}_t} \left[J(f_t(\mathbf{x}_t, \mathbf{u}_t, \mathbf{e}_t)) \right] \right]$$

Optimal cost-to-go

$$V^*(\mathbf{x}_t) = \max_{\mathbf{a}_t} \left[r(\mathbf{x}_t, \mathbf{a}_t) + \mathbb{E}_{\mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1} | \mathbf{x}_t, \mathbf{a}_t)} \left[V^*(\mathbf{x}_{t+1}) \right] \right]$$

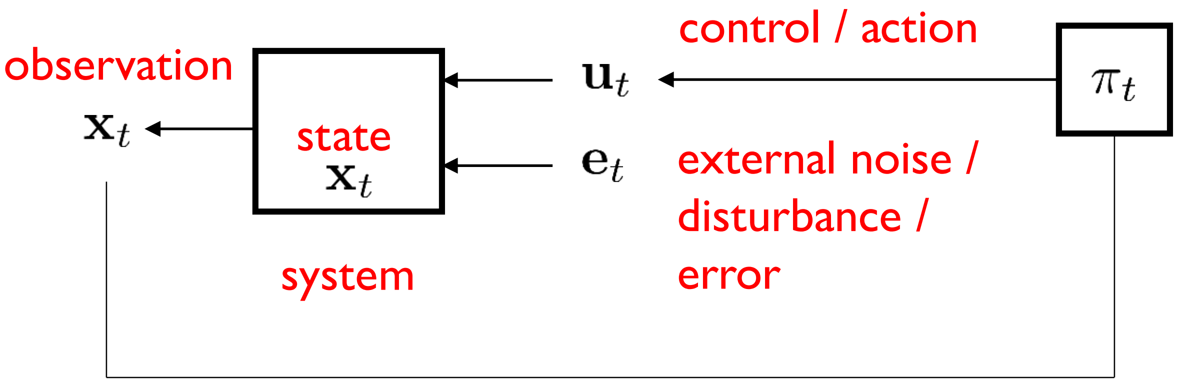
Optimal value function

State-action value function of policy pi: "if you land at state x, and you commit to first execute action a, and then follow policy pi how much reward will you accumulate?"

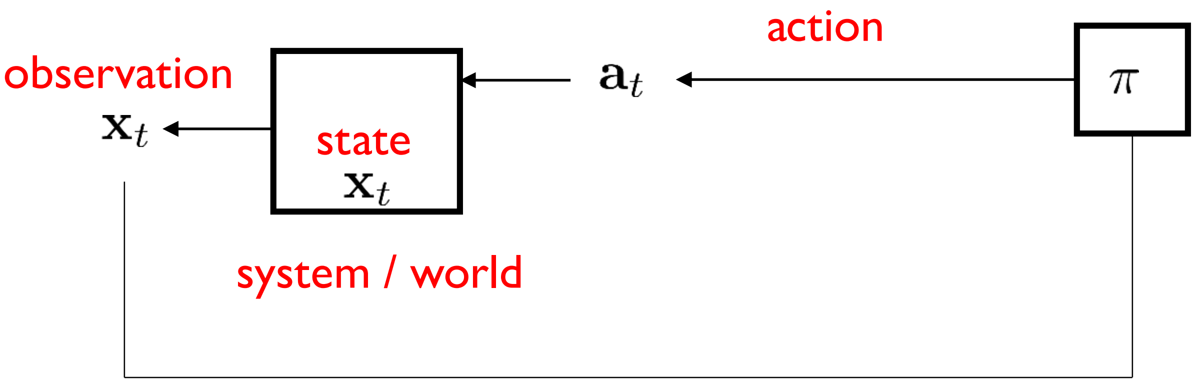
Value function of policy pi

$$V^\pi(\mathbf{x}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi(\mathbf{a} | \mathbf{x}_t)} \left[\overbrace{r(\mathbf{x}_t, \mathbf{a}_t) + \mathbb{E}_{\mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1} | \mathbf{x}_t, \mathbf{a}_t)} \left[V^\pi(\mathbf{x}_{t+1}) \right]}^{Q^\pi(\mathbf{x}_t, \mathbf{a}_t)} \right]$$

Optimal Control



Reinforcement Learning



For finite time horizon

$$J(\mathbf{x}_t) = \min_{\mathbf{u}_t} \left[c(\mathbf{x}_t, \mathbf{u}_t) + \mathbb{E}_{\mathbf{e}_t}\left[J(f_t(\mathbf{x}_t, \mathbf{u}_t, \mathbf{e}_t))\right]\right]$$

Optimal cost-to-go

$$V^*(\mathbf{x}_t) = \max_{\mathbf{a}_t} \underbrace{\left[r(\mathbf{x}_t, \mathbf{a}_t) + \mathbb{E}_{\mathbf{x}_{t+1}\sim p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{a}_t)}\left[V^*(\mathbf{x}_{t+1})\right]\right]}_{Q^*(\mathbf{x}_t, \mathbf{a}_t)}$$

Optimal value function

Optimal state-action value function $Q^*(\mathbf{x}_t, \mathbf{a}_t)$

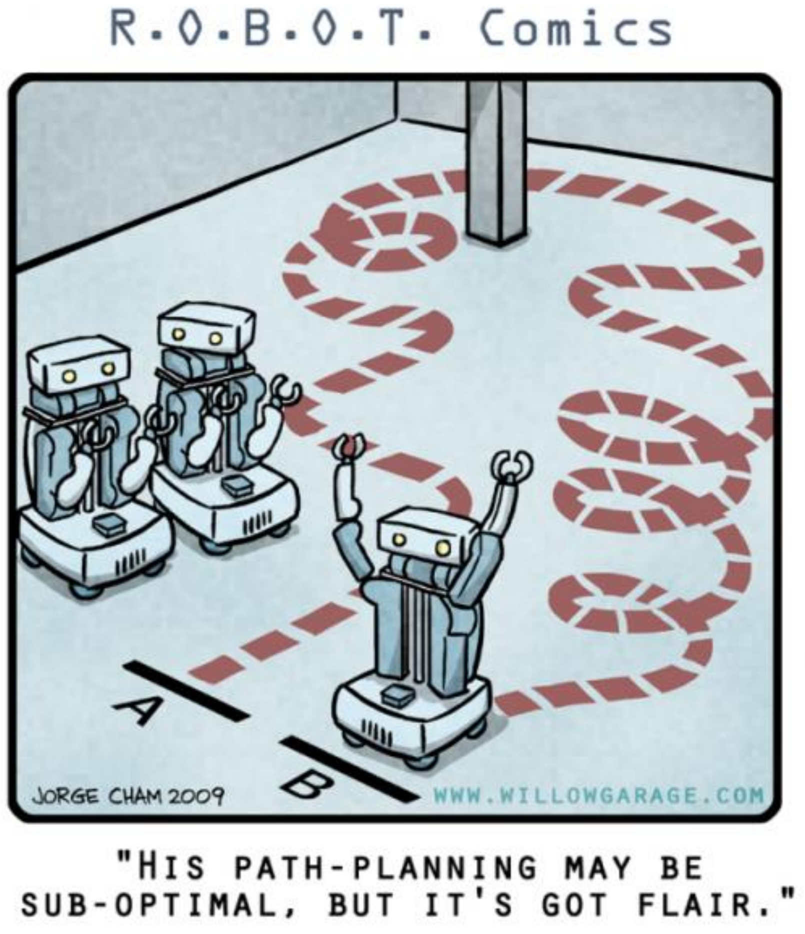
Value function of policy pi

State-action value function of policy pi $Q^\pi(\mathbf{x}_t, \mathbf{a}_t)$

$$V^\pi(\mathbf{x}_t) = \mathbb{E}_{\mathbf{a}_t\sim\pi(\mathbf{a}|\mathbf{x}_t)} \overbrace{\left[r(\mathbf{x}_t, \mathbf{a}_t) + \mathbb{E}_{\mathbf{x}_{t+1}\sim p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{a}_t)}\left[V^\pi(\mathbf{x}_{t+1})\right]\right]}^{Q^\pi(\mathbf{x}_t, \mathbf{a}_t)}$$

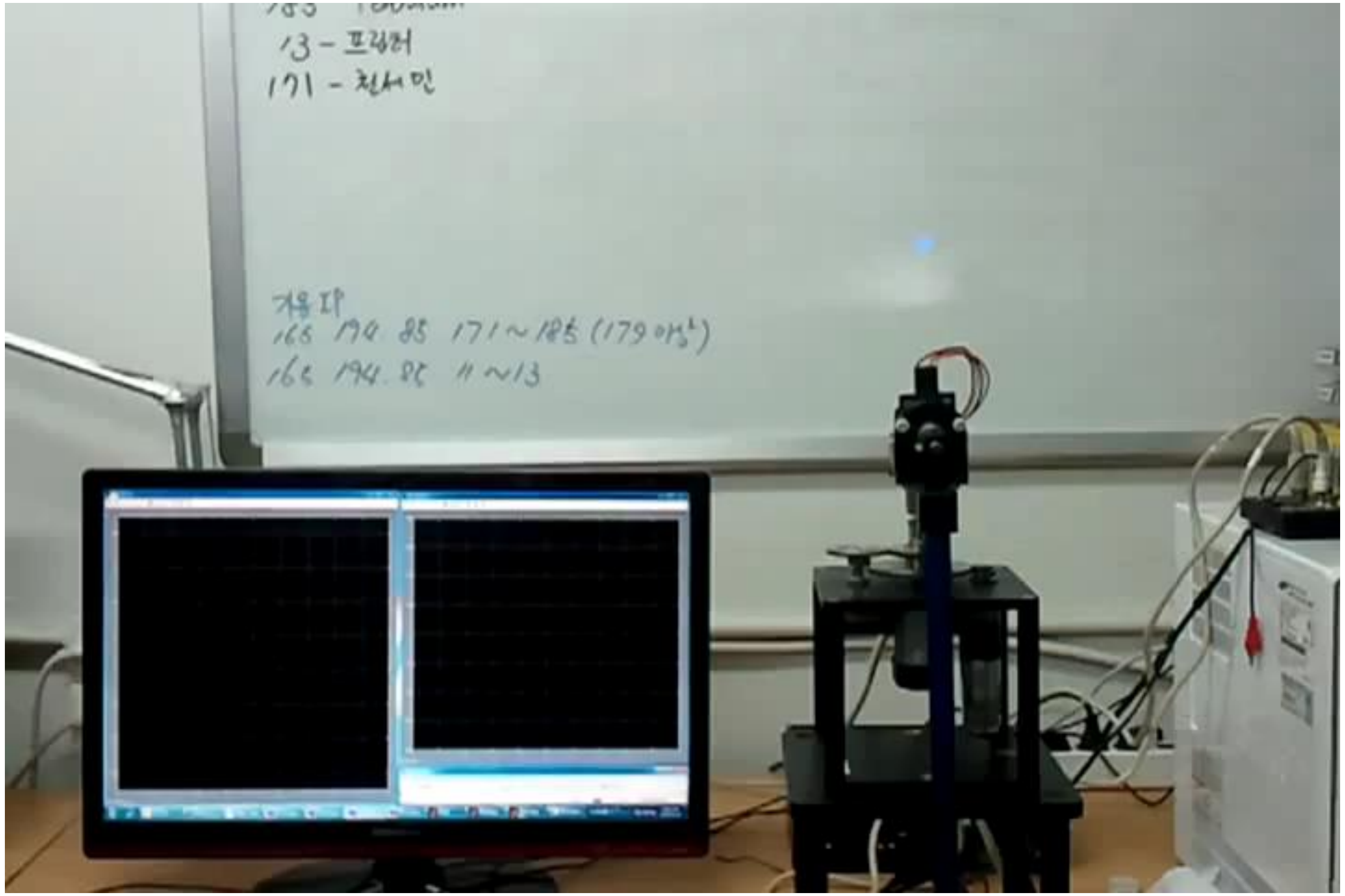
Today's agenda

- Intro to Control & Reinforcement Learning
- Linear Quadratic Regulator (LQR)
- Iterative LQR
- Model Predictive Control
- Learning the dynamics and model-based RL

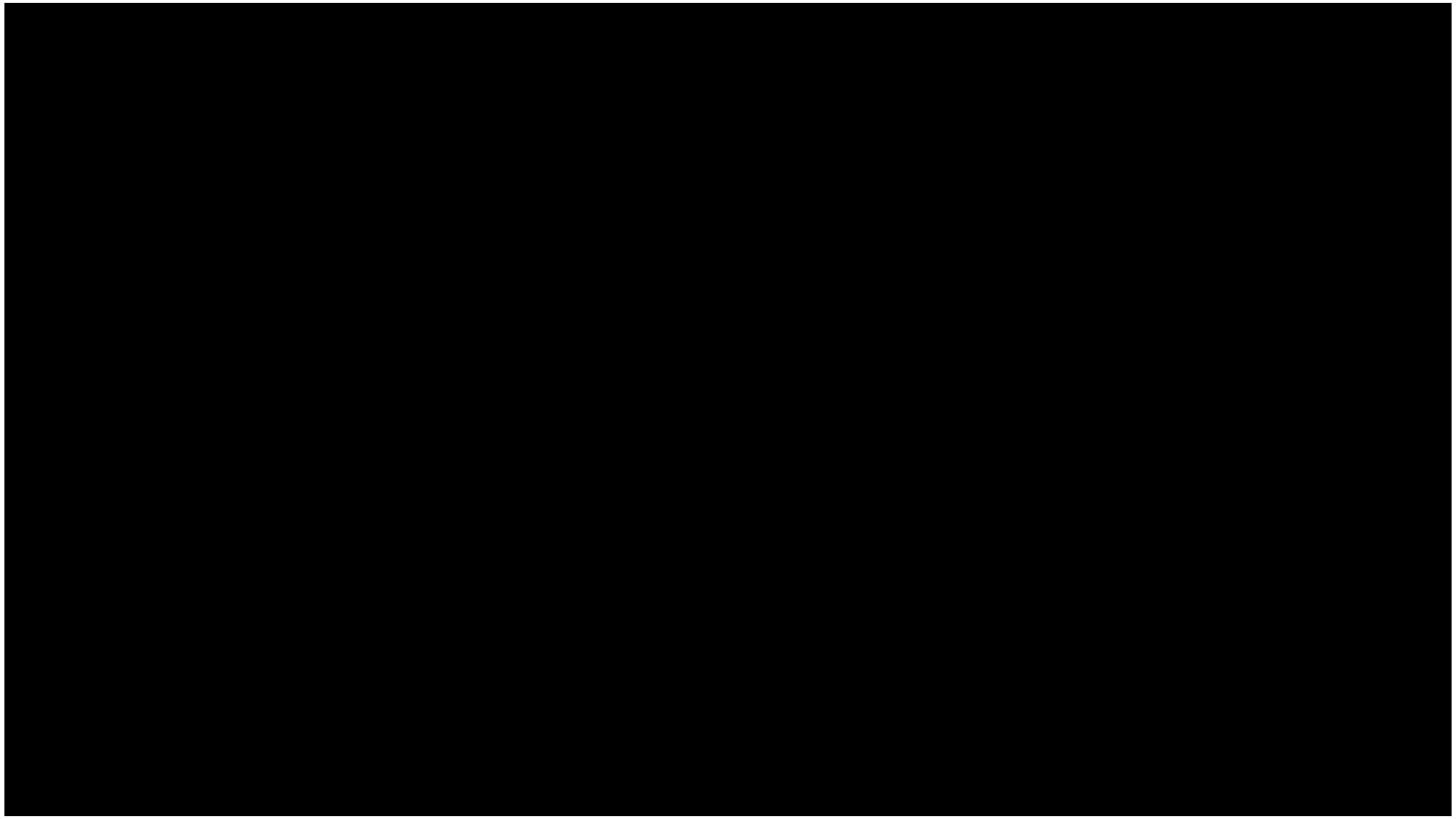


Acknowledgments
Today's slides have been influenced by: Pieter Abbeel (ECE287), Sergey Levine (DeepRL), Ben Recht (ICML'18), Emo Todorov, Zico Kolter

What you can do with LQR control



What you can do with (variants of) LQR control



Pieter Abbeel, Helicopter Aerobatics

LQR: assumptions

- You know the dynamics model of the system
- It is linear: $\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t$

State at the next time step

$$\mathbb{R}^d$$

Control / command / action applied to the system

$$\mathbb{R}^k$$

$$A \in \mathbb{R}^{d \times d}$$

$$B \in \mathbb{R}^{d \times k}$$

Which systems are linear?

- Omnidirectional robot

$$\begin{aligned} x_{t+1} &= x_t + v_x(t)\delta t \\ y_{t+1} &= y_t + v_y(t)\delta t \\ \theta_{t+1} &= \theta_t + \omega_z(t)\delta t \end{aligned}$$

$$\mathbf{x}_{t+1} = I\mathbf{x}_t + \delta t I \mathbf{u}_t$$

$$A = I$$
$$B = \delta t I$$

Which systems are linear?

- Omnidirectional robot

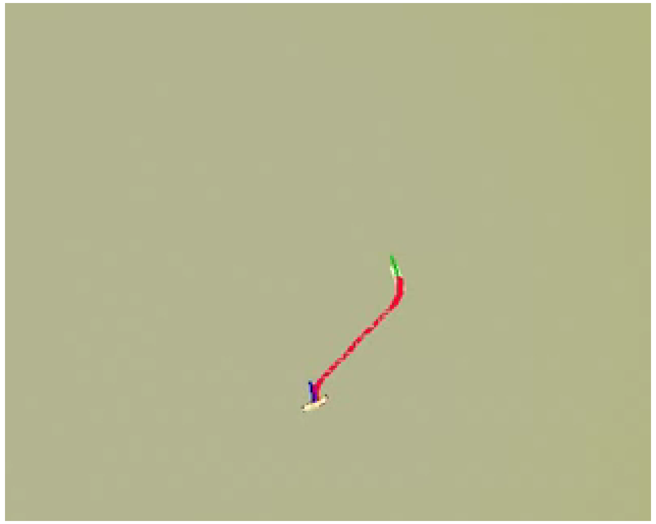
$$\begin{aligned} x_{t+1} &= x_t + v_x(t)\delta t \\ y_{t+1} &= y_t + v_y(t)\delta t \\ \theta_{t+1} &= \theta_t + \omega_z(t)\delta t \end{aligned}$$

$$\mathbf{x}_{t+1} = I\mathbf{x}_t + \delta t I \mathbf{u}_t$$

$$A = I$$
$$B = \delta t I$$

- Simple car

$$\begin{aligned} x_{t+1} &= x_t + v_x(t)\cos(\theta_t)\delta t \\ y_{t+1} &= y_t + v_x(t)\sin(\theta_t)\delta t \\ \theta_{t+1} &= \theta_t + \omega_z\delta t \end{aligned}$$



Which systems are linear?

- Omnidirectional robot

$$\begin{aligned} x_{t+1} &= x_t + v_x(t)\delta t \\ y_{t+1} &= y_t + v_y(t)\delta t \\ \theta_{t+1} &= \theta_t + \omega_z(t)\delta t \end{aligned}$$

$$\mathbf{x}_{t+1} = I\mathbf{x}_t + \delta t I \mathbf{u}_t$$

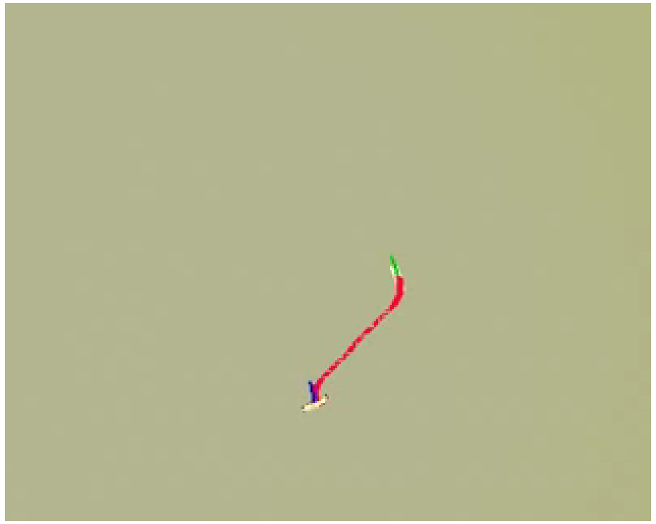
$$A = I$$
$$B = \delta t I$$

- Simple car

$$\begin{aligned} x_{t+1} &= x_t + v_x(t)\cos(\theta_t)\delta t \\ y_{t+1} &= y_t + v_x(t)\sin(\theta_t)\delta t \\ \theta_{t+1} &= \theta_t + \omega_z \delta t \end{aligned}$$

$$\mathbf{x}_{t+1} = I\mathbf{x}_t + \begin{bmatrix} \delta t\cos(\theta_t) & 0 & 0 \\ 0 & \delta t\sin(\theta_t) & 0 \\ 0 & 0 & \delta t \end{bmatrix} \mathbf{u}_t$$

$$A = I$$
$$B = B(\mathbf{x}_t)$$



Which systems are linear?

- Omnidirectional robot

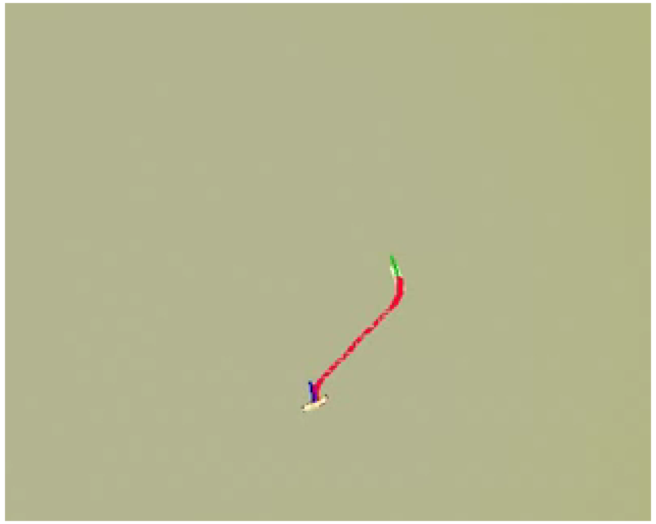
$$\begin{aligned} x_{t+1} &= x_t + v_x(t)\delta t \\ y_{t+1} &= y_t + v_y(t)\delta t \\ \theta_{t+1} &= \theta_t + \omega_z(t)\delta t \end{aligned} \quad \Rightarrow \quad \mathbf{x}_{t+1} = I\mathbf{x}_t + \delta t I \mathbf{u}_t$$

$$A = I$$
$$B = \delta t I$$

- Simple car

$$\begin{aligned} x_{t+1} &= x_t + v_x(t)\cos(\theta_t)\delta t \\ y_{t+1} &= y_t + v_x(t)\sin(\theta_t)\delta t \\ \theta_{t+1} &= \theta_t + \omega_z\delta t \end{aligned} \quad \Rightarrow \quad \mathbf{x}_{t+1} = I\mathbf{x}_t + \begin{bmatrix} \delta t\cos(\theta_t) & 0 & 0 \\ 0 & \delta t\sin(\theta_t) & 0 \\ 0 & 0 & \delta t \end{bmatrix} \mathbf{u}_t$$

$$A = I$$
$$B = B(\mathbf{x}_t)$$



The goal of LQR

- Stabilize the system around state $\mathbf{x}_t = \mathbf{0}$ with control $\mathbf{u}_t = \mathbf{0}$
- Then $\mathbf{x}_{t+1} = \mathbf{0}$ and the system will remain at zero forever

The goal of LQR

If we want to stabilize around x* then let x – x* be the state

- Stabilize the system around state $\mathbf{x}_t = \mathbf{0}$ with control $\mathbf{u}_t = \mathbf{0}$
- Then $\mathbf{x}_{t+1} = \mathbf{0}$ and the system will remain at zero forever

LQR: assumptions

- You know the dynamics model of the system
- It is linear: $\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t$

- There is an instantaneous cost associated with being at state $\mathbf{x}_t$ and taking the action $\mathbf{u}_t$: $c(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t^T Q \mathbf{x}_t + \mathbf{u}_t^T R \mathbf{u}_t$

Quadratic state cost:
Penalizes deviation
from the zero vector

Quadratic control cost:
Penalizes high control
signals

LQR: assumptions

- You know the dynamics model of the system
- It is linear: $\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t$

- There is an instantaneous cost associated with being at state $\mathbf{x}_t$ and taking the action $\mathbf{u}_t$: $c(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t^T Q \mathbf{x}_t + \mathbf{u}_t^T R \mathbf{u}_t$

Square matrices Q and R must be positive definite:

$$Q = Q^T \quad \text{and} \quad \forall x, x^T Q x > 0$$
$$R = R^T \quad \text{and} \quad \forall u, u^T R u > 0$$

i.e. positive cost for ANY nonzero state and control vector

Finite-Horizon LQR

- Idea: finding controls is an optimization problem
- Compute the control variables that minimize the cumulative cost

$$
\begin{aligned}
u_0^*, \ldots, u_{N-1}^* = \quad \underset{u_0, \ldots, u_N}{\operatorname{argmin}} \quad & \sum_{t=0}^{N} c(\mathbf{x}_t, \mathbf{u}_t) \\
\text{s.t.} \quad & \\
\mathbf{x}_1 \quad = \quad & A\mathbf{x}_0 + B\mathbf{u}_0 \\
\mathbf{x}_2 \quad = \quad & A\mathbf{x}_1 + B\mathbf{u}_1 \\
\ldots \quad & \\
\mathbf{x}_N \quad = \quad & A\mathbf{x}_{N-1} + B\mathbf{u}_{N-1}
\end{aligned}
$$

Finite-Horizon LQR

- Idea: finding controls is an optimization problem
- Compute the control variables that minimize the cumulative cost

$$
\begin{aligned}
u_0^*, \ldots, u_{N-1}^* = \quad & \underset{u_0, \ldots, u_N}{\operatorname{argmin}} \quad \sum_{t=0}^{N} c(\mathbf{x}_t, \mathbf{u}_t) \\
& \text{s.t.} \\
& \mathbf{x}_1 = A\mathbf{x}_0 + B\mathbf{u}_0 \\
& \mathbf{x}_2 = A\mathbf{x}_1 + B\mathbf{u}_1 \\
& \qquad \ldots \\
& \mathbf{x}_N = A\mathbf{x}_{N-1} + B\mathbf{u}_{N-1}
\end{aligned}
$$

We could solve this as a constrained nonlinear optimization problem. But, there is a better way: we can find a closed-form solution.

Finite-Horizon LQR

- Idea: finding controls is an optimization problem
- Compute the control variables that minimize the cumulative cost

$$u_0^*, \ldots, u_{N-1}^* = \underset{u_0,\ldots,u_N}{\operatorname{argmin}} \sum_{t=0}^{N} c(\mathbf{x}_t, \mathbf{u}_t)$$

$$\text{s.t.}$$

$$\begin{aligned} \mathbf{x}_1 &= A\mathbf{x}_0 + B\mathbf{u}_0 \\ \mathbf{x}_2 &= A\mathbf{x}_1 + B\mathbf{u}_1 \\ &\ldots \\ \mathbf{x}_N &= A\mathbf{x}_{N-1} + B\mathbf{u}_{N-1} \end{aligned}$$

Open-loop plan!

Given first state compute action sequence

Finding the LQR controller in closed-form by recursion

- Let $J_n(\mathbf{x})$ denote the cumulative cost-to-go starting from state x and moving for n time steps.

- I.e. cumulative future cost from now till n more steps

- $J_0(\mathbf{x}) = \mathbf{x}^T Q \mathbf{x}$ is the terminal cost of ending up at state x, with no actions left to perform. Recall that $c(\mathbf{x}, \mathbf{u}) = \mathbf{x}^T Q \mathbf{x} + \cancel{\mathbf{u}^T R \mathbf{u}}$

Finding the LQR controller in closed-form by recursion

- Let $J_n(\mathbf{x})$ denote the cumulative cost-to-go starting from state x and moving for n time steps.

- I.e. cumulative future cost from now till n more steps

- $J_0(\mathbf{x}) = \mathbf{x}^T Q \mathbf{x}$ is the terminal cost of ending up at state x, with no actions left to perform. Recall that $c(\mathbf{x}, \mathbf{u}) = \mathbf{x}^T Q \mathbf{x} + \cancel{\mathbf{u}^T R \mathbf{u}}$

Q: What is the optimal cumulative cost-to-go function with 1 time step left?

Finding the LQR controller in closed-form by recursion

$$J_0(\mathbf{x}) = \mathbf{x}^T Q \mathbf{x}$$

Finding the LQR controller in closed-form by recursion

$$J_0(\mathbf{x}) = \mathbf{x}^T P_0 \mathbf{x}$$

For notational convenience later on

Finding the LQR controller in closed-form by recursion

$$J_0(\mathbf{x}) = \mathbf{x}^T P_0 \mathbf{x}$$

$$J_1(\mathbf{x}) = \min_{\mathbf{u}} [\underbrace{\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} + J_0(A\mathbf{x} + B\mathbf{u})}_{}]$$

Bellman Update
Dynamic Programming
Value Iteration

In RL this would be the
state-action value function

Finding the LQR controller in closed-form by recursion

$$J_0(\mathbf{x}) = \mathbf{x}^T P_0 \mathbf{x}$$

$$\begin{aligned} J_1(\mathbf{x}) &= \min_{\mathbf{u}} [\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} + J_0(A\mathbf{x} + B\mathbf{u})] \\ &= \min_{\mathbf{u}} [\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} + (A\mathbf{x} + B\mathbf{u})^T P_0 (A\mathbf{x} + B\mathbf{u})] \end{aligned}$$

Q: How do we optimize a multivariable function with respect to some variables (in our case, the controls)?

Finding the LQR controller in closed-form by recursion

$$J_0(\mathbf{x}) = \mathbf{x}^T P_0 \mathbf{x}$$

$$\begin{aligned}
J_1(\mathbf{x}) &= \min_{\mathbf{u}} [\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} + J_0(A\mathbf{x} + B\mathbf{u})] \\
&= \min_{\mathbf{u}} [\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} + (A\mathbf{x} + B\mathbf{u})^T P_0 (A\mathbf{x} + B\mathbf{u})] \\
&= \mathbf{x}^T Q \mathbf{x} + \min_{\mathbf{u}} [\mathbf{u}^T R \mathbf{u} + (A\mathbf{x} + B\mathbf{u})^T P_0 (A\mathbf{x} + B\mathbf{u})]
\end{aligned}$$

Finding the LQR controller in closed-form by recursion

$$J_0(\mathbf{x}) = \mathbf{x}^T P_0 \mathbf{x}$$

$$\begin{aligned}
J_1(\mathbf{x}) &= \min_{\mathbf{u}} [\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} + J_0(A\mathbf{x} + B\mathbf{u})] \\
&= \min_{\mathbf{u}} [\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} + (A\mathbf{x} + B\mathbf{u})^T P_0 (A\mathbf{x} + B\mathbf{u})] \\
&= \mathbf{x}^T Q \mathbf{x} + \min_{\mathbf{u}} [\mathbf{u}^T R \mathbf{u} + (A\mathbf{x} + B\mathbf{u})^T P_0 (A\mathbf{x} + B\mathbf{u})] \\
&= \mathbf{x}^T Q \mathbf{x} + \mathbf{x}^T A^T P_0 A \mathbf{x} + \min_{\mathbf{u}} [\mathbf{u}^T R \mathbf{u} + 2\mathbf{u}^T B^T P_0 A \mathbf{x} + \mathbf{u}^T B^T P_0 B \mathbf{u}]
\end{aligned}$$

Finding the LQR controller in closed-form by recursion

$$J_0(\mathbf{x}) = \mathbf{x}^T P_0 \mathbf{x}$$

$$\begin{aligned}
J_1(\mathbf{x}) &= \min_{\mathbf{u}} [\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} + J_0(A\mathbf{x} + B\mathbf{u})] \\
&= \min_{\mathbf{u}} [\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} + (A\mathbf{x} + B\mathbf{u})^T P_0 (A\mathbf{x} + B\mathbf{u})] \\
&= \mathbf{x}^T Q \mathbf{x} + \min_{\mathbf{u}} [\mathbf{u}^T R \mathbf{u} + (A\mathbf{x} + B\mathbf{u})^T P_0 (A\mathbf{x} + B\mathbf{u})] \\
&= \mathbf{x}^T Q \mathbf{x} + \mathbf{x}^T A^T P_0 A \mathbf{x} + \min_{\mathbf{u}} [\underbrace{\mathbf{u}^T R \mathbf{u}}_{\text{Quadratic term in u}} + \underbrace{2\mathbf{u}^T B^T P_0 A \mathbf{x}}_{\text{Linear term in u}} + \underbrace{\mathbf{u}^T B^T P_0 B \mathbf{u}}_{\text{Quadratic term in u}}]
\end{aligned}$$

A: Take the partial derivative w.r.t. controls and set it to zero. That will give you a critical point.

Finding the LQR controller in closed-form by recursion

$$J_1(\mathbf{x}) \quad = \quad \mathbf{x}^T Q \mathbf{x} + \mathbf{x}^T A^T P_0 A \mathbf{x} + \min_{\mathbf{u}} [\mathbf{u}^T R \mathbf{u} + 2\mathbf{u}^T B^T P_0 A \mathbf{x} + \mathbf{u}^T B^T P_0 B \mathbf{u}]$$

From calculus/algebra:

$$\frac{\partial}{\partial \mathbf{u}}(\mathbf{u}^T M \mathbf{u}) = (M + M^T)\mathbf{u}$$

$$\frac{\partial}{\partial \mathbf{u}}(\mathbf{u}^T M \mathbf{b}) = M\mathbf{b}$$

If M is symmetric:

$$\frac{\partial}{\partial \mathbf{u}}(\mathbf{u}^T M \mathbf{u}) = 2M\mathbf{u}$$

Finding the LQR controller in closed-form by recursion

$$J_1(\mathbf{x}) \quad = \quad \mathbf{x}^T Q\mathbf{x} + \mathbf{x}^T A^T P_0 A\mathbf{x} + \min_{\mathbf{u}}[\mathbf{u}^T R\mathbf{u} + 2\mathbf{u}^T B^T P_0 A\mathbf{x} + \mathbf{u}^T B^T P_0 B\mathbf{u}]$$

The minimum is attained at:

$$2R\mathbf{u} + 2B^T P_0 A\mathbf{x} + 2B^T P_0 B\mathbf{u} = \mathbf{0}$$

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

Q: Is this matrix invertible? Recall R, Po are positive definite matrices.

From calculus/algebra:

$$\frac{\partial}{\partial \mathbf{u}}(\mathbf{u}^T M\mathbf{u}) = (M + M^T)\mathbf{u}$$

$$\frac{\partial}{\partial \mathbf{u}}(\mathbf{u}^T M\mathbf{b}) = M\mathbf{b}$$

If M is symmetric:

$$\frac{\partial}{\partial \mathbf{u}}(\mathbf{u}^T M\mathbf{u}) = 2M\mathbf{u}$$

Finding the LQR controller in closed-form by recursion

$$J_1(\mathbf{x}) \quad = \quad \mathbf{x}^T Q\mathbf{x} + \mathbf{x}^T A^T P_0 A\mathbf{x} + \min_{\mathbf{u}}[\mathbf{u}^T R\mathbf{u} + 2\mathbf{u}^T B^T P_0 A\mathbf{x} + \mathbf{u}^T B^T P_0 B\mathbf{u}]$$

The minimum is attained at:

$$2R\mathbf{u} + 2B^T P_0 A\mathbf{x} + 2B^T P_0 B\mathbf{u} = \mathbf{0}$$

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

Q: Is this matrix invertible? Recall R, Po are positive definite matrices.

$R + B^T P_0 B$ is positive definite, so it is invertible

Finding the LQR controller in closed-form by recursion

$$J_1(\mathbf{x}) \quad = \quad \mathbf{x}^T Q\mathbf{x} + \mathbf{x}^T A^T P_0 A\mathbf{x} + \min_{\mathbf{u}}[\mathbf{u}^T R\mathbf{u} + 2\mathbf{u}^T B^T P_0 A\mathbf{x} + \mathbf{u}^T B^T P_0 B\mathbf{u}]$$

The minimum is attained at:

$$2R\mathbf{u} + 2B^T P_0 A\mathbf{x} + 2B^T P_0 B\mathbf{u} = \mathbf{0}$$

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

So, the optimal control for the last time step is:

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

$$\mathbf{u} = K_1\mathbf{x}$$

Linear controller in terms of the state

Finding the LQR controller in closed-form by recursion

$$J_1(\mathbf{x}) \quad = \quad \mathbf{x}^T Q \mathbf{x} + \mathbf{x}^T A^T P_0 A \mathbf{x} + \min_{\mathbf{u}} [\mathbf{u}^T R \mathbf{u} + 2\mathbf{u}^T B^T P_0 A \mathbf{x} + \mathbf{u}^T B^T P_0 B \mathbf{u}]$$

The minimum is attained at:

$$2R\mathbf{u} + 2B^T P_0 A \mathbf{x} + 2B^T P_0 B \mathbf{u} = \mathbf{0}$$

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

So, the optimal control for the last time step is:

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

$$\mathbf{u} = K_1 \mathbf{x}$$

Linear controller in terms of the state

We computed the location of the minimum.
Now, plug it back in and compute the minimum value

Finding the LQR controller in closed-form by recursion

$$J_0(\mathbf{x}) = \mathbf{x}^T P_0 \mathbf{x}$$

$$\begin{aligned} J_1(\mathbf{x}) &= \mathbf{x}^T Q \mathbf{x} + \mathbf{x}^T A^T P_0 A \mathbf{x} + \min_{\mathbf{u}} [\mathbf{u}^T R \mathbf{u} + 2\mathbf{u}^T B^T P_0 A \mathbf{x} + \mathbf{u}^T B^T P_0 B \mathbf{u}] \\ &= \mathbf{x}^T \underbrace{(Q + K_1^T R K_1 + (A + B K_1)^T P_0 (A + B K_1))}_{P_1} \mathbf{x} \end{aligned}$$

Q: Why is this a big deal?

A: The cost-to-go function remains quadratic after the first recursive step.

Finding the LQR controller in closed-form by recursion

Time N (planning horizon)

$$J_0(\mathbf{x}) = \mathbf{x}^T P_0 \mathbf{x}$$

$$\mathbf{u} = -(R + B^T P_0 B)^{-1} B^T P_0 A \mathbf{x}$$
$$\mathbf{u} = K_1 \mathbf{x}$$

$$\begin{aligned} J_1(\mathbf{x}) &= \mathbf{x}^T (Q + K_1^T R K_1 + (A + BK_1)^T P_0 (A + BK_1))\mathbf{x} \\ &= \mathbf{x}^T P_1 \mathbf{x} \end{aligned}$$

$\ldots$

J remains quadratic in x throughout the recursion

$$\mathbf{u} = -(R + B^T P_{n-1} B)^{-1} B^T P_{n-1} A \mathbf{x}$$
$$\mathbf{u} = K_n \mathbf{x}$$

$$\begin{aligned} J_n(\mathbf{x}) &= \mathbf{x}^T (Q + K_n^T R K_n + (A + BK_n)^T P_{n-1} (A + BK_n))\mathbf{x} \\ &= \mathbf{x}^T P_n \mathbf{x} \end{aligned}$$

u remains linear in x throughout the recursion

$\ldots$

Time 0

Finite-Horizon LQR: algorithm summary

$$P_0 = Q$$

// n is the # of steps left

for n = 1…N

$$K_n = -(R + B^T P_{n-1} B)^{-1} B^T P_{n-1} A$$

$$P_n = Q + K_n^T R K_n + (A + BK_n)^T P_{n-1} (A + BK_n)$$

Optimal control for time t = N – n is $\mathbf{u}_t = K_t \mathbf{x}_t$ with cost-to-go $J_t(\mathbf{x}) = \mathbf{x}^T P_t \mathbf{x}$

where the states are predicted forward in time according to linear dynamics

Finite-Horizon LQR: algorithm summary

$$P_0 = Q$$

// n is the # of steps left

for n = 1…N

$$K_n = -(R + B^T P_{n-1} B)^{-1} B^T P_{n-1} A$$

$$P_n = Q + K_n^T R K_n + (A + BK_n)^T P_{n-1} (A + BK_n)$$

One pass **backward** in time:

Matrix gains are precomputed based on the dynamics and the instantaneous cost

Optimal control for time t = N – n is $\mathbf{u}_t = K_t \mathbf{x}_t$ with cost-to-go $J_t(\mathbf{x}) = \mathbf{x}^T P_t \mathbf{x}$

where the states are predicted forward in time according to linear dynamics

Finite-Horizon LQR: algorithm summary

$$P_0 = Q$$

// n is the # of steps left

for n = 1…N

$$K_n = -(R + B^T P_{n-1} B)^{-1} B^T P_{n-1} A$$

$$P_n = Q + K_n^T R K_n + (A + BK_n)^T P_{n-1} (A + BK_n)$$

One pass **backward** in time:

Matrix gains are precomputed based on the dynamics and the instantaneous cost

Optimal control for time t = N – n is $\mathbf{u}_t = K_t \mathbf{x}_t$ with cost-to-go $J_t(\mathbf{x}) = \mathbf{x}^T P_t \mathbf{x}$

where the states are predicted forward in time according to linear dynamics

One pass **forward** in time

Predict states, compute controls and cost-to-go

Finite-Horizon LQR: algorithm summary

Potential problem for states of dimension >> 100:
Matrix inversion is expensive: O(k^2.3) for the best known algorithm and O(k^3) for Gaussian Elimination.

$$P_0 = Q$$

// n is the # of steps left

for n = 1…N

$$K_n = -(R + B^T P_{n-1} B)^{-1} B^T P_{n-1} A$$

$$P_n = Q + K_n^T R K_n + (A + BK_n)^T P_{n-1} (A + BK_n)$$

Optimal control for time t = N – n is $\mathbf{u}_t = K_t \mathbf{x}_t$ with cost-to-go $J_t(\mathbf{x}) = \mathbf{x}^T P_t \mathbf{x}$

where the states are predicted forward in time according to linear dynamics

LQR: general form of dynamics and cost functions

Even though we assumed $\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t$ $\quad c(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t^T Q\mathbf{x}_t + \mathbf{u}_t^T R\mathbf{u}_t$

we can also accommodate $\mathbf{x}_{t+1} = A_t\mathbf{x}_t + B_t\mathbf{u}_t + \mathbf{b}_t$ $\quad c(\mathbf{x}_t, \mathbf{u}_t) = \begin{bmatrix}\mathbf{x}_t \\ \mathbf{u}_t\end{bmatrix}^T H_t \begin{bmatrix}\mathbf{x}_t \\ \mathbf{u}_t\end{bmatrix} + \begin{bmatrix}\mathbf{x}_t \\ \mathbf{u}_t\end{bmatrix}^T \mathbf{h}_t$

but the form of the computed controls becomes $\mathbf{u}_t = K_t\mathbf{x}_t + \mathbf{k}_t$

LQR with stochastic dynamics

Assume $\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + \mathbf{w}_t$ and $c(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t^T Q \mathbf{x}_t + \mathbf{u}_t^T R \mathbf{u}_t$

zero mean Gaussian

Then the form of the optimal policy is the same as in LQR $\mathbf{u}_t = K_t \mathbf{x}_t$

No need to change the algorithm, as long as you observe the state at each step (closed-loop policy)

LQR with stochastic dynamics

Assume $\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + \mathbf{w}_t$ and $c(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t^T Q \mathbf{x}_t + \mathbf{u}_t^T R \mathbf{u}_t$

zero mean Gaussian

Then the form of the optimal policy is the same as in LQR $\mathbf{u}_t = K_t \mathbf{x}_t$

No need to change the algorithm, as long as you observe the state at each step (closed-loop policy)

Linear Quadratic Gaussian
LQG

LQR summary

- Advantages:
  - If system is linear LQR gives the optimal controller that takes the system's state to 0 (or the desired target state, same thing)

- Drawbacks:

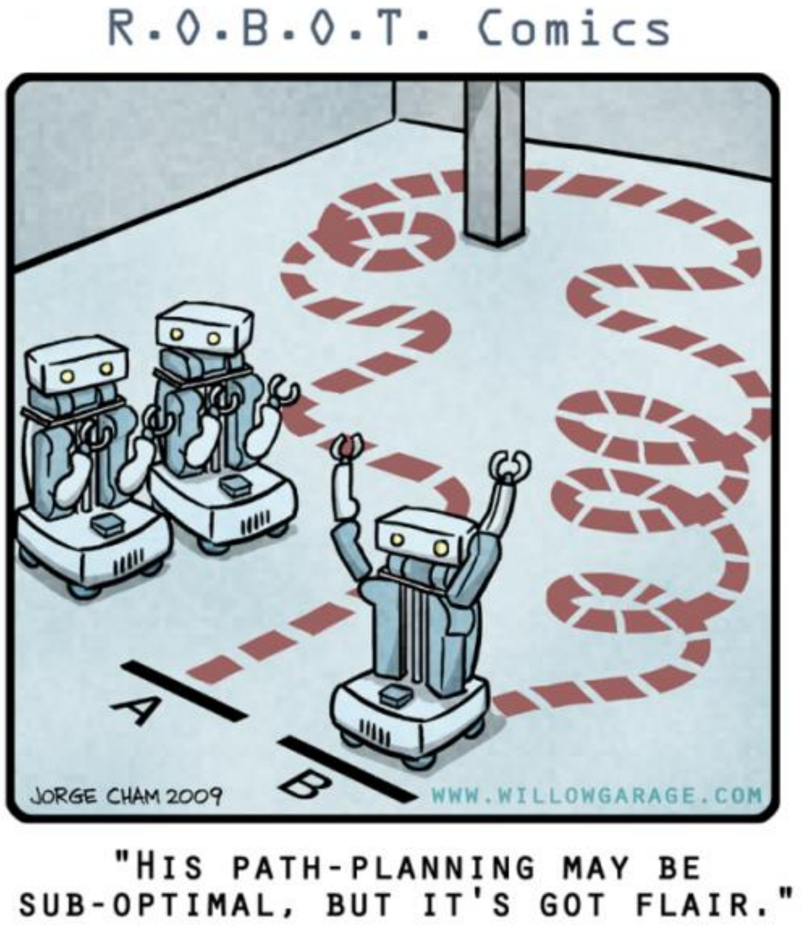
LQR summary

- Advantages:
  - If system is linear LQR gives the optimal controller that takes the system's state to 0 (or the desired target state, same thing)

- Drawbacks:
  - Linear dynamics
  - How can you include obstacles or constraints in the specification?
  - Not easy to put bounds on control values

Today's agenda

- Intro to Control & Reinforcement Learning
- Linear Quadratic Regulator (LQR)
- Iterative LQR
- Model Predictive Control
- Learning the dynamics and model-based RL



What happens in the general nonlinear case?

$$
\begin{aligned}
u_0^*, \ldots, u_{N-1}^* = \quad & \underset{u_0, \ldots, u_N}{\operatorname{argmin}} \quad \sum_{t=0}^{N} c(\mathbf{x}_t, \mathbf{u}_t) \\
& \text{s.t.} \\
\mathbf{x}_1 &= f(\mathbf{x}_0, \mathbf{u}_0) \\
\mathbf{x}_2 &= f(\mathbf{x}_1, \mathbf{u}_1) \\
& \ldots \\
\mathbf{x}_N &= f(\mathbf{x}_{N-1}, \mathbf{u}_{N-1})
\end{aligned}
$$

Arbitrary differentiable functions c, f

What happens in the general nonlinear case?

$$
\begin{aligned}
u_0^*, \ldots, u_{N-1}^* = \quad & \underset{u_0,\ldots,u_N}{\operatorname{argmin}} \quad \sum_{t=0}^{N} c(\mathbf{x}_t, \mathbf{u}_t) \\
& \text{s.t.} \\
\mathbf{x}_1 &= f(\mathbf{x}_0, \mathbf{u}_0) \\
\mathbf{x}_2 &= f(\mathbf{x}_1, \mathbf{u}_1) \\
& \ldots \\
\mathbf{x}_N &= f(\mathbf{x}_{N-1}, \mathbf{u}_{N-1})
\end{aligned}
$$

Arbitrary differentiable functions c, f

Idea: iteratively approximate solution by solving linearized versions of the problem via LQR

Iterative LQR (iLQR)

Given an initial sequence of states $\bar{\mathbf{x}}_0, \ldots, \bar{\mathbf{x}}_N$ and actions $\bar{\mathbf{u}}_0, \ldots, \bar{\mathbf{u}}_N$

Linearize dynamics $$f(\mathbf{x}_t, \mathbf{u}_t) \approx \tilde{f}(\delta\mathbf{x}_t, \delta\mathbf{u}_t) = \underbrace{f(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t)}_{\mathbf{b}_t} + \underbrace{\frac{\partial f}{\partial \mathbf{x}}(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t)}_{A_t}\underbrace{(\mathbf{x}_t - \bar{\mathbf{x}}_t)}_{\delta\mathbf{x}_t} + \underbrace{\frac{\partial f}{\partial \mathbf{u}}(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t)}_{B_t}\underbrace{(\mathbf{u}_t - \bar{\mathbf{u}}_t)}_{\delta\mathbf{u}_t}$$

Iterative LQR (iLQR)

Given an initial sequence of states $\bar{\mathbf{x}}_0, \ldots, \bar{\mathbf{x}}_N$ and actions $\bar{\mathbf{u}}_0, \ldots, \bar{\mathbf{u}}_N$

Linearize dynamics $$f(\mathbf{x}_t, \mathbf{u}_t) \approx \tilde{f}(\delta\mathbf{x}_t, \delta\mathbf{u}_t) = \underbrace{f(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t)}_{\mathbf{b}_t} + \underbrace{\frac{\partial f}{\partial \mathbf{x}}(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t)}_{A_t}\underbrace{(\mathbf{x}_t - \bar{\mathbf{x}}_t)}_{\delta\mathbf{x}_t} + \underbrace{\frac{\partial f}{\partial \mathbf{u}}(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t)}_{B_t}\underbrace{(\mathbf{u}_t - \bar{\mathbf{u}}_t)}_{\delta\mathbf{u}_t}$$

Taylor expand cost $$c(\mathbf{x}_t, \mathbf{u}_t) \approx \tilde{c}(\delta\mathbf{x}_t, \delta\mathbf{u}_t) = c(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t) + \underbrace{\nabla_{\mathbf{x}_t, \mathbf{u}_t} c(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t)}_{\mathbf{h}_t} \begin{bmatrix} \mathbf{x}_t - \bar{\mathbf{x}}_t \\ \mathbf{u}_t - \bar{\mathbf{u}}_t \end{bmatrix} + 1/2 \begin{bmatrix} \mathbf{x}_t - \bar{\mathbf{x}}_t \\ \mathbf{u}_t - \bar{\mathbf{u}}_t \end{bmatrix}^T \underbrace{\nabla^2_{\mathbf{x}_t, \mathbf{u}_t} c(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t)}_{H_t} \begin{bmatrix} \mathbf{x}_t - \bar{\mathbf{x}}_t \\ \mathbf{u}_t - \bar{\mathbf{u}}_t \end{bmatrix}$$

Iterative LQR (iLQR)

Given an initial sequence of states $\bar{\mathbf{x}}_0, \ldots, \bar{\mathbf{x}}_N$ and actions $\bar{\mathbf{u}}_0, \ldots, \bar{\mathbf{u}}_N$

Linearize dynamics $$f(\mathbf{x}_t, \mathbf{u}_t) \approx \tilde{f}(\delta\mathbf{x}_t, \delta\mathbf{u}_t) = \underbrace{f(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t)}_{\mathbf{b}_t} + \underbrace{\frac{\partial f}{\partial \mathbf{x}}(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t)}_{A_t}\underbrace{(\mathbf{x}_t - \bar{\mathbf{x}}_t)}_{\delta\mathbf{x}_t} + \underbrace{\frac{\partial f}{\partial \mathbf{u}}(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t)}_{B_t}\underbrace{(\mathbf{u}_t - \bar{\mathbf{u}}_t)}_{\delta\mathbf{u}_t}$$

Taylor expand cost $$c(\mathbf{x}_t, \mathbf{u}_t) \approx \tilde{c}(\delta\mathbf{x}_t, \delta\mathbf{u}_t) = c(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t) + \underbrace{\nabla_{\mathbf{x}_t, \mathbf{u}_t} c(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t)}_{\mathbf{h}_t} \begin{bmatrix} \mathbf{x}_t - \bar{\mathbf{x}}_t \\ \mathbf{u}_t - \bar{\mathbf{u}}_t \end{bmatrix} + 1/2 \begin{bmatrix} \mathbf{x}_t - \bar{\mathbf{x}}_t \\ \mathbf{u}_t - \bar{\mathbf{u}}_t \end{bmatrix}^T \underbrace{\nabla^2_{\mathbf{x}_t, \mathbf{u}_t} c(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t)}_{H_t} \begin{bmatrix} \mathbf{x}_t - \bar{\mathbf{x}}_t \\ \mathbf{u}_t - \bar{\mathbf{u}}_t \end{bmatrix}$$

Use LQR backward pass on the approximate dynamics $\tilde{f}(\delta\mathbf{x}_t, \delta\mathbf{u}_t)$ and cost $\tilde{c}(\delta\mathbf{x}_t, \delta\mathbf{u}_t)$

Iterative LQR (iLQR)

Given an initial sequence of states $\bar{\mathbf{x}}_0, \ldots, \bar{\mathbf{x}}_N$ and actions $\bar{\mathbf{u}}_0, \ldots, \bar{\mathbf{u}}_N$

Linearize dynamics $$f(\mathbf{x}_t, \mathbf{u}_t) \approx \tilde{f}(\delta\mathbf{x}_t, \delta\mathbf{u}_t) = \underbrace{f(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t)}_{\mathbf{b}_t} + \underbrace{\frac{\partial f}{\partial \mathbf{x}}(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t)}_{A_t}\underbrace{(\mathbf{x}_t - \bar{\mathbf{x}}_t)}_{\delta\mathbf{x}_t} + \underbrace{\frac{\partial f}{\partial \mathbf{u}}(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t)}_{B_t}\underbrace{(\mathbf{u}_t - \bar{\mathbf{u}}_t)}_{\delta\mathbf{u}_t}$$

Taylor expand cost $$c(\mathbf{x}_t, \mathbf{u}_t) \approx \tilde{c}(\delta\mathbf{x}_t, \delta\mathbf{u}_t) = c(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t) + \underbrace{\nabla_{\mathbf{x}_t, \mathbf{u}_t} c(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t)}_{\mathbf{h}_t} \begin{bmatrix} \mathbf{x}_t - \bar{\mathbf{x}}_t \\ \mathbf{u}_t - \bar{\mathbf{u}}_t \end{bmatrix} + 1/2 \begin{bmatrix} \mathbf{x}_t - \bar{\mathbf{x}}_t \\ \mathbf{u}_t - \bar{\mathbf{u}}_t \end{bmatrix}^T \underbrace{\nabla^2_{\mathbf{x}_t, \mathbf{u}_t} c(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t)}_{H_t} \begin{bmatrix} \mathbf{x}_t - \bar{\mathbf{x}}_t \\ \mathbf{u}_t - \bar{\mathbf{u}}_t \end{bmatrix}$$

Use LQR backward pass on the approximate dynamics $\tilde{f}(\delta\mathbf{x}_t, \delta\mathbf{u}_t)$ and cost $\tilde{c}(\delta\mathbf{x}_t, \delta\mathbf{u}_t)$

Do a forward pass to get $\delta\mathbf{u}_t$ and $\delta\mathbf{x}_t$ and update state and action sequence $\bar{\mathbf{x}}_0, \ldots, \bar{\mathbf{x}}_N$ and $\bar{\mathbf{u}}_0, \ldots, \bar{\mathbf{u}}_N$

Iterative LQR: convergence & tricks

- New state and action sequence in iLQR is not guaranteed to be close to the linearization point (so linear approximation might be bad)

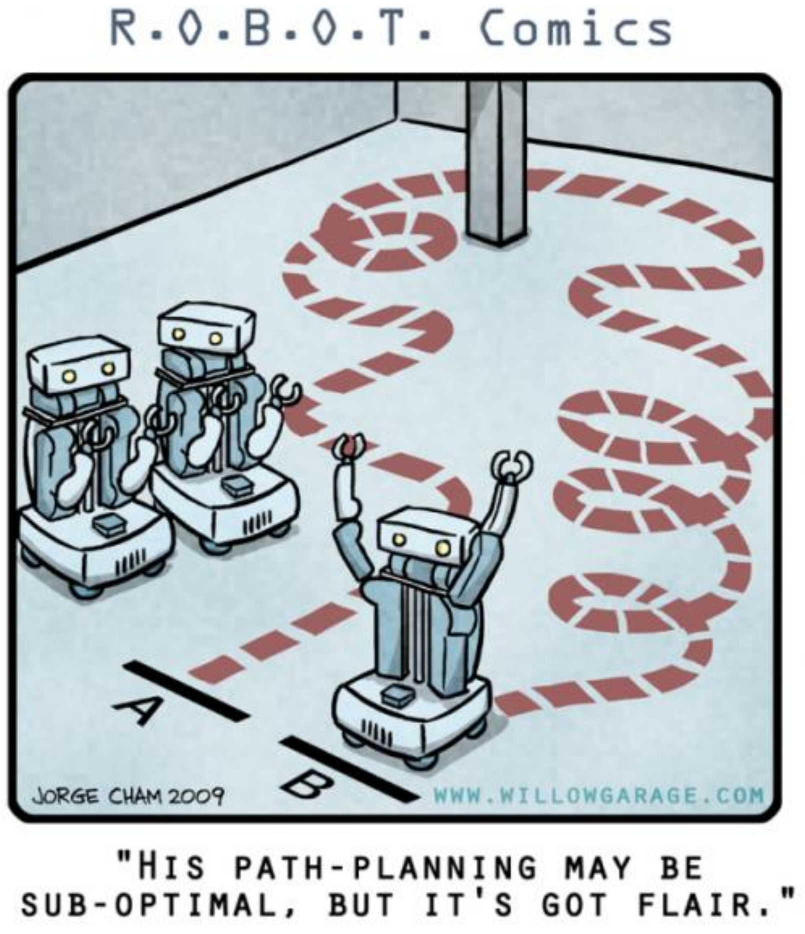
- Trick: try to penalize magnitude of $\delta \mathbf{u}_t$ and $\delta \mathbf{x}_t$

  Replace old LQR linearized cost with $(1-\alpha)\tilde{c}(\delta \mathbf{x}_t, \delta \mathbf{u}_t) + \alpha(||\delta \mathbf{x}_t||^2 + ||\delta \mathbf{u}_t||^2)$

- Problem: Can get stuck in local optima, need to initialize well
- Problem: Hessian might not be positive definite

Today’s agenda

- Intro to Control & Reinforcement Learning
- Linear Quadratic Regulator (LQR)
- Iterative LQR
- Model Predictive Control
- Learning the dynamics and model-based RL



Open loop vs. closed loop

- The instances of LQR and iLQR that we saw were open-loop
- Commands are executed in sequence, without feedback

Open loop vs. closed loop

- The instances of LQR and iLQR that we saw were open-loop
- Commands are executed in sequence, without feedback

- Idea: what if we throw away all commands except the first
- We can execute the first command, and then replan

Takes into account the changing state

Model Predictive Control

while True:

observe the current state $\mathbf{x}_0$

run LQR/iLQR or LQG/iLQG or other planner to get $\mathbf{u}_0, \ldots, \mathbf{u}_{N-1}$

Execute $\mathbf{u}_0$

Model Predictive Control

while True:

observe the current state $\mathbf{x}_0$

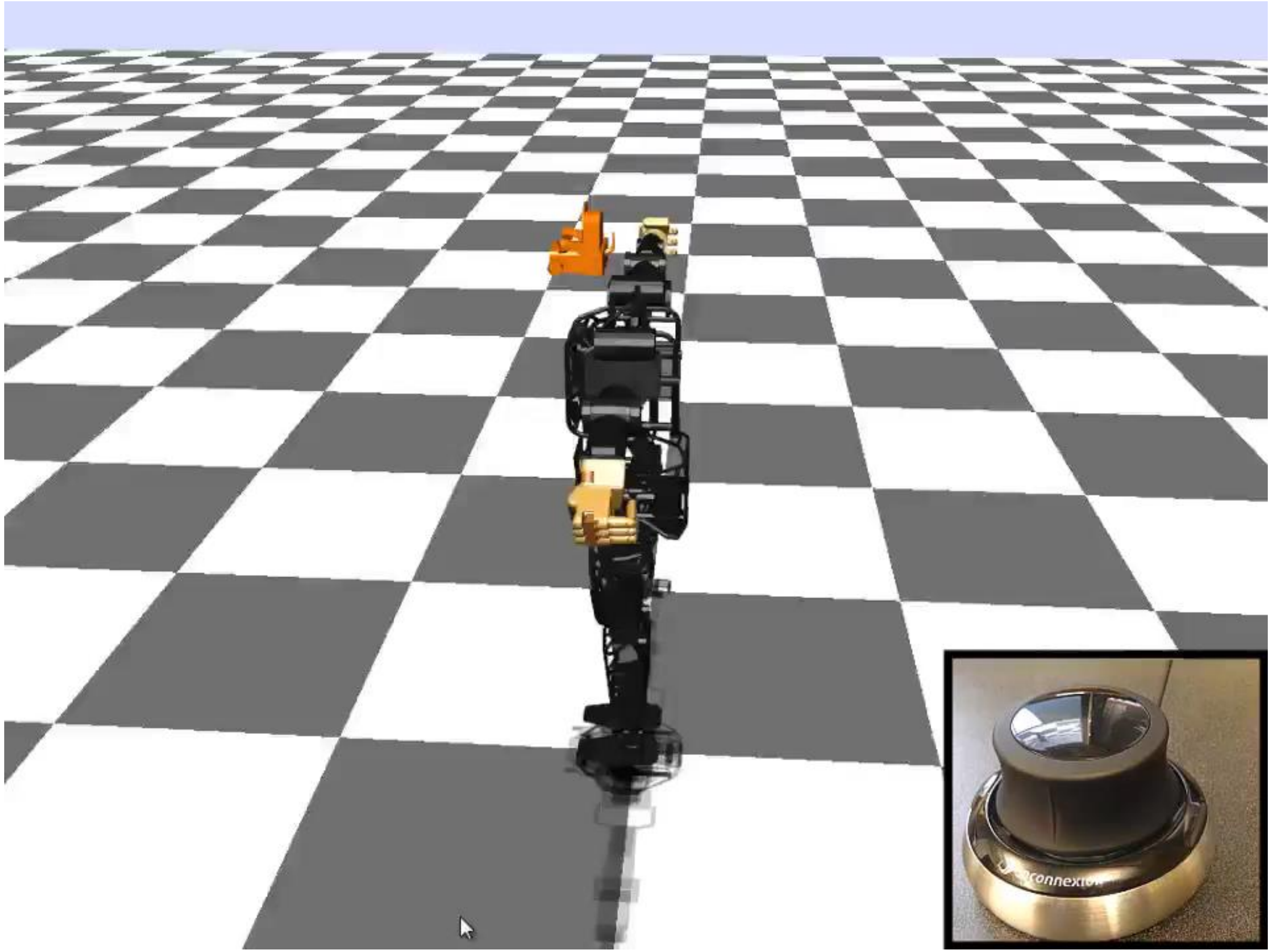
run LQR/iLQR or LQG/iLQG or other planner to get $\mathbf{u}_0, \ldots, \mathbf{u}_{N-1}$

Execute $\mathbf{u}_0$

Possible speedups:

1. Don't plan too far ahead with LQR
2. Execute more than one planned action
3. Warm starts and initialization
4. Use faster / custom optimizer
   (e.g. CPLEX, sequential quadratic programming)

Online trajectory optimization / MPC



Online trajectory optimization / MPC

Synthesis of Complex Behaviors
with
Online Trajectory Optimization

Yuval Tassa, Tom Erez & Emo Todorov

IEEE International Conference
on Intelligent Robots and Systems
2012

Online trajectory optimization / MPC

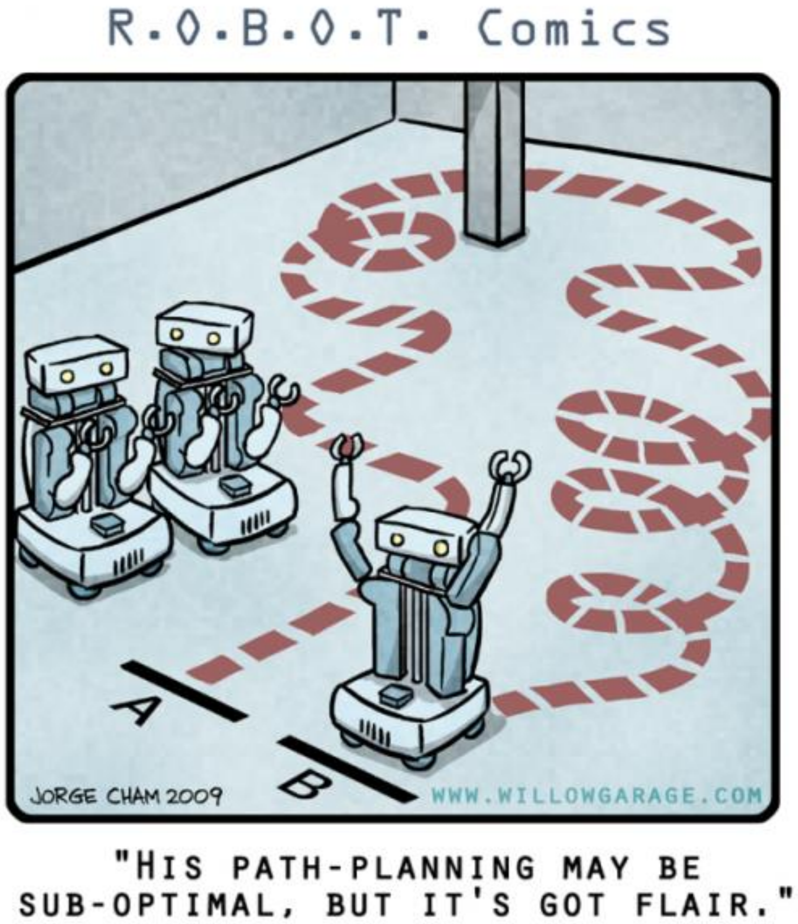
Test 3: Dynamic Maneuvers

ADRL ETH zürich Autonomous Systems Lab

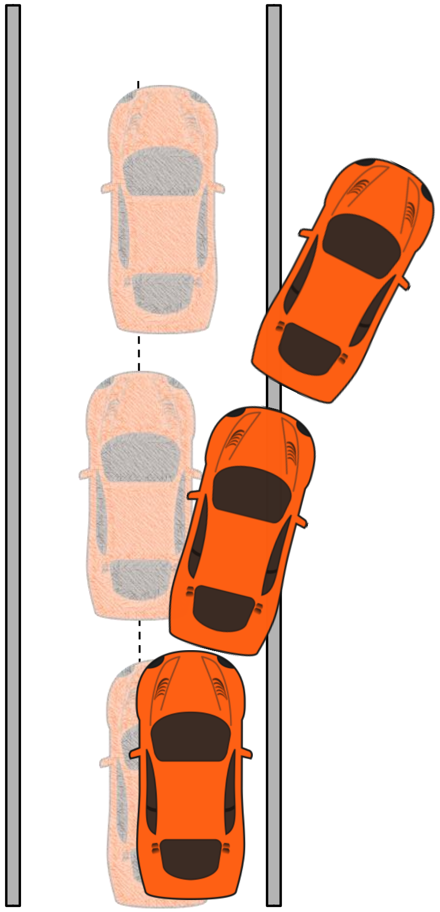
Online trajectory optimization / MPC

Today's agenda

- Intro to Control & Reinforcement Learning
- Linear Quadratic Regulator (LQR)
- Iterative LQR
- Model Predictive Control
- Learning the dynamics and model-based RL



Learning a dynamics model



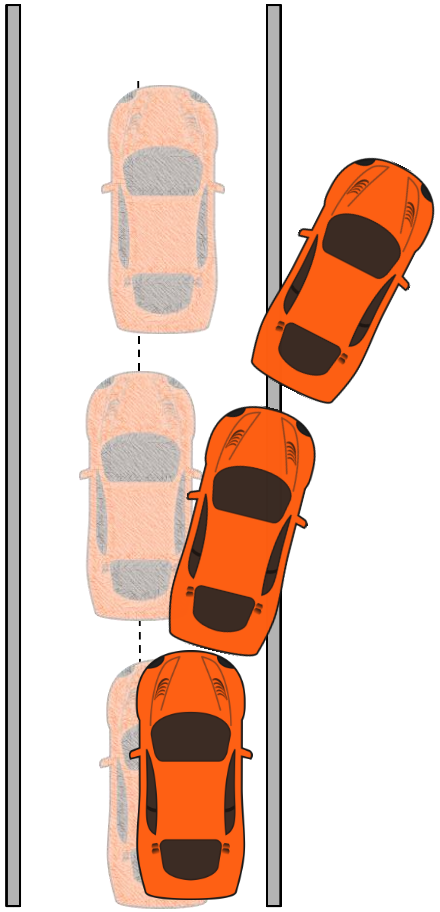
Test distribution is different from training distribution (covariate shift)

Idea #1: Collect dataset $D = \{(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})\}$

do supervised learning to minimize $\sum_t ||f_\theta(\mathbf{x}_t, \mathbf{u}_t) - \mathbf{x}_{t+1}||^2$

and then use the learned model for planning

Learning a dynamics model



Test distribution is different from training distribution (covariate shift)

Idea #1: Collect dataset $D = \{(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})\}$

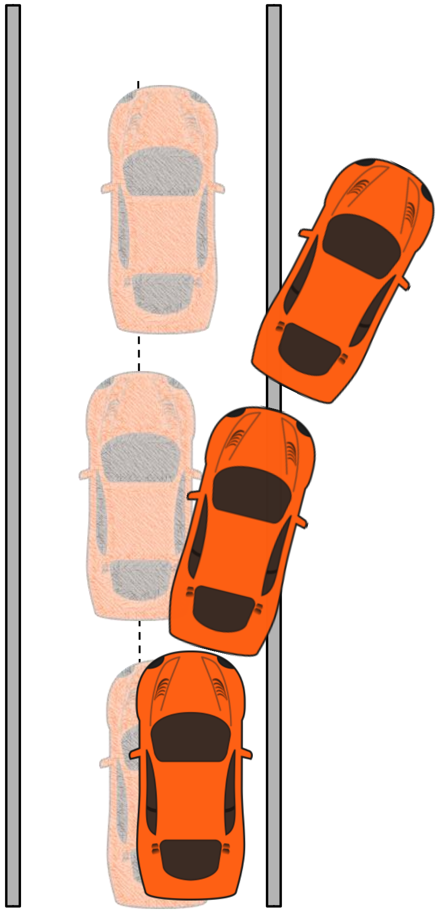
do supervised learning to minimize $\sum_t ||f_\theta(\mathbf{x}_t, \mathbf{u}_t) - \mathbf{x}_{t+1}||^2$

and then use the learned model for planning

Possibly a better idea: instead of minimizing single-step prediction errors, minimize multi-step errors.

See "Improving Multi-step Prediction of Learned Time Series Models" by Venkatraman, Hebert, Bagnell

Learning a dynamics model



Test distribution is different from training distribution (covariate shift)

Idea #1: Collect dataset $D = \{(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})\}$

do supervised learning to minimize $\sum_t ||f_\theta(\mathbf{x}_t, \mathbf{u}_t) - \mathbf{x}_{t+1}||^2$

and then use the learned model for planning

Possibly a better idea: instead of predicting next state predict next change in state.

See "PILCO: A Model-Based and Data-Efficient Approach to Policy Search" by Deisenroth, Rasmussen

Model-based RL

Collect initial dataset $D = \{(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})\}$

Fit dynamics model $f_\theta(\mathbf{x}_t, \mathbf{u}_t)$

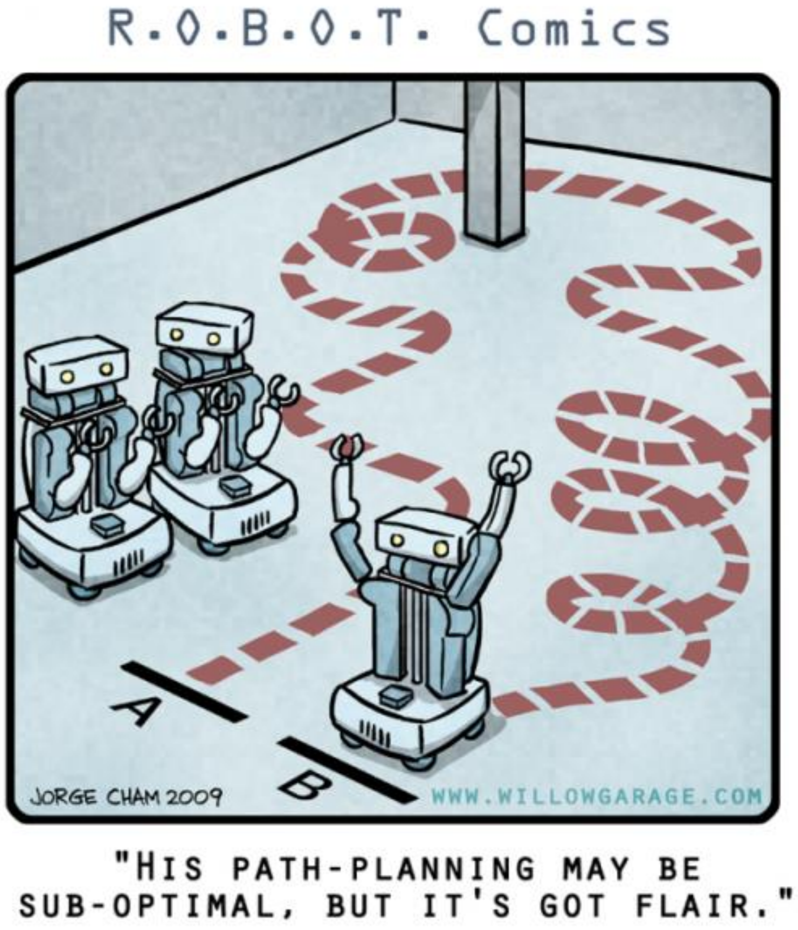
Plan through $f_\theta(\mathbf{x}_t, \mathbf{u}_t)$ to get actions

Execute first action, observe new state $\mathbf{x}_{t+1}$

Append $(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$ to D

Today's agenda

- Intro to Control & Reinforcement Learning
- Linear Quadratic Regulator (LQR)
- Iterative LQR
- Model Predictive Control
- Learning the dynamics and model-based RL
- Appendix



Appendix #1 (optional reading) LQR extensions: time-varying systems

- What can we do when $\mathbf{x}_{t+1} = A_t\mathbf{x}_t + B_t\mathbf{u}_t$ and $c(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t^T Q\mathbf{x}_t + \mathbf{u}_t^T R\mathbf{u}_t$?
- Turns out, the proof and the algorithm are almost the same

$$P_0 = Q_N$$

// n is the # of steps left

for n = 1…N

$$K_n = -(R_{N-n} + B_{N-n}^T P_{n-1} B_{N-n})^{-1} B_{N-n}^T P_{n-1} A_{N-n}$$

$$P_n = Q_{N-n} + K_n^T R_{N-n} K_n + (A_{N-n} + B_{N-n}K_n)^T P_{n-1}(A_{N-n} + B_{N-n}K_n)$$

Optimal controller for n-step horizon is $\mathbf{u}_n = K_n\mathbf{x}_n$ with cost-to-go $J_n(\mathbf{x}) = \mathbf{x}^T P_n \mathbf{x}$

Appendix #2 (optional reading)
Why not use PID control?

- We could, but:

- The gains for PID are good for a small region of state-space.
  - System reaches a state outside this set → becomes unstable
  - PID has no formal guarantees on the size of the set

- We would need to tune PID gains for every control variable.
  - If the state vector has multiple dimensions it becomes harder to tune every control variable in isolation. Need to consider interactions and correlations.

- We would need to tune PID gains for different regions of the state-space and guarantee smooth gain transitions
  - This is called gain scheduling, and it takes a lot of effort and time

Appendix #2 (optional reading) Why not use PID?

LQR addresses these problems

- We could, but:
- The gains for PID are good for a small region of state-space.
  - System reaches a state outside this set → becomes unstable
  - PID has no formal guarantees on the size of the set
- We would need to tune PID gains for every control variable.
  - If the state vector has multiple dimensions it becomes harder to tune every control variable in isolation. Need to consider interactions and correlations.
- We would need to tune PID gains for different regions of the state-space and guarantee smooth gain transitions
  - This is called gain scheduling, and it takes a lot of effort and time

Appendix #3 (optional reading)
Examples of models and solutions with LQR

LQR example #1: omnidirectional vehicle with friction

- Similar to double integrator dynamical system, but with friction:

$$m\ddot{\mathbf{p}} = \mathbf{u} - \alpha\dot{\mathbf{p}}$$

Force applied to the vehicle

Control applied to the vehicle

Friction opposed to motion

LQR example #1: omnidirectional vehicle with friction

- Similar to double integrator dynamical system, but with friction:

$$m\ddot{\mathbf{p}} = \mathbf{u} - \alpha\dot{\mathbf{p}}$$

- Set $\dot{\mathbf{p}} = \mathbf{v}$ and then you get:

$$m\dot{\mathbf{v}} = \mathbf{u} - \alpha\mathbf{v}$$

LQR example #1: omnidirectional vehicle with friction

- Similar to double integrator dynamical system, but with friction:

$$m\ddot{\mathbf{p}} = \mathbf{u} - \alpha\dot{\mathbf{p}}$$

- Set $\dot{\mathbf{p}} = \mathbf{v}$ and then you get:

$$m\dot{\mathbf{v}} = \mathbf{u} - \alpha\mathbf{v}$$

- We discretize by setting

$$\frac{\mathbf{p}_{t+1} - \mathbf{p}_t}{\delta t} \simeq \mathbf{v}_t \qquad\qquad m\frac{\mathbf{v}_{t+1} - \mathbf{v}_t}{\delta t} \simeq \mathbf{u}_t - \alpha\mathbf{v}_t$$

LQR example #1: omnidirectional vehicle with friction

$$\frac{\mathbf{p}_{t+1} - \mathbf{p}_t}{\delta t} \simeq \mathbf{v}_t \qquad m\frac{\mathbf{v}_{t+1} - \mathbf{v}_t}{\delta t} \simeq \mathbf{u}_t - \alpha \mathbf{v}_t$$

- Define the state vector $\mathbf{x}_t = \begin{bmatrix} \mathbf{p}_t \\ \mathbf{v}_t \end{bmatrix}$

Q: How can we express this as a linear system?

LQR example #1: omnidirectional vehicle with friction

$$\frac{\mathbf{p}_{t+1} - \mathbf{p}_t}{\delta t} \simeq \mathbf{v}_t \qquad\qquad m\frac{\mathbf{v}_{t+1} - \mathbf{v}_t}{\delta t} \simeq \mathbf{u}_t - \alpha \mathbf{v}_t$$

- Define the state vector $\mathbf{x}_t = \begin{bmatrix} \mathbf{p}_t \\ \mathbf{v}_t \end{bmatrix}$

$$\mathbf{x}_{t+1} = \begin{bmatrix} \mathbf{p}_{t+1} \\ \mathbf{v}_{t+1} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_t + \delta t \mathbf{v}_t \\ \mathbf{v}_t + \frac{\delta t}{m}\mathbf{u}_t - \frac{\alpha \delta t}{m}\mathbf{v}_t \end{bmatrix} = \begin{bmatrix} \mathbf{p}_t + \delta t \mathbf{v}_t \\ \mathbf{v}_t - \frac{\alpha \delta t}{m}\mathbf{v}_t \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\delta t}{m} & 0 \\ 0 & \frac{\delta t}{m} \end{bmatrix} \mathbf{u}_t$$

LQR example #1: omnidirectional vehicle with friction

$$\frac{\mathbf{p}_{t+1} - \mathbf{p}_t}{\delta t} \simeq \mathbf{v}_t \qquad\qquad m\frac{\mathbf{v}_{t+1} - \mathbf{v}_t}{\delta t} \simeq \mathbf{u}_t - \alpha \mathbf{v}_t$$

- Define the state vector $\mathbf{x}_t = \begin{bmatrix} \mathbf{p}_t \\ \mathbf{v}_t \end{bmatrix}$

$$\mathbf{x}_{t+1} = \begin{bmatrix} \mathbf{p}_{t+1} \\ \mathbf{v}_{t+1} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_t + \delta t \mathbf{v}_t \\ \mathbf{v}_t + \frac{\delta t}{m}\mathbf{u}_t - \frac{\alpha \delta t}{m}\mathbf{v}_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 - \alpha\delta t/m & 0 \\ 0 & 0 & 0 & 1 - \alpha\delta t/m \end{bmatrix} \mathbf{x}_t + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\delta t}{m} & 0 \\ 0 & \frac{\delta t}{m} \end{bmatrix} \mathbf{u}_t$$

LQR example #1: omnidirectional vehicle with friction

$$\frac{\mathbf{p}_{t+1} - \mathbf{p}_t}{\delta t} \simeq \mathbf{v}_t \qquad\qquad m\frac{\mathbf{v}_{t+1} - \mathbf{v}_t}{\delta t} \simeq \mathbf{u}_t - \alpha \mathbf{v}_t$$

- Define the state vector $$\mathbf{x}_t = \begin{bmatrix} \mathbf{p}_t \\ \mathbf{v}_t \end{bmatrix}$$

$$\mathbf{x}_{t+1} = \begin{bmatrix} \mathbf{p}_{t+1} \\ \mathbf{v}_{t+1} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_t + \delta t \mathbf{v}_t \\ \mathbf{v}_t + \frac{\delta t}{m}\mathbf{u}_t - \frac{\alpha \delta t}{m}\mathbf{v}_t \end{bmatrix} = \overset{\textbf{A}}{\begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 - \alpha\delta t/m & 0 \\ 0 & 0 & 0 & 1 - \alpha \delta t/m \end{bmatrix}} \mathbf{x}_t + \overset{\textbf{B}}{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\delta t}{m} & 0 \\ 0 & \frac{\delta t}{m} \end{bmatrix}} \mathbf{u}_t$$

LQR example #1: omnidirectional vehicle with friction

- Define the state vector $\mathbf{x}_t = \begin{bmatrix} \mathbf{p}_t \\ \mathbf{v}_t \end{bmatrix}$

$$\mathbf{x}_{t+1} = \begin{bmatrix} \mathbf{p}_{t+1} \\ \mathbf{v}_{t+1} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_t + \delta t \mathbf{v}_t \\ \mathbf{v}_t + \frac{\delta t}{m}\mathbf{u}_t - \frac{\alpha \delta t}{m}\mathbf{v}_t \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 - \alpha\delta t/m & 0 \\ 0 & 0 & 0 & 1 - \alpha\delta t/m \end{bmatrix}}_{\mathbf{A}} \mathbf{x}_t + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\delta t}{m} & 0 \\ 0 & \frac{\delta t}{m} \end{bmatrix}}_{\mathbf{B}} \mathbf{u}_t$$

- Define the instantaneous cost function

$$\begin{aligned} c(\mathbf{x}, \mathbf{u}) &= \mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} \\ &= \mathbf{x}^T \mathbf{x} + \rho \mathbf{u}^T \mathbf{u} \\ &= ||\mathbf{x}||^2 + \rho ||\mathbf{u}||^2 \end{aligned}$$

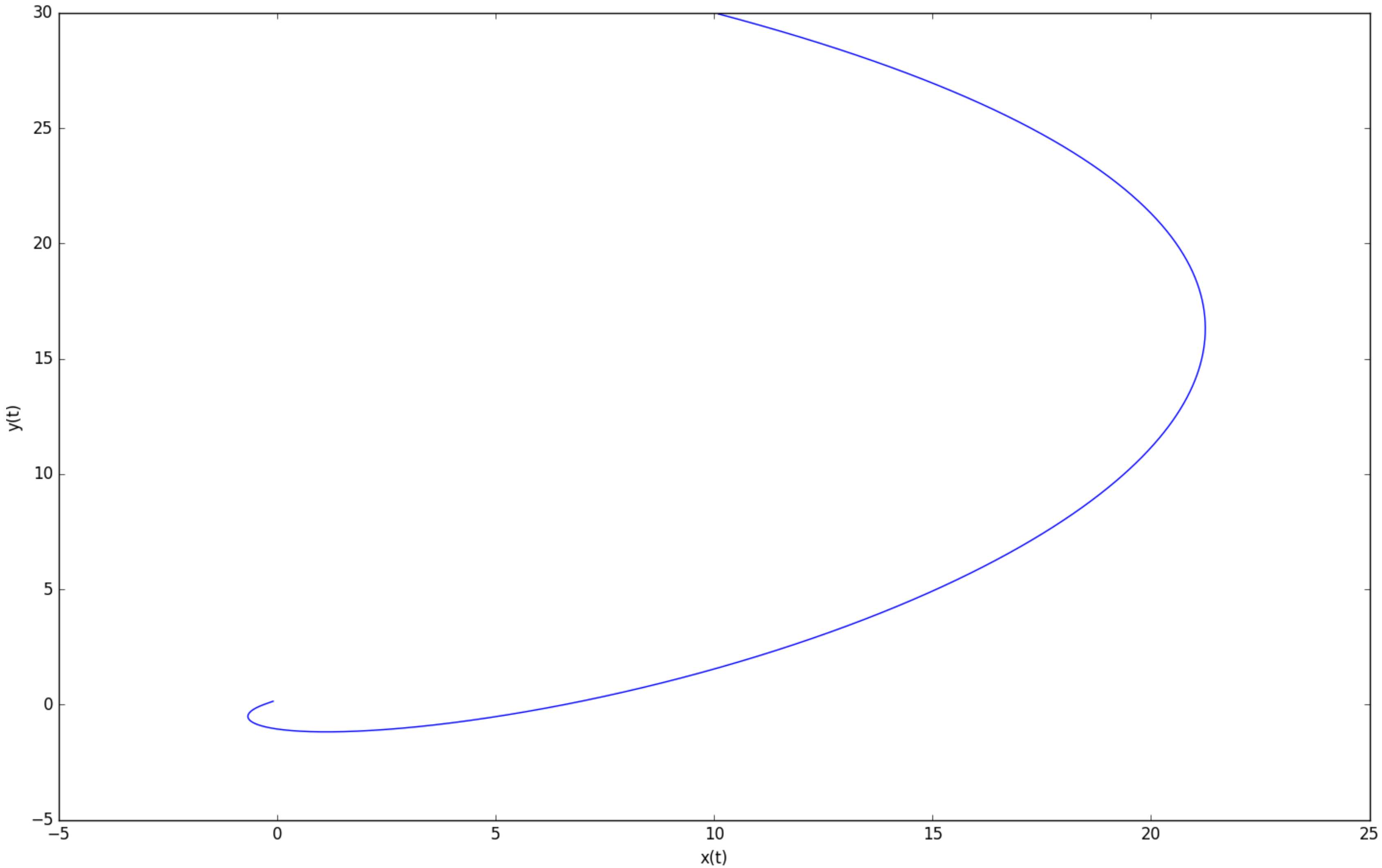
LQR example #1: omnidirectional vehicle with friction

With initial state

$$\mathbf{x}_0 = \begin{bmatrix} 10 \\ 30 \\ 10 \\ -5 \end{bmatrix}$$

Instantaneous cost function

$$c(\mathbf{x}, \mathbf{u}) = ||\mathbf{x}||^2 + 100||\mathbf{u}||^2$$



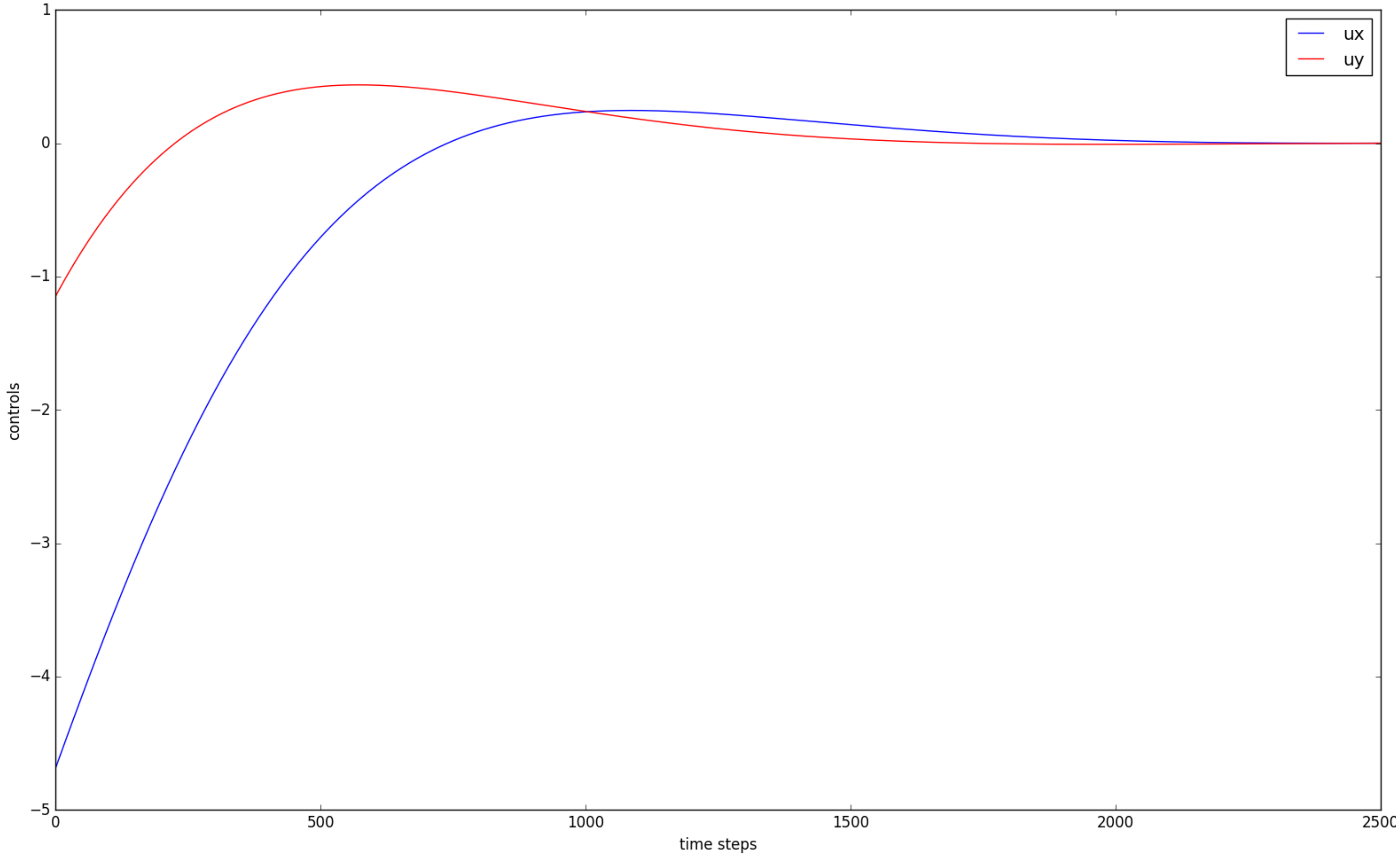
LQR example #1:
omnidirectional vehicle with friction

With initial state

$$\mathbf{x}_0 = \begin{bmatrix} 10 \\ 30 \\ 10 \\ -5 \end{bmatrix}$$

Instantaneous cost function

$$c(\mathbf{x}, \mathbf{u}) = ||\mathbf{x}||^2 + 100||\mathbf{u}||^2$$



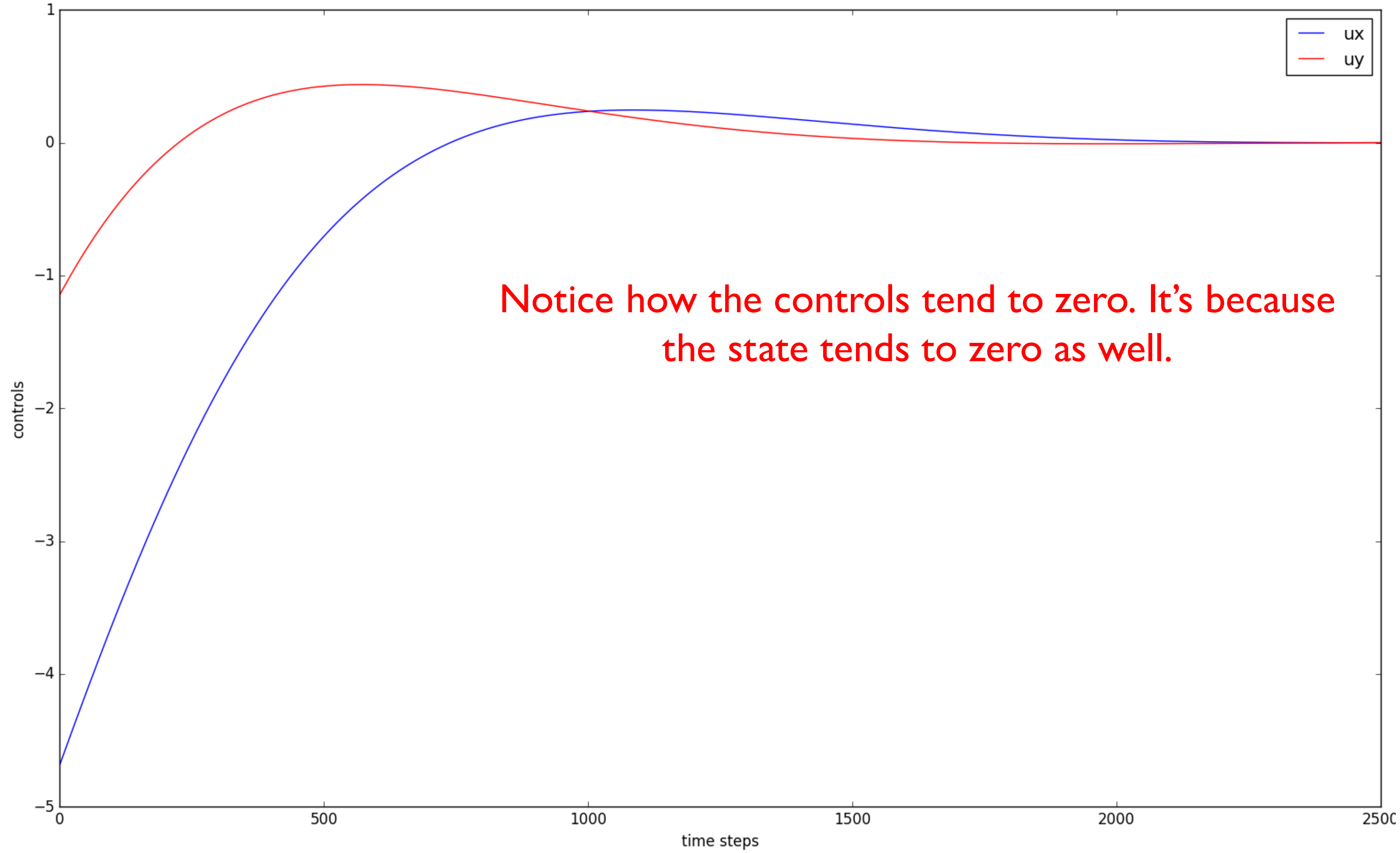
LQR example #1: omnidirectional vehicle with friction

With initial state

$$\mathbf{x}_0 = \begin{bmatrix} 10 \\ 30 \\ 10 \\ -5 \end{bmatrix}$$

Instantaneous cost function

$$c(\mathbf{x}, \mathbf{u}) = ||\mathbf{x}||^2 + 100||\mathbf{u}||^2$$



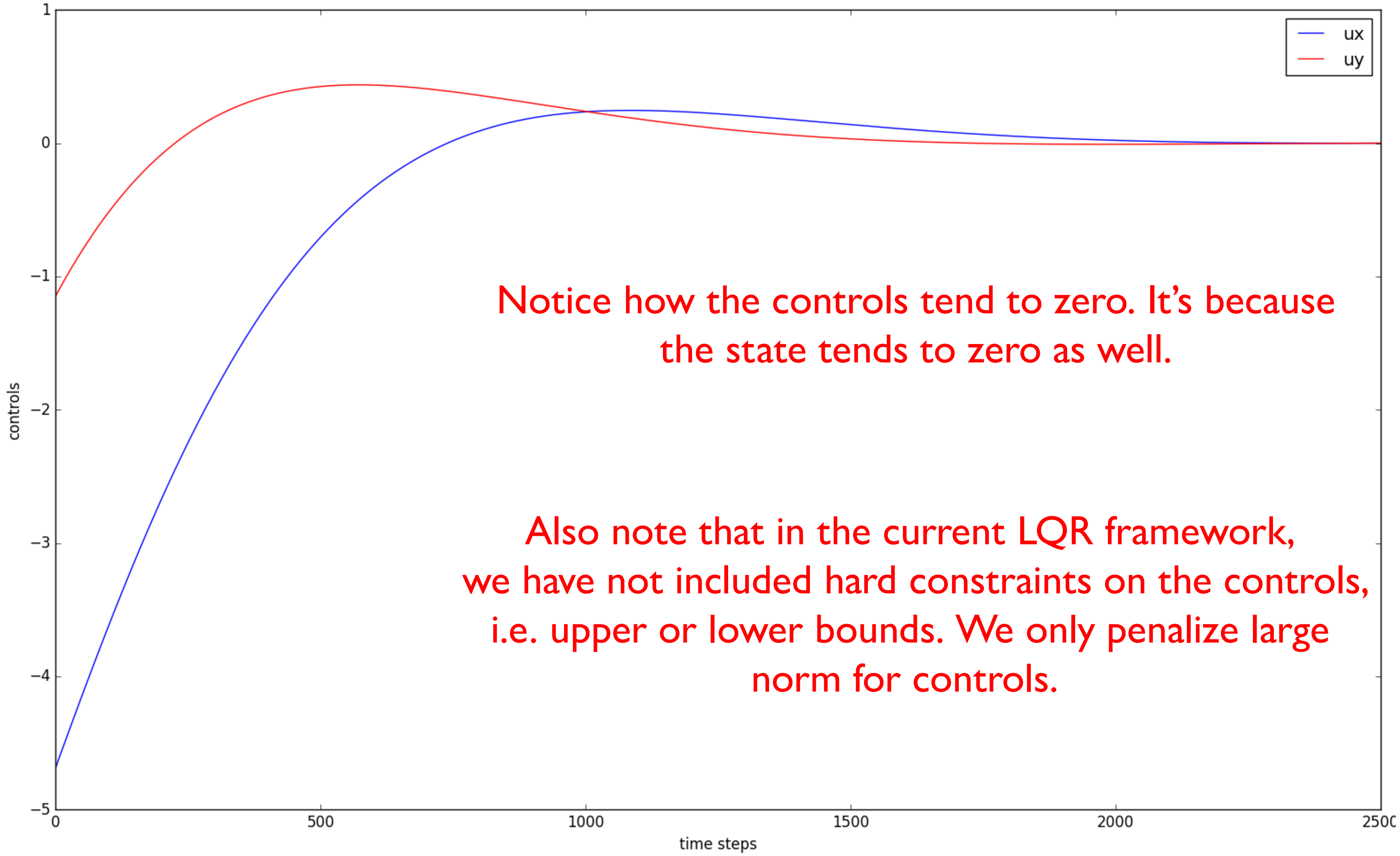
LQR example #1: omnidirectional vehicle with friction

With initial state

$$\mathbf{x}_0 = \begin{bmatrix} 10 \\ 30 \\ 10 \\ -5 \end{bmatrix}$$

Instantaneous cost function

$$c(\mathbf{x}, \mathbf{u}) = ||\mathbf{x}||^2 + 100||\mathbf{u}||^2$$



Notice how the controls tend to zero. It's because the state tends to zero as well.

Also note that in the current LQR framework, we have not included hard constraints on the controls, i.e. upper or lower bounds. We only penalize large norm for controls.

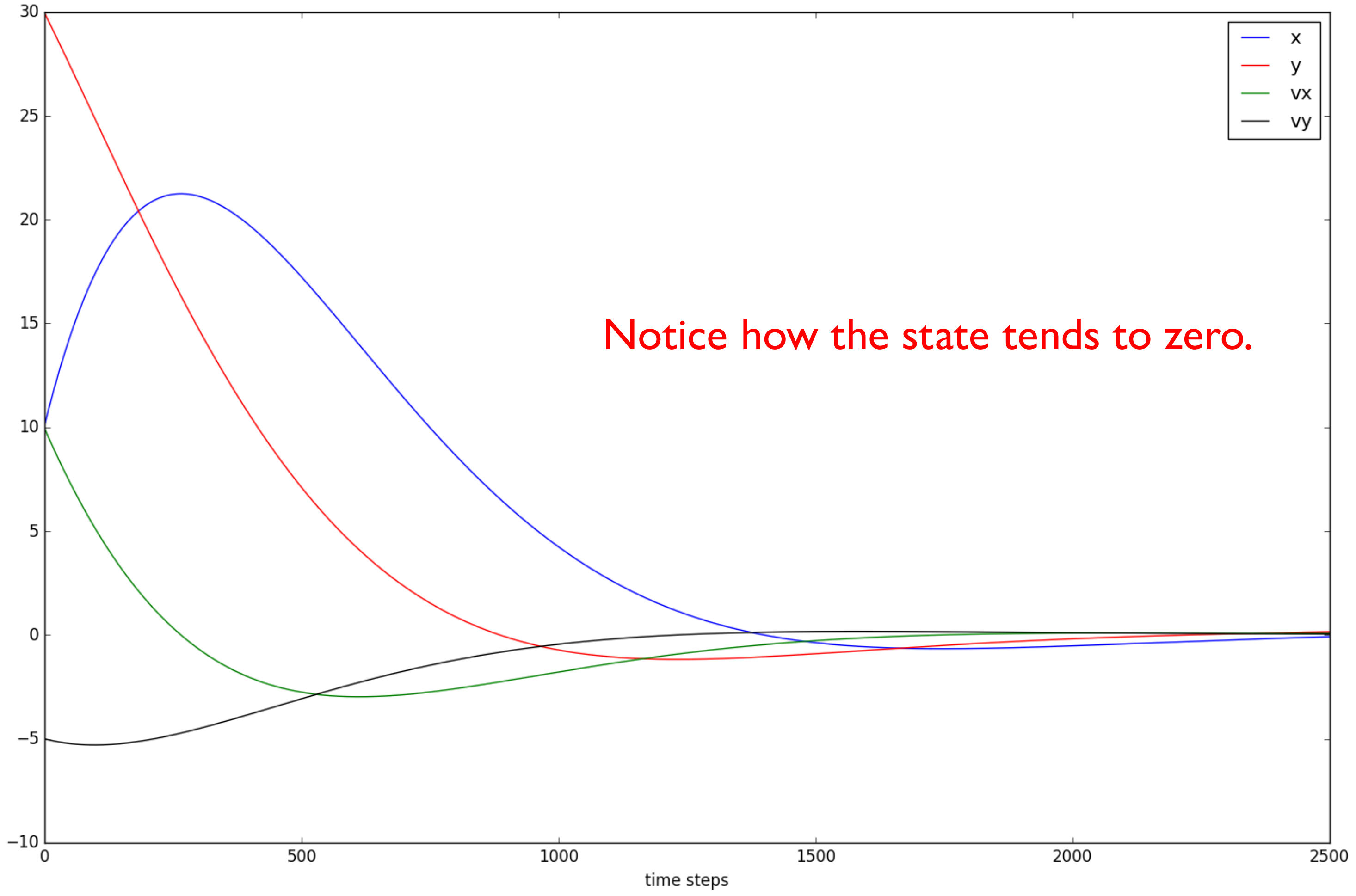
LQR example #1: omnidirectional vehicle with friction

With initial state

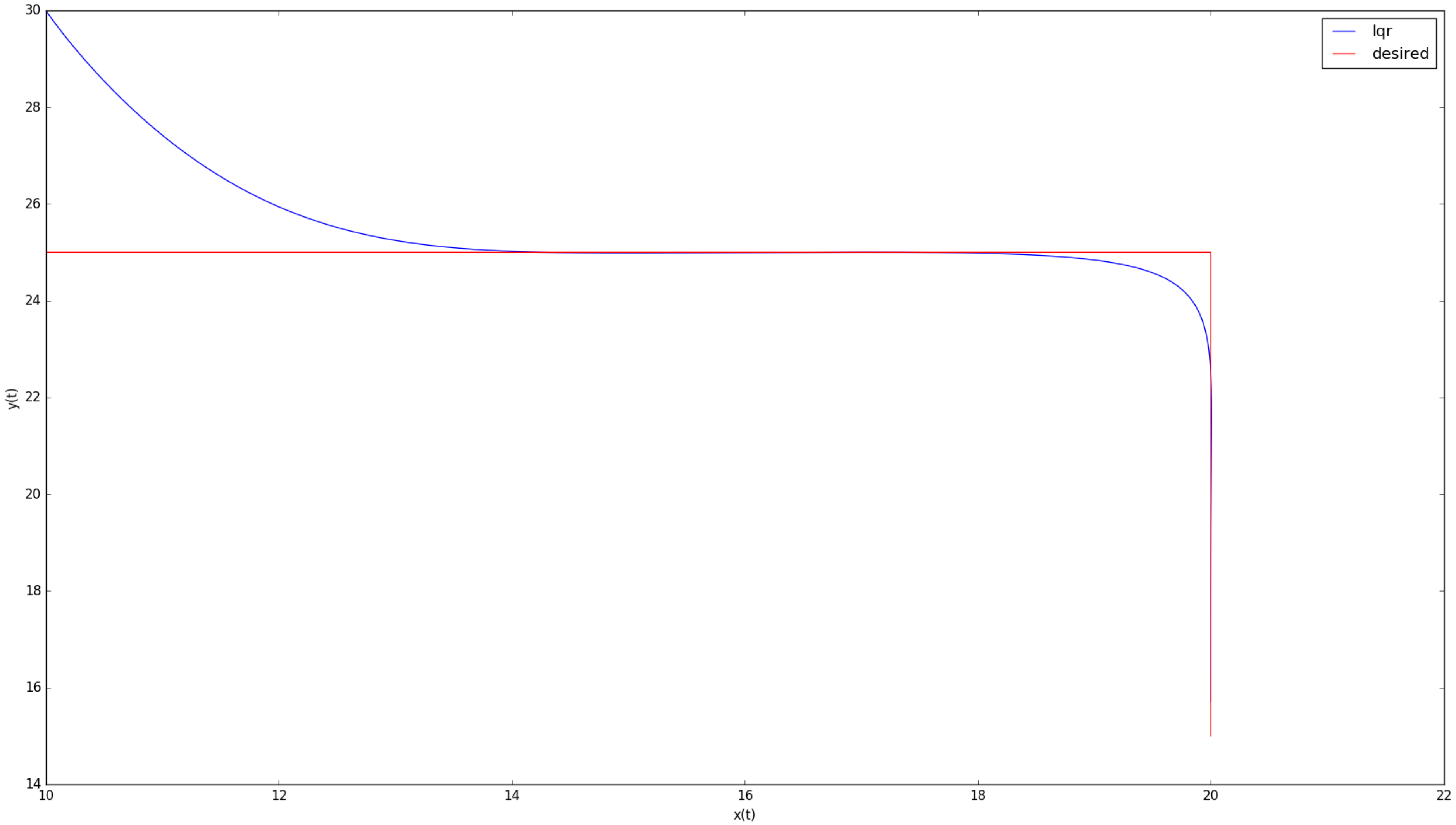
$$\mathbf{x}_0 = \begin{bmatrix} 10 \\ 30 \\ 10 \\ -5 \end{bmatrix}$$

Instantaneous cost function

$$c(\mathbf{x}, \mathbf{u}) = ||\mathbf{x}||^2 + 100||\mathbf{u}||^2$$



LQR example #2: trajectory following for omnidirectional vehicle



LQR example #2: trajectory following for omnidirectional vehicle

$$\mathbf{x}_{t+1} = \begin{bmatrix} \mathbf{p}_{t+1} \\ \mathbf{v}_{t+1} \end{bmatrix} = \overbrace{\begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 - \alpha \delta t / m & 0 \\ 0 & 0 & 0 & 1 - \alpha \delta t / m \end{bmatrix}}^{\mathbf{A}} \mathbf{x}_t + \overbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\delta t}{m} & 0 \\ 0 & \frac{\delta t}{m} \end{bmatrix}}^{\mathbf{B}} \mathbf{u}_t$$

We are given a desired trajectory $\mathbf{p}_0^*, \mathbf{p}_1^*, ..., \mathbf{p}_T^*$

Instantaneous cost $c(\mathbf{x}_t, \mathbf{u}_t) = (\mathbf{p}_t - \mathbf{p}_t^*)^T Q (\mathbf{p}_t - \mathbf{p}_t^*) + \mathbf{u}_t^T R \mathbf{u}_t$

LQR example #2: trajectory following for omnidirectional vehicle

$$\mathbf{x}_{t+1} = \begin{bmatrix} \mathbf{p}_{t+1} \\ \mathbf{v}_{t+1} \end{bmatrix} = \overbrace{\begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1-\alpha\delta t/m & 0 \\ 0 & 0 & 0 & 1-\alpha\delta t/m \end{bmatrix}}^{\mathbf{A}} \mathbf{x}_t + \overbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\delta t}{m} & 0 \\ 0 & \frac{\delta t}{m} \end{bmatrix}}^{\mathbf{B}} \mathbf{u}_t$$

Define

$$\begin{aligned} \bar{\mathbf{x}}_{t+1} &= \mathbf{x}_{t+1} - \mathbf{x}^*_{t+1} \\ &= A\mathbf{x}_t + B\mathbf{u}_t - \mathbf{x}^*_{t+1} \\ &= A\bar{\mathbf{x}}_t + B\mathbf{u}_t - \mathbf{x}^*_{t+1} + A\mathbf{x}^*_t \end{aligned}$$

We want $\bar{\mathbf{x}}_{t+1} = \bar{A}\bar{\mathbf{x}}_t + \bar{B}\mathbf{u}_t$

LQR example #2: trajectory following for omnidirectional vehicle

$$\mathbf{x}_{t+1} = \begin{bmatrix} \mathbf{p}_{t+1} \\ \mathbf{v}_{t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1-\alpha\delta t/m & 0 \\ 0 & 0 & 0 & 1-\alpha\delta t/m \end{bmatrix}}_{\textbf{A}} \mathbf{x}_t + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\delta t}{m} & 0 \\ 0 & \frac{\delta t}{m} \end{bmatrix}}_{\textbf{B}} \mathbf{u}_t$$

Define
$$\begin{aligned} \bar{\mathbf{x}}_{t+1} &= \mathbf{x}_{t+1} - \mathbf{x}^*_{t+1} \\ &= A\mathbf{x}_t + B\mathbf{u}_t - \mathbf{x}^*_{t+1} \\ &= A\bar{\mathbf{x}}_t + B\mathbf{u}_t - \mathbf{x}^*_{t+1} + A\mathbf{x}^*_t \end{aligned}$$

Need to get rid of this additive term

We want $\bar{\mathbf{x}}_{t+1} = \bar{A}\bar{\mathbf{x}}_t + \bar{B}\mathbf{u}_t$

LQR example #2:
trajectory following for omnidirectional vehicle

$$\mathbf{x}_{t+1} = \begin{bmatrix}\mathbf{p}_{t+1}\\ \mathbf{v}_{t+1}\end{bmatrix} = \underbrace{\begin{bmatrix}1 & 0 & \delta t & 0\\ 0 & 1 & 0 & \delta t\\ 0 & 0 & 1-\alpha\delta t/m & 0\\ 0 & 0 & 0 & 1-\alpha\delta t/m\end{bmatrix}}_{\mathbf{A}} \mathbf{x}_t + \underbrace{\begin{bmatrix}0 & 0\\ 0 & 0\\ \frac{\delta t}{m} & 0\\ 0 & \frac{\delta t}{m}\end{bmatrix}}_{\mathbf{B}} \mathbf{u}_t$$

Define
$$\begin{aligned}\bar{\mathbf{x}}_{t+1} &= \mathbf{x}_{t+1} - \mathbf{x}^*_{t+1}\\ &= A\mathbf{x}_t + B\mathbf{u}_t - \mathbf{x}^*_{t+1}\\ &= A\bar{\mathbf{x}}_t + B\mathbf{u}_t \underbrace{- \mathbf{x}^*_{t+1} + A\mathbf{x}^*_t}_{\mathbf{c}}\end{aligned}$$

We want $\bar{\mathbf{x}}_{t+1} = \bar{A}\bar{\mathbf{x}}_t + \bar{B}\mathbf{u}_t$

Need to get rid of this additive term

Redefine state: $\mathbf{z}_{t+1} = \begin{bmatrix}\bar{\mathbf{x}}_{t+1}\\ 1\end{bmatrix} = \begin{bmatrix}A & c\\ 0 & 1\end{bmatrix}\begin{bmatrix}\bar{\mathbf{x}}_t\\ 1\end{bmatrix} + \begin{bmatrix}B\\ 0\end{bmatrix}\mathbf{u}_t = \bar{A}\mathbf{z}_t + \bar{B}\mathbf{u}_t$

LQR example #2: trajectory following for omnidirectional vehicle

$$\mathbf{x}_{t+1} = \begin{bmatrix} \mathbf{p}_{t+1} \\ \mathbf{v}_{t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1-\alpha\delta t/m & 0 \\ 0 & 0 & 0 & 1-\alpha\delta t/m \end{bmatrix}}_{A} \mathbf{x}_t + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{\delta t}{m} & 0 \\ 0 & \frac{\delta t}{m} \end{bmatrix}}_{B} \mathbf{u}_t$$

Define
$$\begin{aligned} \bar{\mathbf{x}}_{t+1} &= \mathbf{x}_{t+1} - \mathbf{x}^*_{t+1} \\ &= A\mathbf{x}_t + B\mathbf{u}_t - \mathbf{x}^*_{t+1} \\ &= A\bar{\mathbf{x}}_t + B\mathbf{u}_t \underbrace{- \mathbf{x}^*_{t+1} + A\mathbf{x}^*_t}_{c} \end{aligned}$$

We want $\bar{\mathbf{x}}_{t+1} = \bar{A}\bar{\mathbf{x}}_t + \bar{B}\mathbf{u}_t$

Need to get rid of this additive term

Idea: augment the state

Redefine state: $\mathbf{z}_{t+1} = \begin{bmatrix} \bar{\mathbf{x}}_{t+1} \\ 1 \end{bmatrix} = \begin{bmatrix} A & c \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{\mathbf{x}}_t \\ 1 \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} \mathbf{u}_t = \bar{A}\mathbf{z}_t + \bar{B}\mathbf{u}_t$

Redefine cost function: $c(\mathbf{z}_t, \mathbf{u}_t) = \mathbf{z}_t^T \bar{Q} \mathbf{z}_t + \mathbf{u}_t^T R \mathbf{u}_t$

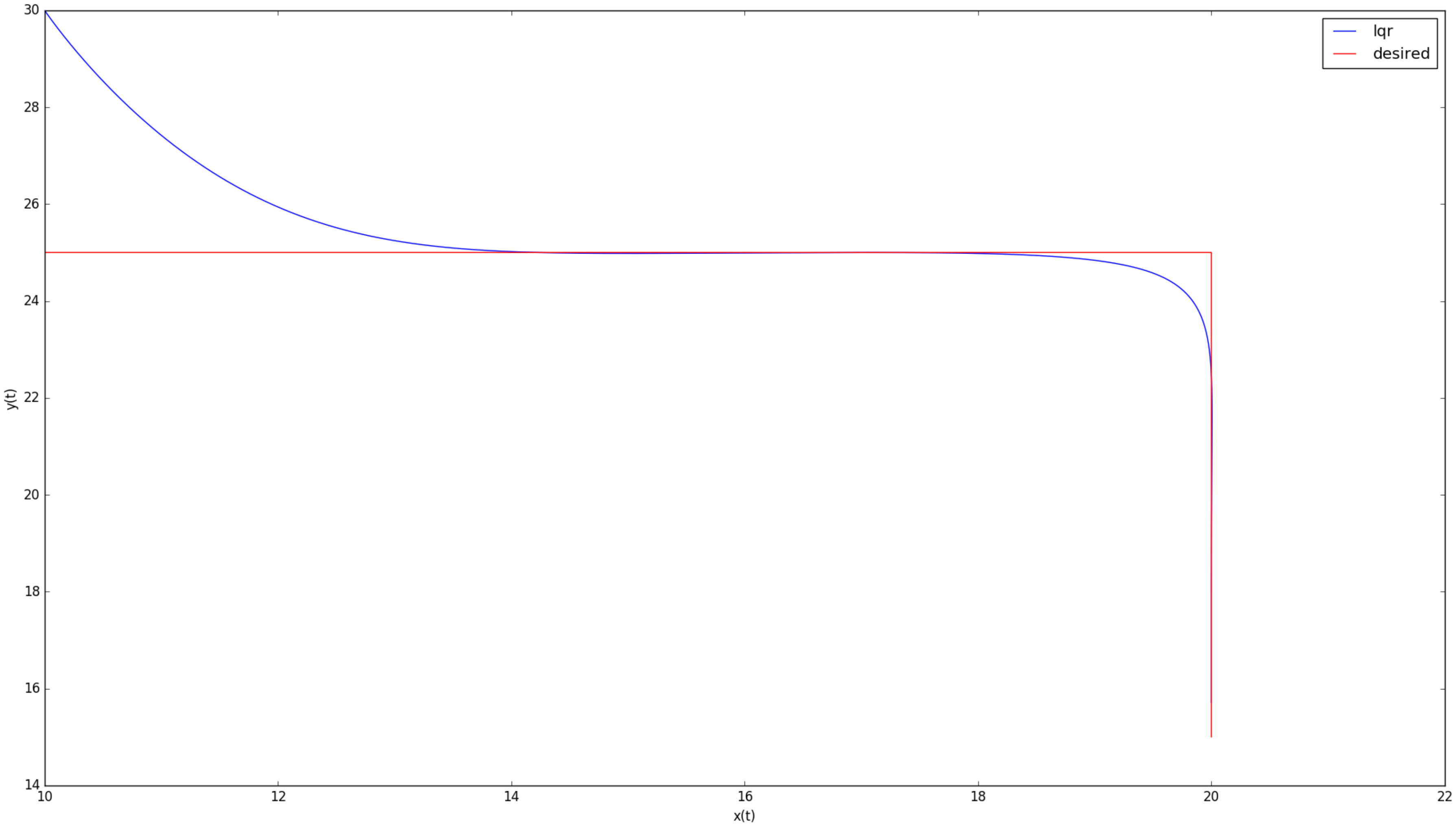
LQR example #2: trajectory following for omnidirectional vehicle

With initial state

$$\mathbf{z}_0 = \begin{bmatrix} 10 \\ 30 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Instantaneous cost function

$$c(\mathbf{z}, \mathbf{u}) = ||\mathbf{z}||^2 + ||\mathbf{u}||^2$$



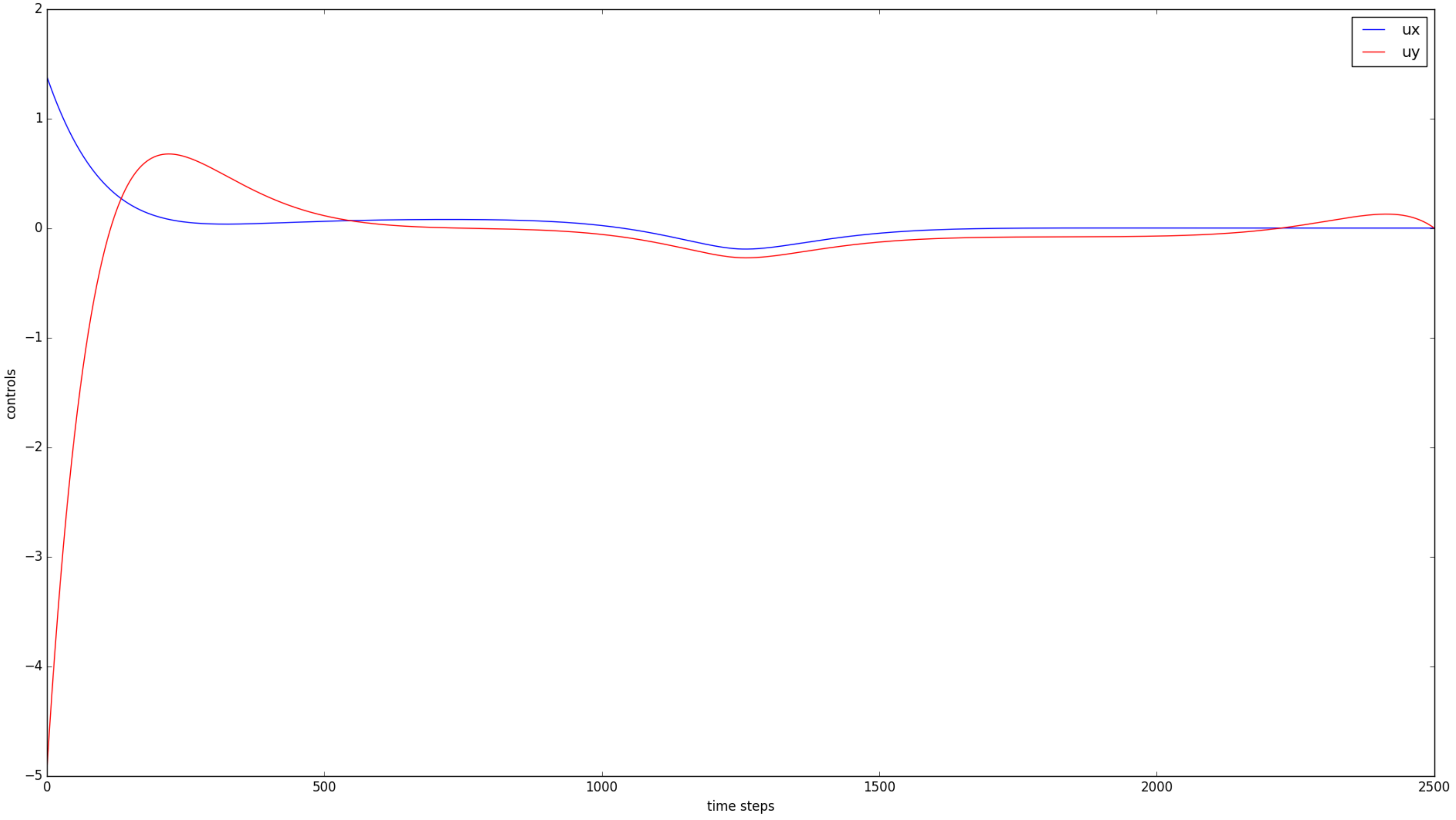
LQR example #2: trajectory following for omnidirectional vehicle

With initial state

$$\mathbf{z}_0 = \begin{bmatrix} 10 \\ 30 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Instantaneous cost function

$$c(\mathbf{z}, \mathbf{u}) = ||\mathbf{z}||^2 + ||\mathbf{u}||^2$$



Appendix #4 (optional reading) LQR extensions: trajectory following

- You are given a reference trajectory (not just path, but states and times, or states and controls) that needs to be approximated

$$\mathbf{x}_0^*, \mathbf{x}_1^*, \ldots, \mathbf{x}_N^* \qquad \mathbf{u}_0^*, \mathbf{u}_1^*, \ldots, \mathbf{u}_N^*$$

Linearize the nonlinear dynamics $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$ around the reference point $(\mathbf{x}_t^*, \mathbf{u}_t^*)$

$$\mathbf{x}_{t+1} \simeq f(\mathbf{x}_t^*, \mathbf{u}_t^*) + \frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}_t^*, \mathbf{u}_t^*)(\mathbf{x}_t - \mathbf{x}_t^*) + \frac{\partial f}{\partial \mathbf{u}}(\mathbf{x}_t^*, \mathbf{u}_t^*)(\mathbf{u}_t - \mathbf{u}_t^*)$$

$$\bar{\mathbf{x}}_{t+1} \simeq A_t \bar{\mathbf{x}}_t + B_t \bar{\mathbf{u}}_t$$
$$c(\mathbf{x}_t, \mathbf{u}_t) = \bar{\mathbf{x}}_t^T Q \bar{\mathbf{x}}_t + \bar{\mathbf{u}}_t^T R \bar{\mathbf{u}}_t$$

where

$$\bar{\mathbf{x}}_t = \mathbf{x}_t - \mathbf{x}_t^*$$
$$\bar{\mathbf{u}}_t = \mathbf{u}_t - \mathbf{u}_t^*$$

Trajectory following can be implemented as a time-varying LQR approximation. Not always clear if this is the best way though.

Appendix #5 (optional reading)
LQR with nonlinear dynamics, quadratic cost

LQR variants: nonlinear dynamics, quadratic cost

What can we do when $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$ but the cost is quadratic $c(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t^T Q \mathbf{x}_t + \mathbf{u}_t^T R \mathbf{u}_t$?

We want to stabilize the system around state $\mathbf{x}_t = \mathbf{0}$
But with nonlinear dynamics we do not know if $\mathbf{u}_t = \mathbf{0}$ will keep the system at the zero state.

LQR variants: nonlinear dynamics, quadratic cost

What can we do when $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$ but the cost is quadratic $c(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t^T Q \mathbf{x}_t + \mathbf{u}_t^T R \mathbf{u}_t$?

We want to stabilize the system around state $\mathbf{x}_t = \mathbf{0}$
But with nonlinear dynamics we do not know if $\mathbf{u}_t = \mathbf{0}$ will keep the system at the zero state.

→ Need to compute $\mathbf{u}^*$ such that $\mathbf{0}_{t+1} = f(\mathbf{0}_t, \mathbf{u}^*)$

LQR variants: nonlinear dynamics, quadratic cost

What can we do when $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$ but the cost is quadratic $c(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t^T Q \mathbf{x}_t + \mathbf{u}_t^T R \mathbf{u}_t$?

We want to stabilize the system around state $\mathbf{x}_t = \mathbf{0}$
But with nonlinear dynamics we do not know if $\mathbf{u}_t = \mathbf{0}$ will keep the system at the zero state.

→ Need to compute $\mathbf{u}^*$ such that $\mathbf{0}_{t+1} = f(\mathbf{0}_t, \mathbf{u}^*)$

Taylor expansion: linearize the nonlinear dynamics around the point $(\mathbf{0}, \mathbf{u}^*)$

$$\mathbf{x}_{t+1} \simeq f(\mathbf{0}, \mathbf{u}^*) + \underbrace{\frac{\partial f}{\partial \mathbf{x}}(\mathbf{0}, \mathbf{u}^*)}_{A}(\mathbf{x}_t - \mathbf{0}) + \underbrace{\frac{\partial f}{\partial \mathbf{u}}(\mathbf{0}, \mathbf{u}^*)}_{B}(\mathbf{u}_t - \mathbf{u}^*)$$

LQR variants: nonlinear dynamics, quadratic cost

What can we do when $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$ but the cost is quadratic $c(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t^T Q \mathbf{x}_t + \mathbf{u}_t^T R \mathbf{u}_t$?

We want to stabilize the system around state $\mathbf{x}_t = \mathbf{0}$
But with nonlinear dynamics we do not know if $\mathbf{u}_t = \mathbf{0}$ will keep the system at the zero state.

→ Need to compute $\mathbf{u}^*$ such that $\mathbf{0}_{t+1} = f(\mathbf{0}_t, \mathbf{u}^*)$

Taylor expansion: linearize the nonlinear dynamics around the point $(\mathbf{0}, \mathbf{u}^*)$

$$\mathbf{x}_{t+1} \simeq f(\mathbf{0}, \mathbf{u}^*) + \frac{\partial f}{\partial \mathbf{x}}(\mathbf{0}, \mathbf{u}^*)(\mathbf{x}_t - \mathbf{0}) + \frac{\partial f}{\partial \mathbf{u}}(\mathbf{0}, \mathbf{u}^*)(\mathbf{u}_t - \mathbf{u}^*)$$

$$\mathbf{x}_{t+1} \simeq A\mathbf{x}_t + B(\mathbf{u}_t - \mathbf{u}^*)$$

Solve this via LQR

LQR examples:
code to replicate these results

- https://github.com/florianshkurti/comp417.git

- Look under comp417/lqr_examples/python