L11: Optimal Control

Hao Su

Spring, 2021

Contents are based on Underactuated Robotics taught at MIT by Prof. Russ Tedrake and CS287 taught at UC Berkeley by Prof. Pieter Abbeel.

Agenda

- Fully-actuated v.s. Under-actuated Systems
- Basic Idea of Optimal Control
- Linear Quadratic Control

click to jump to the section.

Fully-actuated v.s. Under-actuated Systems

Review: Control

- A *desired* trajectory to follow: $(\boldsymbol{q}_d, \dot{\boldsymbol{q}}_d, \ddot{\boldsymbol{q}}_d)$
- Forward dynamics $\ddot{\boldsymbol{q}} = \mathrm{FD}(\boldsymbol{F}; \boldsymbol{q}, \dot{\boldsymbol{q}})$
- Inverse dynamics $\boldsymbol{F} = \mathrm{ID}(\ddot{\boldsymbol{q}}; \boldsymbol{q}, \dot{\boldsymbol{q}})$
- We use **control** to deal with delay, overshoot, or steady-state error, and ensure stability.
- Steady-state error

$$e = q - q_d$$

Review: Feedforward and Feedback Control

- We need some force to match $\ddot{\boldsymbol{q}}_d$. This component is called the **feed-forward component**, which comes from $\mathrm{ID}(\cdot)$:

$$\boldsymbol{F}_{ff} = \mathrm{ID}(\ddot{\boldsymbol{q}}_{\mathrm{d}}; \boldsymbol{q}, \dot{\boldsymbol{q}})$$

- We also need some additional force to correct the steady-state error, which is called the **feedback component**:

$$\boldsymbol{F}_{fb} = M(\boldsymbol{q})(-K_v\dot{e} - K_p e)$$

  where $M(\boldsymbol{q})$ is the inertia of the system.
- The total force we exert to control the system is

$$\boldsymbol{F} = \boldsymbol{F}_{ff} + \boldsymbol{F}_{fb}$$

Review: Computed Torque Control

- Computed Torque Control:

$$\tau = M(\theta)(\ddot{\theta}_d - K_v\dot{e} - K_p e) + C(\theta, \dot{\theta})\dot{\theta} + g(\theta) \tag{1}$$

- By ID, the acceleration under this τ is

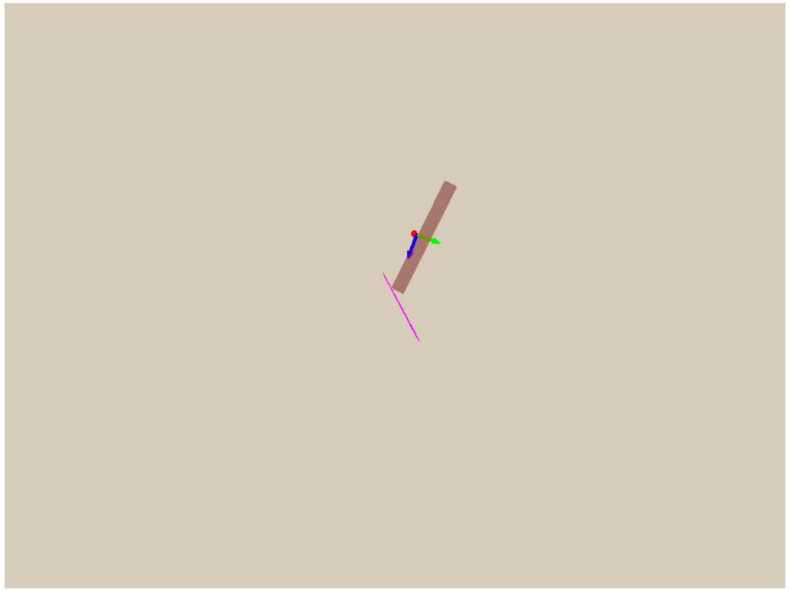
$$\tau = M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + g(\theta) \tag{2}$$

- Subtracting (2) from (1) and cancel $M(\theta)$, we get the error equation:

$$\ddot{e} + K_v\dot{e} + K_p e = 0$$

- Because $K_v, K_p \in \mathbb{S}^+$, by the theory of ODE, $e(t) = \mathcal{O}(e^{\alpha t})$, $\alpha \leq 0$.
- We say that the computed torque control law has **exponential convergence rate**.

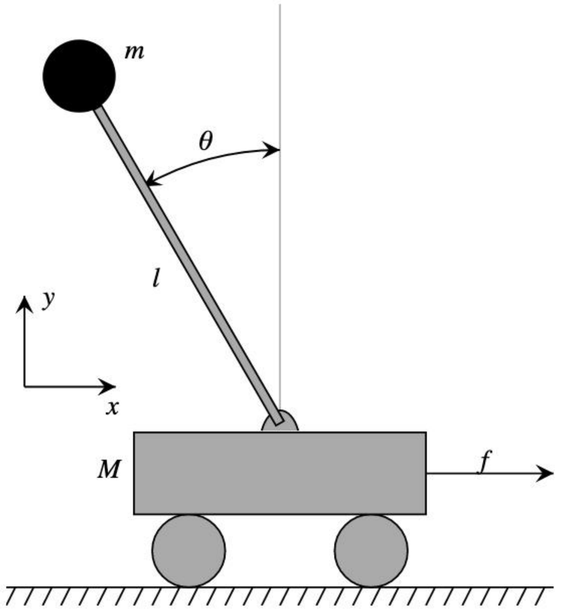
Example of Single-Joint Manipulator (Inverted Pendulum)



Using feedback control (PID), it is very easy to control this swing-up inverted pendulum.

Limitation of PID Control

- Consider the cart-pole example.
- Our desired position is that the rod is upright straight.
- Can we control it by the computed torque law?



A schematic drawing of the inverted pendulum on a cart. The rod is considered massless.

https://en.wikipedia.org/wiki/Inverted_pendulum

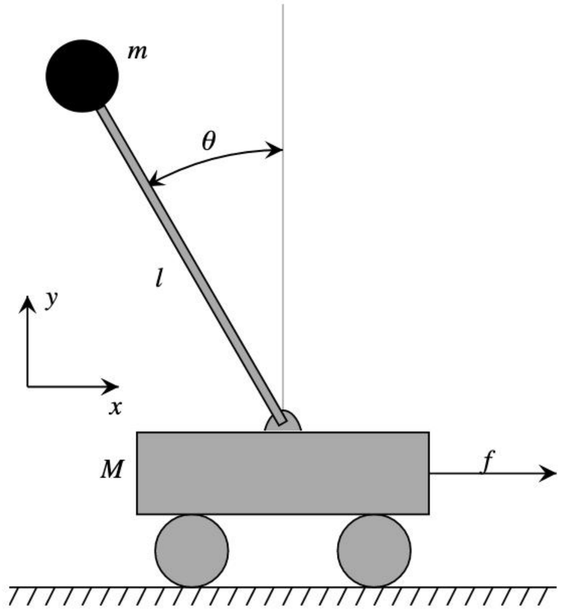
Limitation of PID Control

- Consider the cart-pole example.
- Our desired position is that the rod is upright straight.
- Can we control it by the computed torque law?

- Let us try the feedforward-feedback control law.
- First of all, we can only control the 1D force f. It is also obvious that $f_{feedforward} = 0$.
- The feedback force should be

$$f_{feedback} = M(q)(-K_v \dot{e} - K_p e)$$

- Note that $e = q - q_d = \begin{bmatrix} x \\ \theta \end{bmatrix} - \begin{bmatrix} x_d \\ \theta_d \end{bmatrix}$, thus the feedback force should be 2D. But we can only control the 1D force!
- Our convergence analysis does not apply here.
- Turns out that tuning the PID for cart pole is hard.



A schematic drawing of the inverted pendulum on a cart. The rod is considered massless.

https://en.wikipedia.org/wiki/Inverted_pendulum

Underactuated Control Differential Equations

- Notions:
  - **Second-order** control dynamical system: $\ddot{\boldsymbol{q}} = f(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}, t)$
  - Control vector: $\boldsymbol{u} \in \mathcal{U}$
- **Underactuated Control Differential Equations**:

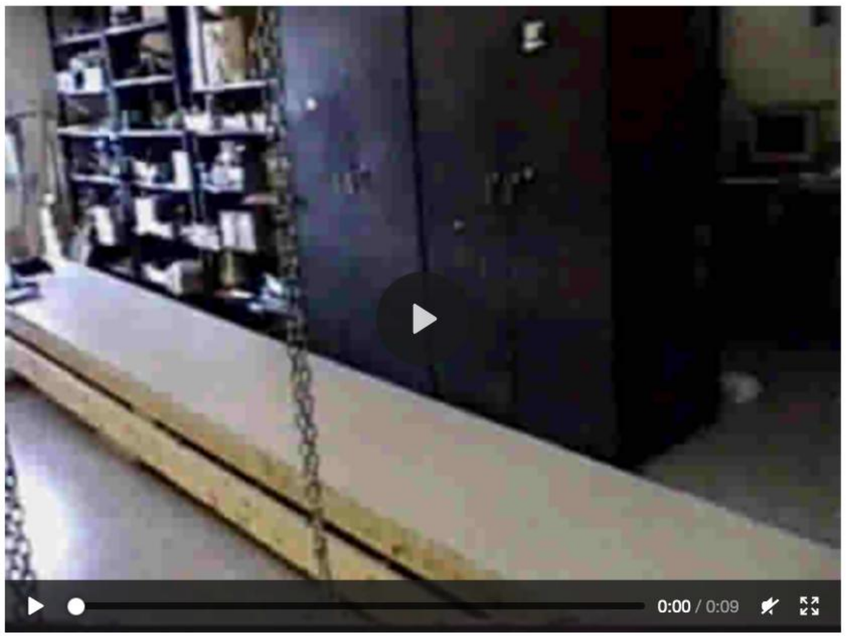
  A second-order control differential equation described by the equations

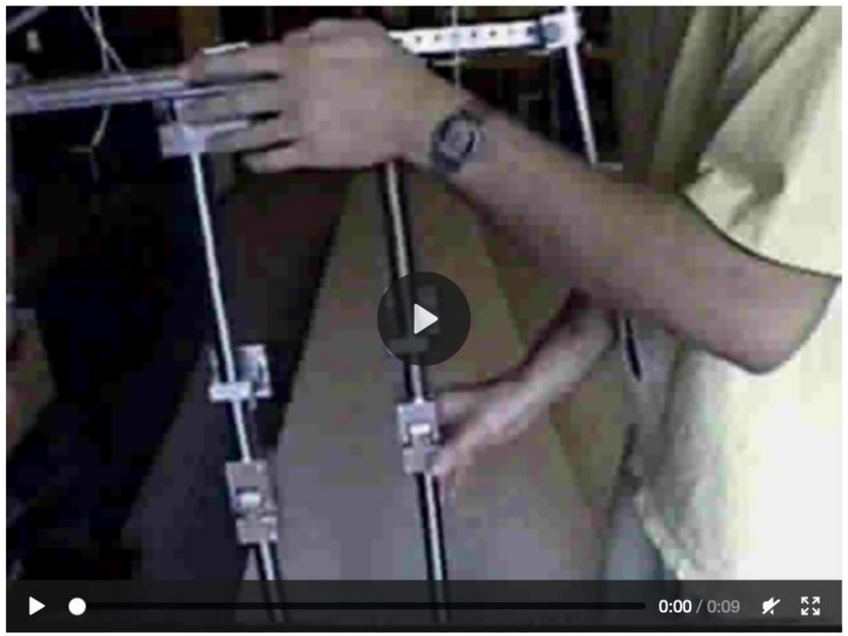
  $$\ddot{\boldsymbol{q}} = f(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}, t)$$

  *is **fully actuated** in state $\boldsymbol{x} = (\boldsymbol{q}, \dot{\boldsymbol{q}})$ and time t if the resulting map f is surjective: for every $\ddot{\boldsymbol{q}}$ there exists a $\boldsymbol{u}$ which produces the desired response. Otherwise, it is **underactuated** (in $\boldsymbol{x}$ at time t).*

- For example, our cart pole system is underactuated (1-D f has to control $\ddot{x}$ and $\ddot{\theta}$).

The Power of Underactuated System: A Passive Dynamic Walker Example





A 3D passive dynamic walker by Steve Collins and Andy Ruina at Cornell.

Many *Interesting* Problems in Robotics are Underactuated

- **Legged robots are underactuated.** Consider a legged machine with N internal joints and N actuators. If the robot is not bolted to the ground, then the degrees of freedom of the system include both the internal joints and the six degrees of freedom which define the position and orientation of the robot in space. Since $\boldsymbol{u} \in \mathbb{R}^N$ and $\boldsymbol{q} \in \mathbb{R}^{N+6}$, the system is underactuated.
- **(Most) Swimming and flying robots are underactuated.** The story is the same here as for legged machines. Each control surface adds one actuator and one DOF. And this is already a simplification, as the true state of the system should really include the (infinite-dimensional) state of the flow.
- **Robot manipulation is (often) underactuated.** Consider a fully-actuated robotic arm. When this arm is manipulating an object with degrees of freedom (even a brick has six), it can become underactuated.

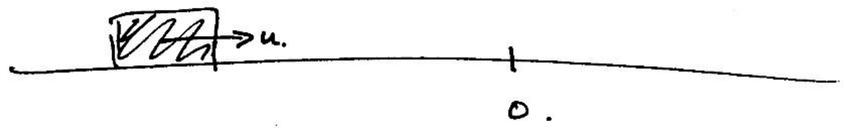
Read by Yourself

Concepts and Main Theoretical Results of Optimal Control

Control as Optimization Problem

- Key idea: **Design an optimization problem whose solution is the control signal**.
- Example
  - Consider the simple second-order control dynamics system:

$$\ddot{q} = u, \quad |u| \leq 1$$



credit: Sec 11 of Underactuated Robotics.

  - The task is to design a control system, $u = \pi(\boldsymbol{x}, t), \boldsymbol{x} = [q, \dot{q}]^T$ to regulate this brick to $\boldsymbol{x} = [0, 0]^T$.
  - Optimization problem:
    - Minimum time: $\min_\pi t_f$ subject to $\boldsymbol{x}(t_0) = \boldsymbol{x}_0, \boldsymbol{x}(t_f) = 0$.
    - Quadratic cost: $\min_\pi \int_{t_0}^{\infty} \boldsymbol{x}^T(t) \boldsymbol{Q} \boldsymbol{x}(t) \mathrm{d}t, \ \boldsymbol{Q} \succ 0$.

Transition Function

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \qquad \text{(transition function)}$$

- Example 1 — Brick:

$$\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u,$$

  which is of $\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}$ form.

- Example 2 — Manipulator :

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & I \\ 0 & M_\theta^{-1} \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ M_\theta^{-1} C_{\theta,\dot{\theta}} \end{bmatrix} \tau + \begin{bmatrix} 0 \\ -M_\theta^{-1} g_\theta \end{bmatrix},$$

  which is of $\dot{\boldsymbol{x}} = A_{\boldsymbol{x}}\boldsymbol{x} + B_{\boldsymbol{x}}\boldsymbol{u} + C_{\boldsymbol{x}}$ form. Let $\boldsymbol{x}'_{\boldsymbol{x}} = \boldsymbol{x} + A_{\boldsymbol{x}}^{-1} C_{\boldsymbol{x}}$, then $\dot{\boldsymbol{x}}' = A_{\boldsymbol{x}}\boldsymbol{x}' + B_{\boldsymbol{x}}\boldsymbol{u}$.

Transition Function

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \qquad \text{(transition function)}$$

- Understanding the linear transition function $\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}$ is fundamentally important, because *locally* it approximates any differentiable transitions.
  - For example, we do Taylor's expansion around stationary state $\boldsymbol{x}^*$, $0 = f(\boldsymbol{x}^*, \boldsymbol{u}^*)$.
    - Stationary state: both velocity and acceleration are 0.
  - Take $\boldsymbol{x}' = \boldsymbol{x} - \boldsymbol{x}^*$, $\boldsymbol{u}' = \boldsymbol{u} - \boldsymbol{u}_*$, plug in the transition function, and use Taylor's expansion to approximate f around $\boldsymbol{x}^*$ and $\boldsymbol{u}^*$, then

$$\dot{\boldsymbol{x}'} \approx \nabla_{\boldsymbol{x}} f(\boldsymbol{x}^*, \boldsymbol{u}^*)\boldsymbol{x}' + \nabla_{\boldsymbol{u}} f(\boldsymbol{x}^*, \boldsymbol{u}^*)\boldsymbol{u}'$$

Additive Cost

$$\int_0^T \ell(\boldsymbol{x}(t), \boldsymbol{u}(t))\mathrm{d}t \qquad \text{(additive cost)}$$

- For example, the quadratic cost for the brick example is an additive cost:

$$\min_{\pi} \int_{t_0}^{\infty} \boldsymbol{x}^T(t)\boldsymbol{Q}\boldsymbol{x}(t)\mathrm{d}t,\ \boldsymbol{Q} \succ 0.$$

- Additive cost is a favorable choice in optimal control, because
  - it admits an optimality condition of elegant form.
  - in discrete case, it also implies a "dynamic programming" solution (we will see in the next lecture of reinforcement learning).

Cost-to-go Function

- Consider a *time-invariant* dynamic system $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$ with an *infinite-horizon* additive cost $\int_0^\infty \ell(\boldsymbol{x}, \boldsymbol{u}) \mathrm{d}t$
  - Time-invariant: f and ℓ do not directly depend on t.
- Suppose that we will control the system using a **policy** $\boldsymbol{u} = \pi(\boldsymbol{x})$
- The **cost-to-go function** at a certain starting state $\boldsymbol{x}_0$ is:

$$J^\pi(\boldsymbol{x}_0) = \int_0^\infty \ell(\boldsymbol{x}(t), \boldsymbol{u}(t)) \mathrm{d}t \qquad \text{(cost-to-go function)}$$

  where $\boldsymbol{x}(0) = \boldsymbol{x}_0$ and $\boldsymbol{u}(t) = \pi(\boldsymbol{x}(t))$.
- You will see that the cost-to-go-function and the value function in reinforcement learning are essentially the same thing, except that one needs to be minimized and the other maximized.

The Hamilton-Jacobi-Bellman Equation (Optimality Condition)

- Consider a time-invariant system $\dot{x} = f(\boldsymbol{x}, \boldsymbol{u})$ with an additive cost $\int_0^\infty \ell(\boldsymbol{x}, \boldsymbol{u}) \mathrm{d}t$, in which $f, \ell \in C^\infty(\mathcal{X} \times \mathcal{U})$, $\mathrm{Hess}(\ell) \succ 0$, $\ell(x) = 0$ iff $\mathbf{x} = \boldsymbol{x}^*$.
- Under some technical conditions on the existence and boundedness of solutions (see *Thm 7.1 in Underactuated Robotics*), a sufficient condition for the existence of optimal policy is:

$$\exists J : \mathcal{X} \to \mathbb{R} \in \mathcal{C}^\infty(\mathcal{X}), \mathrm{Hess}(J) \succ 0 \text{ such that } \forall \boldsymbol{x} \in \mathcal{X},$$

$$\min_{\boldsymbol{u} \in \mathcal{U}} \left[\ell(\boldsymbol{x}, \boldsymbol{u}) + \left(\frac{\partial J(\boldsymbol{x})}{\partial \boldsymbol{x}} \right)^T f(\boldsymbol{x}, \boldsymbol{u}) \right] = 0 \qquad \text{(HJB equation)}$$

- $J(\boldsymbol{x})$ is the **cost-to-go** function.
- The **optimal policy** is $\pi^*(\boldsymbol{x}) = \arg\min_{\boldsymbol{u} \in \mathcal{U}} \left[\ell(\boldsymbol{x}, \boldsymbol{u}) + \left(\frac{\partial J(\boldsymbol{x})}{\partial \boldsymbol{x}} \right)^T f(\boldsymbol{x}, \boldsymbol{u}) \right]$

The Hamilton-Jacobi-Bellman Equation (Optimality Condition)

- The HJB equation is the continuous version of the Bellman equation in discrete dynamic programming/reinforcement learning.
- A (sloppy) justification:
  - The Bellman equation (for optimal policy π^*) of the discrete problem:

$$J(\boldsymbol{x}_t) = \ell(\boldsymbol{x}_t, \boldsymbol{u}_t)\Delta t + J(\boldsymbol{x}_{t+1}), \text{ where } \frac{\boldsymbol{x}_{t+1} - \boldsymbol{x}_t}{\Delta t} = \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t) \text{ and } \boldsymbol{u}_t = \pi^*(\boldsymbol{x}_t)$$

  - Therefore,

$$\begin{aligned} 0 = \ell(\boldsymbol{x}_t, \boldsymbol{u}_t) + \frac{J(\boldsymbol{x}_{t+1}) - J(\boldsymbol{x}_t)}{\Delta t} &\iff 0 = \ell(\boldsymbol{x}_t, \boldsymbol{u}_t) + \frac{J(\boldsymbol{x}_{t+1}) - J(\boldsymbol{x}_t)}{\boldsymbol{x}_{t+1} - \boldsymbol{x}_t}\frac{\boldsymbol{x}_{t+1} - \boldsymbol{x}_t}{\Delta t} \\ &\iff 0 = \ell(\boldsymbol{x}_t, \boldsymbol{u}_t) + \frac{J(\boldsymbol{x}_{t+1}) - J(\boldsymbol{x}_t)}{\boldsymbol{x}_{t+1} - \boldsymbol{x}_t}\boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{u}_t) \end{aligned}$$

  - Let $\Delta t \to 0$, $0 = \ell(\boldsymbol{x}_t, \boldsymbol{u}_t) + \left(\frac{\partial J(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)^T \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$

Linear-Quadratic Regulator

Slides are based on CH8 of Underactuated Robotics

Canonical Form

- Consider a linear time-invariant system in state-space form:

  $$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}$$

  with the objective function to minimize:

  $$\int_0^\infty \ell(t)\mathrm{d}t = \int_0^\infty [\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u}]\mathrm{d}t$$

  in which $\boldsymbol{Q} = \boldsymbol{Q}^T \succeq 0, \boldsymbol{R} = \boldsymbol{R}^T \succ 0$ are two designed matrices.

- Remarks about the objective function:
  - The cost $\ell(\boldsymbol{x}, \boldsymbol{u})$ is quadratic: brings in advantage in doing analysis on the HJB equation.
  - The cost involves penalty for large control signals (the $\boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u}$ term), which favors *low energy* to achieve a certain goal.

State Tracking Example

- Sometimes, we need a bit of derivation to arrive at the canonical form.
- Consider the task of tracking a desired state trajectory $\boldsymbol{x}_d(t)$. It is natural to write down the following optimization problem:

$$\underset{\boldsymbol{u}\in\mathcal{U}}{\text{minimize}} \qquad \int_0^\infty (\boldsymbol{x}-\boldsymbol{x}_d)^T \boldsymbol{Q}(\boldsymbol{x}-\boldsymbol{x}_d) + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u}$$

  in which $\boldsymbol{Q} \succeq 0$, $\boldsymbol{R} \succ 0$, and $\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}$.
- Introduce $\tilde{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x}-\boldsymbol{x}_d \\ \boldsymbol{x}_d \end{bmatrix}$, $\tilde{A} = \begin{bmatrix} A & A \\ 0 & 0 \end{bmatrix}$, $\tilde{B} = \begin{bmatrix} B \\ 0 \end{bmatrix}$, $\tilde{\boldsymbol{Q}} = \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix}$, and the above optimization problem becomes the canonical form:

$$\underset{\boldsymbol{u}\in\mathcal{U}}{\text{minimize}} \qquad \int_0^\infty \tilde{\boldsymbol{x}}^T\tilde{\boldsymbol{Q}}\tilde{\boldsymbol{x}} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u}$$

  in which $\boldsymbol{Q} \succeq 0$, $\boldsymbol{R} \succ 0$, and $\dot{\tilde{\boldsymbol{x}}} = \tilde{A}\tilde{\boldsymbol{x}} + \tilde{B}\boldsymbol{u}$.

LQR Controller (Infinite-horizon, Closed-form)

- Recall the HJB optimality condition:

$$\exists J : \mathcal{X} \to \mathbb{R} \in \mathcal{C}^{\infty}(\mathcal{X}), \text{Hess}(J) \succ 0 \text{ such that } \forall \boldsymbol{x} \in \mathcal{X},$$

$$\min_{\boldsymbol{u} \in \mathcal{U}} L(\boldsymbol{x}, \boldsymbol{u}) = 0, \quad \text{where } L(\boldsymbol{x}, \boldsymbol{u}) = \ell(\boldsymbol{x}, \boldsymbol{u}) + \left(\frac{\partial J(\boldsymbol{x})}{\partial \boldsymbol{x}} \right)^T f(\boldsymbol{x}, \boldsymbol{u})$$

- For our linear-quadratic system setup, it is known that $J(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{S} \boldsymbol{x}$.
- Let us plug this $J(\boldsymbol{x})$ with unknown $\boldsymbol{S}$ into $L(\boldsymbol{x}, \boldsymbol{u})$ to solve $\boldsymbol{S}$.
- Sketch:
  1. For some given $\boldsymbol{x}$, we first find $\boldsymbol{u}^* = \arg\min_{\boldsymbol{u}} L(\boldsymbol{x}, \boldsymbol{u})$;
  2. Then we plug $\boldsymbol{u}^*$ back to $L(\boldsymbol{x}, \boldsymbol{u})$ and obtain an equation system that includes $\boldsymbol{S}$
  3. Solve $\boldsymbol{S}$ from the equation system.

LQR Controller (Infinite-horizon, Closed-form)

Details:

- By $\ell(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{u}^T \boldsymbol{R} \boldsymbol{u}$, $f(\boldsymbol{x}, \boldsymbol{u}) = A\boldsymbol{x} + B\boldsymbol{u}$, and $J(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{S} \boldsymbol{x}$,

$$L(\boldsymbol{x}, \boldsymbol{u}) = \ell(\boldsymbol{x}, \boldsymbol{u}) + \left(\frac{\partial J(\boldsymbol{x})}{\partial \boldsymbol{x}} \right)^T f(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{u}^T \boldsymbol{R} \boldsymbol{u} + 2\boldsymbol{x}^T \boldsymbol{S} A \boldsymbol{x} + 2\boldsymbol{x}^T \boldsymbol{S} B \boldsymbol{u}$$

- To solve $\boldsymbol{u}^*$ given $\boldsymbol{x}$, we have $\frac{\partial L}{\partial \boldsymbol{u}^*} = 2\boldsymbol{R}\boldsymbol{u}^* + 2B^T \boldsymbol{S} \boldsymbol{x} = 0$

$$\therefore \boldsymbol{u}^* = -\boldsymbol{R}^{-1} B^T \boldsymbol{S} \boldsymbol{x} = -\boldsymbol{K} \boldsymbol{x}$$

- Plug $\boldsymbol{u}^*$ back into L and by $\boldsymbol{x}^T \boldsymbol{S} A \boldsymbol{x} = \boldsymbol{x}^T A^T \boldsymbol{S} \boldsymbol{x}$,

$$\min_{\boldsymbol{u}} L(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{x}^T (\boldsymbol{Q} - \boldsymbol{S} B \boldsymbol{R}^{-1} B^T \boldsymbol{S} + \boldsymbol{S} A + A^T \boldsymbol{S}) \boldsymbol{x}$$

- Since $\min_{\boldsymbol{u}} L(\boldsymbol{x}, \boldsymbol{u}) \equiv 0, \forall \boldsymbol{x} \in \mathcal{X}$, it must be true that

$$\boldsymbol{Q} - \boldsymbol{S} B \boldsymbol{R}^{-1} B^T \boldsymbol{S} + \boldsymbol{S} A + A^T \boldsymbol{S} = 0 \quad \text{(algebraic Riccati equation)}$$

Algebraic Riccati Equation

Our goal is to solve $\boldsymbol{S}$ from the algebraic Riccati equation:

$$\boldsymbol{Q} - \boldsymbol{S}B\boldsymbol{R}^{-1}B^T\boldsymbol{S} + \boldsymbol{S}A + A^T\boldsymbol{S} = 0$$

The existence of solution depends on a so-called "controllable" condition.

- Controllable:
  - A system is said to be **controllable** if we can reach any target state $\boldsymbol{x}^*$ from any start state $\boldsymbol{x}_0$.
  - A system is said to be t**-time controllable** if we can reach any target state $\boldsymbol{x}^*$ from any start state $\boldsymbol{x}_0$ within t time period.
- The equation has a single positive-definite solution if and only if the system is controllable.
- There are good numerical methods for finding that solution, even in high-dimensional problems.

A Bit of Retrospection

- Our $\boldsymbol{u}^* = -\boldsymbol{K}\boldsymbol{x}$. Plug in the transition function $\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}$, and

$$\dot{\boldsymbol{x}} = (A - BK)\boldsymbol{x}$$

- The solution takes the form

$$\boldsymbol{x}(t) = e^{(A-B\boldsymbol{K})t}\boldsymbol{x}(0),$$

.

- Plug $\boldsymbol{x}(t)$ in the definition of the cost function, and we see that the cost takes the form

$$J = \boldsymbol{x}^T(0)\boldsymbol{S}\boldsymbol{x}(0),$$

which is a quadratic form, consistent with our assumption that J is quadratic.

iterative LQR (iLQR) for Non-linear System

- Consider the general optimal control problem:

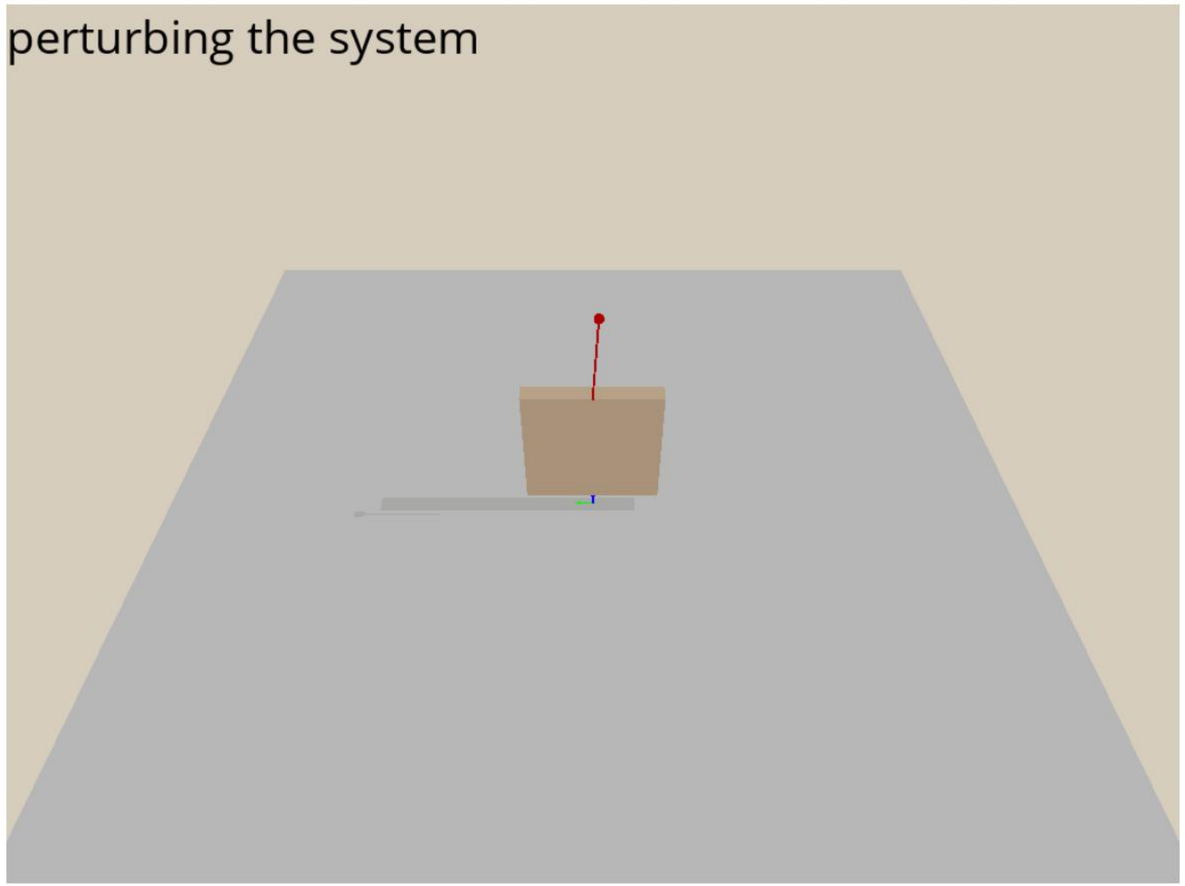
  $$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$$

  with the general objective function to minimize:

  $$\int_0^\infty \ell(t)\mathrm{d}t$$

- We can build a local LQR problem at some key time steps and iteratively apply the LQR controller:
  1. Compute the first-order Taylor expansion of the dynamics model $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ and the second-order Taylor expansion of the cost function $\ell(\boldsymbol{x}, \boldsymbol{u})$;
  2. Use the LQR to solve the optimal control policy and execute the policy for Δt;
  3. Go to (1) and recompute the approximation;

Example: Cart Pole



However, it does not always converge, if the initial perturbation stage gets longer.

Discrete-time LQR

Contents are based on CS287-FA19 by Prof. Pieter Abbeel at UC Berkeley

Problem Setup

- Transition function: $\boldsymbol{x}_{t+1} = A\boldsymbol{x}_t + B\boldsymbol{u}_t$
  $\boldsymbol{x}_t$: state at time t
  $\boldsymbol{u}_t$: input at time t
- 1-step cost function: $\ell(\boldsymbol{x}_t, \boldsymbol{u}_t) = \boldsymbol{x}_t \boldsymbol{Q} \boldsymbol{x}_t + \boldsymbol{u}_t^T \boldsymbol{R} \boldsymbol{u}_t$ with $\boldsymbol{Q} \succeq 0, \boldsymbol{R} \succ 0$
- We consider the **finite-horizon** setup, which needs more delicate treatment compared with the infinite-horizon setup.
- Suppose that our total cost only includes the cost for the first n steps:

$$\sum_{t=0}^{n-1} \ell(\boldsymbol{x}_t, \boldsymbol{u}_t),$$

Finite-horizon Cost-to-go Function

- We introduce the k-step cost-to-go function as:

$$J_k^\pi(\boldsymbol{x}) = \sum_{t=0}^{k-1} \ell(\boldsymbol{x}_t, \boldsymbol{u}_t),$$

  where $\boldsymbol{x}_0 = \boldsymbol{x}$ and $\boldsymbol{u}_t = \pi(\boldsymbol{x}_t)$.
- The optimal policy must take the optimal action at each step. Therefore, the $(k+1)$-step cost-to-go function for the optimal policy is:

$$J_{k+1}^{\pi^*}(\boldsymbol{x}) = \min_{\boldsymbol{u}}[\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u} + J_k^{\pi^*}(A\boldsymbol{x} + B\boldsymbol{u})] \qquad \text{(Bellman equation)}$$

Dynamic Programming

- Final state: When no step can be taken, there is a constant cost: $J_0(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x}$.

$$\boxed{J_0(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{P}_0 \boldsymbol{x}, \quad \text{where } \boldsymbol{P}_0 = \boldsymbol{Q}}$$

- Last step (n-th step):

$$\begin{aligned} J_1(\boldsymbol{x}) &= \min_{\boldsymbol{u}} [\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{u}^T \boldsymbol{R} \boldsymbol{u} + J_0(A\boldsymbol{x} + B\boldsymbol{u})] \\ &= \min_{\boldsymbol{u}} [\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{u}^T \boldsymbol{R} \boldsymbol{u} + (A\boldsymbol{x} + B\boldsymbol{u})^T \boldsymbol{P}_0 (A\boldsymbol{x} + B\boldsymbol{u})] \qquad (1) \end{aligned}$$

  Setting the gradient w.r.t. to $\boldsymbol{u}$ to zero and solve $\boldsymbol{u}_1$:

$$\boldsymbol{u}_1 = -(\boldsymbol{R} + B^T \boldsymbol{Q} B)^{-1} B^T \boldsymbol{P}_0 A \boldsymbol{x} = -\boldsymbol{K}_1 \boldsymbol{x} \qquad (2)$$

  (2) into (1):

$$\boxed{J_1(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{P}_1 \boldsymbol{x}, \quad \text{where } \boldsymbol{P}_1 = \boldsymbol{Q} + \boldsymbol{K}_1^T \boldsymbol{R} \boldsymbol{K}_1 + (A - B\boldsymbol{K}_1)^T \boldsymbol{P}_0 (A - B\boldsymbol{K}_1)}$$

Dynamic Programming (cont')

- $(n-1)$-th step:

$$\boxed{J_2(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{P}_2 \boldsymbol{x}, \quad \text{where } \boldsymbol{P}_2 = \boldsymbol{Q} + \boldsymbol{K}_2^T \boldsymbol{R} \boldsymbol{K}_2 + (A - B\boldsymbol{K}_2)^T \boldsymbol{P}_1 (A - B\boldsymbol{K}_2)}$$

  Control signal: $\boldsymbol{u}_2 = -\boldsymbol{K}_2 \boldsymbol{x}, \quad \text{where } \boldsymbol{K}_2 = (\boldsymbol{R} + B^T \boldsymbol{P}_1 B)^{-1} B^T \boldsymbol{P}_1 A.$

- Notice the similarity between $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$.

Summary of Dynamic Programming for LQR

1. Set $\boldsymbol{P}_0 = \boldsymbol{Q}$
2. for $i = 1, 2, 3, \ldots$

$$\boldsymbol{K}_i = (\boldsymbol{R} + B^T \boldsymbol{P}_{i-1} B)^{-1} B^T \boldsymbol{P}_{i-1} A$$
$$\boldsymbol{P}_i = \boldsymbol{Q} + \boldsymbol{K}_i^T \boldsymbol{R} \boldsymbol{K}_i + (A - B\boldsymbol{K}_i)^T \boldsymbol{P}_{i-1} (A - B\boldsymbol{K}_i)$$

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

$$\pi(\boldsymbol{x}) = \boldsymbol{K}_i \boldsymbol{x}$$

The cost-to-go function for a i-step horizon is given by: $J_i(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{P}_i \boldsymbol{x}$

Some Last Words about discrete LQR

- Solving the infinite-horizon optimal policy in closed-form for discrete LQR is not easy.
- However, if we decide to back-up for infinite steps, DP converges to the infinite-horizon optimal policy *if and only if* the dynamics (A, B) is such that there exists a policy that can drive the state to 0 (*source of information*).
- If converged, it is often most convenient to use the steady-state feedback K for all times.
- Similar to the infinite-horizon LQR example, we can extend our method to more sophisticated scenarios using iterative LQR (iLQR).

What I would like to but do not have the bandwidth to cover

- Stability theory
- Lyapunov analysis with convex optimization
- Trajectory optimization-based control
- Robust control
- Discrete/Continuous Hybrid control
- ...

Recommend to check out Underactuated Robots by Prof. Russ Tedrake if you are interested in these topics.

L11: Optimal Control
Han Su
Spring, 2021


Agenda
Fully-actuated v.s. Under-actuated Systems
Basic Idea of Optimal Control
Linear Quadratic Control

Fully-actuated v.s. Under-actuated Systems

Review: Control


Review: Feedforward and Feedback Control

Review: Computed Torque Control


Example of Single-Joint Manipulator
(Inverted Pendulum)


Limitation of PID Control