Optimal Control for Modern Robotics: Lecture 2

경희대학교
로봇 제어 및 지능 연구실
석박통합과정 2기 강민형

Optimal Control

Hamilton Jacobi Bellman Equation

핵심 내용

지금까지 우리는 optimal control 문제를 풀 때 즉 최적해를 구할 때의 핵심 원리가 Dynamic Programming이고
DP 의 핵심 개념인 최적성의 원리를 식으로 표현한 것이 Hamilton Jacobi Bellman Equation 임을 배웠다.

$$\dot{x} = Ax + Bu. \qquad 0 = \min_u \left[\ell(x,u,t) + \frac{\partial J^*(x,t)}{\partial t} + \nabla_x J^*(x,t)^\top f(x,u,t) \right]$$

최적해라면 반드시 만족해야 하는 optimality의 필요충분 조건

Limitation

- PDE이기 때문에 비선형 시스템에서는 풀 수 없다
- 고차원의 상태 공간에서 계산이 불가능하다

But, 특정한 경우에서는 solvable 하다 → Linear Dynamics + Quadratic Cost 일 경우

Linear Quadratic Regulator

Optimal Control

Linear Quadratic Regulator (LQR)

Linear Dynamics

$$\dot{x} = Ax + Bu$$

$$x \in \mathbb{R}^n, \qquad u \in \mathbb{R}^m$$

Quadratic Cost

$$J = \int_0^\infty \left(x^\top Q x + u^\top R u\right) dt$$

$$Q = Q^\top \succeq 0, \qquad R = R^\top \succ 0$$

Objective: 최적의 cost-to-go function J^* 찾기

HJB

$$\min_u \left[x^\top Q x + u^\top R u + \frac{\partial J^*}{\partial x}(Ax + Bu) \right] = 0$$

이 식을 만족하는 u^*와 J^* 를 찾으면 그것이 바로 최적해가 된다

Optimal Control

Linear Quadratic Regulator (LQR)

Objective: 최적의 cost-to-go function J^* 찾기

HJB

$$\min_u \left[x^\top Q x + u^\top R u + \frac{\partial J^*}{\partial x}(Ax + Bu) \right] = 0$$

이 경우에서 J^*는 항상 quadratic form이다

$$\underbrace{x^\top Q x}_{①} + \underbrace{u^\top R u}_{②} + \nabla_x J^*(x)^\top (Ax + Bu) = 0.$$

①번과 ②번이 소거되려면$\nabla_x J$는 1차 -> $\therefore J$ 는 Quadratic form이어야만 함

$$J^*(x) = x^\top S x, \qquad S = S^\top \succeq 0$$ 라고 가정할 수 있다

Optimal Control

Linear Quadratic Regulator (LQR)

$$J^*(x) = x^\top S x, \qquad S = S^\top \succeq 0$$ 라고 가정

HJB에 대입

$$\min_u \left[x^\top Q x + u^\top R u + (2x^\top S)(Ax + Bu) \right] = 0 \quad \Rightarrow \quad u^\top R u + 2x^\top S B u + \sim$$ u에 대한 convex 함수

$$\frac{\partial}{\partial u}\left(u^\top R u + 2x^\top S B u + \sim\right) = 2Ru + 2B^\top S x = 0.$$

$$2Ru + 2B^\top S x = 0$$

$$u^* = -R^{-1} B^\top S x.$$

$$u^*(x) = -Kx, \qquad K = R^{-1} B^\top S.$$

Optimal Control

Linear Quadratic Regulator (LQR)

이걸 다시 HJB에 대입

$$0 = x^\top \left[Q - SBR^{-1}B^\top S + 2SA \right] x$$

$$\Rightarrow \quad 0 = SA + A^\top S - SBR^{-1}B^\top S + Q$$

Algebric Ricatti Equation

여기서 우리가 모르는 값은 S 하나 뿐 -> S : 우리가 찾아야 할 J^*의 행렬

$$u^* = -Kx, \qquad K = R^{-1}B^\top S.$$

LQR 은 선형 시스템에서 유효하지만 우리의 관심사는 실제 로봇 같은 비선형 시스템 -> How?

Optimal Control

Linear Quadratic Regulator (LQR)

LQR 은 선형 시스템에서 유효하지만 우리의 관심사는 실제 로봇 같은 비선형 시스템 -> How?

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

$$\text{operating point } (x_0, u_0), \qquad f(x_0, u_0) = 0$$

Equilibrium point 또는 steady state

Dynamics가 비선형이어서 사용이 어려운 문제를 steady state를 기준으로 선형화해서 제어 가능

$$\bar{x} = x - x_0, \qquad \bar{u} = u - u_0$$

$$\therefore \quad \dot{\bar{x}} = \dot{x} = f(x, u).$$

Taylor Expansion

$$\dot{\bar{x}} \approx f(x_0, u_0) + \left.\frac{\partial f}{\partial x}\right|_{(x_0, u_0)} (x - x_0) + \left.\frac{\partial f}{\partial u}\right|_{(x_0, u_0)} (u - u_0)$$

Optimal Control

Linear Quadratic Regulator (LQR)

Taylor Expansion

$$\dot{x} \approx f(x_0, u_0) + \left.\frac{\partial f}{\partial x}\right|_{(x_0,u_0)} (x - x_0) + \left.\frac{\partial f}{\partial u}\right|_{(x_0,u_0)} (u - u_0)$$

$$A := \left.\frac{\partial f}{\partial x}\right|_{(x_0,u_0)}, \qquad B := \left.\frac{\partial f}{\partial u}\right|_{(x_0,u_0)}$$

$$\Rightarrow \quad \dot{\bar{x}} \approx A\bar{x} + B\bar{u}$$

선형 및 Quadratic으로 변환 -〉

LQR 적용 가능

$$\therefore \quad \bar{u} = -K\,\bar{x}$$

$$= u_0 - K(x - x_0).$$

$$J = \int_0^\infty \left(x^\top Q x + u^\top R u\right) dt$$

$$x = x_0 + \bar{x}, \qquad u = u_0 + \bar{u}$$

$$x^\top Q x = (x_0 + \bar{x})^\top Q (x_0 + \bar{x}) = x_0^\top Q x_0 + 2x_0^\top Q\bar{x} + \bar{x}^\top Q\bar{x},$$

$$u^\top R u = (u_0 + \bar{u})^\top R (u_0 + \bar{u}) = u_0^\top R u_0 + 2u_0^\top R\bar{u} + \bar{u}^\top R\bar{u}$$

$$\Rightarrow \; J = \int_0^\infty \left[\bar{x}^\top Q\bar{x} + \bar{u}^\top R\bar{u} + 2x_0^\top Q\bar{x} + 2u_0^\top R\bar{u} + x_0^\top Q x_0 + u_0^\top R u_0\right] dt$$

$$J \approx \int_0^\infty \left(\bar{x}^\top Q\bar{x} + \bar{u}^\top R\bar{u}\right) dt$$

Optimal Control

Linear Quadratic Optimal Tracking

지금까지의 LQR은 상태를 0으로 보내는 regulator ➡ 즉 현재 상태가 0에서 얼마나 벗어났는지 penalize

그래서 cost function도 $error = x - 0 = x$ 자체로 사용 $J = \int_0^\infty (x^\top Q x + u^\top R u)\, dt$

But, 실제 문제는 "0"이 아닌 어떤 임의의 목표 궤적 or 상태를 따라야한다

$$x(t) \to x_d(t), \qquad u(t) \to u_d(t)$$

$$\bar{x} = x - x_d(t), \qquad \bar{u} = u - u_d(t)$$

$$\dot{\bar{x}} = A(t)\,\bar{x} + B(t)\,\bar{u}$$

$$\bar{u}(t) = -K(t)\,\bar{x}$$

$$u(t) = u_d(t) - K(t)\big(x - x_d(t)\big)$$

Optimal Control

https://github.com/minhyeong10/optimal_control_tutorial.git

• Linear Quadratic Optimal Tracking

```python
import numpy as np
from scipy.linalg import solve_continuous_are

def continuous_lqr(A, B, Q, R):
    """
    연속시간 LQR: xdot = A x + B u
    cost = ∫ (xᵀ Q x + uᵀ R u) dt

    Riccati 방정식 AᵀP + P A - P B R⁻¹ Bᵀ P + Q = 0
    """
    A = np.array(A, dtype=float)
    B = np.array(B, dtype=float)
    Q = np.array(Q, dtype=float)
    R = np.array(R, dtype=float)

    # 1) CARE 풀기 → P 구함
    P = solve_continuous_are(A, B, Q, R)

    # 2) 최적 게인 K = R⁻¹ Bᵀ P
    R_inv = np.linalg.inv(R)
    K = R_inv @ B.T @ P

    return K, P
```

```python
# ========================================
# Ref trajectory
# ========================================
def joint_reference(t: float):

    # joint1: 1.2 * sin(0.5 t)
    q1 = 1.2 * np.sin(0.5 * t)
    q1d = 1.2 * 0.5 * np.cos(0.5 * t)
    q1dd = -1.2 * (0.5 ** 2) * np.sin(0.5 * t)

    # joint2: -0.5 + 1.0 * sin(0.8 t)
    q2 = -0.5 + 1.0 * np.sin(0.8 * t)
    q2d = 1.0 * 0.8 * np.cos(0.8 * t)
    q2dd = -1.0 * (0.8 ** 2) * np.sin(0.8 * t)

    # joint3: 1.5 * sin(1.2 t)
    q3 = 1.5 * np.sin(1.2 * t)
    q3d = 1.5 * 1.2 * np.cos(1.2 * t)
    q3dd = -1.5 * (1.2 ** 2) * np.sin(1.2 * t)

    q_ref = np.array([q1, q2, q3])
    qd_ref = np.array([q1d, q2d, q3d])
    qdd_ref = np.array([q1dd, q2dd, q3dd])

    return q_ref, qd_ref, qdd_ref
```

```python
def simulate_lqr_tracking(T_total=8.0, dt=0.005):

    nq = N_JOINTS
    nx = 2 * nq

    A, B = build_double_integrator_system(nq)
    Q = np.diag(
        np.concatenate([
            50.0 * np.ones(nq),  # q error
            5.0 * np.ones(nq),   # qdot error
        ])
    )
    R = 0.1 * np.eye(nq)

    K, P = continuous_lqr(A, B, Q, R)
    print("LQR K =\n", K)



    times = np.arange(0.0, T_total, dt)
    N = len(times)

    # 상태 초기값: 정지 상태에서 시작
    x = np.zeros(nx)  # [q; qdot]

    X_trj = np.zeros((N, nx))
    q_ref_trj = np.zeros((N, nq))

    def f(x, t):
        # 참조
        q_ref, qd_ref, qdd_ref = joint_reference(t)
        x_ref = np.concatenate([q_ref, qd_ref])  # (6,)
        u_ff = qdd_ref                           # (3,)

        e = x - x_ref
        u = u_ff - K @ e

        xdot = A @ x + B @ u
        return xdot
```

감사합니다.