Week 2

Inverse-Dynamics Tutorial

Kangmin Lee

This tutorial covers inverse-dynamics from Operational Space Control to QP, HQP, and TSID focusing on the math, intuition, and practical design patterns behind them. Through code and simulation, you'll learn to formulate tasks and constraints and compute constraint-satisfying torques suitable for robots.

Notation & Definitions (기호와 정의)

표기

- 시스템의 상태(state)는 $\mathbf{x}$ 로 표기
- 제어 입력(input)은 $\mathbf{u}$ 로 표기

구동기 구성

- **완전 구동(Fully actuated):** 구동기(액추에이터) 수 = 자유도 수
  - ex) 대부분의 산업용 매니퓰레이터
- **부분/미완전 구동(Underactuated):** 구동기 수 < 자유도 수
  - ex) 다족 보행 로봇, 쿼드로터, 비행기

로봇 모델

- 모델은 로봇의 특성과 과정(task)에 맞춰 선택
  - 복잡한 모델 vs 단순화 모델
  - 접촉, 마찰 등 모델 포함 여부, 제약식 반영 수준

Velocity Control (속도 제어)

- 로봇의 상태 $\mathbf{x}$는 q로 표현된다
- 제어 입력 $\mathbf{u}$는 로봇 관절의 속도 $\mathbf{v}_q$로 표기

$$\mathbf{u} = \mathbf{v}_q = \dot{q} \qquad q = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

- $\mathbf{u}$를 적분하면 q를 알 수 있기 때문에 단순 적분기에 해당 $q = q(0) + \int \mathbf{u}$

유압 액추에이터 :

- 유량 제어를 통한 빠른 속도 응답
- 큰 관성 부하에서도 속도 명령 추종이 우수
- 외란에도 속도 유지 능력↑
- 유압은 본질적으로 유량(=속도)에 민감해 속도 제어 특성이 강함

일부 전기 모터

- 산업용 매니퓰레이터
- 마찰/유연성 영향 작음
  → 속도원 근사 대체로 유효
- 상호작용/충격/유연성이 크면 한계
- 공장 로봇팔처럼 안정된 환경에서 유리

Acceleration Control (가속도 제어)

- $\mathbf{x} = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$
- $\mathbf{u} = \ddot{q}$
- $\dot{\mathbf{q}} = \mathbf{v}_q, \qquad \dot{\mathbf{v}}_q = \mathbf{u}$

$$\begin{bmatrix} \mathbf{v}_q \\ \dot{\mathbf{v}}_q \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} \mathbf{q} \\ \mathbf{v}_q \end{bmatrix} + \underbrace{\begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix}}_{\mathbf{B}} \mathbf{u}$$

전기 모터 :

- 큰 접촉력이 없는 경우 제어가 잘 됨

행렬로 쓰는 이유

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

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \dot{\mathbf{x}}(t)\,\Delta t$$

Torque Control (토크 제어)

- $\mathbf{x} = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$
- $\mathbf{u} = \boldsymbol{\tau}$
- 앞서 배운 운동방정식을 풀어 제어 입력 값인 토크를 직접 구하는 것이 목표

$$M(q)\,\ddot{q} + C(q,\dot{q})\,\dot{q} + g(q) = \tau + J^{\top}(q) f$$

The dynamic equation of a fully-actuated mechanical system

Torque Control (토크 제어)

Under-Actuated Systems

- 자유도 수 대비 구동기 수가 적은 시스템
- **수동 관절** q_u과 **구동 관절** q_a로 분할 : $q = (q_u, q_a)$

- $\mathbf{v}_q = \begin{bmatrix} \mathbf{v}_u \\ \mathbf{v}_a \end{bmatrix}$

- 선택 행렬 **S** 로 "구동 관절 성분"만 뽑기 : $\mathbf{S} \begin{bmatrix} \mathbf{0}_{n_{va} \times n_{vu}} & \mathbf{I}_{n_{va}} \end{bmatrix}$

$$\mathbf{v}_a = \mathbf{S}\,\mathbf{v}_q$$

Torque Control (토크 제어)

Under-Actuated Systems의 운동 방정식

$$\mathbf{M}(\mathbf{q})\,\dot{\mathbf{v}}_q + \mathbf{h}(\mathbf{q}, \mathbf{v}_q) = \mathbf{S}^\top \boldsymbol{\tau}_a + \mathbf{J}(\mathbf{q})^\top \mathbf{f}$$

- 편의를 위해 수동/구동 파트로 분해하여 사용하기도 한다

$$\mathbf{M}_u(\mathbf{q})\,\dot{\mathbf{v}}_q + \mathbf{h}_u(\mathbf{q}, \mathbf{v}_q) = \mathbf{J}_u(\mathbf{q})^\top \mathbf{f}$$

$$\mathbf{M}_a(\mathbf{q})\,\dot{\mathbf{v}}_q + \mathbf{h}_a(\mathbf{q}, \mathbf{v}_q) = \boldsymbol{\tau}_a + \mathbf{J}_a(\mathbf{q})^\top \mathbf{f}$$

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_u \\ \mathbf{M}_a \end{bmatrix}, \quad \mathbf{h} = \begin{bmatrix} \mathbf{h}_u \\ \mathbf{h}_a \end{bmatrix}, \quad \mathbf{J} = \begin{bmatrix} \mathbf{J}_u \\ \mathbf{J}_a \end{bmatrix}$$

어떻게 제어를 할 것인가

제어 목표를 "태스크 함수" e로 정의하고, 이 값을 0으로 수렴시키는 문제로 바꾼다

- 오차(error)가 최소(=0)이 되도록 한다

$$\mathbf{e}(\mathbf{x}, \mathbf{u}, t) = \underbrace{\mathbf{y}(\mathbf{x}, \mathbf{u})}_{\text{real}} - \underbrace{\mathbf{y}^*(t)}_{\text{reference}}$$

- **세가지 종류의 태스크 함수 :**

1. 입력 u에 대한 선형 함수 $\mathbf{e}(\mathbf{u}, t) = A_u\,\mathbf{u} - \mathbf{a}(t)$

2. 속도 v에 대한 비선형 함수 $\mathbf{e}(\mathbf{v}_q, t) = \mathbf{y}(\mathbf{v}_q) - \mathbf{y}^*(t)$

3. 구성 q에 대한 비선형 함수 $\mathbf{e}(\mathbf{q}, t) = \mathbf{y}(\mathbf{q}) - \mathbf{y}^*(t)$

어떻게 제어를 할 것인가

제어 목표를 “태스크 함수” e로 정의하고, 이 값을 0으로 수렴시키는 문제로 바꾼다

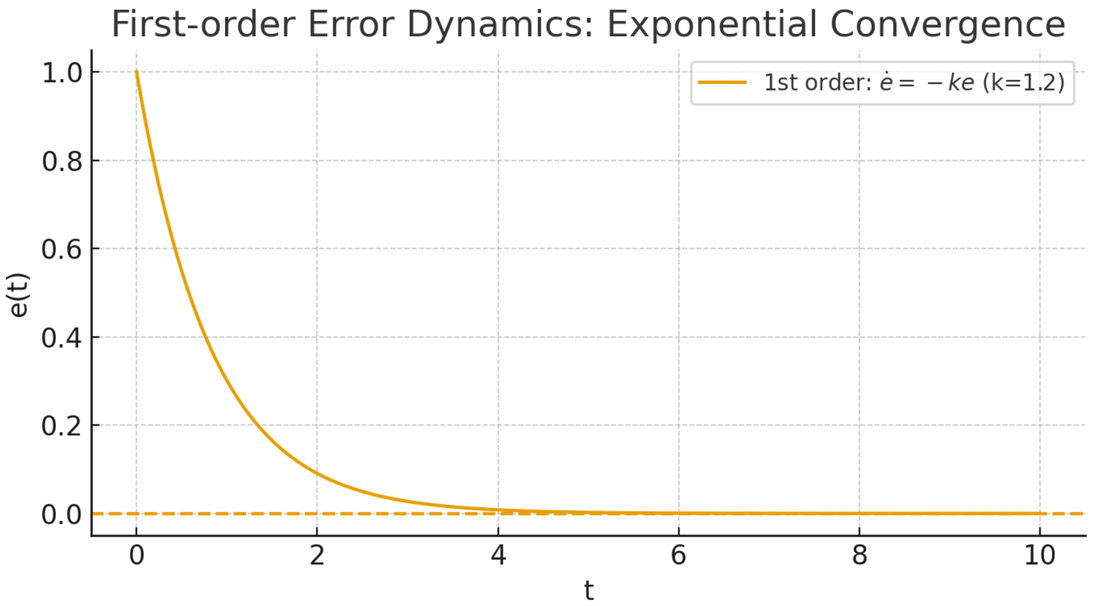
- 오차(error)가 최소(=0)이 되도록 한다

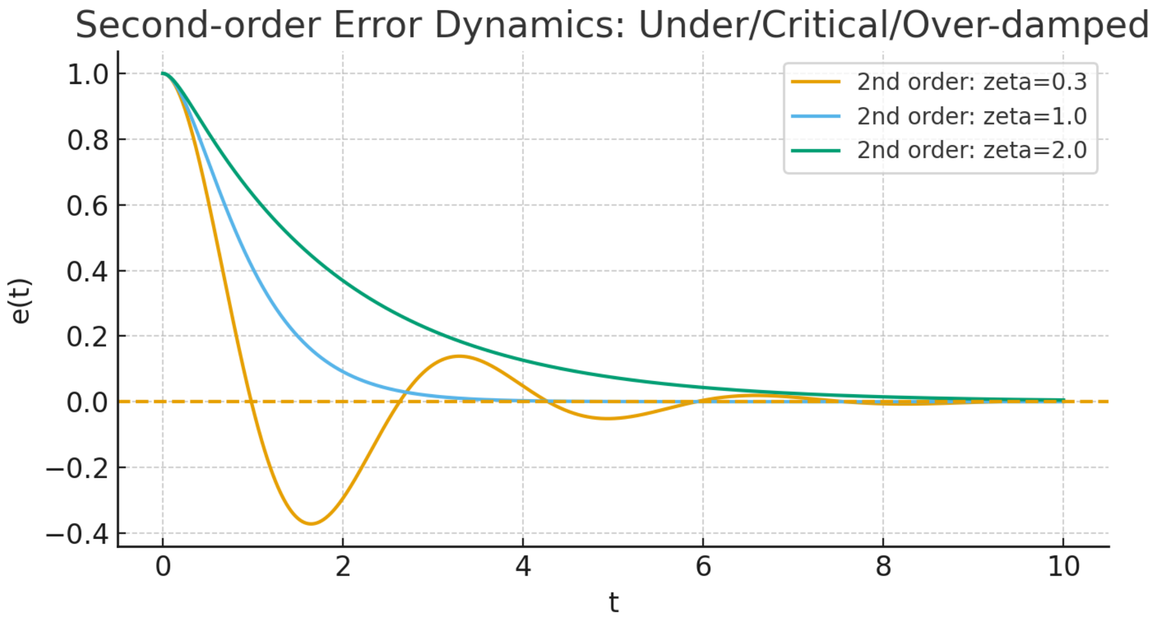
$$\mathbf{e}(\mathbf{x}, \mathbf{u}, t) = \mathbf{y}(\mathbf{x}, \mathbf{u}) - \mathbf{y}^*(t)$$

$\mathbf{e}(\mathbf{x}, \mathbf{u}, t)$ 함수가 $\lim_{t\to\infty} \mathbf{e}(\mathbf{x}, \mathbf{u}, t) = \mathbf{0}$ 가 되도록 시스템을 설계하는 문제로 바뀜

오차동역학 복습 $\lim_{t\to\infty} \mathbf{e}(\mathbf{x}, \mathbf{u}, t) = \mathbf{0}$

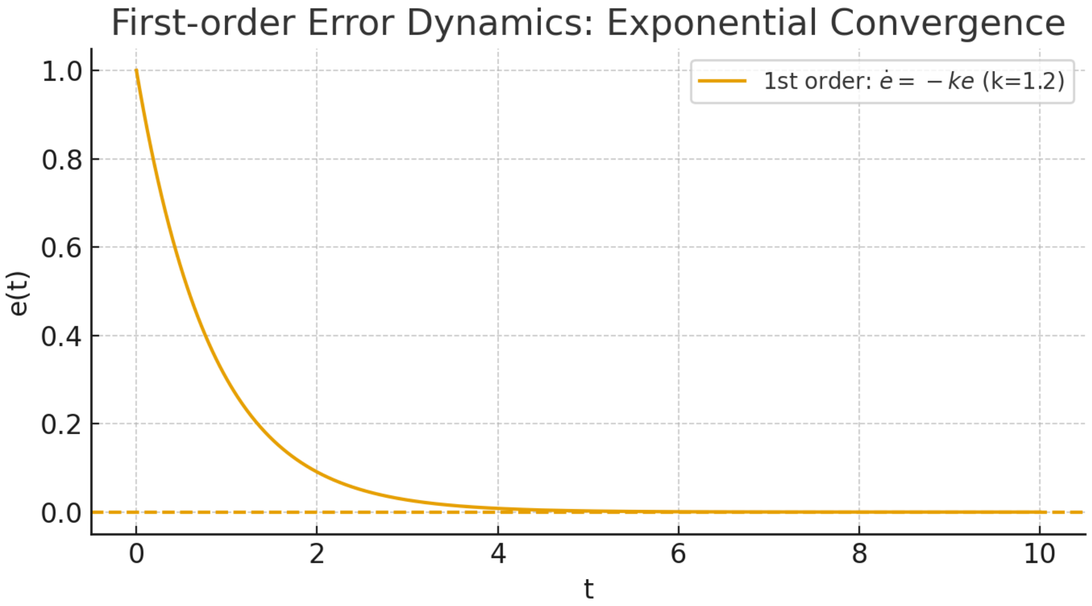
- 공학수학, 기계진동 등에서 수렴을 시키는 시스템을 설계할 때 많이 사용되는 수학적 기법
- 다양한 **선형/비선형** 수렴 형태가 존재 > 대표적으로 1차, 2차 미분방정식이 존재
- **1차, 2차 미분방정식을 만족하는 시스템일 경우 → 수렴성 보장**

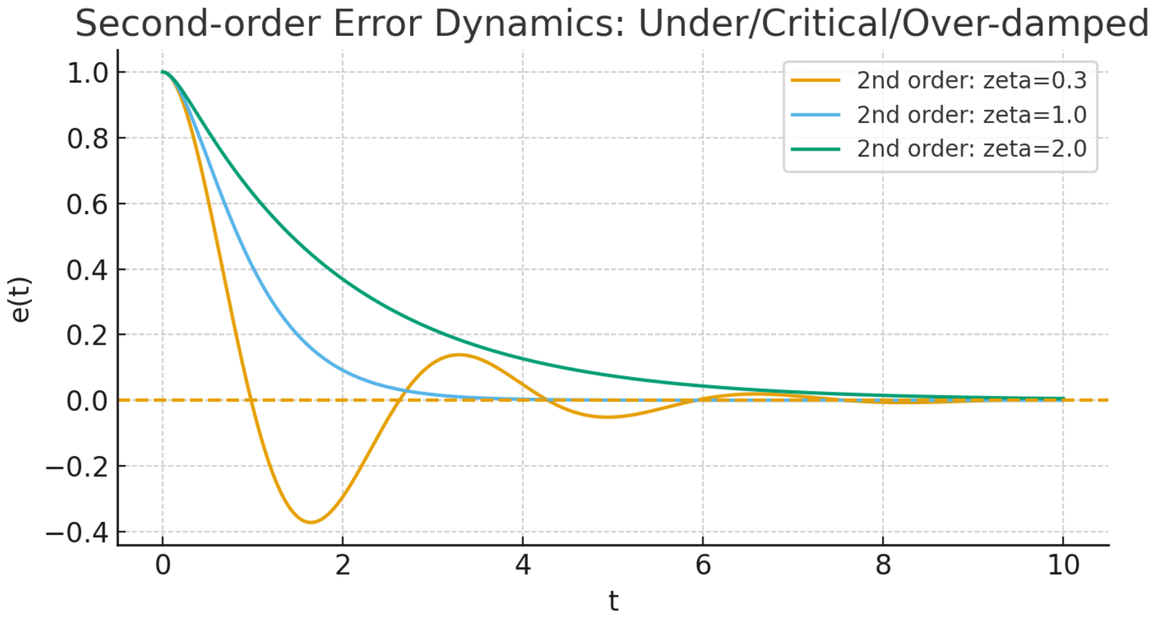




오차동역학 복습 $\lim_{t\to\infty} \mathbf{e}(\mathbf{x},\mathbf{u},t) = \mathbf{0}$

- 따라서 1차, 2차 미분방정식을 만족하는 시스템이라 **가정하고 시작**





$$\dot{e}(t) = -k\,e(t), \qquad k > 0$$

$$\ddot{e}(t) + 2\zeta\omega_n\dot{e}(t) + \omega_n^2 e(t) = 0, \qquad \omega_n > 0,\ \zeta > 0$$

$$e(0) = e_0 \quad \Rightarrow \quad e(t) = e_0\,e^{-kt}$$

Under/Critical/Over damped

OSC (Operational Space Control)

- 목표: 말단(EE)의 원하는 궤적 $(\mathbf{x}_d,\ \dot{\mathbf{x}}_d,\ \ddot{\mathbf{x}}_d)$을 조인트 토크 $\mathbf{u}$로 바로 만들어 추종

- 주요 식 : $\mathbf{e} = \mathbf{x}_d - \mathbf{x}, \qquad \dot{\mathbf{e}} = \dot{\mathbf{x}}_d - \mathbf{J}(\mathbf{q})\,\dot{\mathbf{q}}$

$$y = \mathbf{J}(\mathbf{q})^{-1}\left(\mathbf{a}_x^{\star} - \dot{\mathbf{J}}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}}\right) \qquad \mathbf{a}_x^{\star} = \ddot{\mathbf{x}}_d + \mathbf{K}_D\,\dot{\mathbf{e}} + \mathbf{K}_P\,\mathbf{e}$$

$$\mathbf{u} = \mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q})$$

장점 :

- K 게인 값 등 말단 성능을 직접 설계
- 수식이 간단해 개념 습득·데모가 빠름
- OSC는 이상적·무제한 상황에서 말단 제어를 깔끔히 보여준다

하지만 실제 로봇은 '제약'과 '다중과제'가 핵심 > 따라서 **TSID**(정식화)와 **QP/HQP**(해결 방식)가 필요

OSC 유도

목표 : 말단(end-effector)의 실제 궤적 $x(t)$가 원하는 궤적 $x_d(t)$를 잘 따라가게 하고자 함

- 오차 정의 : $\mathbf{e} = \mathbf{x}_d - \mathbf{x}, \qquad \dot{\mathbf{e}} = \dot{\mathbf{x}}_d - \dot{\mathbf{x}}$
- 오차가 따랐으면 하는 미분방정식 선택(시스템 설계) : 다루기 쉬운 2차(스프링-댐퍼) 시스템

$$\ddot{\mathbf{e}} + \mathbf{K}_D\dot{\mathbf{e}} + \mathbf{K}_P\mathbf{e} = \mathbf{0}$$

- 수식 전개 : $(\ddot{\mathbf{x}}_d - \ddot{\mathbf{x}}) + \mathbf{K}_D(\dot{\mathbf{x}}_d - \dot{\mathbf{x}}) + \mathbf{K}_P(\mathbf{x}_d - \mathbf{x}) = \mathbf{0}$

$$\ddot{\mathbf{x}} = \ddot{\mathbf{x}}_d + \mathbf{K}_D(\dot{\mathbf{x}}_d - \dot{\mathbf{x}}) + \mathbf{K}_P(\mathbf{x}_d - \mathbf{x})$$

$$\Rightarrow \quad \mathbf{a}_x^\star = \ddot{\mathbf{x}}_d + \mathbf{K}_D\dot{\mathbf{e}} + \mathbf{K}_P\mathbf{e}$$

- End-Effector 수식과 연결 : $\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\,\dot{\mathbf{q}}$

$$\ddot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\,\ddot{\mathbf{q}} + \dot{\mathbf{J}}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}}$$

$$\ddot{\mathbf{q}}^\star = \mathbf{J}(\mathbf{q})^{-1}\left(\mathbf{a}_x^\star - \dot{\mathbf{J}}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}}\right)$$

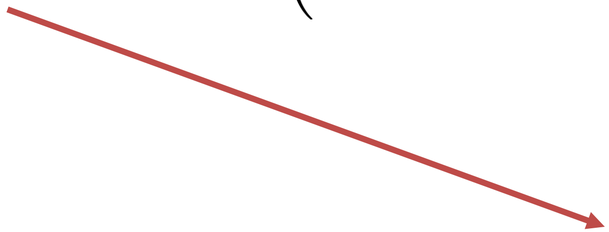
OSC 유도

목표 : 말단(end-effector)의 실제 궤적 $x(t)$가 원하는 궤적 $x_d(t)$를 잘 따라가게 하고자 함

- 최종 식 :

$$\ddot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\,\ddot{\mathbf{q}} + \dot{\mathbf{J}}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}} \qquad \mathbf{a}_x^\star = \ddot{\mathbf{x}}_d + \mathbf{K}_D\dot{\mathbf{e}} + \mathbf{K}_P\mathbf{e}$$

$$\ddot{\mathbf{q}}^\star = \mathbf{J}(\mathbf{q})^{-1}\left(\mathbf{a}_x^\star - \dot{\mathbf{J}}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}}\right)$$



- 대입하여 최종 제어 인풋을 구함 $\mathbf{u} = \mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q})$

OSC 한계

- **가정이 강함 :** J가 정사각/가역, 완전 구동, 접촉·마찰·한계가 거의 없음
  → 실제 전신 로봇/이동로봇과 안 맞는 경우가 많음

- **다중 과업 충돌 :** 한 손 위치, 다른 손 힘, 시선 고정, 몸통 안정 등 여러 목표가 동시에 오면 우선순위/충돌 해결이 필요하지만, 기본 OSC는 이를 체계적으로 보장하기 어려움

- **제약 미포함 :** 조인트 제한(각도/속도/가속도/토크 한계)접촉 가속도 제약), 마찰원뿔 등, 기본 OSC 식에는 이런 부등식/경계 조건을 안전하게 넣기 힘듦

이러한 제한 조건들을 해결하기 위해서 TSID와 QP를 적용

Thank you