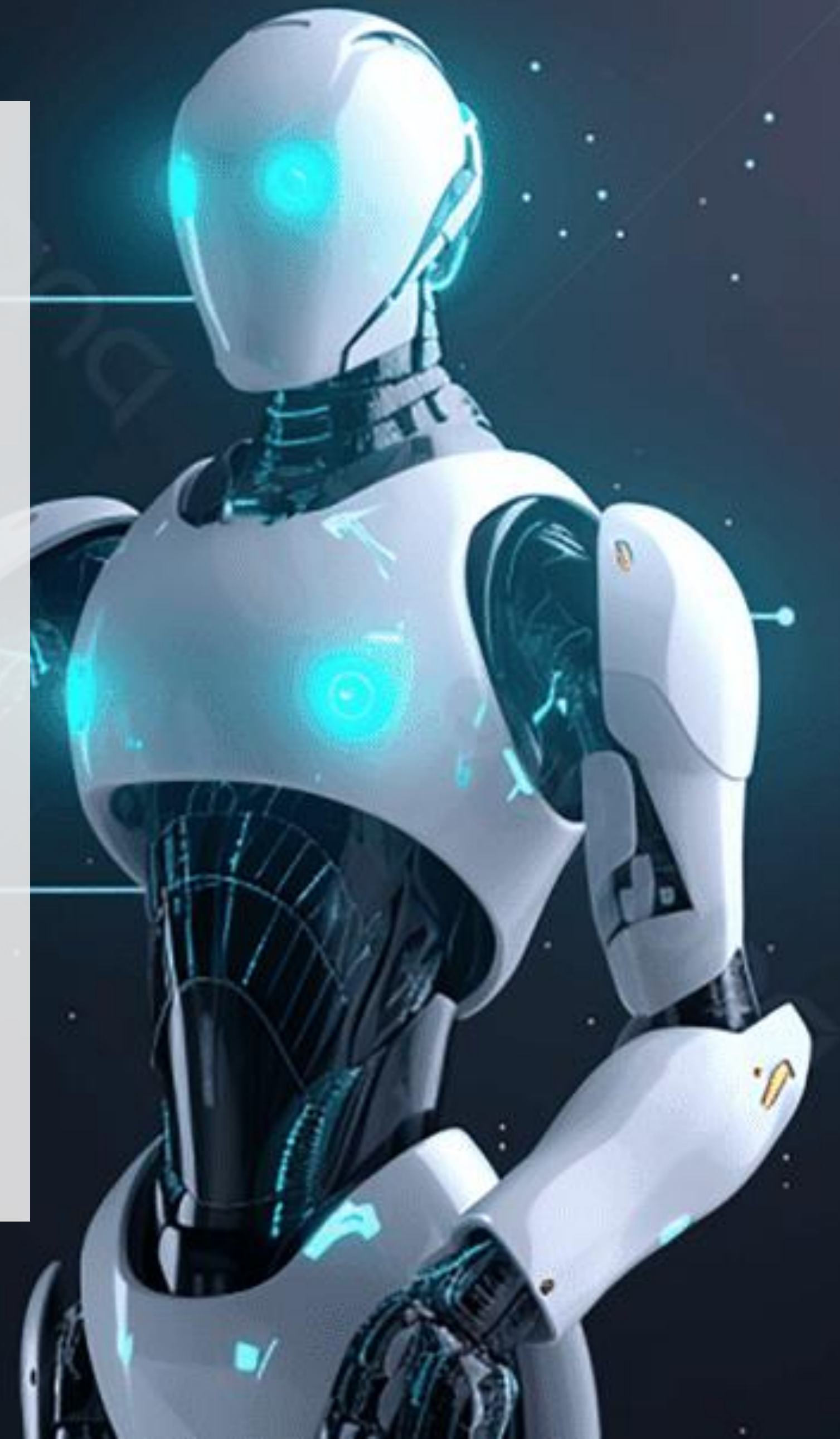


Week 3

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.



TSID

- 마찬가지로 오차방정식 시스템을 만족한다고 가정하고 시작
- 반드시 1차일 필요 없음, 하나의 대표적인 예시

• 세가지 종류의 태스크 함수:

1. 입력 u 에 대한 선형 함수 $e(u, t) = A_u u - a(t)$

2. 속도 v 에 대한 비선형 함수 $e(v_q, t) = y(v_q) - y^*(t)$

3. 구성 q 에 대한 비선형 함수 $e(q, t) = y(q) - y^*(t)$

Task Function $e(v_q, t) = y(v_q) - y^*(t)$

시스템 가정 $\dot{e} = -K e$

전개 $\underbrace{\frac{\partial y}{\partial v_q}}_{\text{Jacobian}} \dot{v}_q - \dot{y}^* = -K e$

$$\underbrace{J}_{A_v} \dot{v}_q = \underbrace{\dot{y}^* - K e}_a$$

TSID

- 마찬가지로 오차방정식 시스템을 만족한다고 가정하고 시작
- 반드시 2차일 필요 없음, 하나의 대표적인 예시

• 세가지 종류의 태스크 함수:

1. 입력 u 에 대한 선형 함수 $e(u, t) = A_u u - a(t)$

2. 속도 v 에 대한 비선형 함수 $e(v_q, t) = y(v_q) - y^*(t)$

3. 구성 q 에 대한 비선형 함수 $e(q, t) = y(q) - y^*(t)$

Task Function $e(q, t) = y(q) - y^*(t)$

시스템 가정 $\ddot{e} = -K e - D \dot{e}$

전개
$$\underbrace{J(q)}_{\text{Jacobian}} \ddot{q} + \dot{J}(q, \dot{q}) \dot{q} - \ddot{y}^* = -K e - D \dot{e}$$

$$\underbrace{J(q)}_{A_v} \dot{v}_q = \underbrace{\ddot{y}^* - \dot{J}(q, \dot{q}) \dot{q} - K e - D \dot{e}}_a$$

TSID 정리

$$\underbrace{J}_{A_v} \dot{v}_q = \underbrace{\dot{y}^* - K e}_a$$

$$\underbrace{J(q)}_{A_v} \dot{v}_q = \underbrace{\ddot{y}^* - \dot{J}(q, \dot{q}) \dot{q} - K e - D \dot{e}}_a$$

- $g(y)$ 함수로 정리 : $g(y) = A_v \dot{v}_q - a \longrightarrow g(y) = 0$

- $g(y)$ 가 0 이 되면 초기 가정한 1, 2차 미분방정식을 만족한다는 것

- > 즉, 수렴성이 보장된다

- > error 가 0이다

- > 제어가 되었다

- 세가지 종류의 태스크 함수:

- 1. 입력 u 에 대한 선형 함수 $e(u, t) = A_u u - a(t)$

- 2. 속도 v 에 대한 비선형 함수 $e(v_q, t) = y(v_q) - y^*(t)$

- 3. 구성 q 에 대한 비선형 함수 $e(q, t) = y(q) - y^*(t)$

TSID 일반화식

$g(y) = A_v \dot{v}_q - a$ 형태로 정리하여 $g(y)$ 가 0이 되도록 함

• 세가지 종류의 태스크 함수:

1. 입력 u 에 대한 선형 함수 $e(u, t) = A_u u - a(t)$
2. 속도 v 에 대한 비선형 함수 $e(v_q, t) = y(v_q) - y^*(t)$
3. 구성 q 에 대한 비선형 함수 $e(q, t) = y(q) - y^*(t)$

입력 u 에 대한 선형 함수까지 포함하면 다음과 같이 일반화 가능

일반화된 $g(y)$:
$$g(y) = \underbrace{\begin{bmatrix} A_v & A_u \end{bmatrix}}_A \underbrace{\begin{bmatrix} \dot{v}_q \\ u \end{bmatrix}}_y - a$$

행렬로 표기했을 때의 장점 :

- 여러 태스크 동시에 다루기 위해 쌓을 수가 있음

$$g(y) = \underbrace{\begin{bmatrix} A_{v,1} & A_{u,1} \\ A_{v,2} & A_{u,2} \\ \vdots & \vdots \\ A_{v,\ell} & A_{u,\ell} \end{bmatrix}}_{=A} \underbrace{\begin{bmatrix} \dot{v}_q \\ u \end{bmatrix}}_{=y} - \underbrace{\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_\ell \end{bmatrix}}_{=a} = Ay - a$$

- 각 태스크별 모든 $g(y)$ 를 0으로 만들기란 어려움 > 따라서 최소가 되는 $g(y)$ 를 찾기 위함

QP (Quadratic Program)

제어 문제를 최적화 문제로 정식화한다

핵심 요소

- 상태 : $x = (q, v_q)$
- 제어 입력 : $u = \tau$
- 동역학 방정식(no contacts) : $M(q) \dot{v}_q + h(q, v_q) = S^\top \tau$
- 최소화할 Task Function : $\|g(y)\|^2 = \|Ay - a\|^2$

QP (Quadratic Program)

- 최소화할 Task Function : $\|g(y)\|^2 = \|Ay - a\|^2$

절댓값을 취한 후, 제곱을 하여 계산하는 이유

- $g(y)$ 자체는 벡터이기 때문에 0에 가까운 최적값을 구하기 위해서 스칼라 형태로 변형
- 제곱을 한 것과 하지 않은 것의 해는 동일
- 제곱을 하게 되면 볼록한 이차 형태가 되어, 오차를 부드럽게 만들 수 있다
- 이차 형태일 경우 더욱 빠르고 안정적이게 되어, QP를 푸는데 실시간성과 신뢰성을 부여할 수 있다
- 제곱을 하지 않고 단순히 절댓값만 사용하면, 미분 불가 지점이 생기며 토크가 커지고 불안정해질 수 있다
 - > 튜닝에 까다로움

QP (Quadratic Program)

- 운동방정식 : $M(\mathbf{q}) \ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{S}^\top \boldsymbol{\tau}$

$$M(\mathbf{q}) \ddot{\mathbf{q}} - \mathbf{S}^\top \boldsymbol{\tau} = -\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \quad \mathbf{y} = \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\tau} \end{bmatrix}$$

→ $\begin{bmatrix} \mathbf{M} & -\mathbf{S}^\top \end{bmatrix} \mathbf{y} = -\mathbf{h}$

- 최종 QP 문제 형태

$$\min_{\mathbf{y}=(\ddot{\mathbf{q}}, \boldsymbol{\tau})} \|\mathbf{A} \mathbf{y} - \mathbf{a}\|^2 \quad \text{s.t.} \quad \begin{bmatrix} \mathbf{M} & -\mathbf{S}^\top \end{bmatrix} \mathbf{y} = -\mathbf{h}$$

- 문제를 최소화할 \mathbf{y} 를 구하는 것!

Contact (접촉 제약) 추가

Soft Contact

- Soft Contact의 경우 바닥이 탄성을 갖고 있어 약간은 움직일 수 있는 상태라 본다
 - > 즉, 속도=0, 가속도=0 과 같은 강한 구속 조건을 추가하지 않음
- 접촉으로 인해 발생하는 힘(외력)을 더해주기만 하면 됨

$$\min_{y=(\ddot{q}, \tau)} \|Ay - a\|^2 \quad \text{s.t.} \quad \begin{bmatrix} M & -S^\top \end{bmatrix} y = -h + J^\top \hat{f}$$

Contact (접촉 제약) 추가

Rigid Contact

- Rigid Contact의 경우 단단한 바닥/벽에 의해 고정되어 움직이지 못하는 상태라 본다
 - > 즉, 속도=0, 가속도=0 과 같은 강한 구속 조건을 추가

기구학적 구속 $c(q) = r_c(q) - r_c^*$ $c(q) = 0$

- $r_c(q)$: 일반화좌표 q 에서 접촉점의 위치/자세
- r_c^* : 고정된(변하지 않는) 목표 위치/자세

Contact (접촉 제약) 추가

속도/가속도 구속

- 1차 미분 : 해당 고정점에서의 속도는 0 이다

$$\frac{d}{dt} c(q(t)) = \frac{\partial c}{\partial q} \dot{q} = \underline{J(q) \dot{q}} = 0$$

- 2차 미분 : 해당 고정점에서의 가속도는 0 이다

$$\frac{d}{dt} (J(q) \dot{q}) = \underline{\dot{J}(q, \dot{q}) \dot{q} + J(q) \ddot{q}} = 0$$

- 운동방정식과 결합한 최종 QP 형태 :

$$\underset{y}{\text{minimize}} \quad \|Ay - a\|^2$$

$$\text{subject to} \quad \begin{bmatrix} J(q) & 0 & 0 \\ M(q) & -J(q)^\top & -S^\top \end{bmatrix} \underset{y}{\begin{bmatrix} \ddot{q} \\ f \\ \tau \end{bmatrix}} = \begin{bmatrix} -\dot{J}(q, \dot{q}) \dot{q} \\ -h(q, \dot{q}) \end{bmatrix}.$$

Contact (접촉 제약) 추가

- 운동방정식과 결합한 최종 QP 형태에서 가속도 조건만 사용하는 이유

$$\begin{aligned} & \underset{y}{\text{minimize}} && \|Ay - a\|^2 \\ & \text{subject to} && \begin{bmatrix} J(q) & 0 & 0 \\ M(q) & -J(q)^\top & -S^\top \end{bmatrix} \begin{bmatrix} \ddot{q} \\ f \\ \tau \end{bmatrix} = \begin{bmatrix} -\dot{J}(q, \dot{q}) \dot{q} \\ -h(q, \dot{q}) \end{bmatrix}. \end{aligned}$$

$$\frac{d}{dt} (J(q) \dot{q}) = \underline{\dot{J}(q, \dot{q}) \dot{q} + J(q) \ddot{q} = 0}$$

- \ddot{q} 에 대한 선형식이기 때문에 결합하기 용이
- 초기 상태(속도)를 0으로 하여 적용

QP 장점

- 지금까지는 QP의 한 형태인 Equality-Constrained LSP (ECLSP)에 대해 배운것

$$\min_y \|Ay - b\|_W^2 \quad \text{s.t.} \quad Ey = d$$

- QP의 일반형 - 일반 이차비용 + **부등식/등식 제약** 모두 가능

$$\min_y \frac{1}{2} y^T H y + c^T y \quad \text{s.t.} \quad Gy \leq h, \quad Ey = d, \quad y_{\min} \leq y \leq y_{\max}$$

제약 :

- ECLSP: 평등제약만(=접촉 가속도 구속처럼 등식)
- QP: 평등 + 불평등(속도/가속도/자세/마찰원뿔 등)을 모두 포함 가능

비용 :

- ECLSP: 오차 제공형 한 종류(가중치로 중요도 조절)
- QP: 오차 제공 + 선형항 등 여러 항을 합성하기 용이

QP 장점

- QP의 가장 큰 장점은 불평등 제약(inequality constraints)을 직접 넣을 수 있다는 것

1) 조인트 토크 한계 $\tau^{\min} \leq \tau \leq \tau^{\max}$

2) 마찰 원뿔 $B f \leq 0$

3) 조인트 position-velocity 한계 $\dot{q}^{\min} \leq \dot{q} \leq \dot{q}^{\max}$

- 실제 로봇은 항상 토크·마찰·가속도·위치/속도 상한이 있어 → **필연적으로 QP**

Thank you