Inverse Dynamics: "Inner Loop / Outer Loop" Approach

(Spong 8.2) ← & Craig 10.4

Lecture 16

Robot Manipulators

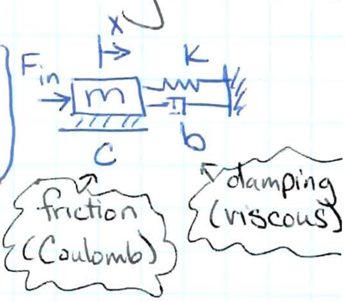
- GEOMETRY affects actuator ⟺ end effector dynamic
- We can use an INNER LOOP to "cancel out" the non-linear robot dynamics

Review!

→ Recall from Lecture 5, "control law partitioning"...

e.g., • Say the EOM (eqn of motion) of a SISO (single-input single-output) system is

$$\boxed{m\ddot{x} = F_{in} - c\cdot \text{sign}(\dot{x}) - b\dot{x} - kx}$$



- To CONTROL F_{in}, break into two parts:

$$\boxed{F_{in} = F_{cancel} + F_{control}},$$

Where

$$F_{cancel} = c\cdot \text{sign}(\dot{x}) + b\dot{x} + kx$$

"cancel out" RHS [righthand side]

leaving:

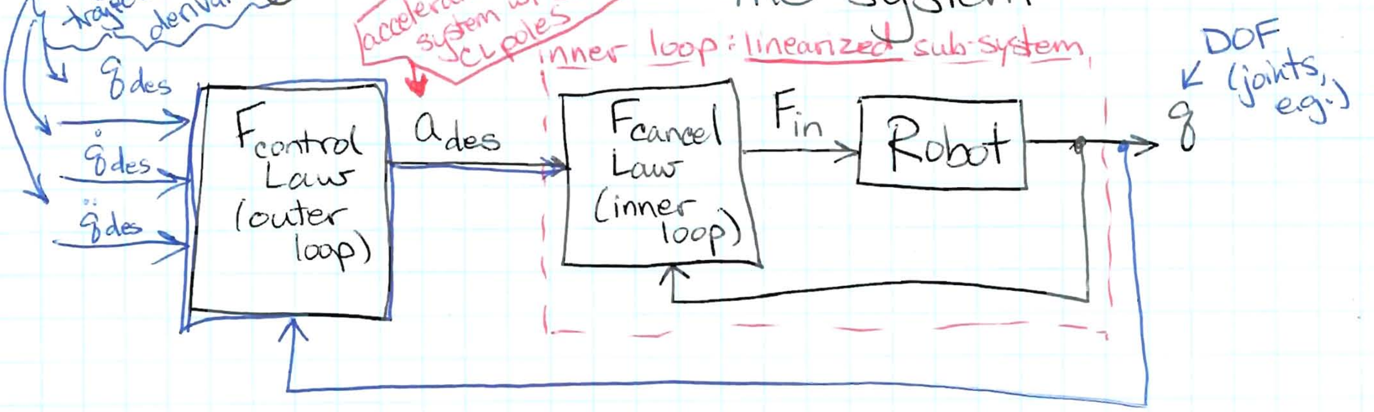
$$m\ddot{x} = (\cancel{F_{cancel}} + F_{control}) \cancel{- c\cdot \text{sign}(\dot{x}) - b\dot{x} - kx}$$

$$\therefore \boxed{m\ddot{x} = F_{control}}$$

1

- This idea can be <u>generalized</u> for the multi-input multi-output (MIMO) case of n actuators controlling n manipulator joints.
- Visually the architecture has:
  - <u>INNER LOOP</u>: to <u>cancel</u> (geometry-dependent) dynamics
  - <u>OUTER LOOP</u>: to set <u>closed-loop poles</u> of the system

pre-computed DESIRED joint trajectories & derivatives



<u>Recall</u>: Jacobian (function of GEOMETRY)

$$\boxed{\dot{\xi}_e = J\dot{q}_a}$$

$$\boxed{\tau_a = J^T F_e}$$

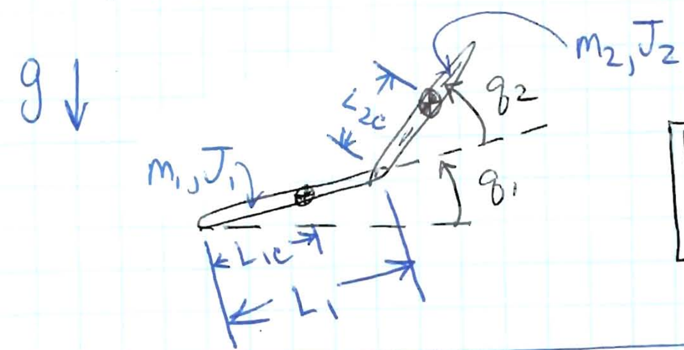
a: actuator
e: end effector

- <u>EOM</u> (e.g., obtained via the <u>LAGRANGIAN</u> approach) can be written in <u>matrix</u> form as:

* Manipulator Eqn's of Motion

$$\boxed{\underbrace{M(q)}_{\text{Inertia (mass) matrix}}\ddot{q} + \underbrace{C(q,\dot{q})\dot{q}}_{\text{Coriolis \& centrifugal forces}} + \underbrace{G(q)}_{\text{Gravity}} + \underbrace{F(q,\dot{q})}_{\text{Friction}} = u}$$

Example: Our familiar two-link arm.



Let:

$$\boxed{h \equiv +m_2 L_1 L_{2c} \sin\theta_2}$$

EOM #1

$$\boxed{M_{11}\ddot{\theta}_1} \rightarrow (m_1 L_{1c}^2 + J_1 + J_2 + m_2(L_1^2 + L_{2c}^2 + 2L_1 L_{2c}\cos\theta_2))\ddot{\theta}_1 + \dots$$

$$\boxed{M_{12}\ddot{\theta}_2} \rightarrow (m_2(L_{2c}^2 + L_1 L_{2c}\cos\theta_2) + J_2)\ddot{\theta}_2 + \dots$$

$$\boxed{G_1} \rightarrow (m_1 L_{1c}\cos\theta_1 + m_2 L_1 \cos\theta_1 + m_2 L_{2c}\cos(\theta_1+\theta_2))g + \dots$$

$$\boxed{C_1\dot{\theta}} \rightarrow (-2h\dot{\theta}_1\dot{\theta}_2 - h\dot{\theta}_2^2) = \tau_1$$

first actuator input, to be set via CONTROL

EOM #2

$$\boxed{M_{21}\ddot{\theta}_1} \rightarrow (J_2 + m_2 L_{2c}^2 + m_2 L_1 L_{2c}\cos\theta_2)\ddot{\theta}_1 + \dots$$

$$\boxed{M_{22}\ddot{\theta}_2} \rightarrow (J_2 + m_2 L_{2c}^2)\ddot{\theta}_2 + \dots$$

$$\boxed{G_2} \rightarrow (m_2 L_{2c}\cos(\theta_1+\theta_2))g + \dots$$

$$\boxed{C_2\dot{\theta}} \rightarrow h\dot{\theta}_1^2 = \tau_2$$

second input (to be set)

In MATRIX form:

$(M_{21} = M_{12})$

$$\underbrace{\begin{bmatrix} M_{11}, & M_{12} \\ M_{12}, & M_{22} \end{bmatrix}}_{M}\underbrace{\begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix}}_{\ddot{q}} + \underbrace{\begin{bmatrix} -h\dot{\theta}_2, & -h(\dot{\theta}_1+\dot{\theta}_2) \\ h\dot{\theta}_1, & 0 \end{bmatrix}}_{C}\underbrace{\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}}_{\dot{q}} + \underbrace{\begin{bmatrix} G_1 \\ G_2 \end{bmatrix}}_{G} = \underbrace{\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}}_{\tau}$$

$$\boxed{M\ddot{q} + C\dot{q} + G = \tau}$$

3

Note:

1. M is symmetric ($M_{12} = M_{21}$).
2. M is <u>invertible</u>! $\boxed{\ddot{q} = M^{-1}(u - C\dot{q} - G)}$ ← EoM
3. Equations depend on geometry (non-linear).
4. Equations tend to be "messy"!

→ $\therefore$ Useful to AUTOMATE generation of the EoM from the Lagrangian (e.g., w/ MATLAB).

(Future lecture topic...)

Inner/Outer Loop Process.

[1] Inner loop – Assume input $\boxed{a_{des}}$ is the "desired joint accelerations," $\ddot{q}$,

– Then, if

$$\boxed{u = M a_{des} + C\dot{q} + G (+F)}$$

← if friction included

"just cancel" dynamics

by inspection, since

$$\boxed{u = M\ddot{q} + C\dot{q} + G + F},$$

the solution will be $\boxed{\ddot{q} = a_{des}}$ given this u.

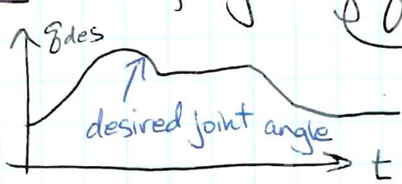
---

[2] Outer loop → $\boxed{\text{How to set } a_{des}?}$

a) Assume we start w/ a <u>desired</u> motion, trajectory q_{des}.

Calculate

$$\boxed{\begin{aligned} \dot{q}_{des} &= \frac{d}{dt} q_{des} \\ \ddot{q}_{des} &= \frac{d}{dt} \dot{q}_{des} \end{aligned}}$$



4

(2: Outer Loop, cont...)

b) Use "PD" control on errors in position & velocity:

Feedback Laws:

Note, if $\dot{q} = \dot{q}_{des}$ & $q = q_{des}$, Then $\ddot{q}_{des} = \ddot{q}_{des}$

$$\boxed{a_{des} = \ddot{q}_{des} + \underbrace{\left\{K_p(q_{des} - q) + K_D(\dot{q}_{des} - \dot{q})\right\}}_{\text{PD feedback for errors...}}}$$

desired

actual (measured, or estimated)

* Recall 2nd-order char. eqn:

$$\boxed{s^2 + \underbrace{2\zeta\omega_n}_{K_d} s + \underbrace{\omega_n^2}_{K_p} = 0}$$

a) Since inner loop ensures $\ddot{q} = a_{des}$ (if NO MODEL ERROR),

$$\ddot{q} = a_{des} = \underbrace{\ddot{q}_{des} + K_p(q_{des} - q) + K_d(\dot{q}_{des} - \dot{q})}_{\text{b) outer loop PD control}}$$

Let error be defined as $\boxed{e = q_{des} - q}$

...Then $\dot{e} = \dot{q}_{des} - \dot{q}$

$\ddot{e} = \ddot{q}_{des} - \ddot{q}$

often, $\tilde{q}$ is used for error, instead of e

Then:

$$\boxed{0 = \ddot{e} + K_d\dot{e} + K_p e}$$

- Pick ζ & ω_n for each joint, e.g., $\zeta = 1$ (or $\zeta = .7$)

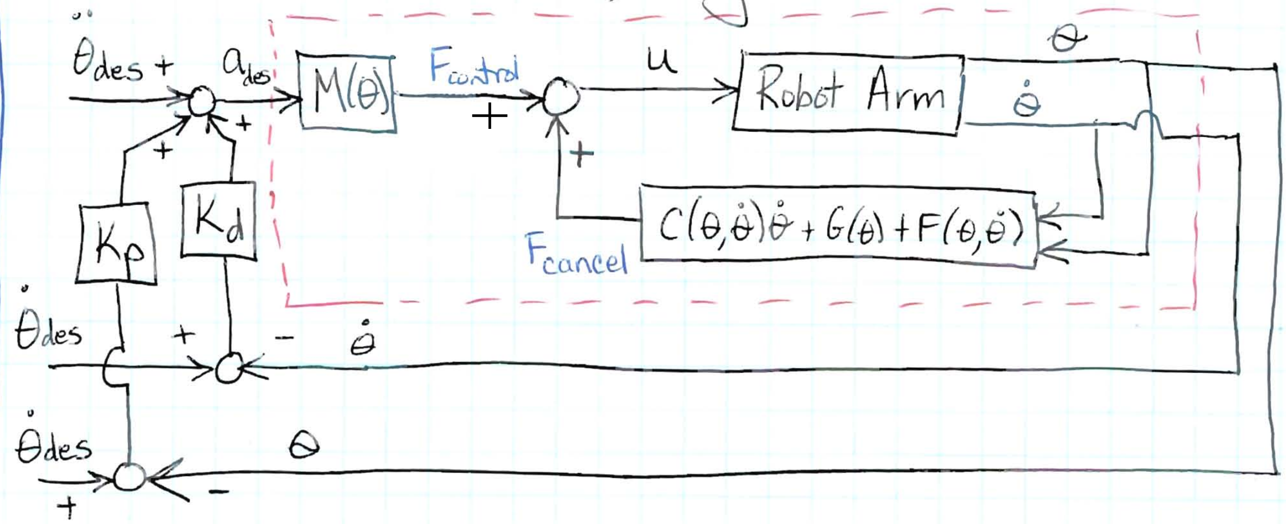
In general, K_p & K_d are $n \times n$ diagonal! $n = 2$ for 2-link arm

$$\boxed{\underset{(n \times n)}{K_p = \begin{bmatrix} \omega_1^2 & 0 \\ 0 & \omega_2^2 \end{bmatrix}}, \quad \underset{(n \times n)}{K_d = \begin{bmatrix} 2\zeta\omega_1 & 0 \\ 0 & 2\zeta\omega_2 \end{bmatrix}}}$$

5

Recall, our control law (inner loop) is $u = M a_{des} + C\dot{q} + G + F$

<u>Non-linear control</u>, diagram: (more detail...)



(Can use "q" or "θ" here. Using θ as a reminder joint vector is <u>usually</u> <u>angles</u>.)

<u>Issues</u>:

(I) <u>Time!</u> <u>Inner loop calculations may be SLOW!</u>

{Imagine DT control with 1 kHz loop... 0.001 sec per step. ☹
May not be able to do the math quickly enough?

$\boxed{\text{Solutions}}$ (perhaps): (A) Assume we <u>pre-compute</u> M, C, G, F for the <u>planned</u> geometries. Then assume small errors, so $M \approx M_d$, $C \approx C_d$, $G \approx G_d$, etc. over time.

(B) Update non-linear matrices at a "slow" rates, but keep dt small (fast servo rate) for u updates (torques).

(II) Unknown model params. e.g. a <u>disturbance torque</u>, τ_d