ECE/ME 179D

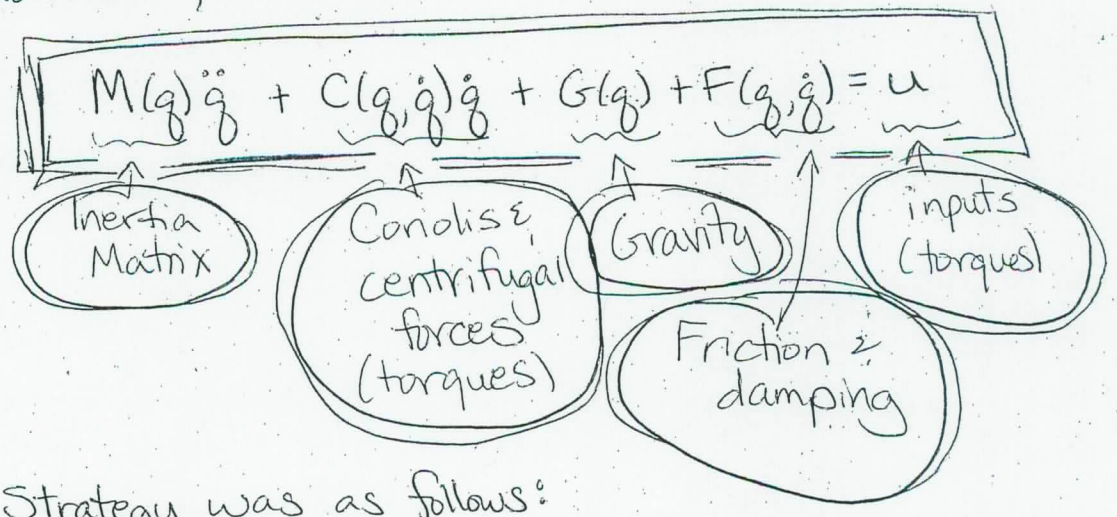
Lecture 17

"Task Space" Inverse Dynamics

Spong 9.3

* Recall from Lecture 16 (& Lect. 5) that (Spong 8.2) we can "cancel out" known plant dynamics.
* Nonlinear, MIMO EOM were written as:

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



* Strategy was as follows:

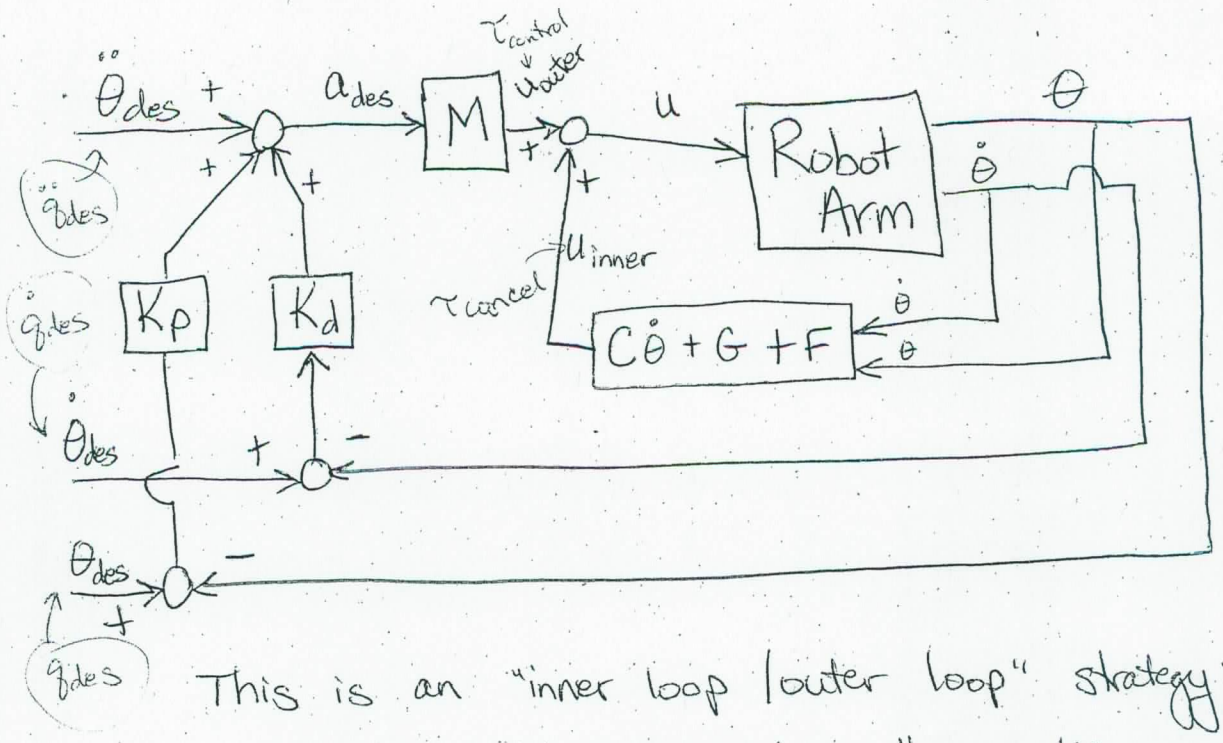
① Set a_{des} via PD control, to track a trajectory (e.g., pre-computed, like the "ball toss" lab)

{ Note: We assume $\theta_d(t)$, $\dot{\theta}_d(t)$ & $\ddot{\theta}_d(t)$ can be "looked up" over time.

② Let the total control involve two parts:

u_{inner}: cancels robot dynamics

u_{outer}: PD control, to drive trajectory errors to zero, via a desired, second-order response.



This is an "inner loop / outer loop" strategy:

- I.L.: inner loop "cancels" the non-linear robot arm dynamics, which depend on configuration (JACOBIAN!) (geometry) & speed.
- O.L.: outer loop ensures we follow a prescribed set of joint trajectories, w/ errors in tracking "dying away" according to the PD gains:

Outer Loop Control Law:

$$a_{des} = \ddot{\theta}_d + K_d(\dot{\theta}_d - \dot{\theta}) + K_p(\theta_d - \theta)$$

Let error, e, be: $e \equiv \theta_d - \theta$, then

$$0 = (\ddot{\theta}_d - a_{des}) + K_d(\dot{\theta}_d - \dot{\theta}) + K_p(\theta_d - \theta)$$

The inner loop ensures that $\ddot{\theta} = a_{des}$

$$0 = \ddot{e} + K_d\dot{e} + K_p e \quad \leftarrow \text{If } \ddot{\theta} = a_{des}$$

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

K_d: $2\zeta\omega_n = K_d$

K_p: $\omega_n^2 = K_p$

Error dynamics, when plant dynamics are truly "Canceled"

$$0 = s^2 + 2\zeta\omega_n s + \omega_n^2$$

Note: These are dynamics of errors in joints, not of the end effector...

→ What about end effector dynamics??

→ What if we want to track in TASK SPACE?

Two options are:

① "inverse Jacobian" (Spong 8.2.2; Craig 10.8)

J^{-1}

→ Use same I.L. control

→ Modify O.L. to track X_{des} (not q_{des}) & set dynamics of errors in X (instead of errors in q)

$$\tilde{X} = X_{des} - X$$

errors in task space

or

(2) "transpose Jacobian" (Spong 9.3.1-9.3.2)

a) plan dynamics in task space (9.3.2)
→ "impedance control" (Craig 10.8)

b) include reaction torque "$J^T F_e$" (9.3.1)
to account for e.e. (end effector) forces, F_e.

$$M\ddot{q} + C\dot{q} + G + J^T F_e = u$$

(add in new terms)

[c) "hybrid" → hybrid impedance control ... (9.3.3 Spong)]

- Jacobian: Recall, the matrix $J(q)$ describes both velocity relationship and STATIC force & torque relationship.

e: end effector
a: actuator

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

$$\tau_a = J^T F_e$$

Requires invertible J! Cannot "blow up".

$$\dot{x} = J\dot{q}$$

$$\ddot{x} = J\ddot{q} + \dot{J}\dot{q}$$

$$\therefore (\ddot{x} - \dot{J}\dot{q}) = J\ddot{q}$$

$$(a_q)_{des} = \ddot{q} = J^{-1}\{(a_x)_{des} - \dot{J}\dot{q}\}$$

Before, we used PD to set a desired $\ddot{q} \rightarrow$ "$(a_q)_{des}$"!!!

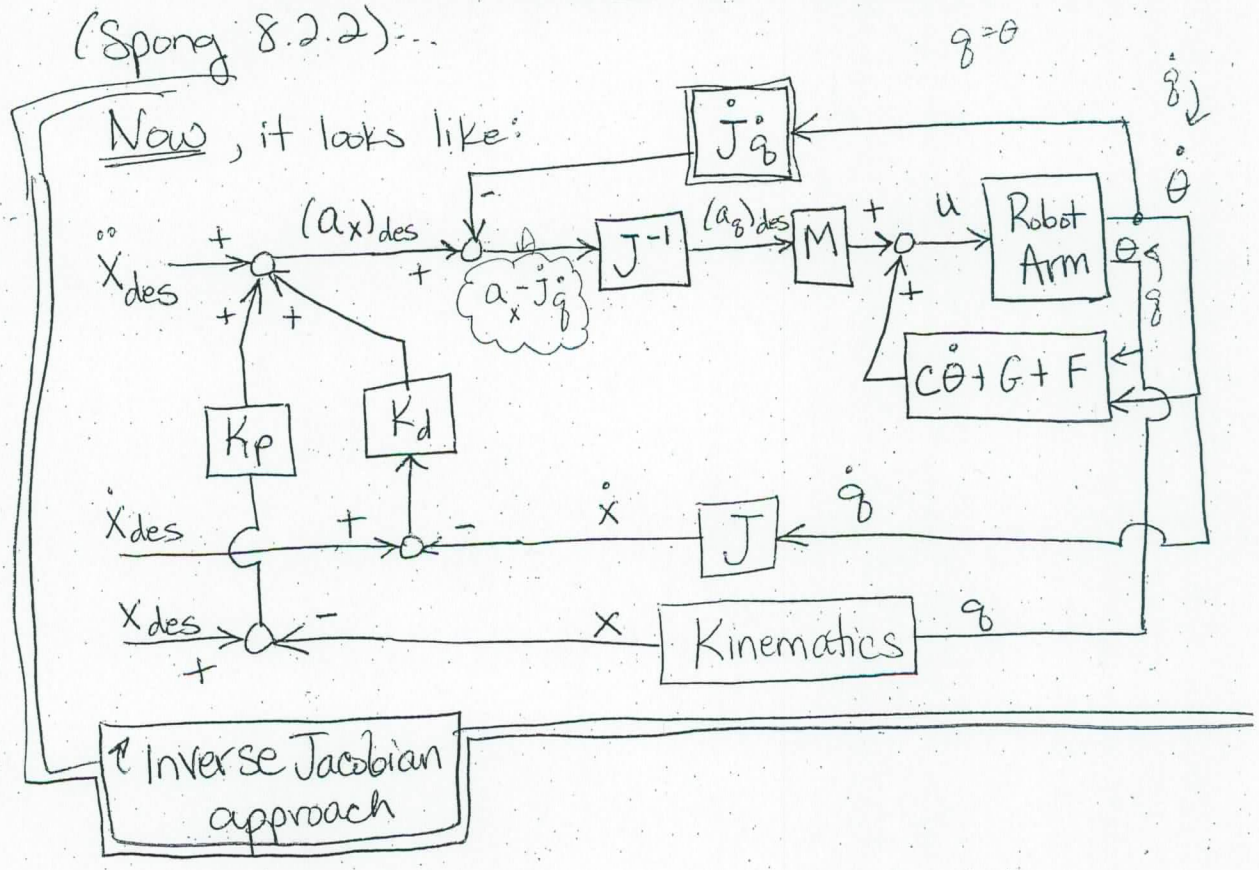
... Now, use PD to set desired $\ddot{x}_e$ dynamics, via $(a_x)_{des}$

- Then, outer loop shapes errors on <u>task space</u> coordinates (instead of joint space variables in q).

$$(a_x)_{des} = \ddot{x}_{des} + K_d(\dot{x}_{des} - \dot{x}) + K_p(x_{des} - x)$$

(Spong 8.2.2)...

<u>Now</u>, it looks like:



* <u>Note</u>: Now K_p & K_d set the dynamics of errors in <u>TASK SPACE</u>.

Transpose Jacobian

a) Plan dynamics in task space:

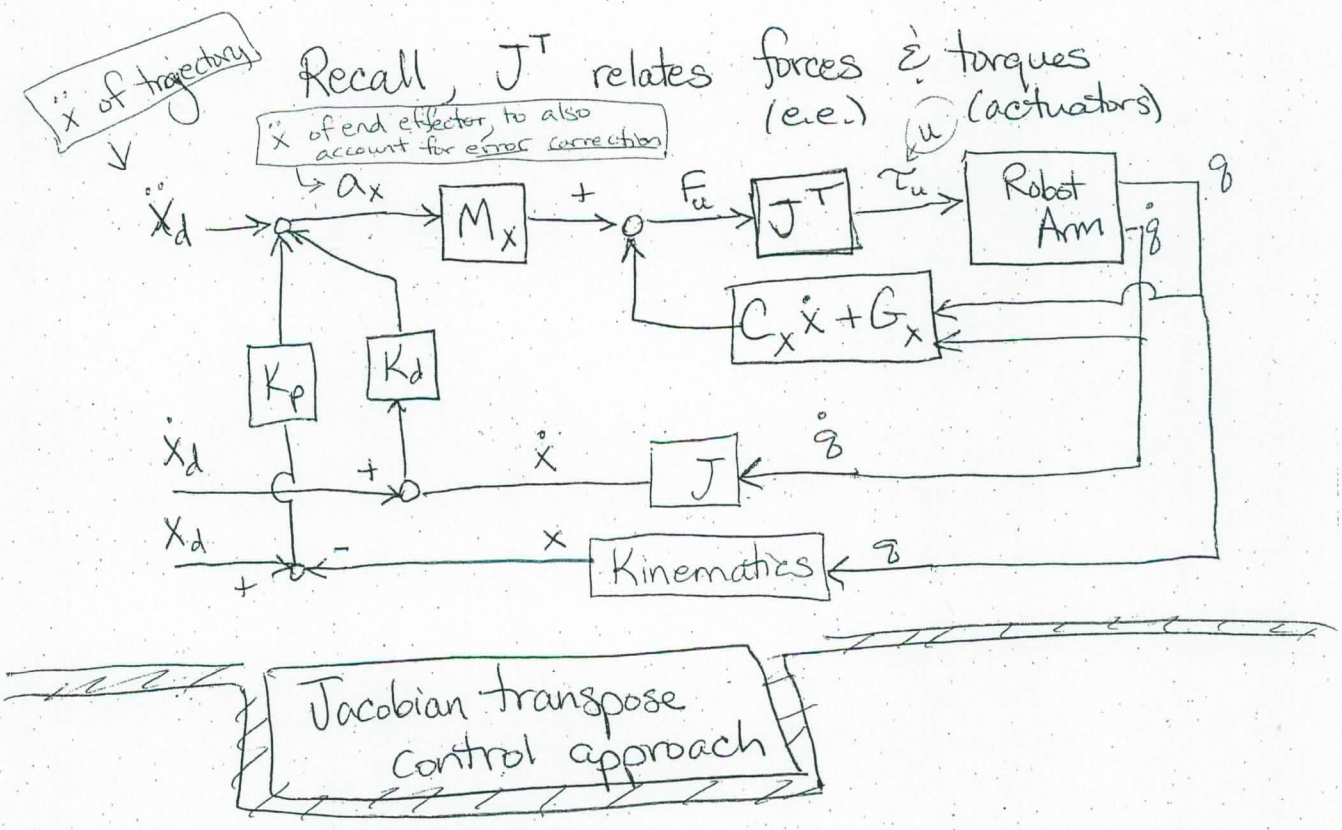
(Spong 8.2.2)
(Craig 10.8)

$$F_u = M_x \ddot{x} + C_x \dot{x} + G_x$$

wrt x, not q!

F_u: Force (not torque)

$\ddot{x}$, $\dot{x}$: task space variables in x (not wrt q)



Jacobian transpose control approach

Side note: These methods can be computationally expensive! In practice, we may only update nonlinear term in θ $\dot{\theta}$ (q $\dot{q}$) at a slower rate than we command updates to torques, u.

(2.5 - Jac. transpose...)

- Finally, what if there is some environment ("disturbance") force, F_e, at the e.e.?
- The torque to overcome this is:

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

Now,

A. $\boxed{M\ddot{q} + C\dot{q} + G = J^T F_e + u}$ EOM

- As before, the "trick" is to plan for some a_q in the outer loop.

Recall: $\dot{x} = J\dot{q}$

1. $\ddot{x} = J\ddot{q} + \dot{J}\dot{q}$
2. $\therefore \boxed{a_x = Ja_q + \dot{J}\dot{q}}$

Combine (1. minus 2.):

$$(\ddot{x} - a_x) = (J\ddot{q} - Ja_q)$$

$$\boxed{\ddot{x} = a_x + J\ddot{q} - Ja_q}$$

Control Law:

B. $\boxed{u = Ma_q + C\dot{q} + G - J^T f_c}$

So, $M\ddot{q} - J^T F_e = Ma_q - J^T f_c$ (from A. & B.)

$$\therefore \boxed{\ddot{q} = a_q + M^{-1}J^T(F_e - f_c)}$$

So $\boxed{J\ddot{q} = Ja_q + (JM^{-1}J^T)(F_e - f_c)}$

actual force; estimated force

So: $\ddot{x} = a_x + J\ddot{q} - Ja_q$

becomes:

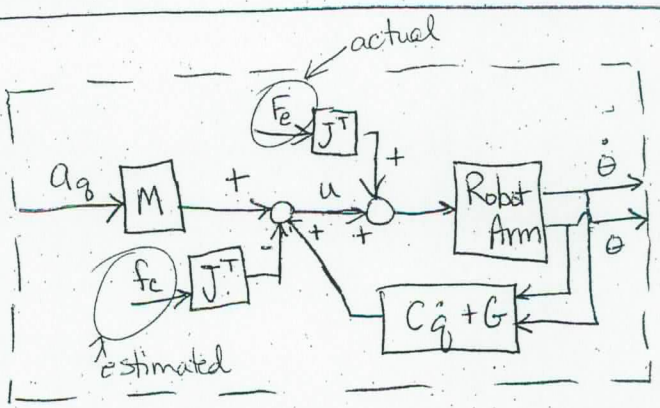
$$\ddot{x} = a_x + Ja_q + (JM^{-1}J^T)(F_e - f_c) - Ja_q$$

mobility tensor → $W \equiv JM^{-1}J^T$

* If $F_e = f_c$, then $\ddot{x} = a_x$

(F_e: actual; f_c: estimated)

Note, a_q can be set as before (either to track θ_{des} or x_{des})...



Issue: When parameters are NOT KNOWN EXACTLY, we will see persistent errors (in joints).

→ We can USE these errors to "adjust" parameters. "Learn" params that drive error to ZERO. This is Adaptive Control.