Material Covered up until now,

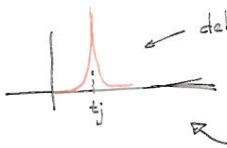
forward kinematics	• how to determine end-effector configuration given joint-configuration
inverse kinematics	• " joint configuration " end-effector configuration
Jacobian (manipulator)	• relate configuration velocities to joint velocities. (end-effector)
exp & ln	• convert configuration differences into velocities/trajectories.

⇒ GOAL: how do we design realistic/feasible trajectories connecting different configurations.

↑ ① joint-configurations.
② group configurations.
③ both levels? → sometimes not possible using only one configuration space. may need to switch between the two.

Trajectory design

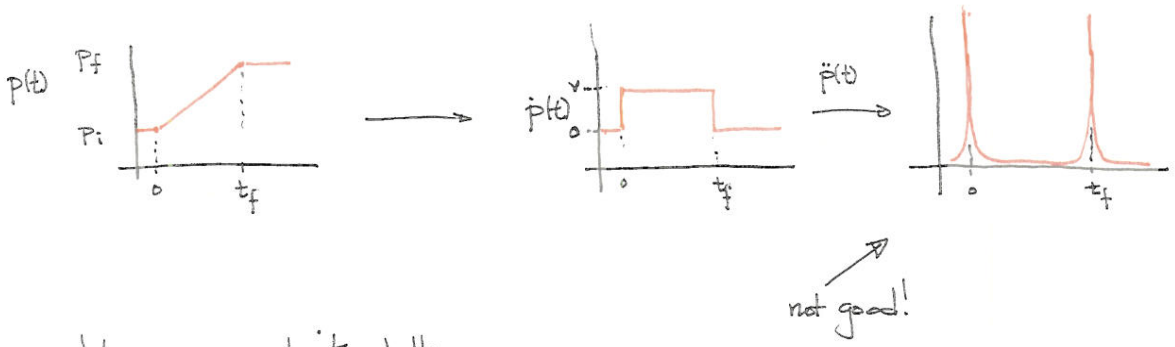
* now, considers the time history: position, velocity, and/or acceleration.
* want sufficiently smooth trajectories (cts. derivatives)
  - no jerky motions → { coarse wear; induce vibrations (excite resonances)



* may require intermediate points between initial and final configurations.
  (AKA · way-points, via-points)

So, how are trajectories generated/designed?

simplest approach is linear (1D)



need to manage velocity better.

let's add velocity constraints on initial & final configurations.

Cubic polynomials/splines

B-Splines

Four constraints on polynomials:

1) initial point p_i
2) final point p_f
3) initial velocity $\dot{p}_i$
4) final velocity $\dot{p}_f$

$\Rightarrow$

Cubic polynomial required (at least)

$$p(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

such that:

$$p(0) = p_i$$

$$p(t_f) = p_f$$

$$\dot{p}(0) = \dot{p}_i = 0$$

$$\dot{p}(t_f) = \dot{p}_f = 0$$

Note:

$$\dot{p}(t) = a_1 t + 2a_2 t + 3a_3 t^2$$

$$\ddot{p}(t) = 2a_2 + 6a_3 t$$

$\Rightarrow$

$$p(0) = a_0 = p_i$$

$$p(t_f) = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 = p_f$$

$$\dot{p}(0) = a_1 = 0$$

$$\dot{p}(t_f) = a_1 + 2a_2 t_f + 3a_3 t_f^2 = 0$$

$\Rightarrow$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \vec{a} = \begin{Bmatrix} p_i \\ p_f \\ 0 \\ 0 \end{Bmatrix} = \vec{P}_0$$

$\Rightarrow$

$$\vec{a} = P(t_f)\,\vec{P}_0 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ -3/t_f^2 & 3/t_f^2 \\ 2/t_f^3 & -2/t_f^3 \end{bmatrix} \begin{Bmatrix} p_i \\ p_f \end{Bmatrix}$$

smaller matrix & vector since last two components vanish.

$a_0 = p_i \qquad a_2 = 3/t_f^2 (p_f - p_i)$

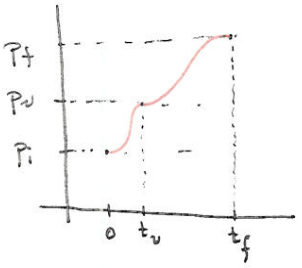
$a_1 = 0 \qquad a_3 = 2/t_f^3 (p_i - p_f)$

What value should t_f take?

- certainly should be some short enough time, so trajectory doesn't take forever
- not too short: actuator limits.
  end-effector velocity limits.
- have decent initial guess based on limits, $v \cdot t = d$ (d: distance, t: time, v: velocity)

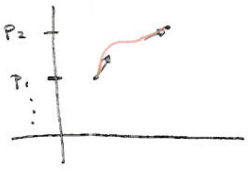
What if there are way points?

- do not want to connect as a series of cubic polynomials w/ zero velocity conditions



← slows down at this point.
is there a need to?

- use non-trivial velocities at way points.



achieve end-point constraints + match velocities

$$p(0) = p_i \qquad p(t_f) = p_f$$
$$\dot{p}(0) = \dot{p}_i \qquad \dot{p}(t_f) = \dot{p}_f$$

$\Rightarrow$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{Bmatrix} = \begin{Bmatrix} p_i \\ p_f \\ \dot{p}_i \\ \dot{p}_f \end{Bmatrix}$$

$\Rightarrow$

$$\begin{Bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3/t_f^2 & 3/t_f^2 & -2/t_f & -1/t_f \\ 2/t_f^3 & -2/t_f^3 & 1/t_f^2 & 1/t_f^2 \end{bmatrix} \begin{Bmatrix} p_i \\ p_f \\ \dot{p}_i \\ \dot{p}_f \end{Bmatrix}$$

$\Rightarrow$

$$\vec{a} = P(t_f)\,\vec{p}_0$$

$$a_0 = p_i \qquad a_1 = \frac{3}{t_f^2}(p_f - p_i) - \frac{2}{t_f}\dot{p}_i - \frac{1}{t_f}\dot{p}_f$$

$$a_2 = \dot{p}_i \qquad a_3 = \frac{2}{t_f^3}(p_i - p_f) + \frac{1}{t_f^2}(\dot{p}_i + \dot{p}_f)$$

- connecting in this manner matches velocities, but not accelerations
  - if accelerations close enough, not a problem
  - if far apart, then may lead to jerky motions.

Let's connect two cubic splines w/a cts acceleration profile.

achieve positions p_i, p_v, p_f at times $0, t_v, t_f$

$$f(t) = \begin{cases} p(t) & \text{if } t \in [0, t_v] \\ q(t - t_v) & \text{if } t \in (t_v, t_f) \end{cases}$$

where,

$$p(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \quad , \quad t \in [0, t_v]$$

$$q(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3 \quad , \quad t \in [0, t_f - t_v]$$

Constraints are:

position:

$$p(0) = p_i$$

$$p(t_v) = p_v$$

$$q(0) = p_v$$

$$q(t_f - t_v) = p_f$$

velocity:

$$\dot{p}(0) = 0$$

$$\dot{q}(t_f - t_v) = 0$$

$$\dot{p}(t_v) = \dot{q}(0)$$

acceleration:

$$\ddot{p}(t_v) = \ddot{q}(0)$$

matching conditions

define: $t_1 = t_v$, $t_2 = t_f - t_v$

$\Rightarrow$

$$p(0) = a_0 = p_i$$

$$p(t_1) = a_0 + a_1 t_1 + a_2 t_1^2 + a_3 t_1^3 = p_v$$

$$q(0) = b_0 = p_v$$

$$q(t_2) = b_0 + b_1 t_2 + b_2 t_2^2 + b_3 t_2^3 = p_f$$

$$\dot{p}(0) = a_1 = 0$$

$$\dot{q}(t_2) = b_1 + 2b_2 t_2 + 3b_3 t_2^2 = 0$$

$$\dot{p}(t_1) - \dot{q}(0) = a_1 + 2a_2 t_1 + 3a_3 t_1^2 - b_1 = 0$$

$$\ddot{p}(t_1) - \ddot{q}(0) = 2a_2 + 6a_3 t_1 - 2b_2 = 0$$

$\Rightarrow$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & t_1 & t_1^2 & t_1^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & t_2 & t_2^2 & t_2^3 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2t_2 & 3t_2^2 \\ 0 & 1 & 2t_1 & 3t_1^2 & 0 & -1 & 0 & 0 \\ 0 & 0 & 2 & 6t_1 & 0 & 0 & -2 & 0 \end{bmatrix} \vec{a} = \begin{Bmatrix} p_i \\ p_v \\ p_v \\ p_f \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} = \vec{p}_0$$

$\Rightarrow$

$$\vec{a} = A^{-1}(t_1, t_2)\,\vec{p}_0 = P(t_1, t_2)\,\vec{p}_0$$

↳ note structure of $\vec{p}_0$ allows for simplification.

if we simplify things, we get

$$\vec{a} = P_{simp}(t_1, t_2) \begin{Bmatrix} p_i \\ p_u \\ p_f \end{Bmatrix}$$

where

$$P_{simp}(t_1, t_2) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ -\frac{3(2t_1+t_2)}{2t_1^2(t_1+t_2)} & \frac{3(t_1+t_2)}{2t_1^2 t_2} & -\frac{3}{2t_2(t_1+t_2)} \\ \frac{4t_1+t_2}{2t_1^3(t_1+t_2)} & -\frac{3t_1+t_2}{2t_1^3 t_2} & \frac{3}{2t_1^2 t_2 + 2t_1 t_2^2} \\ 0 & 1 & 0 \\ -\frac{3t_2}{2t_1(t_1+t_2)} & \frac{3}{2}\left(\frac{1}{t_1} - \frac{1}{t_2}\right) & \frac{3t_1}{2t_2(t_1+t_2)} \\ \frac{3}{t_1^2+t_1 t_2} & -\frac{3}{t_1 t_2} & \frac{3}{t_1 t_2 + t_2^2} \\ -\frac{3}{2t_1^2 t_2 + 2t_1 t_2^2} & \frac{t_1+3t_2}{2t_1 t_2^3} & -\frac{t_1+4t_2}{2t_2^3(t_1+t_2)} \end{bmatrix}$$

if $t_1 = t_2 = t \Rightarrow \quad 0, t, 2t$

we get

$$P_{simp}(t,t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ -\frac{9}{4t^2} & \frac{3}{t^2} & -\frac{3}{4t^2} \\ \frac{5}{4t^3} & -\frac{2}{t^3} & \frac{3}{4t^3} \\ 0 & 1 & 0 \\ -\frac{3}{4t} & 0 & \frac{3}{4t} \\ \frac{3}{2t^2} & -\frac{3}{t^2} & \frac{3}{2t^2} \\ -\frac{3}{4t^3} & \frac{2}{t^3} & -\frac{5}{4t^3} \end{bmatrix}$$

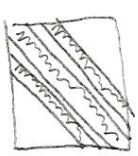
this is much easier to deal with.

* The above procedure can be implemented for n waypoints

Leads to a $[4(n+1) \times 4(n+1)]$ matrix which is invertible.

Has a tri-diagonal form $\Rightarrow$ ① sparse matrix

② there are fast solvers.



Suppose have manipulator:

$\left(\frac{\pi}{3}, -\frac{\pi}{8}, -\frac{\pi}{2}\right)$ $\left(\frac{\pi}{8}, \frac{\pi}{6}, \frac{\pi}{2}\right)$

max speed something like $\frac{\pi}{4}$ rad/sec.

$\Rightarrow$

$$\frac{\frac{\pi}{2}}{\frac{\pi}{4}} = 4 \text{ sec.}$$

$$t_f = 4 \text{ sec.}$$

$\Rightarrow$

$$\theta_1(t) = \frac{\pi}{3} - \frac{5\pi}{128} t^2 + \frac{5\pi}{768} t^3$$

$$\theta_2(t) = -\frac{\pi}{8} + \frac{7\pi}{128} t^2 - \frac{7\pi}{768} t^3$$

$$\theta_3(t) = -\frac{\pi}{2} + \frac{3\pi}{16} t^2 - \frac{\pi}{32} t^3$$

if we examine $\dot{\theta}_3(t)$, it actually goes $> \frac{\pi}{4}$ rad/sec.

need 7 seconds.

$$\theta_1(t) = \frac{\pi}{3} - \frac{5\pi}{392} t^2 + \frac{5\pi}{4116} t^3$$

$$\theta_2(t) = -\frac{\pi}{8} + \frac{\pi}{5} t^2 - \frac{\pi}{588} t^3$$

$$\theta_3(t) = -\frac{\pi}{2} + \frac{3\pi}{49} t^2 - \frac{2\pi}{343} t^3$$

Multiple Waypoints, Equidistant in Time

$$p^i(t) = a_0^i + a_1^i t + a_2^i t^2 + a_3^i t^3$$

$$f(t) = p^k(t - kT) \quad \text{where} \quad k = \left\lfloor \frac{t}{T} \right\rfloor = \text{floor}\left(\frac{t}{T}\right) \cdot T$$

Points are $P_0, P_1, \ldots, P_n, P_{n+1}$

position constraints

$$p^1(0) = a_0^1 = P_0$$
$$p^1(T) = a_0^1 + a_1^1 T + a_2^1 T^2 + a_3^1 T^3 = P_1$$
$$\vdots$$

2(n+1)

$$p^k(0) = a_0^k = P_{k-1}$$
$$p^k(T) = a_0^k + a_1^k T + a_2^k T^2 + a_3^k T^3 = P_k$$
$$\vdots$$
$$p^{n+1}(0) = a_0^{n+1} = P_n$$
$$p^{n+1}(T) = a_0^{n+1} + a_1^{n+1} T + a_2^{n+1} T^2 + a_3^{n+1} T^3 = P_{n+1}$$

velocity constraints:

boundary conditions → $\dot{p}^1(0) = a_0^1 = 0$

n+2

matching:

$$\dot{p}^1(T) - \dot{p}^2(0) = a_1^1 + 2a_2^1 T + 3a_3^1 T - a_1^2 = 0$$
$$\vdots$$
$$\dot{p}^k(T) - \dot{p}^{k+1}(0) = a_1^k + 2a_2^k T + 3a_3^k T - a_1^{k+1} = 0$$
$$\vdots$$
$$\dot{p}^n(T) - \dot{p}^{n+1}(0) = a_1^n + 2a_2^n T + 3a_3^n T - a_1^{n+1} = 0$$

$$\dot{p}^{n+1}(T) = a_1^{n+1} + 2a_2^{n+1} T + 3a_3^{n+1} T^2 = 0$$

acceleration constraints:

n $\quad \ddot{p}^k(T) - \ddot{p}^{k+1}(0) = 2a_2^k + 6a_3^k T - 2a_2^{k+1} = 0$

$\downarrow$

Total = $4(n+1)$ equations $\qquad$ $4(n+1)$ variables

$\Rightarrow$

$[4(n+1) \times 4(n+1)]$ matrix.

This determines a curve with n waypoints between initial and final points.

If velocity is known, then can do differently.

- ORGANIZATION TO OBTAIN TRIDIAGONAL MATRIX.
- SPARSE MATRIX SOLVERS.

Multiple Waypoints w/ desired velocities.

- easier solve, easier setup
- possible discuss acceleration. → allow controller to handle this.

suppose we have $p_0, p_1, \dots, p_n, p_{n+1}$

$0, \dot{p}_1, \dots, \dot{p}_n, 0$

max times: $T_0, T_1, \dots, T_n, T_{n+1}$ of polynomials.

$t_0, t_1, \dots, t_n, t_{n+1}, t_f$ $\quad T_k = t_{k+1} - t_k$

$T_{n+1} = t_f - t_{n+1}$

recall equations

$$p^1(0) = a_0^1 \qquad = p_0$$

$$p^1(T_1) = a_0^1 + a_1^1 T_1 + a_2^1 T_1^2 + a_3^1 T_1^3 \qquad = p_1$$

$$\dot{p}^1(0) = a_1^1 \qquad = 0$$

$$\dot{p}^1(T_1) = a_1^1 + 2a_2^1 T_1 + 3a_3^1 T_1^2 \qquad = \dot{p}_1$$

$$\vdots$$

$$p^k(0) = a_0^k \qquad = p_{k-1}$$

$$p^k(T_k) = a_0^k + a_1^k T_k + a_2^k T_k^2 + a_3^k T_k^3 \qquad = p_k$$

$$\dot{p}^k(0) = a_1^k \qquad = \dot{p}_{k-1}$$

$$\dot{p}^k(T_k) = a_1^k + 2a_2^k T_k + 3a_3^k T_k^2 \qquad = \dot{p}_k$$

$$\vdots$$

also $[4(n+1) \times 4(n+1)]$ matrix.

$$p^{n+1}(0) = a_0^{n+1} \qquad = p_n$$

$$p^{n+1}(T_{n+1}) = a_0^{n+1} + a_1^{n+1} T_{n+1} + a_2^{n+1} T_{n+1}^2 + a_3^{n+1} T_{n+1}^3 \qquad = p_{n+1}$$

$$\dot{p}^{n+1}(0) = a_1^{n+1} \qquad = \dot{p}_n$$

$$\dot{p}^{n+1}(T_{n+1}) = a_1^{n+1} + 2a_2^{n+1} T_{n+1} + 3a_3^{n+1} T_{n+1}^2 \qquad = 0$$

Manipulator Trajectories.

if we have two joint configurations $\vec{\theta}_1$ and $\vec{\theta}_2$,
we can connect with a sufficiently smooth curve;

$$\vec{\theta}(t), \qquad \vec{\theta}_1 = \vec{\theta}(0) \;\&\; \vec{\theta}_2 = \vec{\theta}(t_f)$$

- does curve satisfy actuator limits?
  possible to check by finding peak velocity or acceleration.
  - if not just dilate time until velocity is low enough.

- sometimes need to check that curve does not violate any workspace constraints
  - if so, need to modify curve.
    can add waypoints that force trajectory to respect workspace constraints.

- may need to refer to associated $\dot{g}$ or (ξ^b)

  use Jacobian $\dot{g} = J(\theta)\dot{\theta}$.

What if trajectory is given by $g(t)$?

can convert to trajectory in joint space by using inverse kinematics,

$\vec{\theta}(t) = \theta_e(g(t))$ $\quad \theta_e$: configuration $\rightarrow$ joint angles.

$g_e \circ \theta_e = id$

$\theta_e \circ g_e = id$?

↳ recall that there may be multiple solutions.

may need to break into segments & use a local joint controller.

or if $g(t) = \exp(\hat{\xi} t)$, then

$$\xi^b = J(\theta)\dot{\theta}$$

$$\Rightarrow \quad \dot{\theta} = J^{\#}(\theta)\xi^b \qquad \theta_0 = \theta_e(g_0)$$

↳ pseudo-inverse.

↳ integrate for $t \in [0, t_f]$

- PROBLEM: the further away the point, the worse the trajectory is.

Manipulator
Trajectories

can define a trajectory in joint space or in group (end-effector configuration) space.

recall that velocities are related via Jacobian:

$$\dot{q} = J(\theta)\dot{\theta}$$

↑ will have invertible subspace

① if J is square and invertible, then

$$\dot{\theta} = J^{-1}(\theta)\,\dot{q}$$

else

$$\dot{\theta} = J^{\#}(\theta)\dot{q}$$

↑ pseudo-inverse
"least squares" fit to $\dot{\theta}$ given $\dot{q}$.

$$J^{\#} = (J^T J)^{-1} J^T \quad \text{if on left}$$

$J($

Redundant manipulators

m = dimension of task space

n = # independent variables (joints)

kinematically insufficient if $m > n$.

kinematically redundant if $m < n$.

Given $g_e^*(t)$, trajectory generation problem for robotic manipulators is to find $\vec{\theta}(t)$ such that the actual trajectory $g_e(t)$ matches the desired trajectory $g_e^*(t)$ of the end-effector.

- common approach is to "spline" together the inverse kinematic solution for a set of knot points to get a joint trajectory
  1. solve inverse kinematic problem at each $g^*(t_k)$ to get $\vec{\theta}^*(t_k)$, $t_0, \ldots, t_n$ = # knot points.
  2. interpolate smooth trajectories $\vec{\theta}(t)$ through $\vec{\theta}(t_k)$.

- problems may occur when choosing joint trajectory since inverse is not unique for redundant manipulators.

Inverting the Manipulator Jacobian.

$$\dot{g} = J(\theta)\,\dot{\theta} \qquad \text{OR} \qquad \xi^b = J^b(\theta)\dot{\theta}\,, \quad \xi^s = J^s(\theta)\dot{\theta}$$

↑ better because ξ really is a vector.

or, can use hybrid Jacobian,

$$\xi^h = J^h(\theta)\,\dot{\theta}$$

hybrid

$$\xi^h = \begin{Bmatrix} \dot{p} \\ \omega^s \end{Bmatrix}$$ ← this is position velocity of $J(\theta)$

↑ this is angular velocity vector from spatial Jacobian, $J^s(\theta)$.

- this is what most books use

recall,

$$J^h(\theta) = Ad_R J^b(\theta)\,,$$

where $g_e(\theta) = (p(\theta), R(\theta))$.

let's work in body coordinates,

~~$\xi^b = J^b(\theta)\dot{\theta}$~~ $\qquad \xi^b = J_{body}(\theta)\,\dot{\theta}$

To get joint velocities from body velocities,

$$\dot{\theta} = J_{body}^{-1}(\theta)\,\xi^b$$

- but, is $J_{body}(\theta)$ really invertible?

A: depends on the manipulator

let m = dimension of task space

n = # independent variables (joints)

1) kinematically insufficient if $m > n$
   - ~~need~~ best to reduce dimension of task space.

2) kinematically redundant if $m < n$
   - it's tricky, but we can handle this case.

3) kinematically sufficient if $m = n$
   - but we need to make sure we are in dexterous workspace for maximal mobility/control.
   - e.g. away from singularities.

How do we handle the Jacobian in these cases?

1) reduce dimension until Jacobian is invertible or use pseudo-inverse.

2) use pseudo-inverse (a.k.a. Moore-Penrose inverse)

3) is invertible, so long as we are away from singularity.

Linear Algebra Preliminaries to solution.

a) rowspace of A, denoted row(A), and the null space of A, denoted null(A), are orthogonal.

b) any vector $\vec{y}$ can be decomposed into row(A) & null(A) components,

$$\vec{y} = \vec{y}_{row} + \vec{y}_{null}$$

c) $$A\vec{y} = A(\vec{y}_{row} + \vec{y}_{null}) = A\vec{y}_{row}$$

d) All solutions y^* to $\vec{b} = A\vec{y}^*$ have the same row-space component $\vec{y}^*_{row}$.

- difference between two solutions y_1^* & y_2^* is in the amount of null space contribution.

e) by (a) $$\|\vec{y}\|^2 = \|\vec{y}_{row} + \vec{y}_{null}\|^2 = \|\vec{y}_{row}\|^2 + \|\vec{y}_{null}\|^2$$

f) the Moore-Penrose solution is the solution w/ no null-space component

$$\vec{y} = A^\dagger b, \quad \vec{y} \in \text{row}(A).$$

OK, so what is it?

To compute pseudo-inverse, need to check out different cases.

(A) $m=n$, then $A^{\#} = A^{-1}$.

& A full-rank

(C) $m<n$, then $A^{\#} = A^T(AA^T)^{-1}$

& A is full-rank

note that $AA^{\#}A = AA^T(AA^T)^{-1}A = A$

$A^{\#}AA^{\#} = A^{\#}AA^T(AA^T)^{-1} = A^{\#}$

$(AA^T(AA^T)^{-1})^T = ((AA^T)^{-1})^T A^{\#}A = (A^{\#})^T A^T$

$= (AA^T)^{-1}(AA^T) = I$

$(A^T(AA^T)^{-1}A)^T = A^T(AA^T)^{-1}A = A^{\#}A$

$m[\ ^n\ ]$

$\Rightarrow AA^T = m[\ ^n\ ]\, n[\ ^m\ ] = m[\ ^m\ ]$

(B) $m>n$, then $A^{\#} = (A^TA)^{-1}A^T$

$n[\ \hat{}\ ]$

$A^TA = n[\ ^m\ ]\, m[\ ^n\ ] = n[\ ^n\ ]$

OK, so what is this pseudo-inverse thing?

It is defined by its properties.

Suppose we seek

$$\vec{b} = A\vec{y} \qquad b \in \mathbb{R}^m,\ y \in \mathbb{R}^n,\ A \in \mathbb{R}^{m \times n}$$

Moore & Penrose found a solution A^+ (or $A^\#$) such that:

~~$A^\#$ is the unique matrix with the properties.~~

1) $A A^\# A = A$

2) $A^\# A A^\# = A^\#$

3) $(A A^\#)^T = A A^\#$

4) $(A^\# A)^T = A^\# A$

Interpretation:

a) if $m = n$ & A is full-rank, then $A^\# = A^{-1}$.

b) if $m > n$ the $\vec{y} = A^\# \vec{b}$ is solution that minimizes $\|\vec{b} - A\vec{y}\|_2$

c) if $m < n$ then $\vec{y} = A^\# \vec{b}$ is solution that minimizes $\|y^T y\|$

so, now we have

$$\dot{\vec{\theta}} = J^{\#}_{body}(\vec{\theta}) \cdot \xi^{b}$$

appropriate inverse is chosen based on particulars of $J_{body}(\vec{\theta})$.