ECE 4560

Manipulator Trajectories:

Covered so far:
- given two joint configurations α_i and α_f, they can be connected with a sufficiently smooth curve, described using a polynomial of the appropriate degree.
- the duration of the trajectory (in time) is a design parameter used to satisfy actuation limits.

Reasons to complicate this:
- desired end-points given in the end-effector group space.
  - not a unique curve connecting end-points in a group space.
  - dependence on joint space is non-linear.
- if the workspace has obstacles, need to ensure that they are avoided.
- there may be other workspace constraints (other than obstacles) that limit the space of feasible trajectories

Question: OK, so what if the trajectory is given/assigned in the configuration space?
How does one obtain a trajectory in the joint space?
→ I need to find a joint space trajectory $\alpha^*(t)$ such that, under the forward kinematics, it follows the desired end-effector trajectory $g_e^*(t) = g_e(\alpha^*(t))$.

Answer 1: a common approach is to "spline" together the inverse kinematics solution for a set of waypoints / knotpoints to get a candidate joint trajectory.

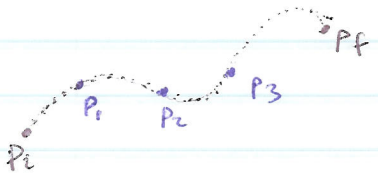
[1] solve inverse kinematics problem for each $q^*(t_k)$ to get $\alpha^*(t_k)$ for $t_0, t_1, \ldots, t_n$ where n = total # knot points.

[2] interpolate/concatenate smooth trajectory $\alpha^*(t)$ through the $\alpha^*(t_k)$

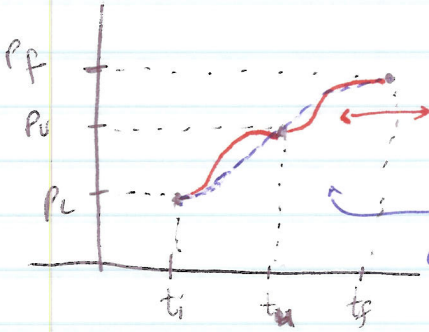
- need to be aware of inverse kinematics solutions, since they may not be unique; could lead to joint configuration flip-flopping.

Let's assume that part [1] is done. Next step is part [2].

Scalar case



* do not want to ~~connect~~ find cubic splines between each pair of adjacent waypoints.



using 2 cubic splines, I stop at (t_u, p_u).

ideally want to avoid slowing down.

2 cubic polynomials as per solution from last time connected knotpoints w/ zero initial and final velocities. Not good.

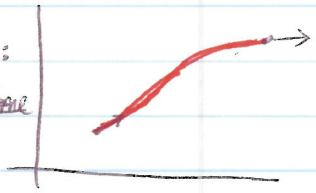
→ we want non-trivial velocities at the way-points.

the constraints generalize to:

$$p(0) = p_i \qquad p(t_f) = p_f$$

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

before:



⇒ Leads to the general problem from ~~last time~~

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

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

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

$$\vec{a} = P(t_f)\,\vec{p}_0 \qquad \text{where } P(t_f) = A^{-1}(t_f)$$

Explicitly this is:

$$a_0 = p_i \qquad a_2 = \frac{3}{t_f^2}(p_f - p_i) - \frac{2}{t_f}\dot{p}_i - \frac{1}{t_f}\dot{p}_f$$

$$a_1 = \dot{p}_i \qquad a_3 = \frac{2}{t_f^3}(p_i - p_f) + \frac{1}{t_f^2}(\dot{p}_i + \dot{p}_f)$$

- Velocities obtained from specification of desired trajectory. (Scalar / easy case)

for manipulators, things are different.

[0] start w/ $g^*(t)$

[1] for waypoint times $t_0, \ldots, t_n$ find $\alpha^*(t_k)$ so that $g^*(t_k) = g_e(\alpha^*(t))$. (using inverse kinematics)

- I also need $\dot{\alpha}^*(t_k)$. I find $\mathcal{V}_e^*(t_k) = (g_e^*(t_k))^{-1} \dot{g}_e^*(t_k)$ then use pseudo inverse of manipulator Jacobian:

$$\dot{\alpha}^*(t_k) = \left(J^b(\alpha^*(t_k))\right)^{\#} \mathcal{V}_e(t_k)$$

ECE 4560

Multiple Waypoints w/ Desired Velocities

- fairly simple to setup and solve
- possible discontinuities in accelerations ⟵ allow joint controller to smooth this out.

Suppose we are given:

$p_0, p_1, \ldots, p_n, p_{n+1}$
$\dot{p}_0, \dot{p}_1, \ldots, \dot{p}_n, \dot{p}_{n+1}$ } initial, intermediate, & final points/velocities.

plus intermediate times $0, t_1, \ldots, t_n, t_f$

$T_0, T_1, \ldots, T_{n-1}, T_n$

where $\begin{cases} T_k = t_{k+1} - t_k & k = 1, \ldots (n-1) \\ T_n = t_f - t_n \end{cases}$

Take velocity matching equations & apply to each segment:

$$
\begin{aligned}
p^0(0) &= a_0^0 &&= p_0 \\
p^0(T_0) &= a_0^0 + a_1^0 T_0 + a_2^0 (T_0)^2 + a_3^0 (T_0)^3 &&= p_1 \\
\dot{p}^0(0) &= a_1^0 &&= 0 \ (= \dot{p}_0) \\
\dot{p}^0(T_0) &= a_1^0 + 2a_2^0 T_0 + 3a_3^0 (T_0)^2 &&= \dot{p}_1 \\
&\vdots && \vdots \\
p^k(0) &= a_0^k &&= p_k \\
p^k(T_k) &= a_0^k + a_1^k T_k + a_2^k (T_k)^2 + a_3^k (T_k)^3 &&= p_{k+1} \\
\dot{p}^k(0) &= a_1^k &&= \dot{p}_k \\
\dot{p}^k(T_k) &= a_1^k + 2a_2^k T_k + 3a_3^k (T_k)^2 &&= \dot{p}_{k+1} \\
&\vdots && \vdots \\
p^n(0) &= a_0^n &&= p_n \\
p^n(T_n) &= a_0^n + a_1^n T_n + a_2^n (T_n)^2 + a_3^n (T_n)^3 &&= p_{n+1} \\
\dot{p}^n(0) &= a_1^n &&= \dot{p}_n \\
\dot{p}^n(T_n) &= a_1^n + 2a_2^n T_n + 3a_3^n (T_n)^2 &&= 0 \ (= \dot{p}_{n+1})
\end{aligned}
$$

This solves for the $(n+1)$ polynomials, each of which is of the form

$$p^k(\tau) = a_0^k + a_1^k \tau + a_2^k \tau^2 + a_3^k \tau^3 \quad \text{for } \tau \in [0, T_k]$$

The complete solution is:

$$p(t) = p^k(t - t_k) \quad \text{where } k = \frac{t}{\lfloor T \rfloor} = \text{floor}(t/T)$$

↑ where one evaluates p in time t determines (n-1) which polynomial to use

This is for $T = T_1 = T_2 = \ldots = T_k = \ldots = T_n$

if any T_k are different, then need to find k such that $t \in [t_k, t_{k+1}]$

* the system of equations solving for the polynomial coefficients is decoupled. Each polynomial can be solved individually. Solution for this "individual" case has been covered.
* basically just need to run a properly coded for loop to get the solution to each polynomial segment.

MATLAB: polyval

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

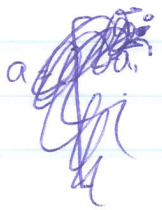
↑ these are coefficients

Matlab prefers $\vec{a} = [a_3 \quad a_2 \quad a_1 \quad a_0]$

$p(t) = \text{polyval}(a, t)$

↑ for one polynomial.

If master solution to multiple waypoints is setup right, you get:



→

$$a = \begin{bmatrix} \vec{a}^0 \\ \vec{a}^1 \\ \vec{a}^2 \\ \vdots \\ \vec{a}^n \end{bmatrix}$$

and suppose $T = T_k$ for $k = 0, \ldots, n$ → be careful about $t = t_f$ since $k = \text{floor}(t/T) = n+1$

```
k = floor(t/T);
p(t) = polyval(a(k,:), t-kt);
```

* if you want to be smart lazy you can generate the solution as a spline structure and use ppval instead.

* connecting in this manner matches velocities, but not accelerations.
  - if accelerations almost match at waypoints, this is not a problem.
  - if accelerations far apart, this may lead to jerky motions.

Let's connect two splines w/ a continuous acceleration
setup: achieve positions p_i, p_v, p_f at times 0, t_v, t_f.

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

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

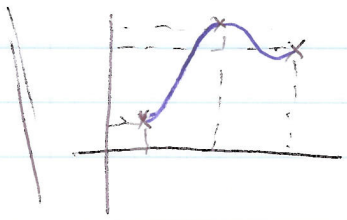
$q(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3 \quad t \in [0, t_f - t_v]$

constraints are: position: $p(0) = p_i$

$p(t_v) = p_v$

$q(0) = p_v$

$q(t_f - t_v) = p_f$



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

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

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

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

ECE 4560

(spline w/ continuous acceleration continued)

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

$$
\begin{aligned}
p(0) &= a_0 && = p_i \\
p(t_1) &= a_0 + a_1 t_1 + a_2 t_1^2 + a_3 t_1^3 && = p_v \\
q(0) &= b_0 && = p_v \\
q(t_2) &= b_0 + b_1 t_2 + b_2 t_2^2 + b_3 t_2^3 && = p_f \\
\dot{p}(0) &= a_1 && = 0 \\
\dot{q}(t_2) &= b_1 + 2b_2 t_2 + 3b_3 t_2^2 && = 0 \\
\dot{p}(t_1) - \dot{q}(0) &= a_1 + 2a_2 t_1 + 3a_3 t_1^2 - b_1 && = 0 \\
\ddot{p}(t_1) - \ddot{q}(0) &= 2a_2 + 6a_3 t_1 - 2b_2 && = 0
\end{aligned}
$$

$$
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & t_1 & t_1^2 & t_1^3 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & t_2 & t_2^2 & t_2^3 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 2t_2 & 3t_2^2 \\
0 & 1 & 2t_1 & 3t_1^2 & 0 & -1 & 0 & 0 \\
0 & 0 & 2 & 6t_1 & 0 & 0 & -2 & 0
\end{bmatrix}
\begin{Bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ b_0 \\ b_1 \\ b_2 \\ b_3 \end{Bmatrix}
=
\begin{Bmatrix} p_i \\ p_v \\ p_v \\ p_f \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}
$$

$A(t_1, t_2)$ $\qquad \vec{a} = \vec{P}_0$

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

looking at $\vec{p}_0$, last four rows are zeros & 2nd/3rd rows are equal, so we can ignore last four columns of $P(t_1, t_2)$ and "join" 2nd/3rd columns to get :

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

$$\vec{a} = P_{simp}(t_1, t_2) \begin{Bmatrix} P_i \\ P_v \\ P_f \end{Bmatrix}$$

where,

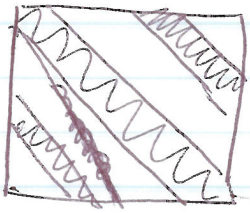
$$P_{simp}(t_1,t_2) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ -\frac{3(2t_1+t_2)}{2t_1^2(t_1+t_2)} & \frac{3(t_1+t_2)}{2t_1^2t_2} & -\frac{3}{2t_2(t_1+t_2)} \\ \frac{4t_1+t_2}{2t_1^3(t_1+t_2)} & -\frac{3t_1+t_2}{2t_1^3t_2} & \frac{3}{2t_1^2t_2+2t_1t_2^2} \\ 0 & 1 & 0 \\ -\frac{3t_2}{2t_1(t_1+t_2)} & \frac{3}{2}\left(\frac{1}{t_1}-\frac{1}{t_2}\right) & \frac{3t_1}{2t_2(t_1+t_2)} \\ \frac{3}{t_1^2+t_1t_2} & -\frac{3}{t_1t_2} & \frac{3}{t_1t_2+t_2^2} \\ -\frac{3}{2t_1^2t_2+2t_1t_2^2} & \frac{t_1+3t_2}{2t_1t_2^3} & -\frac{t_1+4t_2}{2t_2^3(t_1+t_2)} \end{bmatrix}$$

now, if $t = t_1 = t_2 \Rightarrow \underset{0,\ t,\ 2t}{t_i, t_v, t_f}$, then we get

$$P_{simp}(t_1,t_2) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ -9/4t^2 & 3/t^2 & -3/4t^2 \\ 5/4t^3 & -2/t^3 & 3/4t^3 \\ 0 & 1 & 0 \\ -3/4t & 0 & 3/4t \\ 3/2t^2 & -3/t^2 & 3/2t^2 \\ -3/4t^3 & 2/t^3 & -5/4t^3 \end{bmatrix}$$

* 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-diagnol form ⇒ 1) sparse matrix
2) these are fast solvers.



Multiple Waypoints with Matching Velocity/Acceleration & Equidistant in Time

GOAL: find $p^i(t) = a_0^i + a_1^i \tau + a_2^i \tau^2 + a_3^i \tau^3$

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

Solution involves; $2(n+1)$ position constraints
$n+2$ velocity "
n acceleration "
} ⇒ $4(n+1)$ equations

Procedure for organizing matrix into tri-diagnol form:

- arrange rows so that similar/related coefficients are near each other.
- initial & final polynomials have different constraints vs. intermediate polynomials, so their setup differs.

$$p^{\circ}(0) = a_0^{\circ} = P_0$$

$$p^{\circ}(T) = a_0^{\circ} + a_1^{\circ} T + a_2^{\circ} T^2 + a_3^{\circ} T^3 = P_1$$

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

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

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