Topics Covered:

- Note on Adjoint Operation for Twists
- Group vs Joint Space Planning
- Straight-Line Paths

Additional Reading:

- LP Chapter 9

Timeline for the rest of the semester:

1. Today: Last Lecture on Trajectory Design
2. Thursday, November 13th: Wrenches and Forces
3. Tuesday, November 18th: Introduction to Control
4. Thursday, November 20th: Group Presentations
5. Tuesday, November 25th: Dynamics + Control for Manipulators (Not on final exam)
6. Thursday, November 27th: Thanksgiving
7. Tuesday, December 2rd: Final Exam Review

Review of Point-to-Point Trajectory Design

Assume that we only want to move from joint configuration $\vec{\theta}_i$ to $\vec{\theta}_f$ over time $t \in [0, T]$. Here we will use cubic polynomials to create smooth position and velocity profiles:

1. Define starting and ending configuration $\vec{\theta}_i = \vec{\theta}(0)$ and $\vec{\theta}_f = \vec{\theta}(T)$ with $\vec{\theta} = (\theta^1, \theta^2, \ldots, \theta^m)^T$.
2. Define starting and ending joint velocity (only used if we are stitching together multiple configurations: $\dot{\vec{\theta}}_i$ and $\dot{\vec{\theta}}_f$.
3. Solve for polynomial coefficients for each joint j:

$$
\begin{aligned}
a_0^j &= \theta_i^j \\
a_1^j &= 0 \\
a_2^j &= \frac{3}{T^2}(\theta_f^j - \theta_i^j) - \frac{1}{T}(2\dot{\theta}_i^j + \dot{\theta}_f^j) \\
a_3^j &= \frac{-2}{T^3}(\theta_f^j - \theta_i^j) + \frac{1}{T^2}(\dot{\theta}_i^j + \dot{\theta}_f^j)
\end{aligned}
$$

4. Use these coefficients to define $\theta^*(t)$ and $\dot{\theta}^*(t)$ using a cubic polynomial:

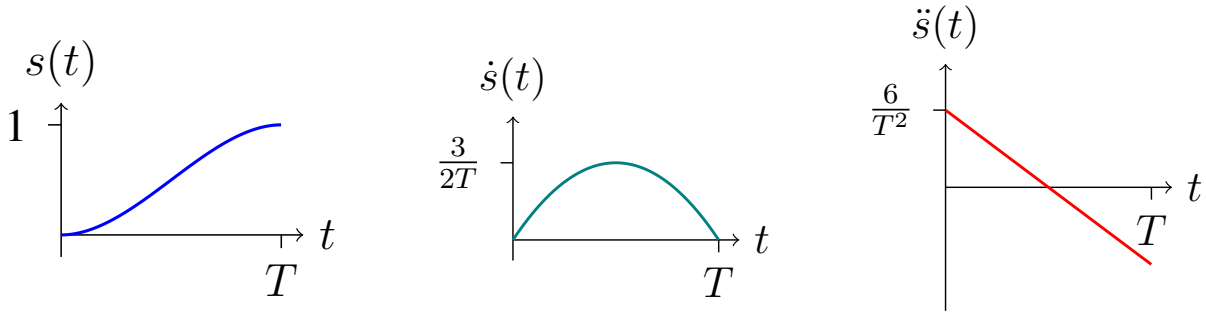
$$\theta^{*j}(t) = a_0^j + a_1^j t + a_2^j t^2 + a_3^j t^3$$
$$\dot{\theta}^{*j}(t) = a_1^j + 2a_2^j t + 3a_3^j t^2$$

Review of Straight-Line Path Trajectory Design with Inverse Kinematics

To achieve straight-line paths in either joint space or task space, we will need to use a time-scaling function to parameterize the straight-line path. The procedure is then as follows:

1. Define starting and ending pose $g_i = g_e^*(0)$ and $g_f = g_e^*(T)$
2. Define the time-scaling function $s(t)$ as a cubic polynomial with boundary conditions:

$$s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$
$$\dot{s}(t) = a_1 + 2a_2 t + 3a_3 t^2$$
$$a_0 = 0, \quad a_1 = 0, \quad a_2 = \frac{3}{T^2}, \quad a_3 = \frac{-2}{T^3}$$

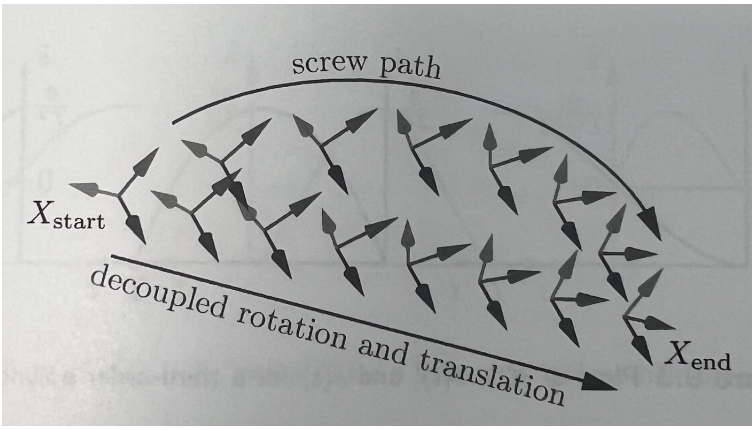


3. Then, we can obtain the desired end-effector path using either a "straight-line" with a constant screw motion (fixed screw axis)

$$g(s(t)) = g_i \exp(\ln(g_i^{-1} g_f) s)$$

This provides a "straight-line" in thes ense that the screw axis is constant. If we want a true straight-line in position space, we must decouple rotation and translation as:

$$p(s(t)) = (1 - s)p_i + s p_f$$
$$R(s(t)) = R_i \exp(\ln(R_i^T R_f) s)$$



Note that in both cases we CANNOT simply do:

$$g(s(t)) = (1-s)g_i + sg_f$$

since this may not produce a valid $SE(3)$ transformation (most likely, $R(s)$ will lose its orthonormality)

4. Perform inverse kinematics at each time step to solve for $\theta^*(t)$
5. Use $\xi_s = \dot{g}(t)g(t)^{-1}$ to compute $\dot{\theta}^*(t)$ using the inverse/pseudo-inverse of the spatial manipulator Jacobian:
$$\dot{\theta}^*(t) = (J_s(\theta(t)))^\dagger \xi_s(t)$$

Review of Resolved Rate Trajectory Design (No Inverse Kinematics)

In general, we have so far introduced the following methodology:

1. Start with $g_e^*(t)$ which either we can vectorize, or assume we are given a collection of waypoints:
$$g_e^*(t_k) \quad \text{for} \quad k = 0, 1, \ldots, n+1$$
2. Use the logarithm to solve for $\xi_e^*(t_k)^*$:

$$\begin{aligned}
(\xi_e^*(t_k))^b &= \ln_{\Delta t}(g_{\text{rel}}) \\
&= \ln_{\Delta t}(g^{-1}(t_k)g(t_{k+1})) && (g_{ab} = g_{sa}^{-1}g_{sb}) \\
&= \text{logm}(g^{-1}(t_k)g(t_{k+1}))/\Delta t \\
(\xi_e^*(t_k))^s &= \text{Ad}_{g(t_k)}(\xi_e^*(t_k))^b \\
&= \text{Ad}_{g(t_k)} \ln_{\Delta t}(g^{-1}(t_k)g(t_{k+1})) \\
&= g(t_k)\text{logm}\left(g^{-1}(t_k)g(t_{k+1})\right)\left(g(t_k)\right)^{-1}/\Delta t \\
&= \text{logm}\left(g(t_{k+1})g(t_k)^{-1}\right)/\Delta t && \text{(Is this true?)}
\end{aligned}$$

3. Option 1: Use inverse kinematics (can be closed-form algorithm, iterative such as Newton-Raphson, or optimization-based) to solve for $\theta^*(t_k)$.
4. Option 2: Use resolved-rate to solve for $\theta^*(t_k)$:

$$\theta(t_{k+1}) = \theta(t_k) + \Delta t (J^b(\theta(t_k)))^\dagger (\xi_e^*(t_k))^b$$

5. Use the same Pseudo-Inverse to solve for Joint-Velocity:

$$\dot{\theta}(t_k) = (J^b(\theta(t_k)))^\dagger (\xi_e^*(t_k))^b$$

6. Use these waypoints to construct cubic polynomials for $\theta^*(t)$ and $\dot{\theta}^*(t)$ to get a smooth trajectory.
7. Track $\theta(t)$ and $\dot{\theta}(t)$ with a controller. For example, our MuJoCo simulations use a PD controller:

$$\vec{u}(t) = K_p(\theta^*(t) - \theta(t)) + K_d(\dot{\theta}^*(t) - \dot{\theta}(t))$$

Fifth-Order (Quintic) Splines

If we want to ensure smooth acceleration profiles as well, we can use fifth-order (quintic) splines. This requires us to introduce two additional boundary conditions on acceleration:

$$\ddot{\theta}(0) = \ddot{\theta}_i$$
$$\ddot{\theta}(T) = \ddot{\theta}_f$$

This results in a fifth-order polynomial of the form:

$$\theta^*(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

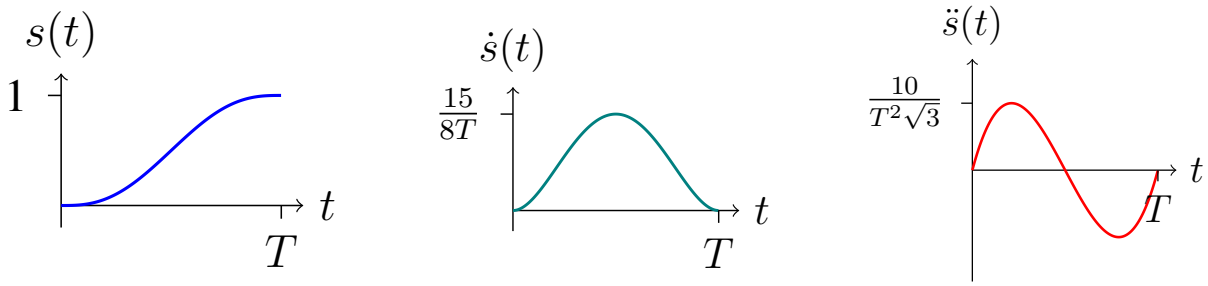
with the polynomial coefficients given by:

$$a_0^j = \theta_i^j$$
$$a_1^j = 0$$
$$a_2^j = 0$$
$$a_3^j = \frac{10}{T^3}(\theta_f^j - \theta_i^j)$$
$$a_4^j = -\frac{15}{T^4}(\theta_f^j - \theta_i^j)$$
$$a_5^j = \frac{6}{T^5}(\theta_f^j - \theta_i^j)$$

This also applies to straight-line paths in which we can define a quintic time-scaling function $s(t)$ with boundary conditions on position, velocity, and acceleration at $t = 0$ and $t = T$. This gives us the coefficients:

$$
\begin{aligned}
a_0^j &= 0 \\
a_1^j &= 0 \\
a_2^j &= 0 \\
a_3^j &= \frac{10}{T^3} \\
a_4^j &= -\frac{15}{T^4} \\
a_5^j &= \frac{6}{T^5}
\end{aligned}
$$

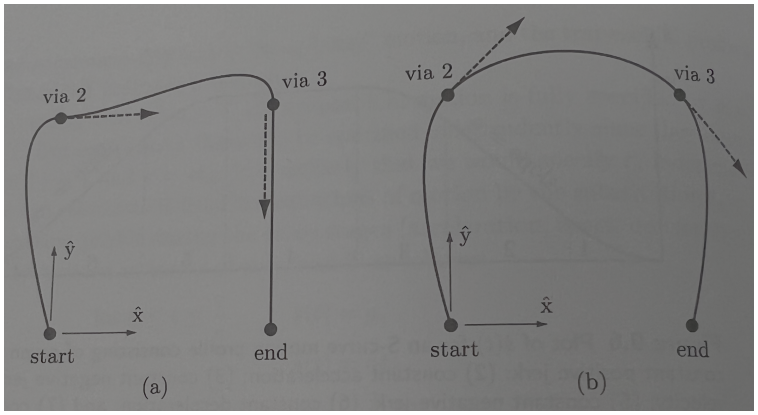
which produces the curves: Of course, this too has it's drawbacks. Namely, there will be infinite



"snap" (derivative of jerk) due to a discontinuity at the endpoints of the jerk profile.

Waypoint Planning

When planning trajectories through multiple waypoints, we can use either cubic or quintic splines to connect each segment. For cubic splines, we can set the velocity at each intermediate waypoint to zero, or we can set it to the average velocity between the two segments. Consider the following example:



Here, example a uses the waypoints at $(0, 0)$, $(0, 1)$ $(1, 1)$, and $(1, 0)$ with velocities $(0, 0)$, $(1, 0)$, $(0, 1)$, and $(0, 0)$. Example b uses the same waypoints but the velocities $(0, 0)$, $(1, 1)$, $(1, -1)$, and $(0, 0)$. Here, our cubic splines between waypoints will be represented as:

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

with t defined over each segment $[t_k, t_{k+1}]$ and recentered to lie on the interval $[0, T_k]$ where $T_k = t_{k+1} - t_k$.