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 14th: Wrenches and Forces
3. Tuesday, November 19th: Introduction to Control
4. Thursday, November 21st: Dynamics + Control for Manipulators (GT football game)
5. Tuesday, November 26th: Group Presentations
6. Thursday, November 28th: Thanksgiving
7. Tuesday, December 3rd: Final Exam Review

Note on Adjoint Operation for Twists

I noticed in Assignment 8 that several students have been using the matrix Adjoint that we introduced in Lecture 16:

$$\mathrm{Ad}_g = \begin{bmatrix} R & [d]_\times R \\ 0 & R \end{bmatrix} \quad \text{assuming } \xi := \begin{bmatrix} v \\ \omega \end{bmatrix} \qquad \text{(Convention in MLS)}$$

or

$$= \begin{bmatrix} R & 0 \\ [d]_\times R & R \end{bmatrix} \quad \text{assuming } \xi := \begin{bmatrix} \omega \\ v \end{bmatrix} \qquad \text{(Convention in LP)}$$

This adjoint was introduced to map body twists to spatial twists:

$$\xi^s = \mathrm{Ad}_g \xi^b$$

This matrix can also be used then for any situation when we want to map a body twist to a spatial twist, so we can use it to compute:

$$\xi_2' = \mathrm{Ad}_{\exp(\hat{\xi}_1 \theta_1)} \xi_2,$$

Thus, if we denote our matrix $e1 = \exp(\hat{\xi}_1\theta_1) = \begin{bmatrix} R_1 & d_1 \\ 0 & 1 \end{bmatrix}$, then we can calculate our previous Adjoint operation as:

$$\begin{aligned} \xi_2' &= \mathrm{Ad}_{e1}\xi_2 \\ &= \begin{bmatrix} R_1 & [d_1]_\times R_1 \\ 0 & R_1 \end{bmatrix} \xi_2 \end{aligned}$$

However, since this matrix does not apply to all situations where we are computing an adjoint, I recommend using the official definition of an adjoint:

$$\begin{aligned} \xi_2' &= \mathrm{Ad}_{\exp(\hat{\xi}_1\theta_1)}\xi_2 \\ &= \left(\exp(\hat{\xi}_1\theta_1)\hat{\xi}_2 \left(\exp(\hat{\xi}_1\theta_1)\right)^{-1}\right)^\vee \end{aligned}$$

For the third twist, this would be:

$$\begin{aligned} \xi_3' &= \mathrm{Ad}_{\exp(\hat{\xi}_1\theta_1)\exp(\hat{\xi}_2\theta_2)}\xi_3 \\ &= \left(\left(\exp(\hat{\xi}_1\theta_1)\exp(\hat{\xi}_2\theta_2)\right)\hat{\xi}_3 \left(\exp(\hat{\xi}_1\theta_1)\exp(\hat{\xi}_2\theta_2)\right)^{-1}\right)^\vee \end{aligned}$$

Otherwise, we would need to keep in mind that when computing the body Jacobian, our procedure for the matrix adjoint is different. Specifically, for the body Jacobian we would do the following:

$$J^b = \begin{bmatrix} \xi_1^\dagger & \xi_2^\dagger & \xi_3^\dagger \end{bmatrix}$$

with each body-twist component computed as:

$$\begin{aligned} \xi_i^\dagger &= \mathrm{Ad}^{-1}_{\underbrace{\exp(\hat{\xi}_i\theta_i)\cdots\exp(\hat{\xi}_n\theta_n)g_0}_{g}}\xi_i \\ &= \mathrm{Ad}_g^{-1}\xi_i \\ &= \mathrm{Ad}_{g^{-1}}\xi_i \\ &= \begin{bmatrix} R^T & -R^T[d]_\times \\ 0 & R^T \end{bmatrix}\xi_i \end{aligned}$$

However, without the matrix we could also compute using our definition of the adjoint operation as:

$$\xi_i^\dagger = \left(\left(\exp(\hat{\xi}_i\theta_i)\cdots\exp(\hat{\xi}_n\theta_n)g_0\right)^{-1}\hat{\xi}_i\left(\exp(\hat{\xi}_i\theta_i)\cdots\exp(\hat{\xi}_n\theta_n)g_0\right)\right)^\vee$$

Back to Trajectory Design

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

Discussion of Joint Space vs. Group (Task) Space

Below is an example from Lynch/Park Section 9.2 that illustrates the difference between planning a straight-line path in joint-space versus in the group (or they call it task) space:

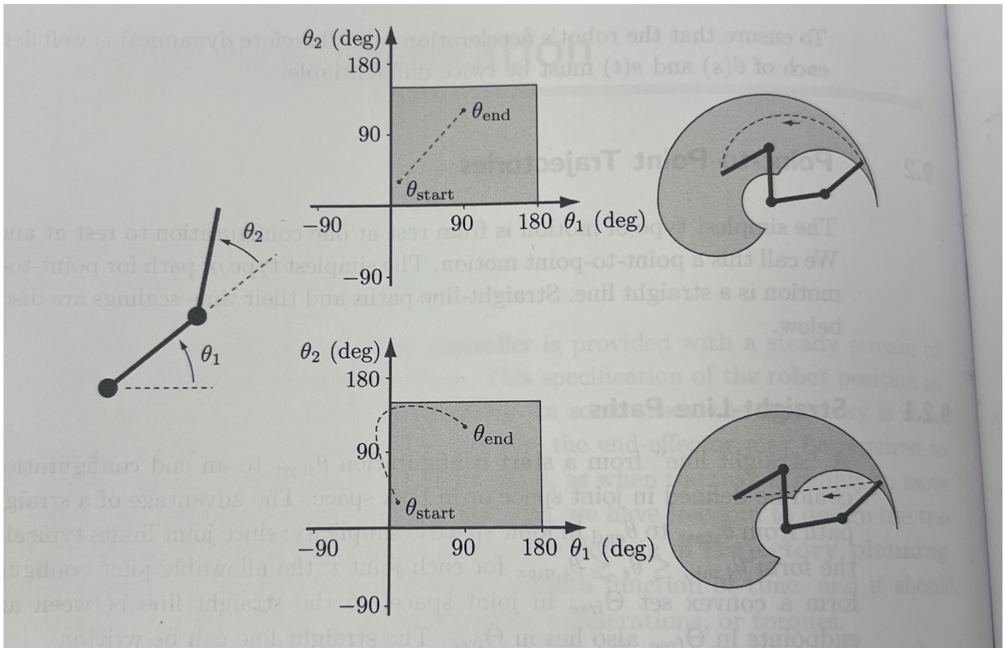

Figure 9.1 (Left) A 2R robot with joint limits $0° \leq \theta_1 \leq 180°$, $0° \leq \theta_2 \leq 150°$. (Top center) A straight-line path in joint space and (top right) the corresponding motion of the end-effector in task space (dashed line). The reachable endpoint configurations, subject to joint limits, are indicated in gray. (Bottom center) This curved line in joint space and (bottom right) the corresponding straight-line path in task space (dashed line) would violate the joint limits.

	Pros	**Cons**
Joint Space Planning	Simpler to compute Joint limits are easier to obey	May result in complex end-effector paths that are hard to predict
Group Space Planning	Ensures end-effector follows desired path in Cartesian space	More computationally intensive Requires solving inverse kinematics

Table 1: Comparison of Joint Space vs. Group Space Planning

Straight-Line Paths

If we want a straight-line path, we will run into the previously mentioned issues with vibrations caused from instantaneous changes in velocity. So, to avoid these vibration issues, we can instead define our polynomial over our time-scaling function $s(t)$ that maps your time interval $[t_i, t_f]$ to $[0, 1]$, i.e., $s : [t_i, t_f] \to [0, 1]$.

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

with the constraints:

$$s(0) = 0, \quad \dot{s} = 0, \quad s(T) = 1, \quad \dot{s}(T) = 0. \qquad (T = t_f - t_i)$$

Note: time-scaling can also just be used with polynomials over $p(t)$, but here the choice of $s(t)$ could be linear:

$$s(t) = \frac{t - t_i}{t_f - t_i}$$

But, using our polynomial function $s(t)$, a twice-differentiable "straight-line" path can be created as the convex combination between two points p_i and p_f:

$$\begin{aligned} p(s) &= (1 - s)p_i + sp_f \\ &= p_i + s(p_f - p_i) \end{aligned}$$

This procedure still gives us a twice-differentiable path with the coefficients:

$$a_0 = 0, \quad a_1 = 0, \quad a_2 = 3/T^2, \quad a_3 = -2/T^3$$

Using this time-scaling, we can now define our straight-line path as the convex combination of two points p_i and p_f. For straight-line paths in the joint space, this is:

$$\theta(s) = \theta_i + s(\theta_f - \theta_i), \quad s \in [0, 1]$$

with the velocity:

$$\begin{aligned} \dot{\theta}(s) &= \dot{s}(\theta_f - \theta_i) \\ &= (a_1 + 2a_2 t + 3a_3 t^2)(\theta_f - \theta_i) \\ &= \left(\frac{6t}{T^2} - \frac{6t^2}{T^3}\right)(\theta_f - \theta_i) \end{aligned}$$

A straight-line path in group space $g = (R, p) \in (SE(3))$ is a bit more complicated since:

$$g(s) = g_i + s(g_f - g_i)$$

does not generally lie in $SE(3)$. In other words, this convex combination does not necessarily follow the physical laws of motion. Instead, we must use our previous knowledge about tranformations to derive our path:

$$g_{if} = g_{wi}^{-1} g_{wf}$$

with i being the initial frame, f being the final frame, and w being the world (or spatial) frame. We can then use our exponential map to obtain specific configurations along this path. This is done by first noting that the twist associated with this transformation is:

$$\xi = \ln(g_{wi}^{-1} g_{wf}),$$

so taking the exponential map gives us:

$$g(s) = \underbrace{g_{wi}}_{g_i^*} \exp(\ln(g_{wi}^{-1} \underbrace{g_{wf}}_{g_f^*})s).$$

Finally, we can construct $g(s)$ by individually constructing paths for the Cartesian position and the rotation:

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

Using this method, the full procedure for generating a straight-line path in group space is:

1. Start with g_i and g_f.
2. Obtain your cartesian path and rotation path $p(s)$ and $R(s)$ from g_i and g_f.
3. Obtain your scaling function $s(t)$ as a cubic polynomial.
4. Conduct inverse-kinematics at either waypoints or continuously along $g(s)$.
5. Interpolate/move along $\theta^*(s)$ and $\dot{\theta}^*(s)$.