Topics Covered:

- Resolved Rate Trajectory Generation Derivation
- Damped Pseudo-Inverse
- Weighted Pseudo-Inverse

Additional Reading:

- LP Chapter 9
- Craig Chapter 7

Review: Trajectory Generation

We've discussed three methods of trajectory generation so far:

1. cubic spline interpolation
2. straight-line paths in joint space
3. straight-line paths in task space

Using the Jacobian Pseudo-Inverse for Manipulators

To find the desired joint-level velocity for straight-line paths in task space, we can utilize the manipulator Jacobian to solve for $\dot{\theta}$. The process is as follows:

1. As before, we start with $g_e^*(t)$ and $\dot{g}_e^*(t)$
2. Then, for waypoint times $t_0, t_1, \ldots, t_n$, we solve the inverse kinematics problem to get $\theta^*(t_k)$.
3. To obtain $\dot{\theta}^*(t_k)$, we can use the psuedo-inverse of the body manipulator Jacobian:

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

with the twist calculated using:

$$\xi_e^b(t_k) = (g_e^*(t_k))^{-1} \dot{g}_e^*(t_k)$$

The problem with the last method is that it requires solving inverse kinematics at various waypoints along the trajectory. This can lead to problems if there's multiple IK solutions (which is especially problematic for kinematically redundant arms).

To avoid these problems, we can leverage an alternative technique called **resolved rate trajectory generation**. This method is based on the idea that we can directly control the end-effector velocity and acceleration.

The general idea is to:

- Specify a desired end-effector trajectory $g_e^*(t)$ and its associated velocity $\dot{g}_e^*(t)$
- Integrate $\dot{\theta}(t) = J^\dagger \xi_e^b(t)$ to get:

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

Notes on Inverting the Manipulator Jacobian

As we saw, the general form thats used with the inverse manipulator Jacobian is:

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

When J^b is not square, we use the **Moore-Penrose Pseudo-Inverse** to obtain the pseudo-inverse. This inverse is defined as:

1. **Kin. Redundant:** If $J^b \in \mathbb{R}^{n\times m}$ is "fat" ($n < m$) and is full (row) rank, then:

$$J^\dagger = J^T(JJ^T)^{-1} \qquad \text{(Right Inverse)}$$

2. **Kin. Insufficient:** If $J^b \in \mathbb{R}^{n\times m}$ is "tall" ($n > m$) and has full (column) rank, then:

$$J^\dagger = (J^T J)^{-1}J^T \qquad \text{(Left Inverse)}$$

Resolved Rate Trajectory Generation

Given a desired end-effector trajectory $g_e^*(t)$ and it's associated velocity $\xi_e^*(t)$, we consider:

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

in order to solve for $\theta(t)$.

This leads to the differential equation:

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

with initial condition

$$\theta(0) = \text{ inv. kin.}(g_e^*(0)) = \theta_0 \qquad (\theta_0 \text{ is typically known})$$

At this point, we just need to integrate to get the solution.

Ideally, let a computational program deal with this (i.e., ode45 in MATLAB), but if the option is not available, here is one option ...

$$\begin{aligned} \dot{\theta}(t) &= (J^b(\theta(t)))^\dagger (\xi_e^*)^b(t) \\ \frac{\theta(t_{k+1}) - \theta(t_k)}{\Delta t} &= (J^b(\theta(t)))^\dagger (\xi_e^*)^b(t) \\ \theta(t_{k+1}) &= \theta(t_k) + \Delta t (J^b(\theta(t)))^\dagger (\xi_e^*(t))^b \end{aligned}$$

Here, we would repeat for all t_k, $k = 0, \ldots, n+1$.

Note:

- This is a first-order accurate integration technique. There are other more accurate methods.
- Assumes we can get $(\xi_e^*)^b(t)$ from a continuous trajectory $g_e^*(t)$.

So, suppose that $g_e^*(t)$ is really just a collection of closely spaced waypoints for which $(\xi_e^*)^b(t)$ can't be found (because we don't have $\dot{g}^*(t)$, we just have $g_e^*(t_k)$).

Naive approach:

If $g_e^*(t)$ is vectorizable, then we can use the standard Jacobian and first-order approximation to end-effector derivative:

$$\begin{aligned} \theta(t_{k+1}) &= \theta(t_k) + \Delta t (J^b(\theta(t_k)))^\dagger \left(\frac{g_e^*(t_{k+1}) - g_e^*(t_k)}{\Delta t} \right) \\ \theta(t_{k+1}) &= \theta(t_k) + (J^b(\theta(t_k)))^\dagger (g_e^*(t_{k+1}) - g_e^*(t_k)) \end{aligned}$$

Geometric Approach:

Use logarithm:

$$(\xi_e^*(t_k))^b = \ln_{\Delta t}(g^{-1}(t_k) g(t_{k+1})) \qquad (\text{need time of ln to be } \Delta t)$$

This logarithm can then be used with:

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

Ultimately, the idea is to use the Jacobian to relate velocities, then integrate up to final configuration of the trajectory (MATLAB and others can do the integration)

Problem 1: If the true trajectory starts to deviate from the desired, then the end-effector may diverge from the desired trajectory (open-loop problem).

Problem 2: What if the trajectory crosses, or passes nearby a singularity? At singularity, the Jacobian loses rank. So what happens to $J^\dagger$?

$$\text{main part of } J^\dagger = (JJ^T)^{-1}$$

$$\det(A) = \prod_1^n \lambda_i$$

$$\text{losing rank} \rightarrow \lambda_i \rightarrow 0$$

so for some i, the inverse matrix blows up! This would result in infinite joint-level velocities.

Damped Pseudo-Inverse

Goal: avoid blow-up near singularities.

The pseudo-inverse solved for $\dot{\theta}$ that minimized:

$$\mathcal{L}(\dot{\theta}) = \|\dot{\theta}\|^2 \quad \text{subject to } \xi = J(\theta)\dot{\theta}$$

Instead, we will propose to minimize:

$$\mathcal{L}(\dot{\theta}) = \frac{1}{2}\|\xi^b - J^b(\theta)\dot{\theta}\|^2 + \frac{1}{2}\rho^2\|\dot{\theta}\|^2$$

The solution is called the damped pseudo-inverse and is:

$$J^\dagger(\theta, \rho) = J^T(JJ^T + \rho^2 I)^{-1}$$

The eigenvalues of $(JJ^T)^{-1}$ change from $\frac{1}{\sigma_i}$ to $\frac{\sigma_i}{\sigma_i^2+\rho^2}$, with σ_i being the singular values of $(JJ^T)^{-1}$.

The issue will be that now the joint velocities are bounded. The exact bound is:

$$\frac{\|\dot{\theta}\|}{\|\xi\|} < \frac{1}{2\rho}$$

So the trajectory may deviate from the desired due to these bounds.

Weighted Pseudo-Inverse

This is yet another alternative formulation of the pseudo-inverse minimization problem. It is the minimization of:

$$\mathcal{L}(\dot{\theta}) = \frac{1}{2}\|\xi - J^b\dot{\theta}\|^2 + \frac{1}{2}\|\dot{\theta}\|_W^2$$

where $\|\cdot\|_W = (\cdot)^T W \cdot$ for some positive definite and symmetric matrix W. For example, $\|v\|_W = v^T W v$.

The solution to this problem (for the kinematically redundant/sufficient case) is:

$$J^\dagger = W^{-1}(J^b)^T(J^b W^{-1}(J^b)^T)^{-1}$$

This allows us to give different joint angle rates different priorities by changing our choice of W.

Side note: Damped-weighted Psuedo-Inverse is:

$$J^\dagger = W^{-1}(J^b)^T(J^b W^{-1}(J^b)^T + \rho^2 I)^{-1}$$

Recap/Summary

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 $\xi_e^*(t_k)^*$:

$$(\xi_e^*(t_k))^b = \ln_{\Delta t}(g^{-1}(t_k)g(t_{k+1}))$$
$$(\xi_e^*(t_k))^s = \ln_{\Delta t}(g(t_{k+1})g^{-1}(t_k))$$

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:

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