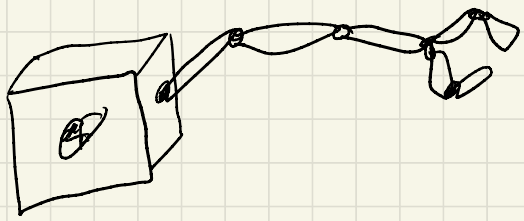
Last Time:

- Forward Kinematics
- Differentiating Quaternions

Today:

- Kinematics in 3D
- Least-Action for rigid bodies
- Floating-base robot dynamics

Floating-Base Kinematics in 3D:



- Not really different from fixed-base. Just account for base pose:

$$
\begin{aligned}
{}^{N}R_0^B &= H^T L(Q_0) R^T(Q_0) H \\
{}^{N}r_0 &= r_0 \\
R_n &= R_{n-1} S(\theta_n) \\
r_n &= r_{n-1} + R_{n-1} b_{n-1} + R_n q_n
\end{aligned}
$$

- Kinematics Function

$$\begin{bmatrix} r_0 \\ Q_0 \\ r_1 \\ Q_1 \\ \vdots \\ r_n \\ Q_n \end{bmatrix} = k(q), \quad q = \begin{bmatrix} r_0 \\ Q_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

"configuration"

- Kinematic Jacobian for Computing Velocities:

$$\begin{bmatrix} \dot{r}_0 \\ \omega_0 \\ \dot{r}_1 \\ \omega_1 \\ \vdots \\ \dot{r}_n \\ \omega_n \end{bmatrix} = \begin{bmatrix} I & & & & \\ & 2G^T(Q_0) & & & \\ & & I & & \\ & & & 2G^T(Q_1) & \\ & & & & \ddots \\ & & & & & 2G^T(Q_n) \end{bmatrix} \frac{\partial k}{\partial q} \begin{bmatrix} I & & & \\ & \frac{1}{2}G(Q_0) & & \\ & & I & \\ & & & \ddots \end{bmatrix} \begin{bmatrix} \dot{r}_0 \\ \omega_0 \\ \dot{\theta}_1 \\ \vdots \\ \dot{\theta}_n \end{bmatrix}$$

$K(q)$ $\qquad$ v

- Note $v \neq \dot{q}$

$$\dot{Q} = \frac{1}{2}G(Q)\omega \Rightarrow \omega = 2G^T(Q)\dot{Q}$$

- Mass Matrix / Kinetic Energy:

$$T = \sum_{n=1}^{n} \frac{1}{2} m_n \dot{r}_n^T \dot{r}_n + \frac{1}{2} \omega_n^T J_n \omega_n = \frac{1}{2} v^T K(q)^T \bar{M} K(q) v$$

$M(q)$

$$\bar{M} = \begin{bmatrix} m_0 I & & & \\ & J_0 & & 0 \\ & & \ddots & \\ 0 & & & m_n I \\ & & & & J_n \end{bmatrix}$$

Floating-Base Systems:

- Anything that isn't bolted to the ground. e.g. mobile (legged/aerial/underwater) space
- Essentially want to combine SE(3) stuff for base w/ Manipulator stuff for other links.
- To ease into this, we're going to start with a single rigid body.

Lagrangian for a "Free" Rigid Body

$$L = \frac{1}{2} m \dot{r}^T \dot{r} + \frac{1}{2} \omega^T J \omega$$

- If we naively apply EL eqn. to ω:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \omega}\right) - J\dot{\omega} = 0$$

- We know the right answer (Euler's Eqn.)
- Standard EL equation doesn't work $\omega = \dot{q}$
- We can still use least-action with a constraint:

$$\min_{\substack{Q(t) \\ \omega(t)}} \int_{t_0}^{t_f} \frac{1}{2} \omega^T J \omega \, dt$$

$$\text{s.t. } \dot{Q} = \frac{1}{2} Q * \hat{\omega} \Rightarrow \omega = 2H^T(Q^+ * \dot{Q})$$

- FON Conditions:

$$\frac{\partial}{\partial \begin{bmatrix} Q \\ \omega \end{bmatrix}} \left[\int_{t_0}^{t_f} \frac{1}{2} \omega^T J \omega + \ell^T \left(2H^T (Q^+ * \dot{Q}) - \omega \right) dt \right] = 0$$

ℓ: Lagrange Multiplier

- Derivative w.r.t. ω is easy:

$$\delta_\omega S = \int_{t_0}^{t_f} \omega^T J \delta\omega - \ell^T \delta\omega \, dt = 0 \quad \forall \, \delta\omega$$

$$\Rightarrow \ell = J\omega$$

ℓ: Angular momentum

- For the next part we need some quaternion tricks:

$$Q^+ * \dot{Q} = L^T(Q)\dot{Q}$$

$$= R(\dot{Q})Q^+ = R(\dot{Q})TQ, \quad T = \begin{bmatrix} 1 & 0 \\ 0 & -I \end{bmatrix}$$

- We also need to differentiate these:

$$\delta Q = \frac{1}{2} G(Q)\delta\phi = \frac{1}{2} L(Q) H \delta\phi$$

$$\Rightarrow \delta Q^+ = T\delta Q = \frac{1}{2} T L(Q) H \delta\phi$$

$$\Rightarrow \delta\dot{Q} = \frac{1}{2} G(Q)\delta\dot{\phi} + \frac{1}{2} G(\dot{Q})\delta\phi$$

$$= \frac{1}{2} L(Q) H \delta\dot{\phi} + \frac{1}{2} L(\dot{Q}) H \delta\phi$$

- Putting these together:

$$\delta_\alpha S = \int_{t_0}^{t_f} \ell^T \left[2H^T (\delta Q^+ \circ \dot{Q} + Q^+ \circ \delta\dot{Q}) \right] dt$$

$$= \int_{t_0}^{t_f} \ell^T \left[H^T R(Q)^T G(Q) \delta\phi + H^T L(Q) (L(Q) H^T \delta\phi \right.$$

(I)

$$\left. + L(Q) H \delta\phi \right] dt$$

$$= \int_{t_0}^{t_f} \underbrace{\ell^T \delta\dot{\phi}}_{-\dot{\ell}^T \delta\phi} + \ell^T \underbrace{\left(H^T L^T(Q) L(\dot{Q}) H + H^T R(\dot{Q})^T L(Q) H \right)}_{\hat{\omega}} \delta\phi \, dt$$

(integration by parts)

($\hat{\omega}$: skew-symmetric cross-product matrix)

$$= \int_{t_0}^{t_f} \delta\phi^T \left[\dot{\ell} + \omega \times \ell \right] dt = 0 \quad \forall \, \delta\phi$$

$$\Rightarrow \dot{\ell} + \omega \times \ell = 0$$

$$\Rightarrow \boxed{J\dot{\omega} + \omega \times J\omega = 0}$$

- This is a special case of the Euler-Poincare equation which generalizes the EL equation to Lie groups.

Floating-Base Dynamics:

- Add more joints + links to the Lagrangian:

$$L(q, v) = \frac{1}{2} v^T M(q) v - U(q)$$

- Introduce explicit quaternion constraint:

$$S = \int_{t_0}^{t_f} L(q, v) + l^T \left[2 H^T (Q_0^+ * \dot{Q}_0) - \omega_0 \right] dt$$

- We need to set derivatives w.r.t. $q(t), v(t), l(t) = 0$

- Derivatives that don't involve Q_0 look like standard EL eqn:

$$\delta r_0 = \int_{t_0}^{t_f} \frac{\partial L}{\partial r_0} \delta r_0 + \frac{\partial L}{\partial \dot{r}_0} \delta \dot{r}_0 \, dt = 0$$

$$= \int_{t_0}^{t_f} \frac{\partial L}{\partial r_0} \delta r_0 - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{r}_0}\right) \delta r_0 \, dt = 0 \quad \forall \delta r_0$$

$$\Rightarrow \boxed{\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{r}_0}\right) - \frac{\partial L}{\partial r_0} = 0} \quad (1)$$

- Same for joint angles:

$$\boxed{\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0} \quad (2)$$

$$\delta_{\omega_0} S = \int_{t_0}^{t_f} \left[\frac{\partial L}{\partial \omega_0} - l^T \right] \delta\omega_0 \, dt$$

$$\Rightarrow \boxed{l = \nabla_{\omega_0} L(q, v)} \quad (3)$$

- Derivative w.r.t. Q_0 is messy but same as before

$$\Rightarrow \dot{l} + \omega \times l - \frac{1}{2} G^T(Q_0) \nabla_{Q_0} L(q, v) = 0$$

$$\Rightarrow \boxed{\frac{d}{dt}\left(\nabla_{\omega_0} L(q, v)\right) + \omega \times \left(\nabla_{\omega_0} L(q, v)\right) - \frac{1}{2} G^T(Q_0) \nabla_{Q_0} L(q, v) = 0} \quad (4)$$

- Equations (1), (2), (4) give the floating-base dynamics of a general robot with joint coordinates.

- This is a special case of Kane's equations and Hamel's equations.

- We can easy write this in manipulator form:

$$M(q)\dot{v} + C(q, v) + G(q) = B(q) u + Y^T(q) F$$

Mass Matrix — Coriolis Term — Potential Term — Input Jacobian

- This is the standard setup in most robotics simulators (e.g. Bullet, MuJoCo, etc.)