Last Time:

- Floating base kinematics + Jacobian
- " " dynamics in joint coordinates

Today:

- Floating base dynamics in Maximal coordinates
- Variational integrators in maximal coordinates

Maximal Coordinates:

- Joint coordinates are standard in robotics
- Conventional reasons: no constraint drift, smaller state vector
- With careful implementation, computational cost is the same
- Variational integrators eliminate constraint drift.
- Essentially combine Newton-Euler dynamics for all links with joint constraints.
- We'll focus on variational integrators because they work better for maximal coordinates.

Newton-Euler Dynamics in Discrete Time:

- We already know how to handle translation:

$$L(q,\dot{q}) = \frac{1}{2} m \dot{r}^T \dot{r} - U(r)$$

$$\Rightarrow L_d(q_n, q_{n+1}) = h\, L\left(\frac{q_n + q_{n+1}}{2}, \frac{q_{n+1} - q_n}{h}\right)$$

$$\Rightarrow D_2 L_d(q_{n-1}, q_n) + D_1 L_d(q_n, q_{n+1}) = 0$$

- Recall that we can handle external forces + constraints:

$$D_2 L_d(q_{n-1}, q_n) + D_1 L_d(q_n, q_{n+1}) + \underbrace{\frac{h}{2} F(t_{n-1} + h/2) + \frac{h}{2} F(t_n + h/2)}_{\text{Force terms}} + \underbrace{h \lambda_{n+1}^T D_1 C(q_n)}_{\text{Constraint term}} = 0$$

$$C(q_{n+1}) = 0$$

- Now we need a variational integrator for the attitude dynamics

- Let's do the un-forced case:

$$L = \frac{1}{2} \omega^T J \omega$$

- We want to write ω_n in terms of Q_n, Q_{n+1}

$$\dot{Q} = \frac{1}{2} L(Q) H\omega \Rightarrow Q_{n+1} \approx Q_n + L(Q_n) H\left(\frac{h}{2}\omega\right)$$
$$\approx Q_n * \left(\begin{bmatrix}1\\0\end{bmatrix} + H\left(\tfrac{h}{2}\omega\right)\right)$$
$$\approx L(Q_n) \begin{bmatrix}1\\ \frac{h}{2}\omega\end{bmatrix}$$

$$\Rightarrow \omega_n \approx \frac{2}{h} H^T L^T(Q_n) Q_{n+1}$$

- Now we write the discrete Lagrangian as:

$$L_d(Q_n, Q_{n+1}) = \frac{h}{2}\left[\left(\frac{2}{h} H^T(Q_n^+ * Q_{n+1})\right)^T J \left(\frac{2}{h} H^T(Q_n^+ * Q_{n+1})\right)\right]$$

- And minimize S_d w.r.t. the midpoint:

$$\min_{Q_2} \; S_d = L_d(Q_1, Q_2) + L_d(Q_2, Q_3)$$

$$\Rightarrow D_2 L_d(Q_1, Q_2) G(Q_2) + D_1 L_d(Q_2, Q_3) G(Q_2) = 0$$

Attitude Jacobian

- It is straightforward to add external torques & constraints like the translation case:

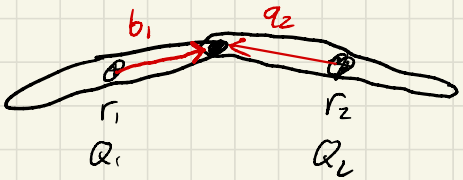
$$\frac{2}{h} G^T(Q_2) L(Q_1) H J H^T L^T(Q_1) Q_2$$
$$+ \frac{2}{h} G^T(Q_2) T R^T(Q_3) H J H^T L^T(Q_2) Q_3 + h\left(D_1 C(Q_2) G(Q_2)\right)^T \lambda_3$$
$$+ \frac{h}{2} \tau(t_1 + h/2) + \frac{h}{2} \tau(t_2 + \tfrac{h}{2}) = 0$$

$$C(Q_3) = 0$$

- Now just stack the translation + rotation equations for all links + joint constraints.

- To reproduce the "floating acrobot" from last time, we need the constraint for a revolute joint.

- For a spherical/ball joint, we set the positions of the joint on both links equal:



$$\Rightarrow r_1 + H^T R^T(Q_1) L(Q_1) H b_1 - r_2 - \underbrace{H^T R^T(Q_2) L(Q_2) H}_{\text{Rotation matrix } (B_2 \to N) \text{ from quaternion}} q_2 = 0$$

- For a revolute joint, we also constrain the relative rotation axis. Let's make it the z-axis (in the link frame(s)):

- The relative rotation between the links is

$$\underbrace{\Delta Q = Q_1^+ * Q_2}_{Q_2 = Q_1 \Delta Q} = \begin{bmatrix} \cos(\theta/2) \\ r \sin(\theta/2) \end{bmatrix}$$

(r: axis of rotation; ΔQ: defined in Q_1 frame)

- To only allow rotation about the z-axis, we set the x and y components of the vector part of ΔQ to zero:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} L^T(Q_1) Q_2 = 0$$

- Now just stack everything.

* Example