Last Time:

- Floating-Base Kinematics in 3D
- " " 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:

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

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

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

- Recall that we can handle external forces + constraints:

$$D_2 L_d(q_{k-1}, q_k) + D_1 L_d(q_k, q_{k+1}) + \frac{h}{2} F(t_{k-1} + \frac{h}{2}) + \frac{h}{2} F(t_k + \frac{h}{2}) + h \lambda_{k+1}^T D_1 C(q_k) = 0$$

Forces

constraint

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

- Now we need a variational integrator for the attitude dynamics.

- Let's do the unforced case:

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

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

$$\hat{\dot{Q}} = \frac{1}{2} L(Q) H \omega \Rightarrow Q_{n+1} \approx Q_n + L(Q_n) H \left(\frac{h}{2}\omega_n\right)$$

$$\approx Q_n * \left(\begin{bmatrix} 1 \\ 0 \end{bmatrix} + H\left(\frac{h}{2}\omega_n\right) \right)$$

$$\approx L(Q_n) \begin{bmatrix} 1 \\ \frac{h}{2}\omega_n \end{bmatrix}$$

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

- Now we write the discrete Lagrangian:

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

- And minimize S_d wrt. the midpoint:

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

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

Attitude Jacobians

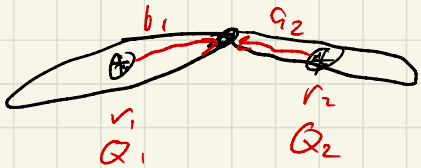
- It is straightforward to add external torques + constraints like the translation case:

$$\frac{2}{h} G^T(Q_2) L(Q_1) H J H^T L^T(Q_1) Q_2$$
$$+ \frac{2}{h} G^T(Q_2) T R^T(Q_3) H J H^T L^T(Q_2) Q_3 \quad \Big\} \text{DEL}$$
$$+ h (D_1 C(Q_2))\, G^T(Q_2))^T \lambda_2 \quad \leftarrow \text{constraints}$$
$$+ \frac{h}{2} \tau(t_1 + \tfrac{h}{2}) + \frac{h}{2} \tau(t_2 + \tfrac{h}{2}) = 0 \quad \leftarrow \text{torques}$$

$$C(Q_3) = 0$$

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

- Example: "floating acrobot":



$$\Rightarrow r_1 + H^T R(Q_1) L(Q_1) H\, b_1 - r_2 - \underbrace{H^T R(Q_2) L(Q_2)}_{\text{rotation matrix } B_2 \to N \text{ from quaternion}} H\, a_2 = 0$$

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

  - Relative rotation between links:

$$\underbrace{\Delta Q = Q_1^+ Q_2}_{Q_2 = Q_1 \circ \Delta Q} = \begin{bmatrix} \cos(\theta/2) \\ \underbrace{\frac{a}{\|a\|}}_{\text{axis of rotation}} \sin(\theta/2) \end{bmatrix}$$

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 stack everything + solve with Newton