Last Time:

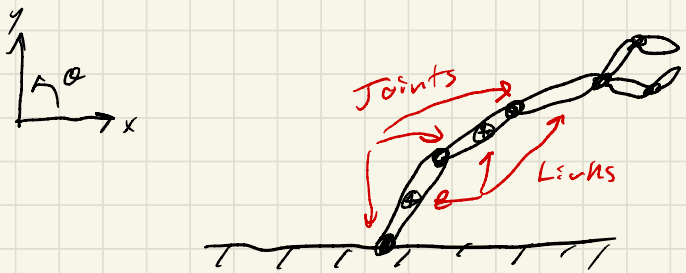
- Coulomb Friction
- Maximum Dissipation Principle
- LCP Methods

Today:

- Fixed-Base Manipulators
- Forward Kinematics
- Floating-Base Systems

Fixed-Base Manipulators:

- Classic serial/open-chain systems e.g. industrial robot arms.



- Note (for now) we won't allow any joints that form closed loops.

- For systems like this, we can easily compute the position + orientation of each link in the world frame from the joint angles. Called "forward kinematics"

- In 2D:

orientation in world frame → $\theta_1 = q_1$ ← joint angle

$$r_1 = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{R(\theta)} a_1$$

r_1: link CoM position in world frame

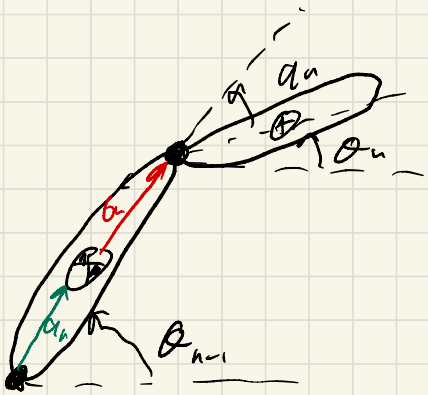
a_1: vector from joint 1 to CoM of link 1 in link frame

from $n = 2:N$

$$\theta_n = \theta_{n-1} + q_n$$

$$r_n = r_{n-1} + R_{n-1} b_{n-1} + R(\theta_n) a_n$$

b_{n-1}: vector from CoM to next joint



- In 3D:

$$
\begin{aligned}
{}^{N}R_1^{B_1} &= S_1(q_1) \\
{}^{N}r_1 &= {}^{N}R_1^{B}\, a_1 \\
R_n &= R_{n-1}\, S_n(q_n) \\
r_n &= r_{n-1} + R_{n-1}\, b_{n-1} + R_n\, a_n
\end{aligned}
$$

- Let's focus on 2D for now.
- Write down kinematics as a function:

$$
\begin{bmatrix} r_1 \\ \theta_1 \\ r_2 \\ \theta_2 \\ \vdots \\ r_n \\ \theta_n \end{bmatrix} = K(q)
$$

↳ $\mathbb{R}^n$ vector of joint angles

- From this we can compute world-frame linear + angular velocities of each link:

$$
\frac{d}{dt}\begin{bmatrix} r_1 \\ \theta_1 \\ \vdots \\ r_n \\ \theta_n \end{bmatrix} = \frac{\partial K(q)}{\partial q}\dot{q}
$$

↳ $K(q) \in \mathbb{R}^{3n \times n}$

"Kinematic Jacobian"

- Given mass + inertia of each link, we can compute kinetic energy:

$$T = \sum_{k=1}^{n} \frac{1}{2} m_k \dot{r}_k^T \dot{r}_k + \frac{1}{2} J_k \dot{\theta}_k^2$$

$$= \frac{1}{2} \begin{bmatrix} \dot{r}_1 \\ \dot{\theta}_1 \\ \vdots \\ \dot{r}_n \\ \dot{\theta}_n \end{bmatrix}^T \underbrace{\begin{bmatrix} m_1 I & & & & \\ & J_1 & & 0 & \\ & & \ddots & & \\ & 0 & & m_n I & \\ & & & & J_n \end{bmatrix}}_{\bar{M}} \begin{bmatrix} \dot{r}_1 \\ \dot{\theta}_1 \\ \vdots \\ \dot{r}_n \\ \dot{\theta}_n \end{bmatrix}$$

$$= \frac{1}{2} \dot{q}^T \underbrace{K(q)^T \bar{M} K(q)}_{M(q)} \dot{q} = \frac{1}{2} \dot{q} M(q) \dot{q}$$

- Similarly, we can compute $V(q)$ and $L(q, \dot{q})$

* Example: Forced double pendulum
  - Joint torques are easy in this setup
  - External forces applied to the links take a little more work

- We can get this from the Lagrange-D'Alembert Principle:

$$\frac{\partial}{\partial q(s)}\left[\int_{t_0}^{t_f} L(q,\dot{q})\,dt + \int_{t_0}^{t_f} F(t)^T y(t)\,dt\right]$$

defined w.r.t. some coordinates y

- If we take the variational derivative:

$$\int_{t_0}^{t_f}\left[\frac{\partial L}{\partial q} - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right)\right]\Delta q\,dt + \int_{t_0}^{t_f} F(t)^T \frac{\partial y}{\partial q}\Delta q\,dt$$

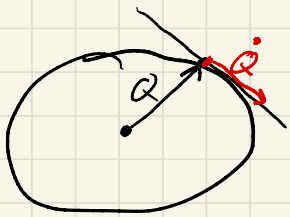
$$\Rightarrow M(q)\ddot{q} + C(q,\dot{q}) + G(q) = Y(q)^T F$$

* Example: double pendulum w/ external force

- To make this work in 3D we need some quaternion tricks.

Differentiating with Quaternions:

- Geometrically, a unit quaternion is confined to the unit sphere in $\mathbb{R}^4$:



- Intuitively, derivatives of Q live in local 3D tangent plane (like ω):

$$\dot{Q} = \tfrac{1}{2} Q * \hat{\omega} = \tfrac{1}{2} L(Q) H \omega$$

$$\hat{\omega} = \begin{bmatrix} 0 \\ \omega \end{bmatrix}$$

- I want a 3D gradient of $f(Q) : \mathbb{H} \to \mathbb{R}$

- We can parameterize a tiny rotation with an axis-angle vector ϕ:

$$\Delta Q = \begin{bmatrix} \cos\left(\frac{1}{2}\|\phi\|\right) \\ \frac{\phi}{\|\phi\|} \sin\left(\frac{1}{2}\|\phi\|\right) \end{bmatrix} \approx \begin{bmatrix} 1 \\ \frac{1}{2}\phi \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{2}\phi \end{bmatrix}$$

$$\frac{1}{2}\hat{\phi}$$

- Now plug this into $f(Q)$:

$$f + \Delta f = f(Q * \Delta Q) \simeq f\left(Q * \begin{bmatrix} 1 \\ \frac{1}{2}\phi \end{bmatrix}\right) = f\left(Q + Q * \tfrac{1}{2}\hat{\phi}\right)$$

$$\approx f(Q) + \frac{\partial f}{\partial Q} L(Q) H \tfrac{1}{2}\phi$$

$$\Rightarrow \Delta f \approx \tfrac{1}{2} \frac{\partial f}{\partial \phi} \underbrace{L(Q) H}_{G(Q)} \phi$$

$G(Q)$ "Attitude Jacobian"

$$\Rightarrow \frac{\partial f}{\partial \phi} = \frac{1}{2} \frac{\partial f}{\partial Q} G(Q)$$

- Whenever we want to diff a function w.r.t. a quaternion input take the standard Jacobian an right multiply by $G(Q)$.

- We can play a similar trick on functions that output a quaternions:

$$Q = f(x) : \mathbb{R}^n \to \mathbb{H}$$

$$Q' = Q * \Delta Q \approx Q + Q * \tfrac{1}{2}\hat{\phi} = Q + L(Q) H \tfrac{1}{2}\phi$$

$$\approx f(x + \Delta x) \approx f(x) + \frac{\partial f}{\partial x}\Delta x$$

$$\Rightarrow \tfrac{1}{2} L(Q) H \phi = \frac{\partial f}{\partial x}\Delta x$$

Orthogonal $\Rightarrow L^T = L^{-1}$

$$\Rightarrow H\phi = 2 L(Q)^T \frac{\partial f}{\partial x}\Delta x$$

$$\Rightarrow \phi = 2 \underbrace{H^T L^T(Q)}_{G^T(Q)} \frac{\partial f}{\partial x}\Delta x$$

- Whenever we want to diff a function with a quaternion output take the standard Jacobian and left multiply by $G^T(Q)$