Last Time:

- Optimization with Quaternions

Today:

- LQR with Quaternions
- Hybrid systems for modeling contact

---

* LQR with Quaternions

- Naively linearizing a system with a quaternion state results in an uncontrollable linear system
- We'll apply our quaternion differentiation tricks to LQR to make this work.
- Given a reference $\bar{x}_n, \bar{u}_n$ for a discrete-time system $f(x, u_n)$:

$$\bar{x}_{n+1} + \Delta x_{n+1} = f(\bar{x}_n + \Delta x_n, \bar{u}_n + \Delta u)$$

$$\approx f(\bar{x}_n, \bar{u}_n) + \underbrace{A_n}_{\left.\frac{\partial f}{\partial x}\right|_{\bar{x},\bar{u}}} \Delta x_n + \underbrace{B_n}_{\left.\frac{\partial f}{\partial u}\right|_{\bar{x},\bar{u}}} \Delta u_n$$

- For the quaternion part of the state, we apply the attitude Jacobian to convert $\Delta q \to \phi$

$$\underbrace{\begin{bmatrix} \Delta x_{n+1}[1:3] \\ \phi_{n+1} \\ \Delta x_{n+1}[8:n] \end{bmatrix}}_{\Delta x_{n+1}} = \underbrace{\begin{bmatrix} I & & \\ & G(\bar{q}_{n+1}) & \\ & & I \end{bmatrix}^T}_{E(\bar{x}_{n+1})} A_n \underbrace{\begin{bmatrix} I & & \\ & G(\bar{q}_n) & \\ & & I \end{bmatrix}}_{E(\bar{x}_n)} \Delta \bar{x}_n + E(\bar{x}_{n+1})^T B_n \Delta u_n$$

- Once we have these "Reduced" Jacobians $\bar{A}_n$, $\bar{B}_n$

$$\bar{A}_n = E^T(\bar{x}_{n+1})\, A_n\, E(\bar{x}_n), \quad \bar{B}_n = E^T(\bar{x}_{n+1})\, B_n$$

we compute the LQR gains as usual.

- When we run the controller, we calculate $\delta\tilde{x}$ before multiplying by K:

given x_n, $\delta\tilde{x} = \begin{bmatrix} x_n[1:3] - \bar{x}_n[1:3] \\ \phi(L(\bar{q})^T q_n) \\ x_n[8:n] - \bar{x}_n[8:n] \end{bmatrix}$

whatever 3-parameter representation you like

$$u_n = \bar{u}_n - K_n\, \delta\tilde{x}_n$$

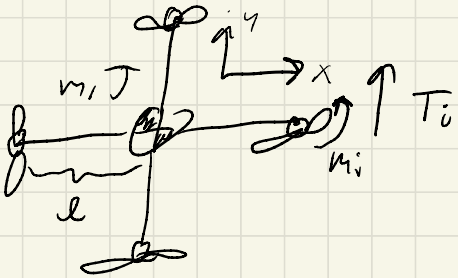
- Computing rotation error:

$${}^{N}_{B_n}Q, \quad {}^{N}_{B_{n+1}}Q \Rightarrow {}^{B_{n+1}}_{B_n}Q = Q_{n+1}^T Q_n$$

$$\delta Q = \bar{Q}^T Q \iff \delta q = \bar{q}^+ * q$$

$$L(\bar{q})^T q$$

* 3D Quadrotor



$T_i = k_T u_i$, $u \in \mathbb{R}^4$

$m_i = k_m u_i$

- State :

$$x = \begin{bmatrix} {}^N r \in \mathbb{R}^3 & \text{position in } N \text{ frame} \\ q \in \mathbb{H} & \text{attitude} \\ {}^B V \in \mathbb{R}^3 & \text{linear velocity in } B \text{ frame} \\ {}^B \omega \in \mathbb{R}^3 & \text{angular velocity in } B \text{ frame} \end{bmatrix}$$

- Kinematics

$${}^N \dot{r} = {}^N V = Q\, {}^B V$$

$$\dot{q} = \frac{1}{2} q * \hat{\omega} = \frac{1}{2} L(q) H\, {}^B \omega$$

- Translation Dynamics :

$$m\, {}^N \dot{V} = {}^N F \quad \leftarrow \text{total force}$$

$\leftarrow$ want ${}^B \dot{V}$

$${}^N V = Q\, {}^B V \Rightarrow {}^N \dot{V} = \dot{Q}\, {}^B V + Q\, {}^B \dot{V} = Q \hat{\omega}\, {}^B V + Q\, {}^B \dot{V}$$

$$\Rightarrow {}^B \dot{V} = Q^T {}^N \dot{V} - {}^B \omega \times {}^B V$$

extra term

$$\Rightarrow {}^B \dot{V} = \frac{1}{m} {}^B F - {}^B \omega \times {}^B V$$

$$^{B}F = Q^{T}\begin{bmatrix}0\\0\\-mg\end{bmatrix} + \begin{bmatrix}0 & 0 & 0 & 0\\0 & 0 & 0 & 0\\k_T & k_T & k_T & k_T\end{bmatrix} u$$

- Rotation Dynamics:

$$\underbrace{J\,{}^{B}\dot{\omega} + {}^{B}\omega \times J\,{}^{B}\omega}_{\text{Euler's Equation}} = {}^{B}\tau \quad \leftarrow \text{Total torque}$$

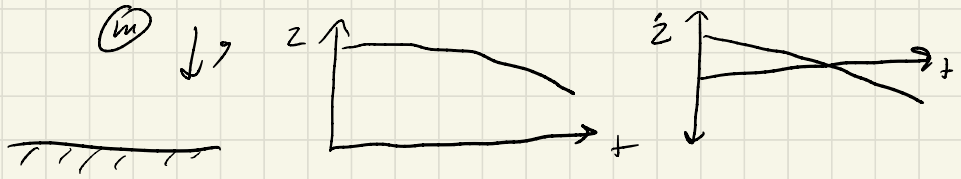
$$^{B}\tau = \begin{bmatrix} l\,k_T\,(u_2 - u_4)\\ l\,k_T\,(u_3 - u_1)\\ k_m\,(u_1 - u_2 + u_3 - u_4)\end{bmatrix}$$

* Example

---

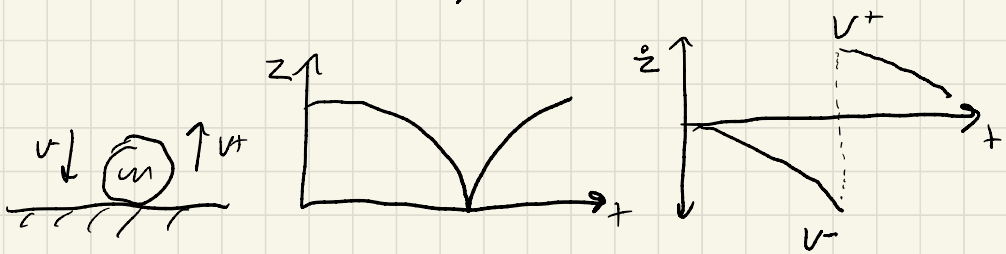
* Contact Dynamics

- Imagine a bouncing ball:



- In the air, dynamics are described by smooth ODE ($m\ddot{z} = -g$)

- When ball hits the ground:



- because of the discontinuity, can't write down dynamics around impact with an ODE.

- Two Options:

1) Event-based / hybrid formulation: Integrate ODE while checking for contact events using a "guard function" (e.g. $z \geq 0$). When contact event happens, execute "jump map" that models discontinuity, then continue integrating ODE.

2) Time-Stepping / Contact implicit formulation: Solve a constrained optimization problem at every time step that enforces no interpenetration between objects ($\phi(x) \geq 0$) by solving for contact forces. (HW 2 brick problem).

- Both are widely used in simulation and have pros/cons.
- In control, hybrid formulation is easy to apply with standard algorithms (e.g. DDP/DIRCOL), and is most common.
- Downside: requires pre-specified contact mode sequence (which parts of the robot are in contact at each knot point)
- Despite this, hybrid methods have been very succesful in locomotion.
- Contact implicit method doesn't need mode sequence pre-specified but the optimization problems are much harder. Active research area.