Last Time:

- SQP
- Direct Collocation

Today:

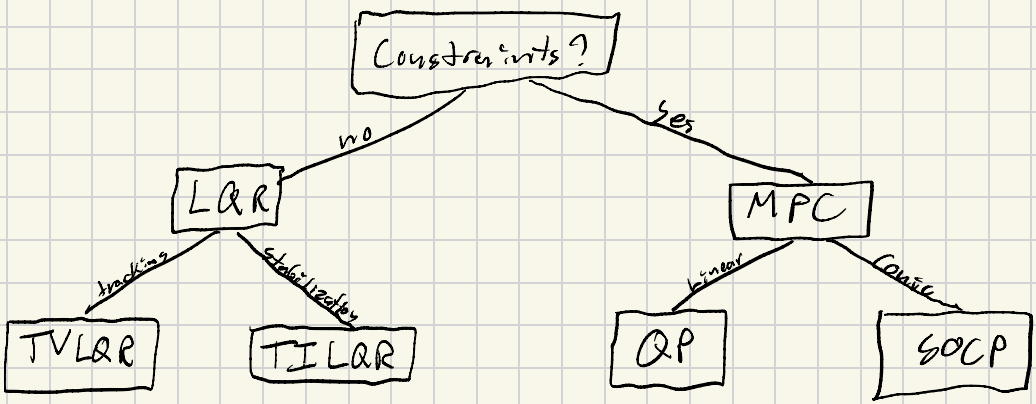
- Algorithm Recap
- Dealing with Attitude (3D rotations)

---

* Recap: Deterministic Optimal Control Algorithms:

- Linear / "Local" Control

≈ "Linearization works"



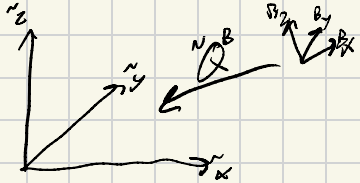
- Nonlinear Trajectory Optimization

DIRCOL	DDP
Only respects dynamics at convergence	Always dynamically feasible
Can use any infeasible guess	can only guess controls
Can handle arbitrary constraints	Hard to state constraints
Tracking controller must be designed separately	TVLQR tracking controller is free
Typically not as fast	Very fast (local) convergence
Difficult to implement large-scale SQP solver	Easy to implement on embedded systems
Numerically robust	Has issues with ill-conditioning

Attitude

- Many robotic systems undergo large rotations (quadrotors, airplanes, spacecraft, underwater vehicles, legged robots)
- Naive angle parameterizations (Euler angles) have singularities that cause failures and/or require hacks
- Rotation matrices and quaternions are singularity free but optimizing over them requires some extra tricks

- What is Attitude?



$$^N v = {}^N Q^B \, {}^B v$$

- Rotation from body frame to world frame
- 3 DOF, but there is no globally nonsingular 3-parameter attitude representation

- Rotation / "Direction-Cosine" Matrix

$$\begin{bmatrix} ^N x_1 \\ ^N x_2 \\ ^N x_3 \end{bmatrix} = \underbrace{\begin{bmatrix} ^B n_1^T \\ ^B n_2^T \\ ^B n_3^T \end{bmatrix}}_{^N Q^B} \begin{bmatrix} ^B x_1 \\ ^B x_2 \\ ^B x_3 \end{bmatrix} = \begin{bmatrix} ^N b_1 & ^N b_2 & ^N b_3 \end{bmatrix} \begin{bmatrix} ^B x_1 \\ ^B x_2 \\ ^B x_3 \end{bmatrix}$$

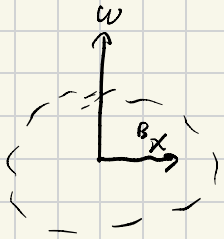
$$\Rightarrow \underbrace{Q^T Q = I}_{\text{"Orthogonal"}}$$

$$\Rightarrow Q^{-1} = Q^T$$

$$\Rightarrow \underbrace{\det(Q) = 1}_{\text{"Special"}}$$

$\Rightarrow Q \in SO(3)$ "Special Orthogonal group in 3D"

- Kinematics (how do I integrate a gyro)



$$^N x = Q_{(4)}\, ^B x \quad , \quad ^N \dot{x} = {}^N\omega \times {}^N x$$

$$= Q(^B\omega \times {}^B x)$$

- Apply product rule:

$$^N x = Q_{(4)}\, ^B x \Rightarrow {}^N\dot{x} = \dot{Q}\, ^B x + Q\, ^B\dot{x}$$

$$\Rightarrow \dot{Q}\, ^B x = Q(^B\omega \times {}^B x)$$

$$\omega \times x = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \hat{\omega}\, x$$

$\hat{\omega}$

$$\Rightarrow \dot{Q}\, ^B x = Q\hat{\omega}\, ^B x \Rightarrow \boxed{\dot{Q} = {}^N Q^B \hat{\omega}}$$

- We can do dynamics with rotation matrices in our state, but has a lot of redundancy
- Quaternions are more compact/efficient

Quaternions

- Define axis of rotation (unit vector) a
- Define angle of rotation (scalar, radians) θ

$$\phi = a\theta$$

"axis-angle" vector $\in \mathbb{R}^3$

$$\|\phi\| = \theta, \quad \frac{\phi}{\|\phi\|} = a$$

- For constant ω (for short h) can think of ϕ as integral of ω:

$$\phi \approx \int_0^h \omega(t)\, dt \qquad \text{(not true in general)}$$

- In terms of ϕ, we can define quaternion:

$$q = \begin{bmatrix} \cos(\theta/2) \\ a \sin(\theta/2) \end{bmatrix} \leftarrow \text{"scalar part"} \leftarrow \text{"vector part"}$$

$q^T q = 1 \Rightarrow$ valid rotations correspond to unit quaternions
Easy to normalize

* q and $-q$ correspond to the same rotation (add 2π to θ)
Called a "double cover"

* Operations on quaternions are analogous to rotation matrices

- Quaternion Multiplication

$$q * q_2 = \begin{bmatrix} s_1 \\ v_1 \end{bmatrix} * \begin{bmatrix} s_2 \\ v_2 \end{bmatrix} = \begin{bmatrix} s_1 s_2 - v_1^T v_2 \\ s_1 v_2 + s_2 v_1 + v_1 \times v_2 \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} s_1 & -v_1^T \\ v_1 & s_1 I + \hat{v}_1 \end{bmatrix}}_{L(q_1)} \begin{bmatrix} s_2 \\ v_2 \end{bmatrix} = \underbrace{\begin{bmatrix} s_2 & -v_2^T \\ v_2 & s_2 I - \hat{v}_2 \end{bmatrix}}_{R(q_2)} \begin{bmatrix} s_1 \\ v_1 \end{bmatrix}$$

- Quaternion Conjugate

$$q^+ = \begin{bmatrix} s \\ -v \end{bmatrix} = T q = \begin{bmatrix} 1 & 0 \\ 0 & -I \end{bmatrix} \begin{bmatrix} s \\ v \end{bmatrix}$$

- Rotate a Vector:

$$\underbrace{\begin{bmatrix} 0 \\ v \\ x \end{bmatrix}}_{Hx} = q * \begin{bmatrix} 0 \\ x \end{bmatrix} * q^+ = \underbrace{\begin{matrix} H^T L(q) R^T(q) H \\ H^T R^T(q) L(q) H \end{matrix}}_{Q(q)} \begin{matrix} \\ x \end{matrix}$$

$$Hx = \begin{bmatrix} 0 \\ x \end{bmatrix} \Rightarrow H = \begin{bmatrix} 0 \\ I \end{bmatrix}$$

- Quaternion Kinematics:

$$\dot{q} = \frac{1}{2} q * \begin{bmatrix} 0 \\ \omega \end{bmatrix} = \frac{1}{2} \underbrace{L(q) H}_{4 \times 3} \omega$$

- Now we can simulate dynamics with rotations