Simulation & Linear control of Newton-Euler equations of motion for a rigid body.

Newton-Euler equations specify how linear velocity & angular velocity evolves with time:

all forces $\sum \vec{f} = m\dot{\vec{v}}$ $\quad \vec{v} \triangleq$ linear velocity of center of mass (CM) (1)

all torques $\sum \vec{\tau} = J\dot{\vec{\omega}} - \vec{\omega} \times J\vec{\omega}$ $\quad \vec{\omega} \triangleq$ angular velocity of body (2)

moment of inertia about CM

We also need to know how position & orientation evolve with time.

position of CM is $\vec{p}$

$$\dot{\vec{p}} = \vec{v} \tag{3}$$

$$\dot{R} = R\vec{\omega}^{\times} \tag{4}$$

$R \triangleq$ orientation of body ($\mathbb{R}^{3\times3}$ matrix for 3D, $\mathbb{R}^{2\times2}$ matrix for 2D)

$\vec{\omega}^{\times}$ is a matrix representation of the cross product operation $\vec{\omega}\times$ ($\mathbb{R}^{3\times3}$ or $\mathbb{R}^{2\times2}$)

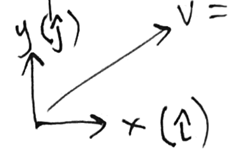
Background on vectors.

a vector $\vec{v}$ exists independent of coordinate system. we will use two representations:

1) as a directed line segment w/direction & mag w/no coordinate system (as above) (for math)

2) as an array of values representing quantities of unit vectors in a given coord. system. useful for simulation.

Example:



$$\vec{v} = v_1\hat{i} + v_2\hat{j} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$ array of values

Two coordinate systems of interest: world frame $(\hat{\imath}, \hat{\jmath}, \hat{k})$

body frame $(\hat{\imath}', \hat{\jmath}', \hat{k}')$

body frame is typically rotated.

slight abuse of notation: $\vec{v}$: is either a pure vector, or vector in world coords

$\vec{v}'$: vector expressed in body attached coords

Rotation matrix R translates between them. $\vec{v} = R\vec{v}'$

2D case: suppose $\hat{\imath} = R_{11}\hat{\imath}' + R_{12}\hat{\jmath}'$
$\hat{\jmath} = R_{21}\hat{\imath}' + R_{22}\hat{\jmath}'$ $\Rightarrow R = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}$

$\det(R) = 1$, $R^{-1} = R^{T}$ (special properties) thus $\vec{v}' = R^{T}\vec{v}$

in 2D, if body is rotated by θ, $R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$

so we can write eq's (1)-(4) in terms of coordinates:

$$\sum \vec{f} = \underset{\text{gravity}}{\vec{f}_g} + \underset{\text{thrust}}{\vec{f}_z} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R\begin{bmatrix} 0 \\ 0 \\ f_z \end{bmatrix} = m\dot{\vec{v}} \qquad \text{(world)}$$

$$\sum \vec{\tau}' = \begin{bmatrix} \tau_x' \\ \tau_y' \\ \tau_z' \end{bmatrix} = J\dot{\vec{\omega}}' + \vec{\omega}' \times J\vec{\omega}' \qquad \text{(body)}$$

Why body? because J is usually known in body-attached coordinates, and can stay fixed.

$\dot{R} = R\vec{\omega}'^{\times}$, $\dot{\vec{p}} = \vec{v}$

LQR Control.

from previous lecture, Know that if $Re\{\lambda_i\} < 0 \ \forall \lambda_i$ ("for all"), for the linearized dynamics, then there exists a neighborhood for which full non linear sys is stable.

choose controller of form $\vec{u} = -K\vec{q}$ to insure linearized dynamics are stable $\Rightarrow$ full sys is stable. How choose K?

LQR control: optimize performance relative to quadratic cost function

$$J = \int_0^\infty (\vec{q}^T Q \vec{q} + \vec{u}^T R \vec{u}) dt$$

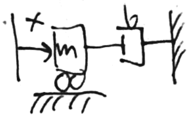
not moment of inertia

given sys $\dot{q} = Aq + Bu$, and $R > 0$ (all $\lambda_i > 0$), $Q \geq 0$ (all $\lambda \geq 0$):

K = control.lqr(A, B, Q, R)[0]

R, Q symmetric, (A, B) controllable.

Example: LQR of mass-damper system.



$$m\ddot{x} = f - b\dot{x} \Rightarrow q = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}, \ \dot{q} = \begin{bmatrix} \dot{x} \\ \frac{f}{m} - \frac{b}{m}\dot{x} \end{bmatrix}$$

$$\Rightarrow A = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{b}{m} \end{bmatrix}, \ B = \begin{bmatrix} 0 \\ 1/m \end{bmatrix}$$

Check: reachable? python: numpy.linalg.det(control.ctrb(A,B)) $\neq 0$ ✓

a) LQR controller design: typically, R is known (energy used by motors) and Q is varied to get desired performance. Q typically diagonal $= \begin{bmatrix} q_1 & \\ & q_2 \end{bmatrix}$

eg. 1 cm error OK $\Rightarrow q_1 = \left(\frac{1}{.01}\right)^2 \Rightarrow q_1 x^2 = 1$ when $x = 1$ cm

10 cm/s error OK $\Rightarrow q_2 = \left(\frac{1}{.1}\right)^2 \Rightarrow q_1 \dot{x}^2 = 1$ when $\dot{x} = 10$ cm/s

K, S, E = control.lqr(A, B, Q, R)

b) track trajectory $\vec{q}(t)$: define $\vec{e}_q = \vec{q}_d - \vec{q}$, then $\vec{u} = -K\vec{e}_q$.