Spacecraft Dynamics & Simulation

* "Dynamics"

$$\dot{x} = f(t, x, u)$$

(t: time; x: state vector; u: control inputs vector)

- Can get this from e.g. Newton
- Can always write as 1st order vector ODE:

$$F = ma, \quad x = \begin{bmatrix} r \\ v \end{bmatrix} \Rightarrow a = \dot{v}$$

(r: position; v: velocity)

$$\dot{x} = \begin{bmatrix} \dot{r} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \\ \frac{1}{m} F \end{bmatrix}$$

* "Simulation"

- Usually we can't solve these ODEs for $x(t)$ analytically:

$$x(t) = x(0) + \underbrace{\int_0^t f(t, x, u)\, dt}_{\text{this is a nasty integral}}$$

- We use numerical methods to approximate the solution
- Many options, but most common are Runge-Kutta
- These fit a polynomial to $x(t)$ over a (short) time step by evaluating f

- "Order" = order of polynomial
- "Stages" = number of f evals
- Classic 4th-order Runge Kutta Method ("RK4")

$$
\begin{aligned}
k_1 &= f(t_n, x_n) \\
k_2 &= f(t_n + \tfrac{1}{2}\Delta t, x_n + \tfrac{1}{2}\Delta t k_1) \\
k_3 &= f(t_n + \tfrac{1}{2}\Delta t, x_n + \tfrac{1}{2}\Delta t k_2) \\
k_4 &= f(t_n + \Delta t, x_n + \Delta t k_3)
\end{aligned}
\quad \} \text{ 4 stages}
$$

$$x_{n+1} = x_n + \tfrac{1}{6}\Delta t (k_1 + 2k_2 + 2k_3 + k_4)$$

- Lots of extensions (e.g. adaptive step size) that you can find in libraries (e.g. ode45 in Matlab)

* Spacecraft Dynamics

- Spacecraft can mostly be modeled as rigid bodies
- Rigid body ⇒ ignore flexible dynamics
- For a cubesat:
  - Orbit dynamics ≈ 1/hour
  - Attitude dynamics ≈ 1 Hz
  - Flexible dynamics ≈ 1 kHz
- This is less true for larger spacecraft

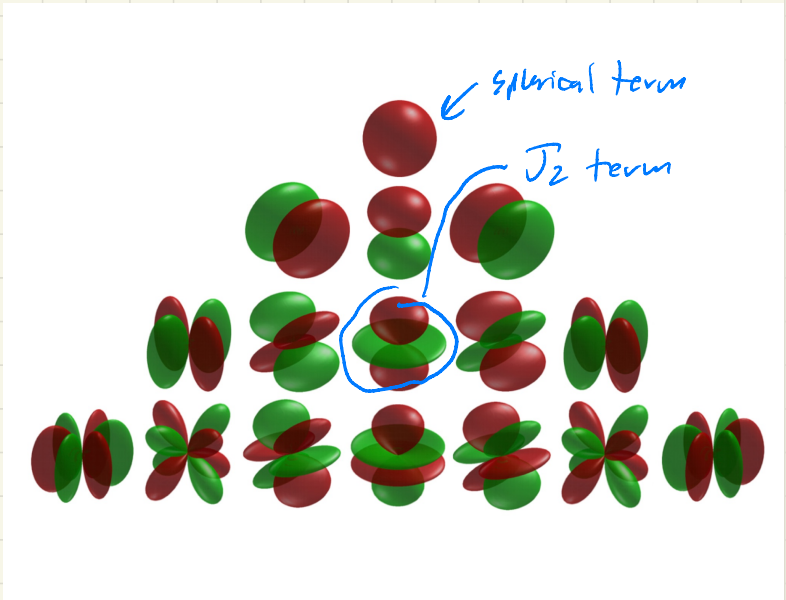
* Orbit Dynamics

- What forces act on a satellite in LEO?
  - Gravity (not just spherical)
  - Atmospheric drag
  - Solar radiation pressure

- Gravity

"Standard Gravitational Parameter" = GM

$$a_g = \underbrace{\frac{-\mu r}{\|r\|^3}}_{\text{"Spherical term"}} + \underbrace{J_2(r)}_{J_2\text{ "doughnut" term}}$$



- In LEO $\|J_2\| \approx \|F_{drag}\|$. Next term is $\approx 1000\times$ smaller.

- Drag:

$$F_{drag} = -\frac{1}{2} C_D \rho A v \|V\|$$

ρ: Atm. Density; V: velocity vector; C_D: Drag coefficient; A: surface area

$C_D \approx 2.2$ for spacecraft in LEO

$\rho \approx 10^{-15} - 10^{-11}$ kg/m^3 (use a model like Harris-Priester)

- Add up all forces to get the dynamics:

$$\underbrace{\begin{bmatrix} \dot{r} \\ \dot{v} \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} v \\ a_g(r) + \frac{1}{m} F_{drag}(r, v) \end{bmatrix}}_{f(x)}$$

---

* Attitude

- What is attitude?
- Rotation from some body-fixed frame to some inertial frame (e.g. ECI)
- We can parameterize this many ways
  - Euler angles (roll, pitch, yaw etc.) (bad)
  - Quaternion (4D unit vector) (good)
  - Axis angle vectors (Okish)
  - Rotation matrices (good)

---

* Rotation Matrices:

- Rotates components of a vector from body to inertial frame:

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

- Rows are n_i basis vectors expressed in B components

$$\begin{bmatrix} {}^N x_1 \\ {}^N x_2 \\ {}^N x_3 \end{bmatrix} = \begin{bmatrix} {}^B n_1^T \\ {}^B n_2^T \\ {}^B n_3 \end{bmatrix} {}^B x$$

- Columns are b_i basis vectors in N components

$${}^N x = \begin{bmatrix} {}^N b_1 & {}^N b_2 & {}^N b_3 \end{bmatrix} \begin{bmatrix} {}^B x_1 \\ {}^B x_2 \\ {}^B x_3 \end{bmatrix}$$

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

$\Rightarrow Q$ is an orthogonal matrix ($Q \in O(3)$)

- $\det(Q) = 1$ called "special"

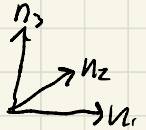
$\Rightarrow Q \in SO(3)$

Rotation Kinematics

- How do we integrate a gyro?

$$\omega(t) \xrightarrow{?} \dot{Q}(t) \longrightarrow Q(t)$$

- Velocities in a rotating frame



$${}^N\dot{r} = {}^N Q^B ({}^B\dot{r} + {}^B\omega \times {}^B r)$$

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

"Kinematic transport theorem"

- Think about a vector fixed in the body frame!

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

($Q\,{}^B\dot{r} = 0$)

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

* Define skew-symmetric "hat" opperator

$$\omega \times r = \hat{\omega} x \Rightarrow \hat{\omega} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$$

$$\Rightarrow \boxed{\dot{Q} = Q\hat{\omega}}$$

* This a linear 1st order ODE. For constant ω:

$$Q(t) = Q_0 e^{t\hat{\omega}}$$

Matrix exponential (expm in Matlab)

* We can also define axis-angle vectors:

$$\hat{\phi} = \log(Q), \qquad \phi = r\theta$$

θ: angle in radians

r: unit vector axis of rotation

Quaternions:

- Naive integrate $\dot{Q} = Q\hat{\omega}$ will lead to $Q \notin SO(3)$ due to numerical error
- Projecting back onto $SO(3)$ is a pain (need SVD)
- Quaternions are nicer for simulation

* Geometry:

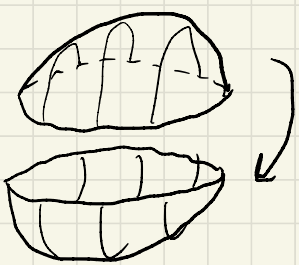
- Set of all possible axis-angle vectors $-\pi \leq \|\phi\| \leq \pi$ is a ball in $\mathbb{R}^3$

- Visualize as a disk in $\mathbb{R}^2$:

- Discontinuous jump when we cross $\pm\pi$ that causes "kinematic singularity"

- We want to get rid of the jump:

1) Stretch disk up out of plane into hemisphere:



2) Make a copy

3) Rotate the copy and glue it on underneath to make a sphere

- Now instead of jumping, we can continue smoothly onto the "southern hemisphere"

- There are 2 quaternions for every rotation matrix ("double cover of $SO(3)$")

* Useful Quaternion Stuff:

* Points on the unit sphere in 4D are given by:

$$q = \begin{bmatrix} r \sin(\theta/2) \\ \cos(\theta/2) \end{bmatrix}, \qquad r = \text{axis (unit vector)}, \quad \theta = \text{angle (radians)}$$

$$= \begin{bmatrix} \underline{v} \\ s \end{bmatrix} \leftarrow \text{"vector part"} \leftarrow \text{"scalar part"}$$

- turns out this is the quaternion exponential

* Identity quaternion

$$q_I = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

* Quaternion "Conjugate" (~ transpose/inverse)

$$q^+ = \begin{bmatrix} r \sin(-\theta/2) \\ \cos(-\theta/2) \end{bmatrix} = \begin{bmatrix} -\underline{v} \\ s \end{bmatrix}$$

* Quaternion Multiplication

$$q_1 * q_2 = \begin{bmatrix} \underline{v}_1 \\ s_1 \end{bmatrix} * \begin{bmatrix} \underline{v}_2 \\ s_2 \end{bmatrix} = \begin{bmatrix} s_1 \underline{v}_2 + s_2 \underline{v}_1 + \underline{v}_1 \times \underline{v}_2 \\ s_1 s_2 - \underline{v}_1^T \underline{v}_2 \end{bmatrix}$$

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

- $L(q^+) = L^T(q)$, $R(q^+) = R^T(q)$

* How to rotate vectors?

$$\begin{bmatrix} {}^{A}\dot{x} \\ 0 \end{bmatrix} = q * \begin{bmatrix} {}^{B}\dot{x} \\ 0 \end{bmatrix} * q^{+}$$

* Quaternions work just like rotation Matrices:

$$Q_3 = Q_2 Q_1 \leftrightarrow q_3 = q_2 * q_1$$

$$Q_1 = Q_2^T Q_3 \leftrightarrow q_1 = q_2^{+} * q_3$$

$${}^{A}\hat{\dot{x}} = Q\, {}^{B}\hat{\dot{x}}\, Q^T \leftrightarrow \begin{bmatrix} {}^{A}\dot{x} \\ 0 \end{bmatrix} = q * \begin{bmatrix} {}^{B}\dot{x} \\ 0 \end{bmatrix} * q^{+}$$

* Quaternion Kinematics:

$$\dot{q} = \frac{1}{2} q * \begin{bmatrix} \omega \\ 0 \end{bmatrix}$$