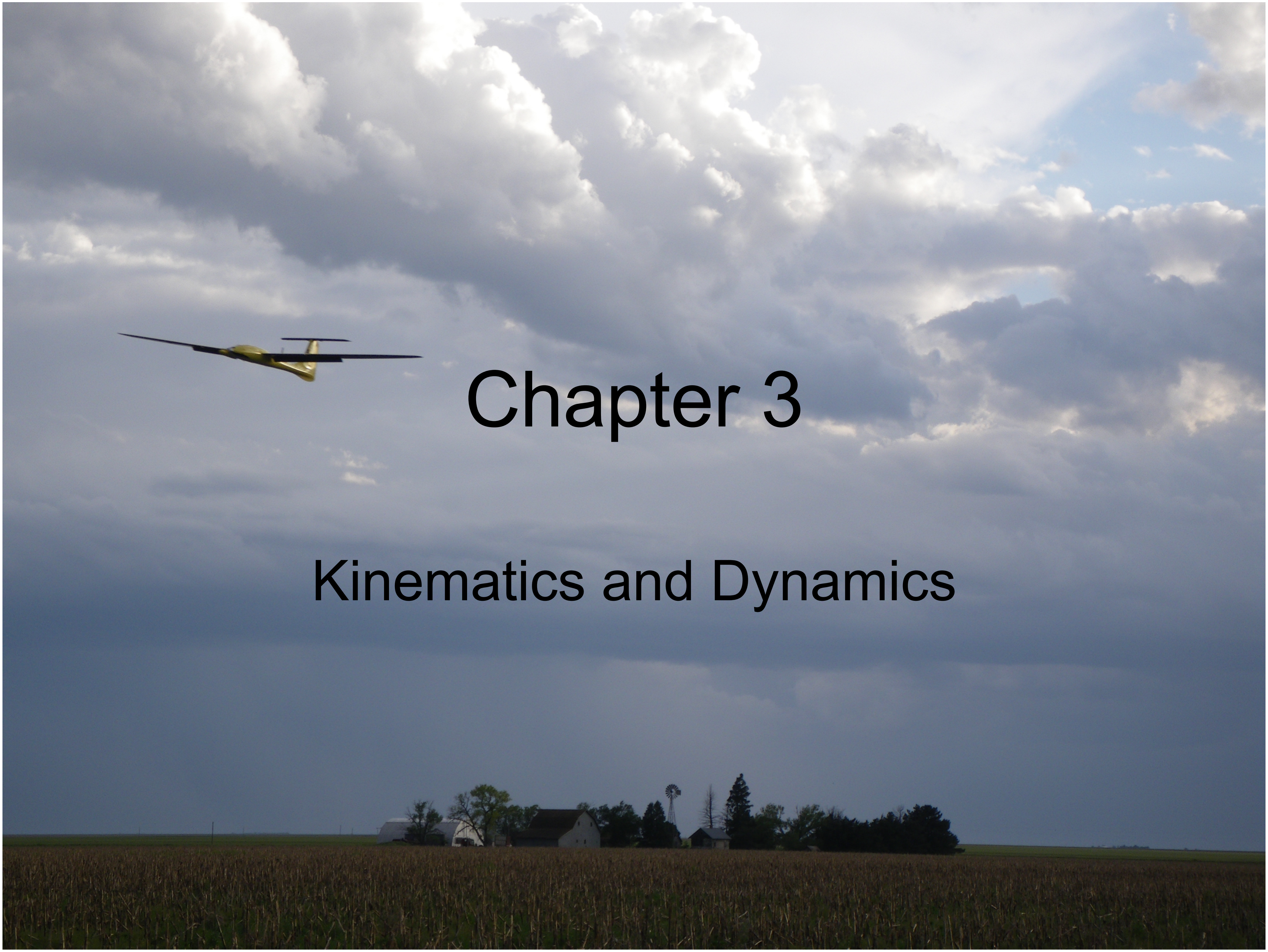
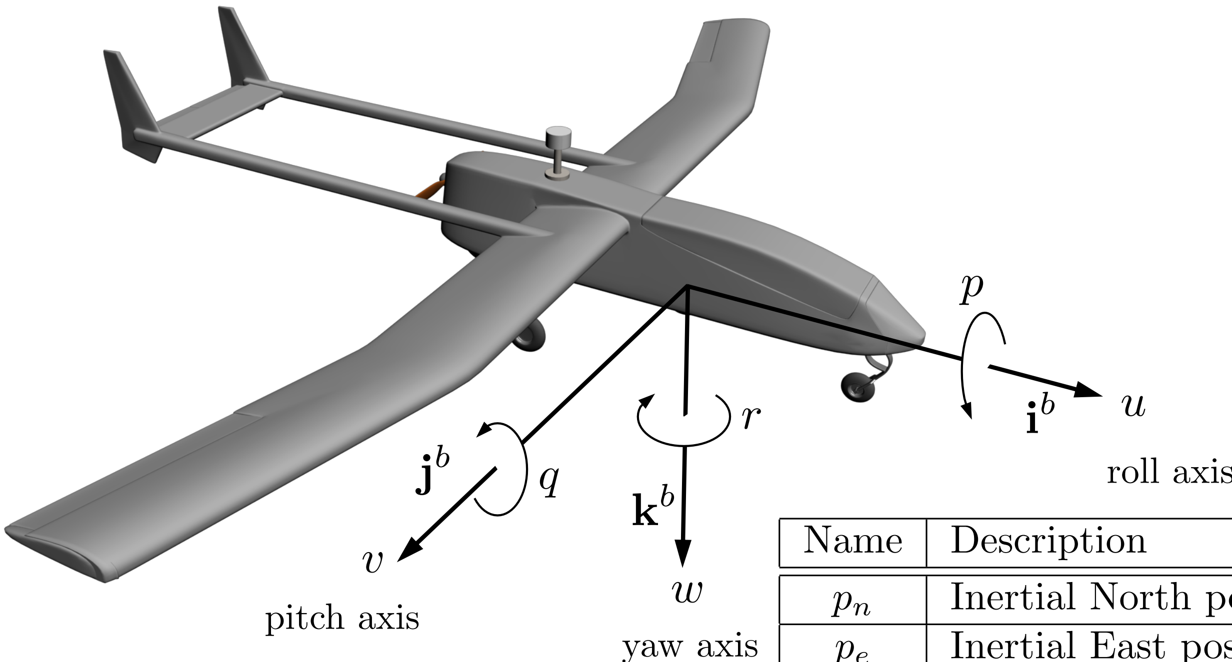


Chapter 3

Kinematics and Dynamics

Aircraft State Variables



Name	Description
p_n	Inertial North position of MAV expressed along $\mathbf{i}^i$ in $\mathcal{F}^i$.
p_e	Inertial East position of MAV expressed along $\mathbf{j}^i$ in $\mathcal{F}^i$.
p_d	Inertial Down position of MAV expressed along $\mathbf{k}^i$ in $\mathcal{F}^i$.
u	Ground velocity expressed along $\mathbf{i}^b$ in $\mathcal{F}^b$.
v	Ground velocity expressed along $\mathbf{j}^b$ in $\mathcal{F}^b$.
w	Ground velocity expressed along $\mathbf{k}^b$ in $\mathcal{F}^b$.
ϕ	Roll angle defined with respect to $\mathcal{F}^{v2}$.
θ	Pitch angle defined with respect to $\mathcal{F}^{v1}$.
ψ	Heading (yaw) angle defined with respect to $\mathcal{F}^v$.
p	Body angular (roll) rate expressed along $\mathbf{i}^b$ in $\mathcal{F}^b$.
q	Body angular (pitch) rate expressed along $\mathbf{j}^b$ in $\mathcal{F}^b$.
r	Body angular (yaw) rate expressed along $\mathbf{k}^b$ in $\mathcal{F}^b$.

Translational Kinematics

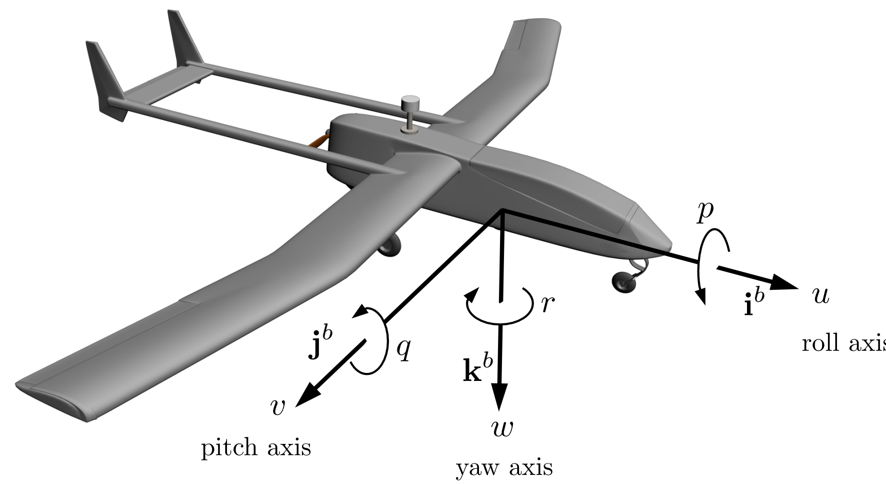
$$
\begin{aligned}
\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{pmatrix} &\stackrel{\triangle}{=} \frac{d}{dt} \begin{pmatrix} p_n \\ p_e \\ p_d \end{pmatrix} = \mathbf{v}^v = R_b^v \mathbf{v}^b \\
&= R_b^v \begin{pmatrix} u \\ v \\ w \end{pmatrix} \\
&= (R_v^b)^\top \begin{pmatrix} u \\ v \\ w \end{pmatrix} \\
&= \begin{pmatrix} \mathrm{c}_\theta \mathrm{c}_\psi & \mathrm{s}_\phi \mathrm{s}_\theta \mathrm{c}_\psi - \mathrm{c}_\phi \mathrm{s}_\psi & \mathrm{c}_\phi \mathrm{s}_\theta \mathrm{c}_\psi + \mathrm{s}_\phi \mathrm{s}_\psi \\ \mathrm{c}_\theta \mathrm{s}_\psi & \mathrm{s}_\phi \mathrm{s}_\theta \mathrm{s}_\psi + \mathrm{c}_\phi \mathrm{c}_\psi & \mathrm{c}_\phi \mathrm{s}_\theta \mathrm{s}_\psi - \mathrm{s}_\phi \mathrm{c}_\psi \\ -\mathrm{s}_\theta & \mathrm{s}_\phi \mathrm{c}_\theta & \mathrm{c}_\phi \mathrm{c}_\theta \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}
\end{aligned}
$$

Rotational Kinematics

$$\begin{pmatrix} p \\ q \\ r \end{pmatrix} = \underbrace{\begin{pmatrix} \dot\phi \\ 0 \\ 0 \end{pmatrix}}_{\dot\phi \text{ is defined in } \mathcal{F}^b} + \underbrace{R_{v2}^b(\phi)\begin{pmatrix} 0 \\ \dot\theta \\ 0 \end{pmatrix}}_{\dot\theta \text{ is defined in } \mathcal{F}^{v2}} + \underbrace{R_{v2}^b(\phi) R_{v1}^{v2}(\theta)\begin{pmatrix} 0 \\ 0 \\ \dot\psi \end{pmatrix}}_{\dot\psi \text{ is defined in } \mathcal{F}^{v1}}$$

$$= \begin{pmatrix} \dot\phi \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} 0 \\ \dot\theta \\ 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ \dot\psi \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{pmatrix} \begin{pmatrix} \dot\phi \\ \dot\theta \\ \dot\psi \end{pmatrix}$$



Inverting gives:

$$\begin{pmatrix} \dot\phi \\ \dot\theta \\ \dot\psi \end{pmatrix} = \begin{pmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix}$$

Kinematic Equations of Motion

Six of the 12 state equations for the MAV come from the kinematic equations relating positions and velocities:

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{pmatrix} = \begin{pmatrix} \mathrm{c}_\theta \mathrm{c}_\psi & \mathrm{s}_\phi \mathrm{s}_\theta \mathrm{c}_\psi - \mathrm{c}_\phi \mathrm{s}_\psi & \mathrm{c}_\phi \mathrm{s}_\theta \mathrm{c}_\psi + \mathrm{s}_\phi \mathrm{s}_\psi \\ \mathrm{c}_\theta \mathrm{s}_\psi & \mathrm{s}_\phi \mathrm{s}_\theta \mathrm{s}_\psi + \mathrm{c}_\phi \mathrm{c}_\psi & \mathrm{c}_\phi \mathrm{s}_\theta \mathrm{s}_\psi - \mathrm{s}_\phi \mathrm{c}_\psi \\ -\mathrm{s}_\theta & \mathrm{s}_\phi \mathrm{c}_\theta & \mathrm{c}_\phi \mathrm{c}_\theta \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}$$

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \sin\phi \tan\theta & \cos\phi \tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi \sec\theta & \cos\phi \sec\theta \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix}$$

The remaining six equations will come from applying Newton's 2nd law to the translational and rotational motion of the aircraft.

Differentiation of a Vector

Define the vector $\mathbf{p}$ in terms of the axes in the body frame:

$$\mathbf{p} = p_x \mathbf{i}^b + p_y \mathbf{j}^b + p_z \mathbf{k}^b.$$

Differentiation with respect to the inertial frame gives

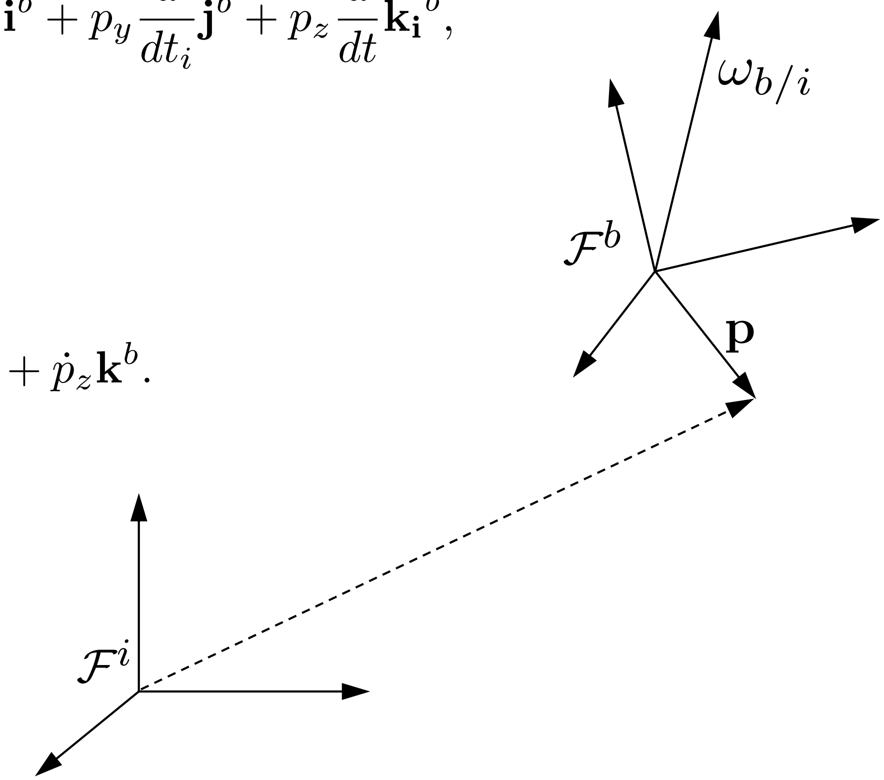
$$\frac{d}{dt_i}\mathbf{p} = \dot{p}_x \mathbf{i}^b + \dot{p}_y \mathbf{j}^b + \dot{p}_z \mathbf{k}^b + p_x \frac{d}{dt_i}\mathbf{i}^b + p_y \frac{d}{dt_i}\mathbf{j}^b + p_z \frac{d}{dt}\mathbf{k_i}^b,$$

where

$$\dot{p}_*^b = \frac{dp_*^b}{dt}.$$

Let

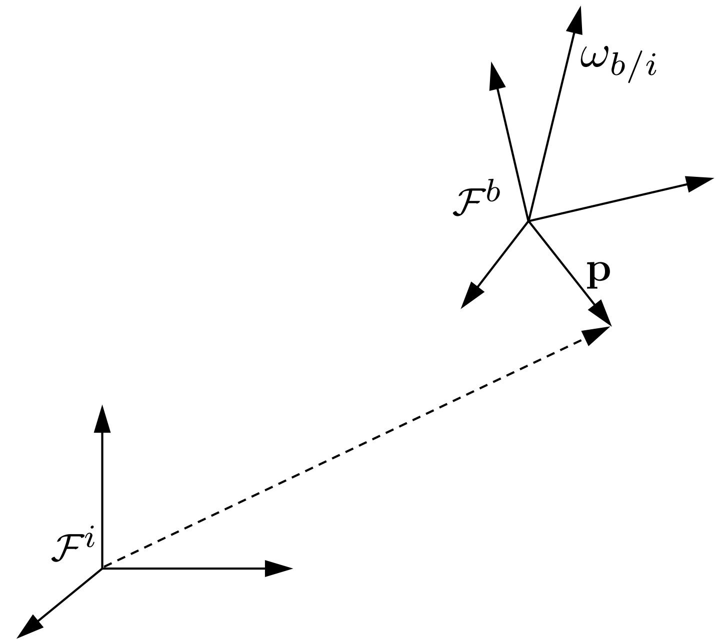
$$\frac{d}{dt_b}\mathbf{p} = \dot{p}_x \mathbf{i}^b + \dot{p}_y \mathbf{j}^b + \dot{p}_z \mathbf{k}^b.$$



Differentiation of a Vector

Recall from physics that for any vector $\mathbf{v}$ fixed in $\mathcal{F}^b$ we have

$$\frac{d}{dt_i}\mathbf{v} = \boldsymbol{\omega}_{b/i} \times \mathbf{v}.$$



Therefore

$$\begin{aligned}
\frac{d}{dt_i}\mathbf{i}^b &= \boldsymbol{\omega}_{b/i} \times \mathbf{i}^b \\
\frac{d}{dt_i}\mathbf{j}^b &= \boldsymbol{\omega}_{b/i} \times \mathbf{j}^b \\
\frac{d}{dt_i}\mathbf{k}^b &= \boldsymbol{\omega}_{b/i} \times \mathbf{k}^b.
\end{aligned}$$

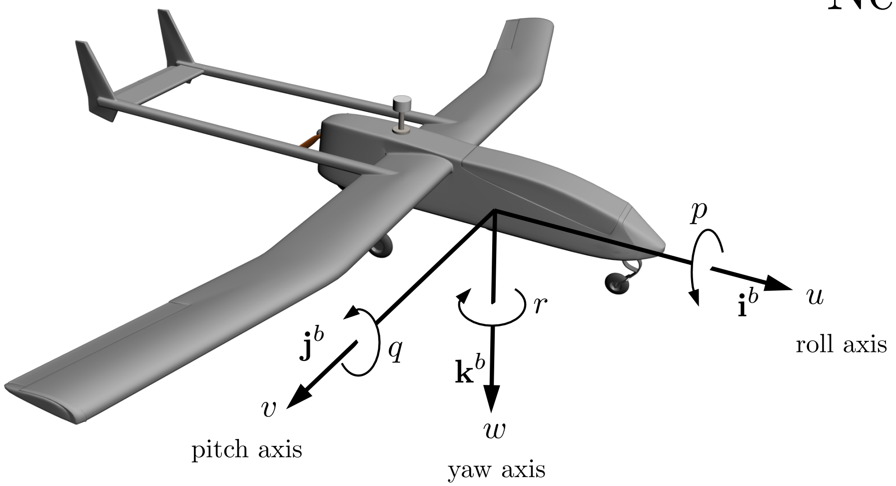
Therefore

$$\begin{aligned}
p_x\frac{d}{dt_i}\mathbf{i}^b + p_y\frac{d}{dt_i}\mathbf{j}^b + p_z\frac{d}{dt_i}\mathbf{k}^b &= p_x\boldsymbol{\omega}_{b/i} \times \mathbf{i}^b + p_y\boldsymbol{\omega}_{b/i} \times \mathbf{j}^b + p_z\boldsymbol{\omega}_{b/i} \times \mathbf{k}^b \\
&= \boldsymbol{\omega}_{b/i} \times \left(p_x\mathbf{i}^b + p_y\mathbf{j}^b + p_z\mathbf{k}^b\right) \\
&= \boldsymbol{\omega}_{b/i} \times \mathbf{p},
\end{aligned}$$

resulting in

$$\frac{d}{dt_i}\mathbf{p} = \frac{d}{dt_b}\mathbf{p} + \boldsymbol{\omega}_{b/i} \times \mathbf{p}.$$

Translational Dynamics



Newton's 2nd Law:

$$\mathsf{m}\frac{d\mathbf{V}_g}{dt_i} = \mathbf{f}$$

- $\mathbf{f}$ is the sum of all external forces
- m is the mass of the aircraft
- Time derivative taken wrt inertial frame

Using the expression

$$\frac{d\mathbf{V}_g}{dt_i} = \frac{d\mathbf{V}_g}{dt_b} + \boldsymbol{\omega}_{b/i} \times \mathbf{V}_g$$

gives

$$\mathsf{m}\left(\frac{d\mathbf{V}_g}{dt_b} + \boldsymbol{\omega}_{b/i} \times \mathbf{V}_g\right) = \mathbf{f}$$

Translational Dynamics

Expressing $\mathsf{m}\left(\frac{d\mathbf{V}_g}{dt_b} + \boldsymbol{\omega}_{b/i} \times \mathbf{V}_g\right) = \mathbf{f}$ in the body frame gives

$$\mathsf{m}\left(\frac{d\mathbf{V}_g^b}{dt_b} + \boldsymbol{\omega}_{b/i}^b \times \mathbf{V}_g^b\right) = \mathbf{f}^b,$$

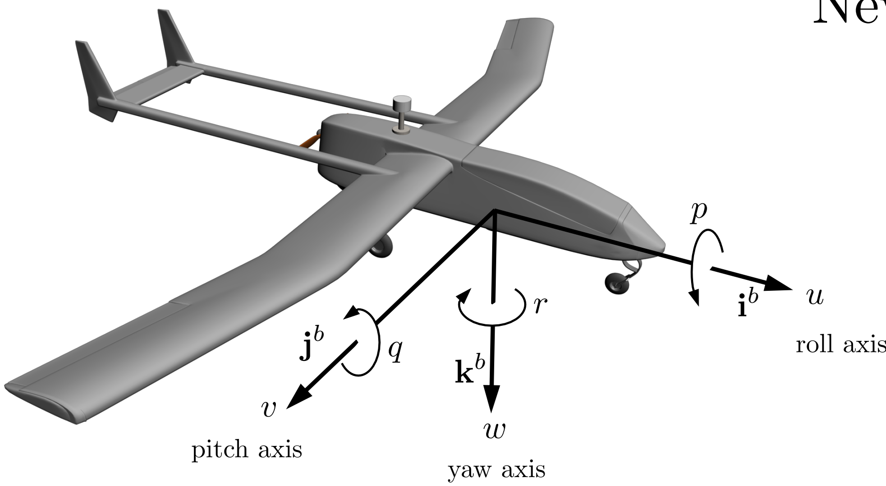
where

$$\mathbf{V}_g^b = \begin{pmatrix} u \\ v \\ w \end{pmatrix} \qquad \boldsymbol{\omega}_{b/i}^b = \begin{pmatrix} p \\ q \\ r \end{pmatrix} \qquad \boldsymbol{f}^b = \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix}.$$

Since $\frac{d\mathbf{V}_g^b}{dt_b} = \begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix}$ we have that

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} = -\begin{pmatrix} p \\ q \\ r \end{pmatrix} \times \begin{pmatrix} u \\ v \\ w \end{pmatrix} + \frac{1}{\mathsf{m}}\begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} = \begin{pmatrix} rv - qw \\ pw - ru \\ qu - pv \end{pmatrix} + \frac{1}{\mathsf{m}}\begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix}$$

Rotational Dynamics



Newton's 2nd Law:

$$\frac{d\mathbf{h}}{dt_i} = \mathbf{m}$$

- $\mathbf{h}$ is the angular momentum vector
- $\mathbf{m}$ is the sum of all external moments
- Time derivative taken wrt inertial frame

Therefore we have

$$\frac{d\mathbf{h}}{dt_i} = \frac{d\mathbf{h}}{dt_b} + \boldsymbol{\omega}_{b/i} \times \mathbf{h} = \mathbf{m}.$$

Expressing in the body frame gives

$$\frac{d\mathbf{h}^b}{dt_b} + \boldsymbol{\omega}^b_{b/i} \times \mathbf{h}^b = \mathbf{m}^b.$$

Rotational Dynamics

For a rigid body, angular momentum is defined as the product of the inertia matrix and the angular velocity vector:

$$\mathbf{h}^b \triangleq \mathbf{J}\boldsymbol{\omega}^b_{b/i}$$

where

$$\begin{aligned}
\mathbf{J} &= \begin{pmatrix} \int (y^2 + z^2)\, d\mathsf{m} & -\int xy\, d\mathsf{m} & -\int xz\, d\mathsf{m} \\ -\int xy\, d\mathsf{m} & \int (x^2 + z^2)\, d\mathsf{m} & -\int yz\, d\mathsf{m} \\ -\int xz\, d\mathsf{m} & -\int yz\, d\mathsf{m} & \int (x^2 + y^2)\, d\mathsf{m} \end{pmatrix} \\
&\triangleq \begin{pmatrix} J_x & -J_{xy} & -J_{xz} \\ -J_{xy} & J_y & -J_{yz} \\ -J_{xz} & -J_{yz} & J_z \end{pmatrix}
\end{aligned}$$

Diagonal elements are called moments of inertia. Off-diagonal elements are called products of inertia.

$\mathbf{J}$ determined from mass properties in CAD program or measured experimentally using a bifilar pendulum.

Rotational Dynamics

$$\frac{d\mathbf{h}^b}{dt_b} + \boldsymbol{\omega}_{b/i}^b \times \mathbf{h}^b = \mathbf{m}^b$$

Because $\mathbf{J}$ is unchanging in the body frame, $\frac{d\mathbf{J}}{dt_b} = 0$ and

$$\mathbf{J}\frac{d\boldsymbol{\omega}_{b/i}^b}{dt_b} + \boldsymbol{\omega}_{b/i}^b \times \left(\mathbf{J}\boldsymbol{\omega}_{b/i}^b\right) = \mathbf{m}^b$$

Rearranging we get

$$\dot{\boldsymbol{\omega}}_{b/i}^b = \mathbf{J}^{-1}\left[-\boldsymbol{\omega}_{b/i}^b \times \left(\mathbf{J}\boldsymbol{\omega}_{b/i}^b\right) + \mathbf{m}^b\right]$$

where

$$\dot{\boldsymbol{\omega}}_{b/i}^b = \frac{d\boldsymbol{\omega}_{b/i}^b}{dt_b} = \begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix}$$

Rotational Dynamics

If the aircraft is symmetric about the $\mathbf{i}^b$-$\mathbf{k}^b$ plane, then $J_{xy} = J_{yz} = 0$ and

$$\mathbf{J} = \begin{pmatrix} J_x & 0 & -J_{xz} \\ 0 & J_y & 0 \\ -J_{xz} & 0 & J_z \end{pmatrix}$$

This symmetry assumption helps to simplify the analysis. The inverse of $\mathbf{J}$ becomes

$$\begin{aligned} \mathbf{J}^{-1} &= \frac{\mathrm{adj}(\mathbf{J})}{\det(\mathbf{J})} = \frac{\begin{pmatrix} J_y J_z & 0 & J_y J_{xz} \\ 0 & J_x J_z - J_{xz}^2 & 0 \\ J_{xz} J_y & 0 & J_x J_y \end{pmatrix}}{J_x J_y J_z - J_{xz}^2 J_y} \\ &= \begin{pmatrix} \frac{J_z}{\Gamma} & 0 & \frac{J_{xz}}{\Gamma} \\ 0 & \frac{1}{J_y} & 0 \\ \frac{J_{xz}}{\Gamma} & 0 & \frac{J_x}{\Gamma} \end{pmatrix} \end{aligned}$$

where

$$\Gamma \triangleq J_x J_z - J_{xz}^2$$

Rotational Dynamics

Define

$$\mathbf{m}^b \triangleq \begin{pmatrix} l \\ m \\ n \end{pmatrix}$$

Then

$$\dot{\boldsymbol{\omega}}^b_{b/i} = \mathbf{J}^{-1}\left[-\boldsymbol{\omega}^b_{b/i} \times \left(\mathbf{J}\boldsymbol{\omega}^b_{b/i}\right) + \mathbf{m}^b\right]$$

can be expressed as

$$\begin{aligned}
\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} &= \begin{pmatrix} \frac{J_z}{\Gamma} & 0 & \frac{J_{xz}}{\Gamma} \\ 0 & \frac{1}{J_y} & 0 \\ \frac{J_{xz}}{\Gamma} & 0 & \frac{J_x}{\Gamma} \end{pmatrix} \left[\begin{pmatrix} 0 & r & -q \\ -r & 0 & p \\ q & -p & 0 \end{pmatrix} \begin{pmatrix} J_x & 0 & -J_{xz} \\ 0 & J_y & 0 \\ -J_{xz} & 0 & J_z \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} + \begin{pmatrix} l \\ m \\ n \end{pmatrix} \right] \\
&= \begin{pmatrix} \frac{J_z}{\Gamma} & 0 & \frac{J_{xz}}{\Gamma} \\ 0 & \frac{1}{J_y} & 0 \\ \frac{J_{xz}}{\Gamma} & 0 & \frac{J_x}{\Gamma} \end{pmatrix} \left[\begin{pmatrix} J_{xz}pq + (J_y - J_z)qr \\ J_{xz}(r^2 - p^2) + (J_z - J_x)pr \\ (J_x - J_y)pq - J_{xz}qr \end{pmatrix} + \begin{pmatrix} l \\ m \\ n \end{pmatrix} \right] \\
&= \begin{pmatrix} \Gamma_1 pq - \Gamma_2 qr + \Gamma_3 l + \Gamma_4 n \\ \Gamma_5 pr - \Gamma_6 (p^2 - r^2) + \frac{1}{J_y} m \\ \Gamma_7 pq - \Gamma_1 qr + \Gamma_4 l + \Gamma_8 n \end{pmatrix}
\end{aligned}$$

where Γ's are functions of moments and products of inertia.

Equation of Motion Summary

The equations of motion are a system of 12 first order ODE's:

$$\begin{pmatrix}\dot{p}_n\\ \dot{p}_e\\ \dot{p}_d\end{pmatrix} = \begin{pmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{pmatrix} \begin{pmatrix} u\\ v\\ w\end{pmatrix}$$

$$\begin{pmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{pmatrix} = \begin{pmatrix} rv - qw\\ pw - ru\\ qu - pv\end{pmatrix} + \frac{1}{\mathrm{m}}\begin{pmatrix} f_x\\ f_y\\ f_z\end{pmatrix},$$

$$\begin{pmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{pmatrix} = \begin{pmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta}\end{pmatrix}\begin{pmatrix} p\\ q\\ r\end{pmatrix}$$

$$\begin{pmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{pmatrix} = \begin{pmatrix} \Gamma_1 pq - \Gamma_2 qr \\ \Gamma_5 pr - \Gamma_6 (p^2 - r^2) \\ \Gamma_7 pq - \Gamma_1 qr \end{pmatrix} + \begin{pmatrix} \Gamma_3 l + \Gamma_4 n \\ \frac{1}{J_y} m \\ \Gamma_4 l + \Gamma_8 n\end{pmatrix}$$

where

$$\Gamma_1 = \frac{J_{xz}(J_x - J_y + J_z)}{\Gamma}, \qquad \Gamma_2 = \frac{J_z(J_z - J_y) + J_{xz}^2}{\Gamma}, \qquad \Gamma_3 = \frac{J_z}{\Gamma},$$

$$\Gamma_4 = \frac{J_{xz}}{\Gamma}, \qquad \Gamma_5 = \frac{J_z - J_x}{J_y}, \qquad \Gamma_6 = \frac{J_{xz}}{J_y},$$

$$\Gamma_7 = \frac{(J_x - J_y)J_x + J_{xz}^2}{\Gamma}, \qquad \Gamma_8 = \frac{J_x}{\Gamma}, \qquad \Gamma = J_x J_z - J_{xz}^2.$$

Quaternions

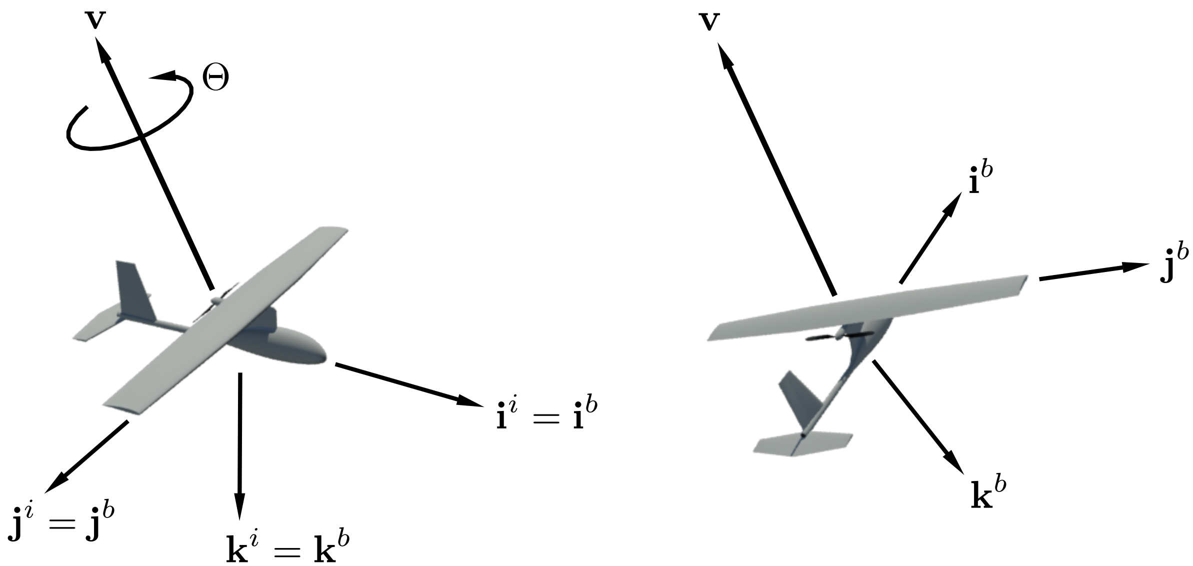
The attitude of a rigid body can be represented by a unit quaternion, which is a 4-vector

$$e = \begin{pmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{pmatrix},$$

where e_0, e_1, e_2, and e_3 are scalars, and where $\|e\| = 1$.

- e_0 is called the scaler part of the quaternion.
- $(e_1, e_2, e_3)^\top$ is called the vector part of the quaternion.

Quaternions



For a rotation of Θ about unit vector $\mathbf{v}$ the scalar part is defined as

$$e_0 = \cos\left(\frac{\Theta}{2}\right).$$

and the vector part is defined as

$$\mathbf{v}\sin\left(\frac{\Theta}{2}\right) = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix}.$$

Quaternions

Conversion Between Euler Angles and Quaternions

$$
\begin{aligned}
\phi &= \operatorname{atan2}\left(2(e_0e_1 + e_2e_3), (e_0^2 + e_3^2 - e_1^2 - e_2^2)\right) \\
\theta &= \operatorname{asin}\left(2(e_0e_2 - e_1e_3)\right) \\
\psi &= \operatorname{atan2}\left(2(e_0e_3 + e_1e_2), (e_0^2 + e_1^2 - e_2^2 - e_3^2)\right),
\end{aligned}
$$

From the yaw, pitch, and roll Euler angles (ψ, ϕ, θ), the corresponding quaternion elements are

$$
\begin{aligned}
e_0 &= \cos\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2} \\
e_1 &= \cos\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} - \sin\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} \\
e_2 &= \cos\frac{\psi}{2}\sin\frac{\theta}{2}\cos\frac{\phi}{2} + \sin\frac{\psi}{2}\cos\frac{\theta}{2}\sin\frac{\phi}{2} \\
e_3 &= \sin\frac{\psi}{2}\cos\frac{\theta}{2}\cos\frac{\phi}{2} - \cos\frac{\psi}{2}\sin\frac{\theta}{2}\sin\frac{\phi}{2}.
\end{aligned}
$$

Quaternions

Conversion Between Quaternion and Rotation Matrix If the quaternion $\mathbf{e}_b^i = (e_0, e_1, e_2, e_3)^\top$ represents a rotation from the body to the inertial frame, then the corresponding rotation matrix is

$$R_b^i = \begin{pmatrix} e_1^2 + e_0^2 - e_2^2 - e_3^2 & 2(e_1 e_2 - e_3 e_0) & 2(e_1 e_3 + e_2 e_0) \\ 2(e_1 e_2 + e_3 e_0) & e_2^2 + e_0^2 - e_1^2 - e_3^2 & 2(e_2 e_3 - e_1 e_0) \\ 2(e_1 e_3 - e_2 e_0) & 2(e_2 e_3 + e_1 e_0) & e_3^2 + e_0^2 - e_1^2 - e_2^2 \end{pmatrix}.$$

Equation of Motion Summary

The equations of motion are a system of 12 first order ODE's:

$$\begin{pmatrix}\dot{p}_n\\ \dot{p}_e\\ \dot{p}_d\end{pmatrix} = \begin{pmatrix}c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi\\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi\\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta\end{pmatrix}\begin{pmatrix}u\\ v\\ w\end{pmatrix}$$

$$\begin{pmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{pmatrix} = \begin{pmatrix}rv - qw\\ pw - ru\\ qu - pv\end{pmatrix} + \frac{1}{\mathsf{m}}\begin{pmatrix}f_x\\ f_y\\ f_z\end{pmatrix},$$

$$\begin{pmatrix}\dot{e}_0\\ \dot{e}_1\\ \dot{e}_2\\ \dot{e}_3\end{pmatrix} = \frac{1}{2}\begin{pmatrix}0 & -p & -q & -r\\ p & 0 & r & -q\\ q & -r & 0 & p\\ r & q & -p & 0\end{pmatrix}\begin{pmatrix}e_0\\ e_1\\ e_2\\ e_3\end{pmatrix}$$

$$\begin{pmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{pmatrix} = \begin{pmatrix}\Gamma_1 pq - \Gamma_2 qr\\ \Gamma_5 pr - \Gamma_6(p^2 - r^2)\\ \Gamma_7 pq - \Gamma_1 qr\end{pmatrix} + \begin{pmatrix}\Gamma_3 l + \Gamma_4 n\\ \frac{1}{J_y} m\\ \Gamma_4 l + \Gamma_8 n\end{pmatrix}$$

where

$$\Gamma_1 = \frac{J_{xz}(J_x - J_y + J_z)}{\Gamma}, \qquad \Gamma_2 = \frac{J_z(J_z - J_y) + J_{xz}^2}{\Gamma}, \qquad \Gamma_3 = \frac{J_z}{\Gamma},$$

$$\Gamma_4 = \frac{J_{xz}}{\Gamma}, \qquad \Gamma_5 = \frac{J_z - J_x}{J_y}, \qquad \Gamma_6 = \frac{J_{xz}}{J_y},$$

$$\Gamma_7 = \frac{(J_x - J_y)J_x + J_{xz}^2}{\Gamma}, \qquad \Gamma_8 = \frac{J_x}{\Gamma}, \qquad \Gamma = J_x J_z - J_{xz}^2.$$

Quaternions

In Python, e needs to be normalized after applying the RK4 update step to ensure that $\|e\| = 1$.

In Simulink, we can ensure that $\|e\| \approx 1$ by appending the quaternion kinematics as

$$\begin{pmatrix} \dot{e}_0 \\ \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{pmatrix} \begin{pmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{pmatrix} - \lambda \frac{\partial J}{\partial e}$$

$$= \frac{1}{2} \begin{pmatrix} \lambda(1 - \|e\|^2) & -p & -q & -r \\ p & \lambda(1 - \|e\|^2) & r & -q \\ q & -r & \lambda(1 - \|e\|^2) & p \\ r & q & -p & \lambda(1 - \|e\|^2) \end{pmatrix} \begin{pmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{pmatrix},$$

where $J = \frac{1}{8}(1 - \|e\|^2)^2$ and where $\lambda > 0$. The second term forces $\|e\| \to 1$. In our experience, a value of $\lambda = 1000$ seems to work well, but a stiff solver like ODE15s must be used.

Runge-Kutta Integration

The obective is the numerically solve the differential equation

$$\frac{dx}{dt}(t) = f(x(t), u(t)), \quad x(t_0) = x_0.$$

Integrating both sides gives

$$x(t) = x(t_0) + \int_{t_0}^{t} f(x(\tau), u(\tau))\, d\tau.$$

Note: $x(t)$ appears on both sides of the equation!! Therefore, approximation is required.

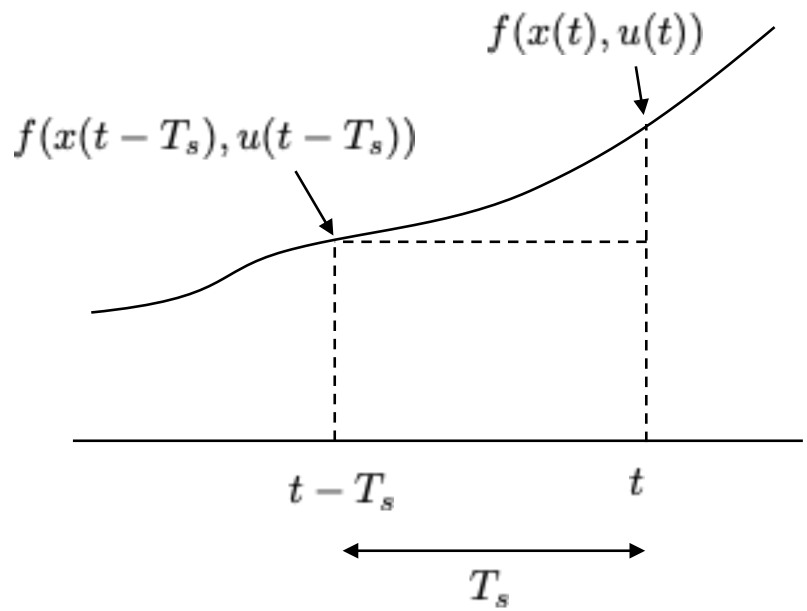
Re-write solution integral as

$$\begin{aligned} x(t) &= x(t_0) + \int_{t_0}^{t-T_s} f(x(\tau), u(\tau))\, d\tau + \int_{t-T_s}^{t} f(x(\tau), u(\tau))\, d\tau \\ &= x(t - T_s) + \int_{t-T_s}^{t} f(x(\tau), x(\tau))\, d\tau. \end{aligned}$$

RK1 Algorithm

The integral on the previous page can be approximated as

$$\int_{t-T_s}^{t} f(x(\tau), u(\tau))\, d\tau \approx T_s f(x(t-T_s), u(t-T_s)).$$



Resulting in

$$x_k = x_{k-1} + T_s f(x_{k-1}, u_{k-1}), \quad x_0 = x(t_0).$$

This is the Runge-Kutta first order method, or RK1.

RK1 Implementation

```
classdef system_rk1 < handle
    properties
        Ts
        state
    end
    methods
        %---constructor------------------------
        function self = system_rk1(Ts)
            self.Ts = Ts;  % sample rate
            self.state = [0; 0];  % initial state
        end
        %---------------------------
        function y = update(self, u)
            % external method that takes u
            % as input and returns y
            self.rk1_step(u);
            y = self.h();
        end
```

```
        function self = rk1_step(self, u)
            F1 = self.f(self.state, u);
            self.state = self.state + self.Ts * F1;
        end
        function xdot = f(self, state, u)
            % Return xdot = f(x,u),
            xdot = [0; 0];
        end
        function y = h(self)
            % return y = h(x)
            y = 0;
        end
    end
end
```

RK2 Algorithm

To improve accuracy, the integral can be approximated using the trapezoidal rule

$$\int_a^b f(\xi)d\xi \approx (b-a)\left(\frac{f(a)+f(b)}{2}\right).$$

Accordingly,

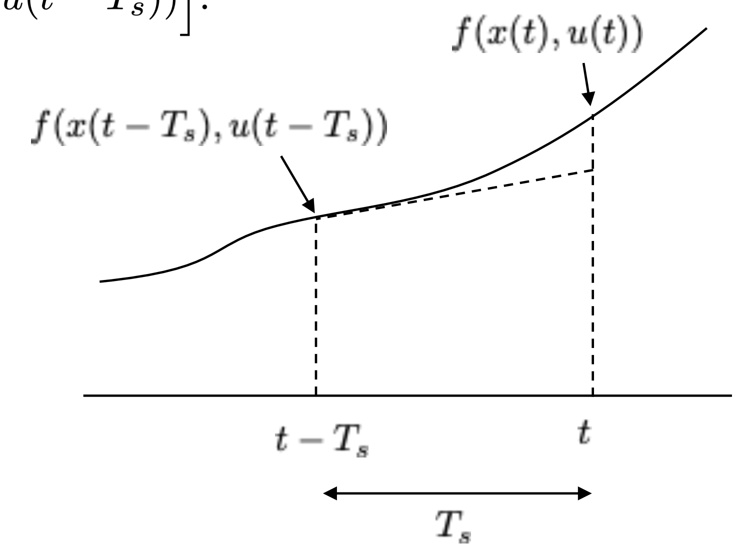
$$\int_{t-T_s}^{t} f(x(\tau),u(\tau))d\tau \approx \frac{T_s}{2}\big[f(x(t-T_s),u(t-T_s)) + f(x(t),u(t))\big].$$

$x(t)$ is unknown so use RK1 to approximate it as

$$\int_{t-T_s}^{t} f(x(\tau),u(\tau))d\tau \approx \frac{T_s}{2}\Big[f(x(t-T_s),u(t-T_s)) \\ + f(x(t-T_s) + T_s f(x(t-T_s),u(t-T_s)),u(t-T_s))\Big].$$

The numerical integration routine can be written as

$$\begin{aligned} x_0 &= x(t_0) \\ X_1 &= f(x_{k-1},u_{k-1}) \\ X_2 &= f(x_{k-1}+T_s X_1,u_{k-1}) \\ x_k &= x_{k-1} + \frac{T_s}{2}\left(X_1+X_2\right). \end{aligned}$$



This is the Runge-Kutta second order method or RK2.

RK2 Implementation

```
function self = rk2_step(self, u)
    F1 = self.f(self.state, u);
    F2 = self.f(self.state + self.Ts/2*F1, u);
    self.state = self.state + self.Ts/2 * (F1 + F2);
end
```

RK4 Algorithm

An even better approximation is obtained using Simpson's Rule (area under a parabola):

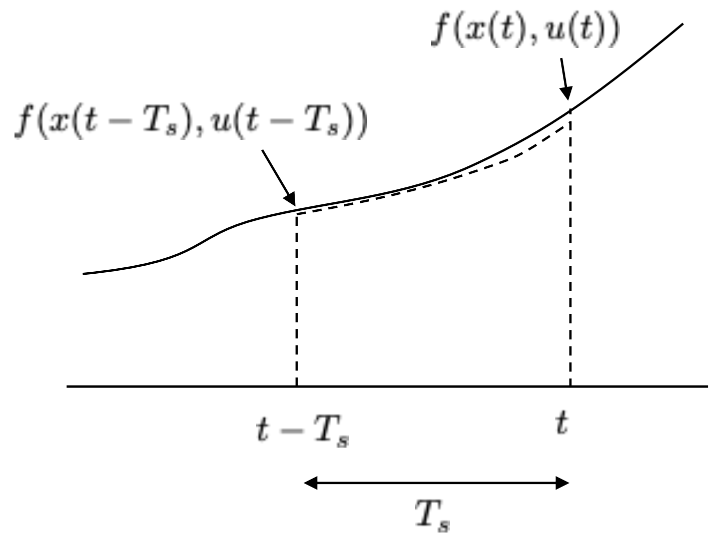
$$\int_a^b f(\xi)d\xi \approx \left(\frac{b-a}{6}\right)\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right),$$

The standard strategy is to write the approximation as

$$\int_a^b f(\xi)\xi \approx \left(\frac{b-a}{6}\right)\left(f(a) + 2f\left(\frac{a+b}{2}\right) + 2f\left(\frac{a+b}{2}\right) + f(b)\right),$$

Define

$$\begin{aligned}
X_1 &= f(a) \\
X_2 &= f(a + \frac{T_s}{2}X_1) \approx f\left(\frac{a+b}{2}\right) \\
X_3 &= f(a + \frac{T_s}{2}X_2) \approx f\left(\frac{a+b}{2}\right) \\
X_4 &= f(a + T_s X_3) \approx f(b),
\end{aligned}$$



RK4 Algorithm

The numerical integration routine can be written as

$$\begin{aligned}
x_0 &= x(t_0) \\
X_1 &= f(x_{k-1}, u_{k-1}) \\
X_2 &= f(x_{k-1} + \frac{T_s}{2} X_1, u_{k-1}) \\
X_3 &= f(x_{k-1} + \frac{T_s}{2} X_2, u_{k-1}) \\
X_4 &= f(x_{k-1} + T_s X_3, u_{k-1}) \\
x_k &= x_{k-1} + \frac{T_s}{6} \left(X_1 + 2X_2 + 2X_3 + X_4 \right).
\end{aligned}$$

This is the Runge-Kutta fourth order method or RK4.

RK4 Implementation

```
 function self = rk4_step(self, u)
    F1 = self.f(self.state, u);
    F2 = self.f(self.state + self.Ts/2*F1, u);
    F3 = self.f(self.state + self.Ts/2*F2, u);
    F4 = self.f(self.state + self.Ts*F3, u);
    self.state = self.state + self.Ts/6 * (F1 + 2*F2 + 2*F3 + F4);
end
```

Project

1. Implement the MAV equations of motion given in Equations (3.14) through (3.17). Assume that the inputs to the block are the forces and moments applied to the MAV in the body frame. Changeable parameters should include the mass, the moments and products of inertia, and the initial conditions for each state. Use the parameters given in Appendix E.

2. Verify that the equations of motion are correct by individually setting the forces and moments along each axis to a nonzero value and convincing yourself that the motion is appropriate.

3. Since J_{xz} is non-zero, there is gyroscopic coupling between roll and yaw. To test your simulation, set J_{xz} to zero and place nonzero moments on l and n and verify that there is no coupling between the roll and yaw axes. Verify that when J_{xz} is not zero, there is coupling between the roll and yaw axes.