Quadcopter Differential Equations of Motion

Jonathan Dorsey
https://github.com/JonnyD1117/

I. Introduction

These are my notes on the derivations and formulation for the 6-DOF Quadcopter rigid body dynamics equations of motion, making notes on all salient aspects concerning details from derivation to implementation in C++.

II. Coordinate Systems and Frames of Reference

A. Inertial Frame of Reference

The inertial reference frame is defined as a coordinate system that is not accelerating or rotating. Typically, the earth is chosen to be the inertial reference frame of the vehicle.

B. Euler Angles

Euler Angles are a series of sequential rotations for a given coordinate system. This formulation is commonly used to convert between different coordinate systems.

The most common symbols for **Euler Angles** in aircraft dynamics are θ (pitch), ψ(yaw), ϕ(rolls)

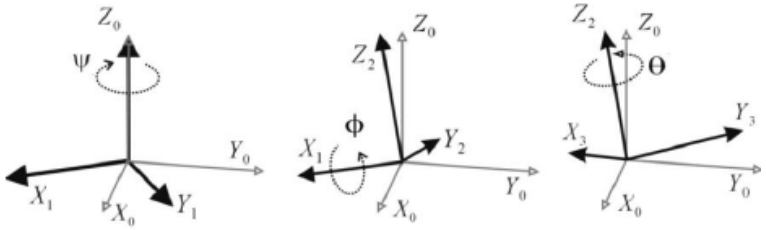


Fig. 1: Sequential Rotations of Euler Angles

1) Euler Rotation Matrix: By using trigonometry, between each pair of coordinates above it is possible to show that rotations between frames can be expressed as the product of elementary rotation matrices obtained from each rotation about intermediate axes.

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi) & c(\phi) \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ 0 & 1 & 0 \\ -s(\theta) & 0 & (\theta) \end{bmatrix} \quad R_z(\psi) = \begin{bmatrix} c(\psi) & -s(\psi) & 0 \\ s(\psi) & c(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

These elementary rotation matrices can be combined to to produce a single rotation matrix.

$$R_{zyx}(\phi, \theta, \psi)\Big|_B^I = R_x(\phi) \cdot R_y(\theta) \cdot R_z(\psi)$$

$$R_{zyx}(\phi, \theta, \psi)\Big|_B^I = \begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) + s(\theta)s(\psi) \\ c(\theta)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\theta)c(\psi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{bmatrix}$$

It should also be noted that since this rotation matrix is orthogonal, it's inverse can be found by taking the transpose...

$$R^{-1} = R^T$$

2) Euler Rates to Angular Velocity Components (& Vise Versa):

$$T(\phi,\theta,\psi) = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & \frac{s(\phi)}{c(\theta)} & \frac{c(\phi)}{c(\theta)} \end{bmatrix}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{x} \end{bmatrix} = R(\phi,\theta,\psi) \cdot \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad , \quad \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = T(\phi,\theta,\psi) \cdot \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

It should be noted that the body frame angular velocities are not equal to the Euler rates. This is due to the fact that the angular velocity components exist simultaneously as components of the angular velocity vector $\vec{\omega}$. However, Euler angles do not exist at the same time, they constitute a sequence of three consecutive rotations around different axes of rotations.

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R() \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R()R() \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}$$

These transformations convert between the angular velocity and the euler rates (and vise versa)

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s(\theta) \\ 0 & c(\phi) & s(\phi)c(\theta) \\ 0 & -s(\phi) & c(\phi)c(\theta) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & \frac{s(\phi)}{c(\theta)} & \frac{c(\phi)}{c(\theta)} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

3) Problem with Euler Angles: The main drawback of using Euler angles is the presence of gimbal lock where there does not exist a unique mathematical description of the orientation given the angles.

C. Quaternions

The most commonly used alternative to Euler angles is the quaternion formulation.

$$q = q_0 + q_1\hat{i} + q_2\hat{j} + q_3\hat{k}$$
$$q = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T$$

1) Quaternion Properties: The product of two quaternions is defined as the Kronecker product as defined below.

$$p \times q = \begin{bmatrix} p_0q_0 - p_1q_1 - p_2q_2 - p_3q_3 \\ p_0q_1 + p_1q_0 + p_2q_3 - p_3q_2 \\ p_0q_2 - p_1q_3 + p_2q_0 + p_3q_1 \\ p_0q_3 + p_1q_2 - p_2q_1 + p_3q_0 \end{bmatrix}$$

A quaternion can be represented as a $4x4$ matrix given $q = [q_0, q_1, q_2, q_3]$...

$$\begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix}$$

Quaternion multiplication can be cast into a Matrix vector product by using this expansion of the quaternion into a matrix

$$R = q \otimes p = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

Or if desired, can be expressed as matrix multiplication

$$p \times q = Q(p)q = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & -p_3 & p_2 \\ p_2 & p_3 & p_0 & -p_1 \\ p_3 & -p_2 & p_1 & p_0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

The conjugate of a quaternion is simply the negated vector components of the quaternion.

$$q^* = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \end{bmatrix}$$

One of the most important properties that needs to be known for simulations that use quaternions is how to define the derivative of a quaternion so that an update law can be defined.

2) Using the Quaterion:

$$w = q \times \begin{bmatrix} 0 \\ v \end{bmatrix} \times q^*$$

D. Quaternion Rotation Matrix

Quaternions can be used as a drop-in replacement for Euler-angles by the use of a quaternion based rotation matrix.

$$R_i^b(q_i^b) = \begin{bmatrix} {q_0}^2 + {q_1}^2 - {q_2}^2 - {q_3}^2 & 2(q_1q_2) - 2(q_0q_3) & 2(q_1q_3) - 2(q_0q_2) \\ 2(q_1q_2) + 2(q_0q_3) & {q_0}^2 - {q_1}^2 + {q_2}^2 - {q_3}^2 & 2(q_2q_3) - 2(q_0q_1) \\ 2(q_1q_3) - 2(q_0q_3) & 2(q_2q_3) + 2(q_0q_1) & {q_0}^2 - {q_1}^2 - {q_2}^2 + {q_3}^2 \end{bmatrix}$$

However, this usage is computationally inefficient compared to direct conversion using quaternion multiplication directly, which is the preferred method of computing transformations.

E. Derivatives of Quaternions

Like Euler rates relate to Euler Angles, it is necessary to be able to define and compute the derivatives of quaternions as these process allows for us to evolve and align the quaternion in time with the body frame of the quadcopter.

Angular Velocity based Quaternion Derivatives

Direct derivatives can be computed using the angular velocity of the system.

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

Expanded this shows...

$$\begin{bmatrix} \dot{q_0} \\ \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

1) Numerical methods :

F. Numerical Integration of Quaternions

1) Digital Integration of Quaternions:

$$q_{k+1} = q_k + \int_{t_{k+1}}^{t_k} \frac{d}{dt} q(\tau)\, d\tau$$

G. Quaternion Normalization for Numerical Methods

Unlike Euler-Rates, in order for quaternions to retain some of their desireable properties, they must be normalized to 1. Therefore, for either method of computing the quaternion derivative above, numerical errors will build up and begin to diverge the norm of the quaternion from unity. Therefore a robust numerical implementation of the quaternion derivatives requires periodic normalization.

The most simple way of accomplishing this is by brute-force normalizing the quaternion after every time-step in the update law.

$$q_{k+1} \rightarrow \frac{q_{k+1}}{\|q_{k+1}\|}$$

1) Runge-Kutta Integration of Quaternions:

$$f(q,t) = \frac{1}{2}\Omega(\omega(t))q$$

$$q_{k+1} = q_k + \sum_{i=1}^{s} b_i k^{(i)}$$

$$k^{(i)} = f(q^{(i),t_k+c_i\delta t})$$

$$q^{(i)} = q_k + \delta \sum_{j=1}^{i-1} a_{ij} k^{(j)}$$

III. Newtons Equations

According to Newton's Second Law of Motion, the time rate of change of the linear momentum is equal to the sum of external forces on a dynamic system in translation.

$$\sum \vec{F} = \dot{\vec{L}} \quad \text{Where} \quad \vec{L} = m\vec{V}$$

$$\sum \vec{F} = m\dot{\vec{V}} = m\dot{\vec{V}} + \dot{m}\vec{V}$$

When taken in a rotating reference frame...

$$\left.\frac{d(\vec{V})}{dt}\right|_{inertial} = \left.\dot{\vec{V}}\right|_{body} + \omega \times \vec{V}$$

This equation allows us to express the derivatives in terms of the velocity of the body frame and the angular velocity of the body frame with respect to the inertial frame.

$$\sum \vec{F} = m(\dot{\vec{V}} + \omega \times \vec{V}) + \dot{m}\vec{V}$$

If the rate of change of mass is zero or negligible then...

$$\sum \vec{F} = m\dot{V} + \dot{m}\vec{V} = m\vec{a}$$

Leaving the expression

$$\sum \vec{F} = m(\dot{\vec{V}} + \omega \times \vec{V})$$

IV. Eulers Equations

Euler's Equations of motion are the rotational analogue of Newton's Equation of Motion. It can be shown that the summation of torques acting on a dynamics system is equal to the time rate of change of the angular momentum of the system. However, unlike translational dynamics where mass is a scalar valued property, the inertial properties of an object in rotation change with the geometry about how mass is distributed through the system. Therefore we use the inertial tensor $\tilde{I}$ to encapsulate the full 3-dimensional nature of rotational mass.

$$\sum \vec{\tau} = \dot{\vec{H}} \quad \text{Where} \quad \vec{H} = \tilde{I}\vec{\omega}$$

$$\sum \vec{\tau} = \tilde{I}\dot{\vec{\omega}} + \dot{\tilde{I}}\vec{H}$$

As with translation dynamics, when considering the usecase of rotating reference frames the time derivative of the angular velocity is as follows....

$$\left.\frac{d(\vec{\omega})}{dt}\right|_{inertial} = \left.\dot{\vec{\omega}}\right|_{body} + \omega \times \vec{\omega}$$

$$\sum \vec{\tau} = \tilde{I}\left(\dot{\vec{\omega}} + \omega \times \vec{\omega}\right) + \dot{\tilde{I}}\vec{\omega}$$

If the inertial properties of the system are constant with respect to time, we can simplify the expression as follows...

$$\sum \vec{\tau} = \tilde{I}(\dot{\vec{\omega}} + \omega \times \vec{\omega})$$

V. Newton-Euler Equations

By combining both Newton's and Euler's equations, we derive a 6-DOF differential equations of motion of a rigid body dynamic system for coupled translation and rotational dynamics.

$$\sum \vec{F} = m(\dot{\vec{V}} + \omega \times \vec{V})$$

$$\sum \vec{\tau} = \tilde{I}\dot{\vec{\omega}} + \omega \times (\tilde{I}\vec{\omega})$$

Where $\sum F$ is the summation of external forces acting on the center of gravity of the system and likewise, $\sum \tau$ is the summation of external torques acting on the system. From this formulation of the Newton-Euler equations, we can derive the rigid body dynamics of the system by defining the linear and angular velocities and accelerations, as well as defining the forces and torques acting on the system.

These equations can be generalized and simplified to make them easier to implement in simulation.

Inertial Frame Quantities:

$$\vec{P} = \begin{bmatrix} x & y & z \end{bmatrix}^T \quad \vec{V_{ref}} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T$$

Body Frame Quantities:

$$V_{body} = \begin{bmatrix} u & v & w \end{bmatrix}^T \quad , \omega^{body} = \begin{bmatrix} p & q & r \end{bmatrix}^T$$

A. Newton-Euler Simplification

1) Newton Equation Cross Product: It is possible to simplify the cross product contained within Newton's Equation by expanding then reducing the cross product to its vector form.

$$\vec{\omega} \times \vec{V_{body}} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ p & q & r \\ u & v & w \end{vmatrix} = \hat{i} \cdot (qw - rv) - \hat{j} \cdot (pw - ru) + \hat{k} \cdot (pv - qu)$$

Yielding...

$$\vec{\omega} \times \vec{V_{body}} = \begin{bmatrix} qw - rv \\ ru - pw \\ pv - qu \end{bmatrix}$$

2) Euler's Equation Cross Product: As with Newton's Equation, we can simplify by expanding the compact cross product form into a determinant and using the Inertial tensor. It should be noted that in the case of a quadcopter, its symmetrical construction allows for a simplification of the Inertial tensor to a diagonal matrix.

$$\tilde{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \rightarrow \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

$$\tilde{I} \cdot \vec{\omega} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \cdot \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} I_{xx}p \\ I_{yy}q \\ I_{zz}r \end{bmatrix}$$

$$\vec{\omega} \times (\tilde{I} \cdot \vec{\omega}) = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ p & q & r \\ I_{xx}p & I_{yy}q & I_{zz}r \end{vmatrix} = \hat{i} \cdot (qI_{zz}r - rI_{yy}q) - \hat{j} \cdot (pI_{zz}r - rI_{xx}p) + \hat{k} \cdot (pI_{yy}q - qI_{xx}p)$$

$$\vec{\omega} \times (\tilde{I} \cdot \vec{\omega}) = \begin{bmatrix} (I_{zz} - I_{yy})qr \\ (I_{xx} - I_{zz})pr \\ (I_{yy} - I_{xx})pq \end{bmatrix}$$

3) Newton-Euler Simplifications: Now that we have simplified the cross product terms, we can substitute these terms back into the governing equations...

$$\sum \vec{F} = m(\dot{\vec{V}} + \omega \times \vec{V}) \qquad \sum \vec{\tau} = \tilde{I}\dot{\vec{\omega}} + \omega \times (\tilde{I}\vec{\omega})$$

$$\vec{\omega} \times \vec{V_{body}} = \begin{bmatrix} qw - rv \\ ru - pw \\ pv - qu \end{bmatrix} \quad \& \quad \vec{\omega} \times (\tilde{I} \cdot \vec{\omega}) = \begin{bmatrix} (I_{zz} - I_{yy})qr \\ (I_{xx} - I_{zz})pr \\ (I_{yy} - I_{xx})pq \end{bmatrix}$$

Substitute simplified terms...

$$\sum \vec{F} = m\dot{\vec{V}} + m \begin{bmatrix} qw - rv \\ ru - pw \\ pv - qu \end{bmatrix} \quad \rightarrow \quad \sum \vec{F} = m \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + m \begin{bmatrix} qw - rv \\ ru - pw \\ pv - qu \end{bmatrix}$$

$$\sum \vec{\tau} = \tilde{I}\dot{\vec{\omega}} + \begin{bmatrix} (I_{zz} - I_{yy})qr \\ (I_{xx} - I_{zz})pr \\ (I_{yy} - I_{xx})pq \end{bmatrix} \quad \rightarrow \quad \sum \vec{\tau} = \tilde{I} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} (I_{zz} - I_{yy})qr \\ (I_{xx} - I_{zz})pr \\ (I_{yy} - I_{xx})pq \end{bmatrix}$$

These equations more useful in later steps, it is desirable to reform these equations by solving for accelerations.

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \frac{1}{m} \sum \vec{F} - \begin{bmatrix} qw - rv \\ ru - pw \\ pv - qu \end{bmatrix}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \tilde{I}^{-1} \sum \vec{\tau} - \tilde{I}^{-1} \begin{bmatrix} (I_{zz} - I_{yy})qr \\ (I_{xx} - I_{zz})pr \\ (I_{yy} - I_{xx})pq \end{bmatrix}$$

As written Newton's equation is fully simplified with the exception of the extern forces F; however, before we can completely simplify Euler's equation it is necessary to define the following definition.

$$\tilde{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \qquad \tilde{I}^{-1} = \begin{bmatrix} \frac{1}{I_{xx}} & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix}$$

It can be shown that the inverse of a diagonal matrix is simply another diagonal matrix where the elements are the reciprocals of the original matrix. Therefore...

$$\tilde{I}\tilde{I}^{-1} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \frac{1}{I_{xx}} & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix} = I_{3x3}$$

This definition allows us to easily define the inverse of the inertial tensor and to further simplify Euler's equation.

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{1}{I_{xx}} & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix} \sum \vec{\tau} - \begin{bmatrix} \frac{1}{I_{xx}} & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix} \begin{bmatrix} (I_{zz} - I_{yy})qr \\ (I_{xx} - I_{zz})pr \\ (I_{yy} - I_{xx})pq \end{bmatrix}$$

Resulting in...

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{\sum \tau_1}{I_{xx}} \\ \frac{\sum \tau_2}{I_{yy}} \\ \frac{\sum \tau_3}{I_{zz}} \end{bmatrix} - \begin{bmatrix} \frac{(I_{zz} - I_{yy})}{I_{xx}} qr \\ \frac{(I_{xx} - I_{zz})}{I_{yy}} pr \\ \frac{(I_{yy} - I_{xx})}{I_{zz}} pq \end{bmatrix} \qquad \text{where} \quad \sum \vec{\tau} = \begin{bmatrix} \sum \tau_1 \\ \sum \tau_2 \\ \sum \tau_3 \end{bmatrix}$$

VI. Simplified Form - Newton-Euler Equations

After all of this manipulation and simplification, we arrive at the final form of the Newton-Euler equations for rigid body motion in 6 Degrees of Freedom, shown below.

$$\boxed{\begin{bmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{bmatrix} = \frac{1}{m}\sum \vec{F} - \begin{bmatrix}qw - rv\\ ru - pw\\ pv - qu\end{bmatrix}}$$

$$\boxed{\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}\frac{\sum \tau_1}{I_{xx}}\\ \frac{\sum \tau_2}{I_{yy}}\\ \frac{\sum \tau_3}{I_{zz}}\end{bmatrix} - \begin{bmatrix}\frac{(I_{zz}-I_{yy})}{I_{xx}}qr\\ \frac{(I_{xx}-I_{zz})}{I_{yy}}pr\\ \frac{(I_{yy}-I_{xx})}{I_{zz}}pq\end{bmatrix} \quad \text{where} \quad \sum \vec{\tau} = \begin{bmatrix}\sum \tau_1\\ \sum \tau_2\\ \sum \tau_3\end{bmatrix}}$$

The only remaining task is to compute the external forces and torques into the formulation above so that system of differential equations of motion can be constructed for the quadcopter.

VII. External & Aerodynamic Forces/Torques and Actuator Dynamics

With the Newton-Euler equations defined as simplified as much as possible, the next step is to determine the external forces and torques acting on the system that are unique to the application (e.g. quadcopter dynamics). As with most dynamic simulations, these external forces and torques are either do not fully encapsulate the complete dynamics of the systems or are intentionally simplified.

A. External & Aerodynamics Forces

1) Force Due to Gravity: The force of gravity is relative to the inertial reference frame and is proportional to the mass of a system by the gravitational constant.

$$F_{gravity}\Big|_{inertial} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix}$$

In the body frame...

$$\boxed{F_{gravity}\Big|_{body} = R_{zyx}(\phi,\theta,\psi)\Big|_{I}^{B} \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix}}$$

2) Net Thrust Force: From Conservation of Energy and Blade Momentum Theory it can be determined that the thrust force generated by a single rotating propellor is...

$$f_{thrust_i}\Big|_{body} = \left(\frac{K_v K_\tau \sqrt{2\rho A}}{K_t}\omega\right)^2 \qquad \rightarrow \qquad f_{thrust_i}\Big|_{body} = k_{lift} \cdot \omega_i^2$$

When considering the net thrust generated by the four propellors of a quadcopter, (assuming the propellors and motors driving them are identical) we can say...

$$F_{thrust}\Big|_{body} = \sum_{i=1}^{4} f_{thrust_i} \qquad \rightarrow \boxed{F_{thrust}\Big|_{body} = k_{lift} \begin{bmatrix} 0 \\ 0 \\ \sum \omega_i^2 \end{bmatrix}}$$

3) Drag Force: There are many different types of drag forces that can be modeled for the quadcopter; however for simplicity we are only going to cover a "general" drag term that is proportional to the velocity of the vehicle. The standard force of drag due to fluid friction is written below.

$$F_{drag}\Big|_{inertial} = \frac{1}{2}\rho C_D A \vec{v}^2$$

By collapsing all constants into a single term, the drag force can be expressed as ...

$$\boxed{F_{drag}\Big|_{inertial} = \begin{bmatrix} -k_d\dot{x} \\ -k_d\dot{y} \\ -k_d\dot{z} \end{bmatrix}}$$

4) Total External Forces:

$$\sum F_{ext} = F_{gravity}\Big|_{body} + F_{thrust}\Big|_{body} \quad F_{thrust}\Big|_{body}$$

$$\boxed{\sum F_{ext}\Big|_{body} = R_{zyx}(\phi,\theta,\psi)\Big|_{I}^{B} \left(\begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \begin{bmatrix} -k_d\dot{x} \\ -k_d\dot{y} \\ -k_d\dot{z} \end{bmatrix} \right) + k_{lift} \begin{bmatrix} 0 \\ 0 \\ \sum \omega_i^2 \end{bmatrix}}$$

B. External & Aerodynamic Torques

Modeling torques is important to study and simulation of quadcopters as controlling the balance of torque in the system is the only means of translational and rotational control that is available to a quadcopter. This is accomplished by combining torques about the CG of the quadcopter.

1) Torque Due to Drag: According to $F_{drag}|_{inertial} = \frac{1}{2}\rho C_D A \vec{v}^2$, there is a force due to drag; however, if this forces line of action is acting at some distance r from the center of gravity of the quadcopter, there is a corresponding torque to the action of this force.

$$\tau_{drag} = \frac{1}{2}\rho C_D A(r \perp \omega)^2 \qquad where \quad V = \vec{r} \times \vec{\omega}$$

Again, collapsing the constant terms gives us...

$$\tau_{drag} = b \cdot \omega^2$$

2) Motor Torques: Several torques are generated by the motor, including torques due to drag, inertial effects, and gyroscopic effects. In this paper, we will ignore both the gyroscopic and inertial torques.

$$\tau_{M_i} = b \cdot \omega_i^2 + \Gamma_{gyro} + I_M \dot{\omega_i}$$

In addition to the simplifications described above, since quadcopters spin two propellers clockwise and the other two propellers counter-clockwise, there is a size change for the direction of the angular velocity of the propeller depending on which propeller is being observed. This can be captured with the following equation.

$$\tau_{M_i} = (-1)^{i+1} b \cdot \omega_i^2$$

3) Yaw Torque: The formulation for the total torque around the yaw axis is described as the sum of all motor torques (accounting for sign changes)

$$\tau_\psi = b(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)$$

4) Roll Torque: The roll torque about the CG is a product of two motors (about the roll axis) mismatching torques and producing a net torque about that axis.

$$\tau_\phi = \sum r \times F_{thrust} = Lk(\omega_1^2 - \omega_3^2)$$

5) Pitch Torque: As will roll torques, pitch torques work the same but around the pitch axis of the body.

$$\tau_\theta = \sum r \times F_{thrust} = Lk(\omega_2^2 - \omega_4^2)$$

C. Total Body Torques

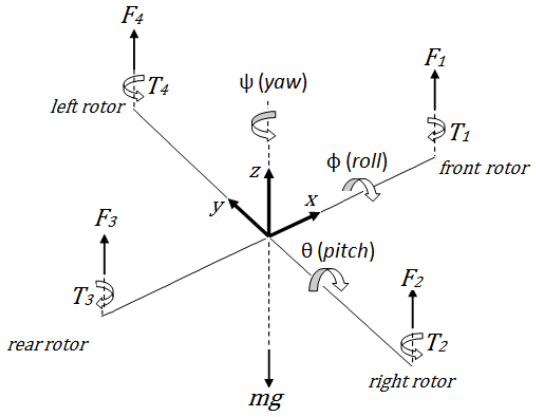
All of the torques acting on the body and be combined to form the following expression for the total torque acting on the body.

$$\boxed{\tau_{body} = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} Lk(\omega_1^2 - \omega_3^2) \\ Lk(\omega_2^2 - \omega_4^2) \\ b(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix}}$$

VIII. Quadcopter Dynamics

A quadcopter is a system of multibladed helicopter that is inherently unstable. It has a set of four rotors that are all coupled in both translational and rotational dynamics. The objective of these notes are to clearly derive governing equations of motion from first principles, and then to cast them in a way that is tractable and useful for the reader to be able to take and implement directly via simulation.

A. Free Body Diagram



B. Governing Differential Equations of Motion

Inertial Frame Quantities:

$$\vec{P} = \begin{bmatrix} x & y & z \end{bmatrix}^T \quad \vec{V_{ref}} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T$$

Body Frame Quantities:

$$V_{body} = \begin{bmatrix} u & v & w \end{bmatrix}^T \quad , \omega^{body} = \begin{bmatrix} p & q & r \end{bmatrix}^T$$

$$\boxed{\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \frac{1}{m} \sum \vec{F} - \begin{bmatrix} qw - rv \\ ru - pw \\ pv - qu \end{bmatrix}}$$

$$\boxed{\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{\sum \tau_1}{I_{xx}} \\ \frac{\sum \tau_2}{I_{yy}} \\ \frac{\sum \tau_3}{I_{zz}} \end{bmatrix} - \begin{bmatrix} \frac{(I_{zz} - I_{yy})}{I_{xx}} qr \\ \frac{(I_{xx} - I_{zz})}{I_{yy}} pr \\ \frac{(I_{yy} - I_{xx})}{I_{zz}} pq \end{bmatrix} \quad \text{where} \quad \sum \vec{\tau} = \begin{bmatrix} \sum \tau_1 \\ \sum \tau_2 \\ \sum \tau_3 \end{bmatrix}}$$

$$\boxed{\sum F_{ext}\Big|_{body} = R_{zyx}(\phi, \theta, \psi)\Big|_I^B \left(\begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \begin{bmatrix} -k_d \dot{x} \\ -k_d \dot{y} \\ -k_d \dot{z} \end{bmatrix} \right) + k_{lift} \begin{bmatrix} 0 \\ 0 \\ \sum \omega_i^2 \end{bmatrix}}$$

$$\boxed{\tau_{body} = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} Lk(\omega_1^2 - \omega_3^2) \\ Lk(\omega_2^2 - \omega_4^2) \\ b(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix}}$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s(\theta) \\ 0 & c(\phi) & s(\phi)c(\theta) \\ 0 & -s(\phi) & c(\phi)c(\theta) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & \frac{s(\phi)}{c(\theta)} & \frac{c(\phi)}{c(\theta)} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

IX. Simulation

Now that the dynamics of the system have been handled, we can turn to the simulation by preparing a State Space model of the quadcopter and then applying a feedback control system such that we can control the pose of the quadcopter in space and guide it to desirable positions.

A. State Space Formulation

1) System States & Inputs: While it is possible to further model the system by including the dynamics of the actuators (four electric motors), for simplicity this simulation will assume direct control to command exact motor speeds

$$\vec{U} = [\omega_1, \omega_2, \omega_3, \omega_4]^T$$

$$\vec{Q} = \begin{bmatrix} u, v, w, p, q, r, \bar{q}, x, y, z \end{bmatrix}$$

2) Canonical State Space Form:

$$\dot{\vec{Q}} = \tilde{A}()\vec{Q} + \tilde{B}()\vec{U}$$

3) Quadcopter State Space:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R(\bar{q}) \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \frac{1}{m} \left(R(\bar{q}) \Big|_I^B \left(\begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \begin{bmatrix} -k_d \dot{x} \\ -k_d \dot{y} \\ -k_d \dot{z} \end{bmatrix} \right) + k_{lift} \begin{bmatrix} 0 \\ 0 \\ \sum \omega_i^2 \end{bmatrix} \right) - \begin{bmatrix} qw - rv \\ ru - pw \\ pv - qu \end{bmatrix}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{1}{I_{xx}} & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix} \begin{bmatrix} Lk(\omega_1^2 - \omega_3^2) \\ Lk(\omega_2^2 - \omega_4^2) \\ b(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix} - \begin{bmatrix} \frac{(I_{zz} - I_{yy})}{I_{xx}} qr \\ \frac{(I_{xx} - I_{zz})}{I_{yy}} pr \\ \frac{(I_{yy} - I_{xx})}{I_{zz}} pq \end{bmatrix}$$

B. Discretization Scheme

C.