# quadcopter package

dinay-kingkiller

## I. Introduction

THE quadcopter package is filled with obtuse and often confusing code derived from various physics equations and mathematical formulas. This paper hopes to bridge the gap between quadcopter theory and the src files included. This paper can then also be used as a reference for later expansion of this package.

## II. The Model Node

While an actual quadcopter would be helpful, a simulation is the next best thing. The model node simulates the physics of a quadcopter. The physics in this section also provide insight on how motor input and sensor output behave.

### A. Rotational Kinematics

Quaternions are integral in understanding the implementation of the model node. Derivations of quaternion operations can be found elsewhere, including on my GitHub. This paper will then focus on the implementation in the model node. There are multiple definitions for quaternion multiplication. The model node uses the same definition as the tf2 ROS package.

$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ w_1 \end{pmatrix} \otimes \begin{pmatrix} x_2 \\ y_2 \\ z_2 \\ w_2 \end{pmatrix} = \begin{pmatrix} w_1x_2 + x_1w_2 + y_1z_2 - z_1y_2 \\ w_1y_2 - x_1z_2 + y_1w_2 + z_1x_2 \\ w_1z_2 + x_1y_2 - y_1x_2 + z_1w_2 \\ w_1w_2 - x_1x_2 - y_1y_2 - z_1z_2 \end{pmatrix} \tag{1}$$

The unit quaternion $\mathrm{q}$ describes the orientation of the robot by describing a transform between two frames

$$\begin{pmatrix} \mathbf{v}^B \\ 0 \end{pmatrix} = \mathrm{q} \otimes \begin{pmatrix} \mathbf{v}^N \\ 0 \end{pmatrix} \otimes \mathrm{q}^* \tag{2}$$

where $\mathbf{v}^N$ and $\mathbf{v}^B$ are the same vector described in an inertial frame $N$ and the robot body frame $B$ respectively. A rotation matrix is another description of rotations:

$$\mathbf{v}^B = \mathrm{R}\left(\mathbf{v}^N\right) \tag{3}$$

Importantly for the model node, rotation matrices collect the dot product of basis vectors: $\mathrm{R}_{ij} = \hat{\mathbf{b}}_i \cdot \hat{\mathbf{n}}_j$. In terms of quaternion $(x, y, z, w)$

$$\mathrm{R} = \begin{pmatrix} x^2-y^2-z^2+w^2 & 2xy-2zw & 2xz+2yw \\ 2xy+2zw & -x^2+y^2-z^2+w^2 & -2xw+2yz \\ 2xz-2wy & 2xw+2yz & -x^2-y^2+z^2+w^2 \end{pmatrix} \tag{4}$$

That rate of change of the quaternion can be calculated from the angular velocity

$$\dot{\mathrm{q}} = \frac{1}{2} \begin{pmatrix} \boldsymbol{\omega}_B^N \\ 0 \end{pmatrix} \otimes \mathrm{q} \tag{5}$$

where $\boldsymbol{\omega}_B^N$ is the angular velocity of the robot body $B$ in the inertial frame $N$ described in the body basis:

$$\boldsymbol{\omega}_B^N = \omega_x \hat{\mathbf{b}}_x + \omega_y \hat{\mathbf{b}}_y + \omega_z \hat{\mathbf{b}}_z \tag{6}$$

### B. Linear Equations of Motion

The linear equations of motion are pretty apparent upon starting our analysis. First, the motion of the quadcopter center of mass $c$ from its origin point $o$ is defined simply:

$$\mathbf{p}_{c/o} = p_x \hat{\mathbf{b}}_x + p_y \hat{\mathbf{b}}_y + p_z \hat{\mathbf{b}}_z \tag{7}$$

Instead of calculating the acceleration in the inertial frame, we can use v2pt theory to expand the derivatives

$$\mathbf{v}_c^N = \mathbf{v}_c^B + \boldsymbol{\omega}_B^N \times \mathbf{p}_{c/o} \tag{8}$$

$$\begin{aligned} \mathbf{a}_c^N = {} & \mathbf{a}_c^B + \boldsymbol{\alpha}_B^N \times \mathbf{p}_{c/o} \\ & + 2\boldsymbol{\omega}_B^N \times \mathbf{v}_c^B + \boldsymbol{\omega}_B^N \times \left(\boldsymbol{\omega}_B^N \times \mathbf{p}_{c/o}\right) \end{aligned} \tag{9}$$

The first term is the linear acceleration observed by the robot.

$$\mathbf{a}_c^B = \ddot{p}_x \hat{\mathbf{b}}_x + \ddot{p}_y \hat{\mathbf{b}}_y + \ddot{p}_z \hat{\mathbf{b}}_z \tag{10}$$

The second term is the Euler acceleration

$$\begin{aligned} \mathbf{a}_{euler} = {} & \left(p_z\dot{\omega}_y - p_y\dot{\omega}_z\right) \hat{\mathbf{b}}_x \\ & + \left(p_x\dot{\omega}_z - p_z\dot{\omega}_x\right) \hat{\mathbf{b}}_y \\ & + \left(p_y\dot{\omega}_x - p_x\dot{\omega}_y\right) \hat{\mathbf{b}}_z \end{aligned} \tag{11}$$

The third term is the Coriolis acceleration

$$\begin{aligned} \mathbf{a}_{coriolis} = {} & \left(2\omega_y\dot{p}_z - 2\omega_z\dot{p}_y\right) \hat{\mathbf{b}}_x \\ & + \left(2\omega_z\dot{p}_x - 2\omega_x\dot{p}_z\right) \hat{\mathbf{b}}_y \\ & + \left(2\omega_x\dot{p}_y - 2\omega_y\dot{p}_x\right) \hat{\mathbf{b}}_z \end{aligned} \tag{12}$$

And the final term is the centrifugal acceleration

$$\begin{aligned} \mathbf{a}_{center} = {} & \left(\omega_x\omega_y p_y + \omega_x\omega_z p_z - \omega_y^2 p_x - \omega_z^2 p_x\right) \hat{\mathbf{b}}_x \\ & + \left(\omega_x\omega_y p_x + \omega_y\omega_z p_z - \omega_x^2 p_y - \omega_z^2 p_y\right) \hat{\mathbf{b}}_y \\ & + \left(\omega_x\omega_z p_x + \omega_y\omega_z p_y - \omega_x^2 p_z - \omega_y^2 p_z\right) \hat{\mathbf{b}}_z \end{aligned} \tag{13}$$

For our model we assume each motor puts out a force proportional to its square. If we let $m$ be the total mass of the robot and $k$ be the proportionality constant:

$$m\mathbf{a}_c^N = k\left(\omega_f^2 + \omega_l^2 + \omega_b^2 + \omega_r^2\right) \hat{\mathbf{b}}_z - mg\hat{\mathbf{n}}_z \tag{14}$$

Where $f$, $l$, $b$, and $r$ stand for the front, right, back and left motors respectively.

### C. Rotational Equations of Motion

Supposing that most of the mass of the quadcopter is in the motors, the inertia dyadics (around the center of mass $c$) of the four motors are

$$\mathbf{I}_{f/c} = \frac{mr^2}{4}\hat{\mathbf{b}}_y\hat{\mathbf{b}}_y + \frac{mr^2}{4}\hat{\mathbf{b}}_z\hat{\mathbf{b}}_z \tag{15}$$

$$\mathbf{I}_{l/c} = \frac{mr^2}{4}\hat{\mathbf{b}}_x\hat{\mathbf{b}}_x + \frac{mr^2}{4}\hat{\mathbf{b}}_z\hat{\mathbf{b}}_z \tag{16}$$

$$\mathbf{I}_{b/c} = \frac{mr^2}{4}\hat{\mathbf{b}}_y\hat{\mathbf{b}}_y + \frac{mr^2}{4}\hat{\mathbf{b}}_z\hat{\mathbf{b}}_z \tag{17}$$

$$\mathbf{I}_{r/c} = \frac{mr^2}{4}\hat{\mathbf{b}}_x\hat{\mathbf{b}}_x + \frac{mr^2}{4}\hat{\mathbf{b}}_z\hat{\mathbf{b}}_z \tag{18}$$

where the mass of each motor is $m/4$ and $r$ is the radius of the robot. The total inertial dyadic of the robot is then

$$\mathbf{I}_{B/c} = \frac{mr^2}{2}\left(\hat{\mathbf{b}}_x\hat{\mathbf{b}}_x + \hat{\mathbf{b}}_y\hat{\mathbf{b}}_y + 2\hat{\mathbf{b}}_z\hat{\mathbf{b}}_z\right) \tag{19}$$

The angular velocity was given in equation 6.

$$\boldsymbol{\omega}_B^N = \omega_x\hat{\mathbf{b}}_x + \omega_y\hat{\mathbf{b}}_y + \omega_c\hat{\mathbf{b}}_z$$

The angular acceleration comes out simple because of our choice of reference frame

$$\begin{aligned}\boldsymbol{\alpha}_B^N &= \frac{{}^B d}{dt}\boldsymbol{\omega}_B^N + \boldsymbol{\omega}_B^N \times \boldsymbol{\omega}_B^N \\ &= \dot{\omega}_x\hat{\mathbf{b}}_x + \dot{\omega}_y\hat{\mathbf{b}}_y + \dot{\omega}_z\hat{\mathbf{b}}_z\end{aligned} \tag{20}$$

And the angular momentum around the center of mass $c$ is pretty nice too

$$\begin{aligned}\mathbf{H}_{B/c}^N &= \mathbf{I}_{B/c}\cdot\boldsymbol{\omega}_B^N \\ &= \frac{mr^2}{2}\omega_x\hat{\mathbf{b}}_x + \frac{mr^2}{2}\omega_y\hat{\mathbf{b}}_y + mr^2\omega_z\hat{\mathbf{b}}_z\end{aligned} \tag{21}$$

And the derivative is

$$\begin{aligned}\frac{{}^N d}{dt}\mathbf{H}_{B/c}^N &= \mathbf{I}\cdot\boldsymbol{\alpha}_B^N + \boldsymbol{\omega}_B^N \times \mathbf{H}_{B/c}^N \\ &= \frac{mr^2}{2}\dot{\omega}_x\hat{\mathbf{b}}_x + \frac{mr^2}{2}\dot{\omega}_y\hat{\mathbf{b}}_y + mr^2\dot{\omega}_y\hat{\mathbf{b}}_z \\ &\quad + \frac{mr^2}{2}\omega_y\omega_z\hat{\mathbf{b}}_x - \frac{mr^2}{2}\omega_x\omega_z\hat{\mathbf{b}}_y\end{aligned} \tag{22}$$

Quadcopters move laterally by rolling or pitching (rotation around the $x$ and $y$ axes respectively) and increasing thrust to go "up". A simple rotation around a single axis is generated by increasing the speed on one propeller and decreasing its opposite. The moment of these forces provides a torque to the robot. All forces point up, so

$$\begin{aligned}\mathbf{M}_{B/c} &= r\hat{\mathbf{b}}_x \times k\omega_f^2\hat{\mathbf{b}}_z - r\hat{\mathbf{b}}_x \times k\omega_b^2\hat{\mathbf{b}}_z \\ &\quad + r\hat{\mathbf{b}}_y \times k\omega_l^2\hat{\mathbf{b}}_z - r\hat{\mathbf{b}}_y \times k\omega_r^2\hat{\mathbf{b}}_z \\ &= kr\left(\omega_l^2 - \omega_r^2\right)\hat{\mathbf{b}}_x + kr\left(\omega_b^2 - \omega_f^2\right)\hat{\mathbf{b}}_y\end{aligned} \tag{23}$$

For sanity sake, let's show that the other force acting on the motors, gravity, does not affect the rotation

$$\begin{aligned}-r\hat{\mathbf{b}}_x \times \frac{mg}{4}\hat{\mathbf{n}}_z + r\hat{\mathbf{b}}_x \times \frac{mg}{4}\hat{\mathbf{n}}_z = \mathbf{0} \\ -r\hat{\mathbf{b}}_y \times \frac{mg}{4}\hat{\mathbf{n}}_z + r\hat{\mathbf{b}}_y \times \frac{mg}{4}\hat{\mathbf{n}}_z = \mathbf{0}\end{aligned}$$

As long as the weight of the quadcopter is balanced along the axes, gravity should not affect the rotation. Adjacent quadcopter propellers have reverse pitch, so that together they can provide a yaw torque (around the $z$ axis) in either direction; without it quadcopters would have to choose to fall or rotate. Choosing the front and rear motors to provide positive torque

$$\boldsymbol{\tau} = b\left(\omega_f^2 - \omega_r^2 + \omega_b^2 - \omega_l^2\right)\hat{\mathbf{b}}_z \tag{24}$$

The last couple equations combine to form the rotational equations of motion

$$\frac{{}^N d}{dt}\mathbf{H}_{B/c}^N = \mathbf{M}_{B/c} + \boldsymbol{\tau} \tag{25}$$

### D. Numerical Integration

The first differential equations are the easiest differential equations. Define the velocities $v_i = \dot{r}_i$. Then $\dot{v}_i = \ddot{p}_i$ and

$$\dot{p}_x = v_x \tag{26}$$

$$\dot{p}_y = v_y \tag{27}$$

$$\dot{p}_z = v_z \tag{28}$$

Using the linear equation of motion (equation 14), we can take the dot product with respect to inerial basis $N$. Recall that the rotation matrix given in equation 4 collects the dot products between the two bases. Then, the second set of differential equations becomes

$$\dot{v}_x = \omega_z v_y - \omega_y v_z - \omega_x \tag{29}$$

$$\dot{v}_y = 2Tq_xq_w + 2q_yq_z \tag{30}$$

$$\dot{v}_z = -Tq_x^2 - Tq_y^2 + Tq_z^2 + Tq_w^2 \tag{31}$$

where $T$ is the specific motor thrust.

$$T = \frac{k}{m}\left(\omega_f^2 + \omega_l^2 + \omega_b^2 + \omega_r^2\right) \tag{32}$$

The next set of differential equations come from combining the rotational equations of motion (equations 22-25). Again, we use the rotation matrix given in equation 4, but this time we take the dot products in the robot frame $B$. The rotational equations then come out not terribly complicated

$$\dot{\omega}_x = \omega_y\omega_z + \frac{2k}{mr}\left(\omega_l^2 - \omega_r^2\right) \tag{33}$$

$$\dot{\omega}_y = \omega_x\omega_z + \frac{2k}{mr}\left(\omega_b^2 - \omega_f^2\right) \tag{34}$$

$$\dot{\omega}_z = \frac{b}{mr^2}\left(\omega_f^2 - \omega_r^2 + \omega_b^2 - \omega_l^2\right) \tag{35}$$

Expanding the quaternion differential equation (equation 5) we get the final four differential equations.

$$\dot{q}_x = -\omega_z q_y + \omega_y q_z + \omega_x q_w \tag{36}$$

$$\dot{q}_y = \omega_z q_x - \omega_x q_z + \omega_y q_w \tag{37}$$

$$\dot{q}_z = -\omega_y q_x + \omega_x q_y + \omega_z q_w \tag{38}$$

$$\dot{q}_w = -\omega_x q_x - \omega_y q_y - \omega_z q_z \tag{39}$$

The quadcopter model uses most simple integration method (Euler's Method), which is fast to implement, but more accurate methods like Runge-Kutta or Crouch-Grossman could

improve the accuracy. However that shouldn't be needed. Those algorithms can be more computationally intensive, and the model node just needs to approximate the motion of a quadcopter. For all the non-linear differential equations $\dot{u} = f[\mathbf{u}]$, the discrete solutions are modeled as

$$u(t + \Delta t) = u(t) + f[\mathbf{u}(t)]\Delta t \tag{40}$$

### *E. Constant Controller*

This section is for deriving a simple controller that provides constant thrust to the quadcopter. It's used for testing out if the physics of the model are set up correctly. All the controllers are based on a reference balanced output. From a zeroed position the quadcopter should hover in place with the balanced output. First we must find a way to remove any rotational acceleration ($\frac{d\mathbf{H}}{dt} = \mathbf{0}$).

$$\begin{aligned} kr\left(\omega_l^2 - \omega_r^2\right) &= 0 \\ kr\left(\omega_b^2 - \omega_f^2\right) &= 0 \\ b\left(\omega_f^2 - \omega_r^2 + \omega_b^2 - \omega_l^2\right) &= 0 \end{aligned}$$

In the above it can be seen with equal motor thrusts the torques on the quadcopter *should* zero. Removing any rotational motion, zeroing the acceleration becomes

$$4km\omega_i^2 - mg = 0$$

Or

$$\omega_i = \frac{1}{2}\sqrt{\frac{g}{k}} \tag{41}$$

### *F. Sensor Model*

The model node also simulates sensors onboard a typical quadcopters. While there are more advanced sensors that can be added later (i.e. cameras, GPS, altimeter), for now, this section only contains a three degrees of freedom (3-DoF) gyroscope, a 3-DoF accelerometer, and a 3-DoF magnetometer. Thanks to the magic that is quaternion integration, the three sensor values of a gyroscope are the same as the angular velocity described in basis $B$. From the angular velocity defintion in equation 6, we get

$$g_i = \boldsymbol{\omega}_B^N \cdot \hat{\mathbf{b}}_i = \omega_i \tag{42}$$

An accelerometer at the most basic level is a known mass that's allowed to move with respect to its own equations of motion.

$$\mathbf{a}_c^N \cdot \hat{\mathbf{b}}_i = -g\hat{\mathbf{n}}_y \cdot \hat{\mathbf{b}}_i \tag{43}$$

From the definition of acceleration in equation **??**, our definition of $v_i$ in equations 26-28, and using the rotation matrix given in equation 4

$$\begin{aligned} a_x = {} & \dot{v}_x q_x^2 - \dot{v}_x q_y^2 - \ddot{v}_x q_z^2 + \dot{v}_x q_w^2 \\ & + 2\dot{v}_y q_x q_y - 2\dot{v}_y q_z q_w + 2\dot{v}_z q_x q_z + 2\dot{v}_z q_y q_w \\ & + 2g q_x q_y - 2g q_z q_w \end{aligned} \tag{44}$$

One example of an IMU is the ICM20948. From the datasheet,

$$\text{X\_acceleration} = \text{ACCEL\_XOUT}/\text{Accel\_Sensitivity} \tag{45}$$

Or

$$\text{ACCEL\_iOUT} = (\mathbf{a} \cdot \mathbf{b}_i)\text{Accel\_Sensitivity} \tag{46}$$

for $i \in \{X, Y, Z\}$. The acceleration word length is 16 bits output in two's complement format. We let C++ manage this with `static_cast`.

$$\text{ACCEL\_iOUT\_H} = (\text{ACCEL\_iOUT} \ll 8)\ \&\ \texttt{0xFF} \tag{47}$$

$$\text{ACCEL\_iOUT\_L} = \text{ACCEL\_iOUT}\ \&\ \texttt{0xFF} \tag{48}$$

## III. Localization Node