Quadrotor Mathematics For Simpletons

Cody Lewis
srlm@anzhelka.com

Luke De Ruyter
ilukester@anzhelka.com

June 12, 2012

And when he [Herod] had apprehended him [Peter], he put him in prison, and delivered him to four quaternions of soldiers to keep him; intending after Easter to bring him forth to the people.
–Acts 12:14, King James Bible, Cambridge Edition

Abstract

This paper describes in detail the dynamics of a quadrotor platform. Most current research papers simply explain the equations without explaining derivations or reasoning behind the numbers. This paper will explain and derive the equations for a quadrotor aircraft based an the system found in [1].

1 Introduction

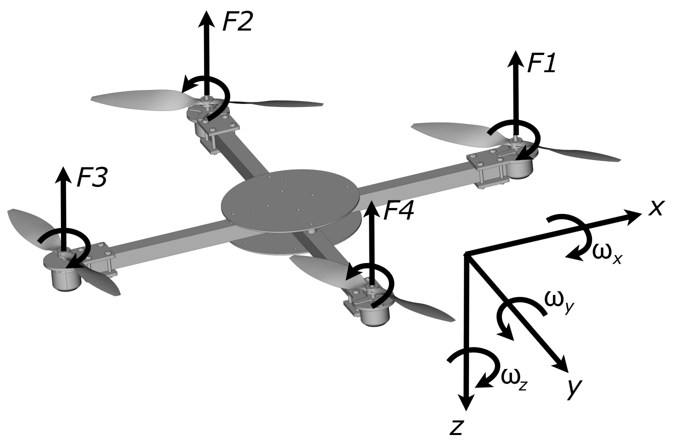

Figure 1: Orientation of the body axis with respect to the motor numbers and rotations.

A quadrotor is a simple aerial vehicle much like a helicopter, but instead of a single main rotor a quadrotor has four rotors. This allows for a much simpler mechanical system, since each rotor can be directly mounted to it's motor without any linkages to change the pitch of a rotor.

Quadrotor dynamics are very complex. It's difficult because you have four motors, all pointed in the same direction, and that's it. But yet the vehicle can move on three axes in translational motion, and can rotate about another three axes. A quadrotor must be able to convert the four motor inputs into these six different (and often conflicting) motions.

The simplest case is hovering in a single location, and will be dealt with in this paper. Initially it is assumed that the quadrotor will have negligible translational motion in the global x-y plane, although eventually the translational equations may be added to this paper.

The quadrotor used in this paper is assumed to be fairly simple. The platform must be able to measure the following components:

* rotation (direction and velocity)
* translation (velocity and accleration)
* motor rotational speed
* air density

Note that if air density is assumed to be constant than performance may suffer at different altitudes or weather. In addition, the following constants must be empirically measured for each quadrotor vehicle:

* mass (kg)
* moment of inertia tensor ($kg \cdot m^2$)
* offset of each rotor from the center of mass (m)
* thrust coefficient (see subsection 5.1)
* torque coefficient (see subsection 5.1)
* rotor diameter (m)

The quadrotor system is complicated by the fact that a quadrotor will need to be able to balance itself in flight. If a gust of wind blows from the side and rolls the quadrotor, it will no longer be in a stable hover. The controller will need to increase the speed of some motors and decrease the speed of others so that the vehicle is rotated back into a vertical hovering state.

There is still a problem: each motor is producing torque on the body as well as thrust. This is manifested by the motor "attempting" to rotate the body due to the spinning mass of the motor and propeller, and the air resistance the propeller encounters. On a quadrotor frame we have two motors spinning clockwise and two spinning counter clockwise, and the torques cancel each other out. But if a motor's speed in increased to account for a tilt, then it's torque on the frame will increase as well, causing the whole body to rotate.

And so now, when a gust of wind blows on the frame, we have to somehow correct the tilt without allowing the torque to yaw the frame. At other times we may want to yaw without tilting, and sometimes we want to do both simultaneously. There is so much more beyond just the basic thrust, tilt, and yaw: we have to account for many factors such as free stream velocity (or air, in layman's terms) through the propellers, blade flapping, sensor failure, and the many small equations required to support the main equations.

2 Symbols Used

2.1 Variables

Variables with the subscript i denote association with motor i.

$\boldsymbol{\omega}_b$	angular velocity in the body frame (rad/s)
$\mathbf{q}$	attitude quaternion
$\tilde{\mathbf{q}}$	error quaternion
$\mathbf{q}_d$	desired quaternion
$\boldsymbol{V}_b$	velocity in the body frame (m/s)
$\boldsymbol{R}_e$	position in the inertial frame (m)
F_z	total force of the rotors on the z axis (N)
$M_x - M_z$	total rotor moments along each axis ($N \cdot m$)
$F_1 - F_4$	thrust force of each rotor (N)
Q	rotor torque ($N \cdot m$)
T	rotor thrust force (N)
n_i	rotation frequency of the rotors (Hz)
Ω_i	rotation speed of the rotors (rad/s)
Ω_{di}	desired rotation speed of the rotors (rad/s)
Ω_{imeas}	measured rotation speed of the rotors (rad/s)
$\hat{\Omega}_i$	expected speed (rad/s)
u_i	motor command (???)
a_i	motor gain (no units)
ψ	yaw angle (???)
γ_H	tilt direction in the x-y plane (???)
α_H	tilt amount around γ_H (???)
$\boldsymbol{q}_V$	yaw error
$\boldsymbol{q}_H$	tilt error
$\begin{bmatrix} r_x \\ r_y \end{bmatrix}$	tilt axis in the $x - y$ plane (???)
β_H	direction of the tilt axis in the $x - y$ plane (???)
K_{tilt}	amount of tilt in the system? (from altitude controller)

2.2 Constants

$\boldsymbol{I}_{nb}$	moment of inertia tensor ($kg \cdot m^2$)
D	rotor diameter (m)
d	offset of each rotor from the center of mass (m)
ρ	air density (kg/m^3)
K_t	thrust coefficient (???)
K_q	torque coefficient (???)
K_g	the speed adaptable gain for a_i (???)
$f_{mot}(\cdot)$	motor response function (???)
τ	time constant, step (???)
K_{PH}	tilt moment proportional constant
K_{DH}	tilt moment derivative constant
K_{Pz}	yaw moment proportional constant
K_{Dz}	yaw moment derivative constant
F_{PIDz}	proportional altitude controller (N ?)
m	quadrotor mass (kg ?)
g	local gravity (m/s^2)
K_{P_z}	Proportional PID gain for F_z
K_{I_z}	Integral PID gain for F_z
K_{I_z}	Derivative PID gain for F_z
K_{P_i}	Motor Proportional PID gain (for $1 \leq i \leq 4$)
K_{I_i}	Motor Integral PID gain (for $1 \leq i \leq 4$)

3 Introduction to Quaternions

This paper represents orientation using quaternions. There are other methods of representing rotation, but a quaternion has several advantages that lend it well to use on the quadrotor:

* Quaternions avoid gimbal lock (where two rotation axes align on each other during the course of a rotation, producing unnatural results)

* Quaternions have smaller memory requirements (4 floats) than other representations

* Quaternions have much more efficient mathematical operations, and don't require orthonormalization (making the axes orthogonal after an imprecise computation) like a rotation matrix does.

* Quaternions allow for smooth interpolation between two orientations.

* Orientation updates (updating a refernce orientation with ω, for example) do not require trigonometric functions.

A quaternion is a way of representing a rotation. This is similar to Euler angles, but has the advantage of a more concise notation (and math), and not having singularities like Euler angles have. Quaternions are difficult to visualize from the numbers, but are easy enough to understand and use.

A quaternion is geometrically defined as follows:

$$\boldsymbol{q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos(\alpha/2) \\ \sin(\alpha/2) \cdot \boldsymbol{r} \end{bmatrix}$$

This means that the rotation is around the axis $\boldsymbol{r}$ and by amount α. Note that in the second representation the second element is q_1, q_2, and q_3 (it's three because of the $\cdot\boldsymbol{r}$, which is a three element vector). To add some terminology, the first term q_0 is called the scalar part, while q_1 through q_3 is called the vector part.

A quaternion can be thought of a number, just like any other. So, we can do several operations such as add, multiply, invert (divide), take the norm, and of course test for equality. We'll be focusing on rotation quaternions for the rest of this paper. A rotation quaternion is the specific form of quaternion used to represent rotation, and must have a norm equal to one. It should be noted here that a rotation quaternion is of length one (unit) only by convention and because it simplifies some proofs, but that any non-zero real multiple of q gives the the same rotation (although the interpretation of rotating by Ω about v is lost for non-unit quaternions).

We define the norm of quaternion as

$$\|q\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}$$

The norm can be thought of as the "length" of a quaternion (or vector, etc). A $\|q\|$ value of 1 is called a unit quaternion. In quaternion literature there seems to be a discrepancy: some authors define the quaternion norm as above, with a square root, while others omit the square root (e.g. [3]). In this paper we will use the norm with a square root. This is called the L2-norm, or more commonly the Euclidean norm. I couldn't find anything on a norm that squared it's components but had no root, so we'll ignore it (for more information on norms, see [2]).

We also define the conjugate and inverse of a rotation quaternion as

$$q^* = q' = (q_0, -q_1, -q_2, -q_3) = q^{-1}$$

Note this equality is only true for unit quaternions (rotation quaternions), and does not hold for other quaternions.

The only binary operation that we will be using for rotation quaternions is multiplication, denoted by $\otimes$. Note that quaternion multiplication is not commutative. Multiplying $q_1 \otimes q_2 \neq q_2 \otimes q_1$. Multiplying rotation quaternions produces a third rotation quaternion that is the result of rotating each quaternion in reverse order starting with the right hand side quaternion. To rotate a standard vector from one coordinate system to another simply place it into the vector part of the quaternion and set the scalar part to zero. Then, for transforming the vector r_e to the vector r_b:

$$\begin{bmatrix} 0 \\ r_b \end{bmatrix} = \boldsymbol{q}^* \otimes \begin{bmatrix} 0 \\ r_e \end{bmatrix} \otimes \boldsymbol{q} \tag{3.1}$$

$$\begin{bmatrix} 0 \\ r_e \end{bmatrix} = \boldsymbol{q} \otimes \begin{bmatrix} 0 \\ r_b \end{bmatrix} \otimes \boldsymbol{q}^* \tag{3.2}$$

$$\tag{3.3}$$

For the rest of this paper, if a vector is rotated by a quaternion it will be written as a vector but assumed to be in the vector part of a quaternion. For example,

$$\omega_b' = \boldsymbol{q} \otimes \omega_b, \qquad \text{where} \quad \omega_b = \begin{bmatrix} 0 \\ \omega_b \end{bmatrix} \tag{3.4}$$

The vector ω_b is rotated by quaternion $\boldsymbol{q}$ to get the rotated vector ω_b'.

The product of two quaternions is another quaternion and is defined as:

$$\begin{aligned} p \otimes q = (&p_0q_0 - p_1q_1 - p_2q_2 - p_3q_3, \\ &p_0q_1 + p_1q_0 + p_2q_3 - p_3q_2, \\ &p_0q_2 + p_2q_0 + p_3q_1 - p_1q_3, \\ &p_0q_3 + p_3q_0 + p_1q_2 - p_2q_1) \end{aligned} \tag{3.5}$$

One thing to note is that a quaternion multiplication has 12 scalar additions and 16 scalar multiplies. Since the components are floating point numbers, each quaternion multiplication consumes 28 floating point operations.

4 Basic Equations

In this section we will cover in detail all of the equations necessary to fly. This section explains the derivation and reasoning behind each equation used, and any assumptions that are made.

4.1 Roll, Pitch, and Yaw Errors

The control of a quadrotor must first consider maintaining a desired attitude. For hover, the attitude would be with no roll or pitch, and possibly some yaw. We start by generating a desired orientation $\boldsymbol{q}_d$. This can be either a from some software routine (autonomous), or perhaps from a user control (ie, the orientation of a control board). Given the current orientation $\boldsymbol{q}$, the quadrotor must be turned through some rotation $\tilde{\boldsymbol{q}}$. Hence,

$$\boldsymbol{q}_d = \tilde{\boldsymbol{q}} \otimes \boldsymbol{q} \tag{4.1}$$

$$\tilde{\boldsymbol{q}} = \boldsymbol{q}_d \otimes \boldsymbol{q}^* \tag{4.2}$$

Rewritten using the geometric definition of a quaternion,

$$\tilde{\boldsymbol{q}} = \begin{bmatrix} \cos(\alpha/2) \\ \sin(\alpha/2) \cdot \boldsymbol{r}_e \end{bmatrix} \tag{4.3}$$

Equation (4.3) says that the error quaternion is composed of an angle α and a vector $\boldsymbol{r}_e$ (remember, the e subscript denotes the earth reference frame). From equation (4.2) we have the value of $\tilde{\boldsymbol{q}}$, so equation (4.3) should be rewritten as

$$\begin{bmatrix} \cos(\alpha/2) \\ \sin(\alpha/2) \cdot \boldsymbol{r}_{e_1} \\ \sin(\alpha/2) \cdot \boldsymbol{r}_{e_2} \\ \sin(\alpha/2) \cdot \boldsymbol{r}_{e_3} \end{bmatrix} = \begin{bmatrix} \tilde{q_0} \\ \tilde{q_1} \\ \tilde{q_2} \\ \tilde{q_3} \end{bmatrix} \tag{4.4}$$

This puts the four known $\tilde{q_i}$ on the right hand side and the unknown α and three $\boldsymbol{r}_{e_j}$ on the left hand side. Solving, we have

$$\alpha = 2 \arccos(\tilde{q_0}) \tag{4.5}$$

And, using the calculated α we solve the following:

$$\boldsymbol{r}_{e_1} = \frac{\tilde{q_1}}{\sin \alpha/2} \tag{4.6}$$

$$\boldsymbol{r}_{e_2} = \frac{\tilde{q_2}}{\sin \alpha/2} \tag{4.7}$$

$$\boldsymbol{r}_{e_3} = \frac{\tilde{q_3}}{\sin \alpha/2} \tag{4.8}$$

$$\tag{4.9}$$

The vector $\boldsymbol{r}_e$ is in earth coordinates, so using the properties of quaternion rotation we must rotate it into body coordinates to determine which movements are necessary.

$$\boldsymbol{r}_b = \boldsymbol{q}^* \otimes \boldsymbol{r}_e \otimes \boldsymbol{q} \tag{4.10}$$

And now we can create the error quaternion in body coordinates $\tilde{\boldsymbol{q}}_b$. Note that α is the same as in equation (4.5). Even though the two vectors $\boldsymbol{r}_e$ and $\boldsymbol{r}_b$ differ numerically, they are really identical except for being in different coordinate systems. So the rotation α is the same.

$$\tilde{\boldsymbol{q}}_b = \begin{bmatrix} \cos(\alpha/2) \\ \sin(\alpha/2) \cdot \boldsymbol{r}_b \end{bmatrix} \tag{4.11}$$

Instead of defining three different angles of rotation (ie, roll, pitch, and yaw) this paper will define the body rotation as tilt (encompassing both roll and pitch) and yaw. This can be written as

$$\tilde{\boldsymbol{q}}_b = \boldsymbol{q}_H \otimes \boldsymbol{q}_V \tag{4.12}$$

where $\boldsymbol{q}_H$ is the tilt error and $\boldsymbol{q}_V$ is the yaw error. From the definition of the quaternion, we can rewrite (4.12) in the geometric form and expand the quaternion multiplication.

$$\begin{aligned}
\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} &= \begin{bmatrix} \cos(\alpha_H/2) \\ \sin(\alpha_H/2)\cdot r_x \\ \sin(\alpha_H/2)\cdot r_y \\ \sin(\alpha_H/2)\cdot 0 \end{bmatrix} \otimes \begin{bmatrix} \cos(\psi/2) \\ \sin(\psi/2)\cdot 0 \\ \sin(\psi/2)\cdot 0 \\ \sin(\psi/2)\cdot 1 \end{bmatrix} \\
&= \begin{bmatrix} \cos(\alpha_H/2) \\ \sin(\alpha_H/2)\cdot r_x \\ \sin(\alpha_H/2)\cdot r_y \\ 0 \end{bmatrix} \otimes \begin{bmatrix} \cos(\psi/2) \\ 0 \\ 0 \\ \sin(\psi/2)\cdot 1 \end{bmatrix} \\
&= \begin{bmatrix} \cos(\alpha_H/2)\cos(\psi/2) - 0 - 0 - 0 \\ 0 + \sin(\alpha_H/2)\cdot r_x\cos(\psi/2) + \sin(\alpha_H/2)\cdot r_y\sin(\psi/2) + 0 \\ 0 + \sin(\alpha_H/2)\cdot r_y\cos(\psi/2) + 0 - \sin(\alpha_H/2)\cdot r_x\sin(\psi/2) \\ \cos(\alpha_H/2)\sin(\psi/2) + 0 + 0 + 0 \end{bmatrix} \qquad (4.13)
\end{aligned}$$

α_H is the amount of tilt, positive by convention (note it is not the same as α in 4.11 and earlier equations). r_x and r_y is the tilt axis in the body $x-y$ plane, and ψ is the yaw angle. Next, using simple algebra we can solve for the four unknowns (α_H, ψ, r_x, and r_y) with the four knowns (q_i). This gives the following four results:

$$\alpha_H = \arccos\left[1 - 2\left(q_1^2 + q_2^2\right)\right] \qquad 0 \le \alpha_H < \pi \qquad (4.14)$$

$$\psi = 2\arctan_2(q_3, q_0) \qquad -\pi \le \psi < \pi \qquad (4.15)$$

$$r_x = \frac{\cos(\psi/2)\, q_1 - \sin(\psi/2)\, q_2}{\sin(\alpha_H/2)} \qquad (4.16)$$

$$r_y = \frac{\sin(\psi/2)\, q_1 - \cos(\psi/2)\, q_2}{\sin(\alpha_H/2)} \qquad (4.17)$$

We can now find the direction of the tilt axis β_H:

$$\beta_H = \arctan_2(r_y, r_x) \qquad (4.18)$$

With α_H, ψ, and β_H we are now have enough information to stabilize the quadrotor platform. We now turn in the next section to creating a PID control.

4.2 Attitude Stabilization

Attitude is controlled via a very simple PID loop. The main thing to note here is the use of β_H to weight the α_H term to the correct axes. The ω terms (angular rate) can be measured directly from onboard gyroscopes.

$$M_x = K_{PH}\alpha_H \cos\beta_H - K_{DH}\omega_x \tag{4.19}$$
$$M_y = K_{PH}\alpha_H \sin\beta_H - K_{DH}\omega_y \tag{4.20}$$
$$M_z = K_{Pz}\psi - K_{Dz}\omega_z \tag{4.21}$$
$$\tag{4.22}$$

4.3 Altitude Stabilization

The current altitude can be measured in any number of ways. Many quadrotors use an ultrasonic sensor mounted on the bottom of the platform to measure distance to the ground. Alternatively, a system could use an IR sensor, altimeter module, precision differential GPS, or any system that gives a single scalar number as the altitude. This section assumes that whatever measurement was used, the current altitude can be represented a scalar number.

To stabilize the altitude, we define the altitude error as

$$\tilde{\boldsymbol{R}}_{e_z} = \boldsymbol{R}_{e_z} - desired\left(\boldsymbol{R}_{e_z}\right) \tag{4.23}$$

Then we have a simple PID equation for the altitude:

$$F_{PIDz} = K_{Pz}\tilde{\boldsymbol{R}}_{e_z} + K_{Iz}\Sigma\tilde{\boldsymbol{R}}_{e_z} + K_{Dz}\dot{\tilde{\boldsymbol{R}}}_{e_z} \tag{4.24}$$

In order to obtain the final force along the body z axis, equation (4.24) must be augmented with information about the weight of the quadrotor and the angle the quadrotor is tilted at:

$$F_z = \frac{mg + F_{PIDz}}{K_{tilt}} \tag{4.25}$$

$$K_{tilt} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \cdot \left(q \otimes \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \otimes q^* \right) \tag{4.26}$$

With these equations, the more the body is tilted the F_z will become and the faster the quadrotor will move in the $x - y$ plane, without a mathematical bound. In reality, at some point the motors will become saturated (running at maximum speed) and the quadrotor will not be able to simultaneously maintain a commanded altitude and attitude.

4.4 Forces to Motor Speeds

In previous sections we have calculated the desired moments M_x, M_y, and M_z, and the desired force F_z. In this section we will take those moments and convert it to the desired rotation speed of each motor.

First, in equation (4.27) we have a standard conversion from rotational frequency to angular frequency .

$$\Omega_i = 2\pi n_i \tag{4.27}$$

Each motor generates a thrust in the direction of the motor shaft axis, and a torque around the motor shaft axis. The calculation of K_T and K_Q is discussed in subsection 5.1.

$$T = K_T \rho n^2 D^4 \tag{4.28}$$

$$Q = K_Q \rho n^2 D^5 \tag{4.29}$$

These two equations are used to define T and Q. The constants K_T and K_Q are determined experimentally. A motor test stand must be able to measure T, Q, ρ, n, and D, and generate the constants for a particular motor and propeller combination. Note the exponents in the equations: rotational speed and propeller diameter have a significant effect on the thrust and torque.

For use onboard the quadrotor, equation (4.28) will be used with equation (4.27) to connect thrust to rotational speed. This derivation is done in equation (4.30). By this combination we will be able to determine what rotational speed we need to run the motors at to achieve a specified thrust. From equations (4.28) and (4.27) we rewrite and combine:

$$\begin{aligned} n^2 &= \frac{T}{\rho K_T D^4} \\ n &= \frac{1}{D^2}\sqrt{\frac{T}{\rho K_T}} \\ \Omega_{di} &= 2\pi n \\ \Omega_{di} &= \frac{2\pi}{D^2}\sqrt{\frac{T}{\rho K_T}} \end{aligned} \tag{4.30}$$

With equation (4.30) we have an equation that will tie the thrust required to how fast the rotors must spin. This can later be fed into a PI controller for each motor, described in section 4.5 .

Finally, we need to determine what the thrust and torque of each motor for the given M and F_z:

$$\begin{bmatrix} F_z \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -d & 0 & d \\ -d & 0 & d & 0 \\ c & -c & c & -c \end{bmatrix} \cdot \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{4.31}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -d & 0 & d \\ -d & 0 & d & 0 \\ c & -c & c & -c \end{bmatrix}^{-1} \cdot \begin{bmatrix} F_z \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{4.32}$$

$$\tag{4.33}$$

where

$$c = \frac{K_Q D}{K_T} \tag{4.34}$$

Solving the matrix inversion and multiplication, we arrive at the final four equations:

$$F_1 = +\frac{M_z}{4c} - \frac{M_y}{2d} + \frac{F_z}{4} \tag{4.35}$$

$$F_2 = -\frac{M_z}{4c} - \frac{M_x}{2d} + \frac{F_z}{4} \tag{4.36}$$

$$F_3 = +\frac{M_z}{4c} + \frac{M_y}{2d} + \frac{F_z}{4} \tag{4.37}$$

$$F_4 = -\frac{M_z}{4c} + \frac{M_x}{2d} + \frac{F_z}{4} \tag{4.38}$$

$$\tag{4.39}$$

Now, we have the force (thrust) that each motor must produce in order to maintain the desired heading, and can use equation (4.30) to determine the desired rotor speed.

4.5 Motor control

This subsection describes the equations used to set the rotational speed of the motors. Note that this differs from [1]: in that paper the authors used an empirically measured function (f_{mot}) to estimate the rotational speed based on the voltage of the motor and the command given. This required a dynamically adjustable constant a_i to calibrate each motor for small differences in manufacturing. In this paper it is assumed that actual motor rotational speed is able to be measured. By measuring the rotational speed of the motors we are able to ignore the manufacturing differences between motors, and create a closed loop control. If a motor is not spinning at the correct speed the PI control loop described next will be able to correct until the desired rotation is achieved.

The following two equations are explicitly defined a discrete time t. For clarity, the subscript i denoting a specific motor is implied but not written:

$$\Omega\left(t\right)_{Ierror} = \Omega\left(t-1\right)_{Ierror} + \left(\Omega_d - \Omega_i\right) \tag{4.40}$$

$$u_i = K_P\left(\Omega_d - \Omega_i\right) + K_I \Omega\left(t\right)_{Ierror} \tag{4.41}$$

4.6 Quadrotor Motion Model

The equations in this section describe the motion of the quadrotor body through space. In essence, these equations are taken directly from any introductory physics textbook and simply expanded a small amount. These equations are necessary for translational motion, but are not required to simply hover

$$\dot{\boldsymbol{V}}_b = \frac{1}{m}\begin{bmatrix} 0 \\ 0 \\ F_z \end{bmatrix} + \boldsymbol{q}^* \otimes \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \otimes \boldsymbol{q} - \boldsymbol{\omega}_b \times \boldsymbol{V}_b \tag{4.42}$$

Equation (4.42) describes the body acceleration of the quadrotor. The equation follows the form of $F = ma$ from traditional physics (rewritten for $a = \frac{F}{m}$). The first accounts for the acceleration due to the force of the motors. The second term accounts for the acceleration due to gravity, rotated based on the vehicle orientation. The third term accounts for fictitious forces (forces due to the non-inertial reference frame of the body).

$$\dot{\boldsymbol{\omega}}_b = \boldsymbol{I}_{nb}^{-1}\left(\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} - \boldsymbol{\omega_b} \times \left(\boldsymbol{I}_{nb}\boldsymbol{\omega}_b\right)\right) \tag{4.43}$$

Equation (4.43) describes the angular acceleration of the quadrotor. The equation follows the form of $\tau = I\alpha$ from traditional physics (rewritten for $\alpha = \frac{\tau}{I}$. The first term (with the moments M_{xyz}) accounts for the rotational force of the propellers. The second term accounts for the fictitious forces.

$$\dot{\boldsymbol{R}}_e = \boldsymbol{q} \otimes \boldsymbol{V}_b \otimes \boldsymbol{q}^* \tag{4.44}$$

Equation (4.44) relates velocity in the body reference frame to velocity in the earth reference frame.

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes \boldsymbol{\omega}_b \tag{4.45}$$

Note that equation (4.45) describes the derivative of a quaternion.

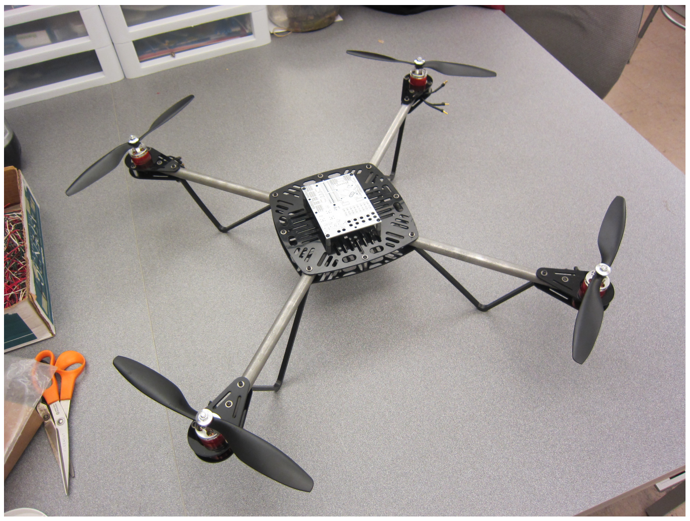

Figure 2: The Elev-8 quadrotor frame.

5 Implementation Details

This section describes all the equations necessary to have a stable hovering platform, and their order of implementation. It is broken down into three logical blocks: moment calculations (5.2), z axis force calculations (5.3), and motor speed calculations (5.4).

It is assumed that all math is done with floating point operations, and that the variables t_i are temporary variables that can be used to hold a value for a short duration of time.

5.1 Thrust and Torque Coefficient Measurements

In the equations to calculate the thrust (4.28) and torque (4.29) of each motor we have two constants, K_T and K_Q that need to be measured empirically. The two equations are reproduced below:

$$K_T = \frac{T}{\rho n^2 D^4} \tag{5.1}$$

$$K_Q = \frac{Q}{\rho n^2 D^5} \tag{5.2}$$

Our test stand must be able measure thrust, torque, rotational speed, rotor diameter, and air density.

5.2 Moment Calculations

This subsection calculates the moments M_x, M_y, and M_z from the current orientation $\boldsymbol{q}$ and the desired orientation $\boldsymbol{q}_d$. The full derivation can be found in sections 4.1 and 4.2.

Input variables:	$\boldsymbol{\omega}_b$	angular velocity in the body frame (rad/s)
	$\mathbf{q}$	attitude quaternion
	$\mathbf{q}_d$	desired quaternion

Input constants:	K_{PH}	tilt moment proportional constant
	K_{DH}	tilt moment derivative constant
	K_{Pz}	yaw moment proportional constant
	K_{Dz}	yaw moment derivative constant

Output:	$M_x - M_z$	total rotor moments along each axis $(N \cdot m)$

$$\tilde{\boldsymbol{q}} = \boldsymbol{q}_d \otimes \boldsymbol{q}^* \tag{5.3}$$

$$\alpha = 2 \arccos\left(\tilde{q_0}\right) \tag{5.4}$$

$$t_1 = \sin \alpha/2 \tag{5.5}$$

$$\boldsymbol{r}_{e_1} = \frac{\tilde{q_1}}{t_1} \tag{5.6}$$

$$\boldsymbol{r}_{e_2} = \frac{\tilde{q_2}}{t_1} \tag{5.7}$$

$$\boldsymbol{r}_{e_3} = \frac{\tilde{q_3}}{t_1} \tag{5.8}$$

$$\boldsymbol{r}_b = \boldsymbol{q}^* \otimes \boldsymbol{r}_e \otimes \boldsymbol{q} \tag{5.9}$$

$$\tilde{\boldsymbol{q}}_b = \begin{bmatrix} \cos\left(\alpha/2\right) \\ \sin\left(\alpha/2\right) \cdot \boldsymbol{r}_b \end{bmatrix} \tag{5.10}$$

$$\alpha_H = \arccos\left[1 - 2\left(q_1^2 + q_2^2\right)\right] \tag{5.11}$$

$$\psi = 2 \arctan_2\left(q_3, q_0\right) \tag{5.12}$$

$$t_1 = \cos\left(\psi/2\right) \tag{5.13}$$

$$t_2 = \sin\left(\psi/2\right) \tag{5.14}$$

$$t_3 = \sin\left(\alpha_H/2\right) \tag{5.15}$$

$$r_x = \frac{t_1 q_1 - t_2 q_2}{t_3} \tag{5.16}$$

$$r_y = \frac{t_2 q_1 - t_1 q_2}{t_3} \tag{5.17}$$

$$\beta_H = \arctan_2\left(r_y, r_x\right) \tag{5.18}$$

$$M_x = K_{PH} \alpha_H \cos \beta_H - K_{DH} \omega_x \tag{5.19}$$

$$M_y = K_{PH} \alpha_H \sin \beta_H - K_{DH} \omega_y \tag{5.20}$$

$$M_z = K_{Pz}\psi - K_{Dz}\omega_z \tag{5.21}$$

5.3 Z Axis Force Calculations

This subsection calculates the necessary total thrust along the quadrotor z axis. This section takes the desired altitude and the current orientation and outputs the resultant F_z necessary. The relevant equation derivations can be found in section 4.3.

Input variables:	$\mathbf{q}$	attitude quaternion
	$\boldsymbol{R}_{e_z}$	position in the inertial frame (m)
	F_z	total force of the rotors on the z axis (N)
	$desired\,(\boldsymbol{R}_{e_z})$	Desired z position (altitude) (m)

Input Constants:	m	quadrotor mass $(kg\ ?)$
	g	local gravity (m/s^2)
	K_{P_z}	Proportional PID gain for F_z
	K_{I_z}	Integral PID gain for F_z
	K_{I_z}	Derivative PID gain for F_z

Output Variables:	F_z	total force of the rotors on the z axis (N)

$$K_{tilt} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \cdot \left(q \otimes \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \otimes q^* \right) \tag{5.22}$$

$$\tilde{\boldsymbol{R}}_{e_z} = \boldsymbol{R}_{e_z} - desired\,(\boldsymbol{R}_{e_z}) \tag{5.23}$$

$$F_{PIDz} = K_{Pz}\tilde{\boldsymbol{R}}_{e_z} + K_{Iz}\Sigma\tilde{\boldsymbol{R}}_{e_z} + K_{Dz}\dot{\tilde{\boldsymbol{R}}}_{e_z} \tag{5.24}$$

$$F_z = \frac{mg + F_{PIDz}}{K_{tilt}} \tag{5.25}$$

5.4 Motor Speed Calculations

This subsection calculates the output command (i.e. motor speed) for each of the motors. The derivations for these equations can be found in sections 4.4 and 4.5.

Input variables:	F_z	total force of the rotors on the z axis (N)
	M_x, M_y, M_z	total rotor moments along each axis $(N \cdot m)$
	n_{imeas}	rotation frequency of the rotors (Hz)

Input Constants:	D	rotor diameter (m)
	d	offset of each rotor from the center of mass (m)
	ρ	air density (kg/m^3)
	K_t	thrust coefficient
	K_q	torque coefficient
	K_{P_i}	Motor Proportional PID gain (for $1 \leq i \leq 4$)
	K_{I_i}	Motor Integral PID gain (for $1 \leq i \leq 4$)
	π	pi

Output Variables:	u_i	motor command

$$c = \frac{K_Q D}{K_T} \tag{5.26}$$

$$t_1 = \frac{M_z}{4c} \tag{5.27}$$

$$t_2 = \frac{M_y}{2d} \tag{5.28}$$

$$t_3 = \frac{M_x}{2d} \tag{5.29}$$

$$t_4 = \frac{F_z}{4} \tag{5.30}$$

$$F_1 = +t_1 - t_2 + t_4 \tag{5.31}$$

$$F_2 = -t_1 - t_3 + t_4 \tag{5.32}$$

$$F_3 = +t_1 + t_2 + t_4 \tag{5.33}$$

$$F_4 = -t_1 + t_3 + t_4 \tag{5.34}$$

$$t_1 = \frac{2\pi}{D^2} \tag{5.35}$$

$$t_2 = \rho K_T \tag{5.36}$$

$$\Omega_{d1} = t_1 \sqrt{F_1/t_2} \tag{5.37}$$

$$\Omega_{d2} = t_1 \sqrt{F_2/t_2} \tag{5.38}$$

$$\Omega_{d3} = t_1 \sqrt{F_3/t_2} \tag{5.39}$$

$$\Omega_{d4} = t_1 \sqrt{F_4/t_2} \tag{5.40}$$

$$\Omega\left(t\right)_{Ierror1} = \Omega\left(t-1\right)_{Ierror1} + \left(\Omega_{d1} - \Omega_{i1}\right) \tag{5.41}$$

$$\Omega\left(t\right)_{Ierror2} = \Omega\left(t-1\right)_{Ierror2} + \left(\Omega_{d2} - \Omega_{i2}\right) \tag{5.42}$$
$$\Omega\left(t\right)_{Ierror3} = \Omega\left(t-1\right)_{Ierror3} + \left(\Omega_{d3} - \Omega_{i3}\right) \tag{5.43}$$
$$\Omega\left(t\right)_{Ierror4} = \Omega\left(t-1\right)_{Ierror4} + \left(\Omega_{d4} - \Omega_{i4}\right) \tag{5.44}$$
$$u_1 = K_{P1}\left(\Omega_{d1} - \Omega_{i1}\right) + K_{I1}\Omega\left(t\right)_{Ierror1} \tag{5.45}$$
$$u_2 = K_{P2}\left(\Omega_{d2} - \Omega_{i2}\right) + K_{I2}\Omega\left(t\right)_{Ierror2} \tag{5.46}$$
$$u_3 = K_{P3}\left(\Omega_{d3} - \Omega_{i3}\right) + K_{I3}\Omega\left(t\right)_{Ierror3} \tag{5.47}$$
$$u_4 = K_{P4}\left(\Omega_{d4} - \Omega_{i4}\right) + K_{I4}\Omega\left(t\right)_{Ierror4} \tag{5.48}$$

From here, we are able to send the command u_i to the motor ESCs via a standard PWM signal. Note that the u_i will have to be adjusted to fit within the standard servo type PWM signal standard. Typically, this means that the signal is set so that a motor speed of 0 corresponds to 1000uS, and a full motor speed corresponds to 2000uS, with a pulse every 20mS (approximately).

6 Basic Equations from [1]

The follow equations are exactly as written in [1]. The equations in previous sections follow the general guidelines of these equations, but some exceptions were made. They are here for reference only.

6.1 Quadrotor Model

$$\dot{\boldsymbol{V}}_b = \frac{1}{m}\begin{bmatrix} 0 \\ 0 \\ F_z \end{bmatrix} + \boldsymbol{q}^* \otimes \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \otimes \boldsymbol{q} - \boldsymbol{\omega}_b \times \boldsymbol{V}_b \tag{6.1}$$

$$\dot{\boldsymbol{\omega}}_b = \boldsymbol{I}_{nb}^{-1}\left(\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} - \boldsymbol{\omega_b} \times \left(\boldsymbol{I}_{nb}\boldsymbol{\omega}_b\right)\right) \tag{6.2}$$

$$\dot{\boldsymbol{R}}_e = \boldsymbol{q} \otimes \boldsymbol{V}_b \otimes \boldsymbol{q}^* \tag{6.3}$$

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes \boldsymbol{\omega}_b \tag{6.4}$$

$$K_T = \frac{T}{\rho n^2 D^4} \tag{6.5}$$

$$K_Q = \frac{Q}{\rho n^2 D^5} \tag{6.6}$$

$$\Omega_i = 2\pi n_i \tag{6.7}$$

$$\tau\dot{\Omega}_i + \Omega_i = a_i f_{mot}\left(u_i, V_{batt}\right) \tag{6.8}$$

$$\begin{bmatrix} F_z \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -d & 0 & d \\ -d & 0 & d & 0 \\ c & -c & c & -c \end{bmatrix} \cdot \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{6.9}$$

where

$$c = \frac{K_Q D}{K_T} \tag{6.10}$$

$$\Omega_{di} = \frac{2\pi}{D^2}\sqrt{\frac{-F_{di}}{K_T \rho}} \tag{6.11}$$

$$u_i = f_{mot}^{-1}\left(\frac{1}{a_i}\Omega_{di}, V_{batt}\right) \tag{6.12}$$

6.2 Motor Gain Estimation

$$\dot{a}_i = K_g\left(\Omega_{imeas} - \hat{\Omega}_i\right) \tag{6.13}$$

6.3 Yaw and Tilt Errors

$$\boldsymbol{q}_d = \tilde{\boldsymbol{q}} \otimes \boldsymbol{q} \tag{6.14}$$

$$\tilde{\boldsymbol{q}} = \boldsymbol{q}_d \otimes \boldsymbol{q}^* \tag{6.15}$$

$$\tilde{\boldsymbol{q}} = \begin{bmatrix} \cos\left(\alpha/2\right) \\ \sin\left(\alpha/2\right) \cdot \boldsymbol{r}_e \end{bmatrix} \tag{6.16}$$

$$\boldsymbol{r}_b = q^* \otimes \boldsymbol{r}_e \otimes \boldsymbol{q} \tag{6.17}$$

$$\tilde{\boldsymbol{q}}_b = \begin{bmatrix} \cos\left(\alpha/2\right) \\ \sin\left(\alpha/2\right) \cdot \boldsymbol{r}_b \end{bmatrix} \tag{6.18}$$

$$\tilde{\boldsymbol{q}}_b = \boldsymbol{q}_H \otimes \boldsymbol{q}_V \tag{6.19}$$

$$\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos\left(\alpha_H/2\right) \\ \sin\left(\alpha_H/2\right) \cdot r_x \\ \sin\left(\alpha_H/2\right) \cdot r_y \\ \sin\left(\alpha_H/2\right) \cdot 0 \end{bmatrix} \otimes \begin{bmatrix} \cos\left(\psi/2\right) \\ \sin\left(\psi/2\right) \cdot 0 \\ \sin\left(\psi/2\right) \cdot 0 \\ \sin\left(\psi/2\right) \cdot 1 \end{bmatrix} \tag{6.20}$$

$$\alpha_H = \arccos\left[1 - 2\left(q_1^2 + q_2^2\right)\right] \qquad 0 \leq \alpha_H < \pi \tag{6.21}$$

$$\psi = 2\arctan_2\left(q_3, q_0\right) \qquad -\pi \leq \psi < \pi \tag{6.22}$$

$$r_x = \frac{\cos(\psi/2)\, q_1 - \sin(\psi/2)\, q_2}{\sin(\alpha_H/2)} \tag{6.23}$$

$$r_y = \frac{\sin(\psi/2)\, q_1 - \cos(\psi/2)\, q_2}{\sin(\alpha_H/2)} \tag{6.24}$$

$$\beta_H = \mathrm{arctan}_2\,(r_y, r_x) \tag{6.25}$$

$$\gamma_H = \mathrm{arctan}_2\,(r_x, r_y) \tag{6.26}$$

6.4 Attitude stabilization

$$\begin{align} M_x &= K_{PH}\alpha_H \cos\beta_H - K_{DH}\omega_x \tag{6.27}\\ M_y &= K_{PH}\alpha_H \sin\beta_H - K_{DH}\omega_y \tag{6.28}\\ M_z &= K_{Pz}\psi - K_{Dz}\omega_z \tag{6.29}\\ & \tag{6.30} \end{align}$$

6.5 Altitude Control

$$F_z = \frac{mg + F_{PIDz}}{K_{tilt}} \tag{6.31}$$

$$K_{tilt} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \cdot \left(q \otimes \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \otimes q^* \right) \tag{6.32}$$

References

[1] E. Stingu, F. Lewis, "Design and Implementation of a Structured Flight Controller for a 6DoF Quadrotor Using Quaternions," Mediterranean Conference on Control & Automation, pp. 1233-1238, June 2009.

[2] M. Malek, "Vector and Matrix Norms", California State University, East Bay, May 2009.

[3] K. Shoemake, "Quaternions", Department of Computer and Information Science, University of Pennsylvania.