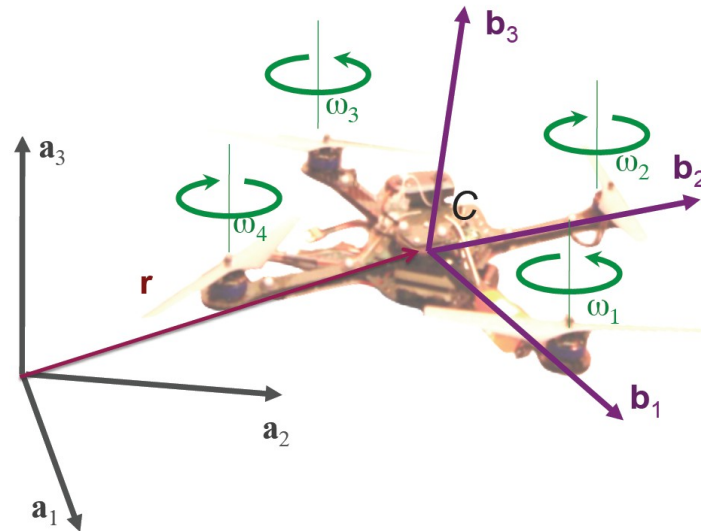


## 3-D Quadrotor Control

Having looked at the 2-dimensional quadrotor, we now turn our attention to the dynamics of the 3-dimensional model, and we'll talk about trajectory planning and control.

Recall that we have a body-fixed frame attached to the quadrotor and we have an inertial frame:



The state of the quadrotor consists of position and orientation. The position is the position-vector of the centre-of-mass, and the roll, pitch and yaw angles tell us the orientation. This state is composed of a six-dimensional vector and its rate of change (in other words, the velocity).

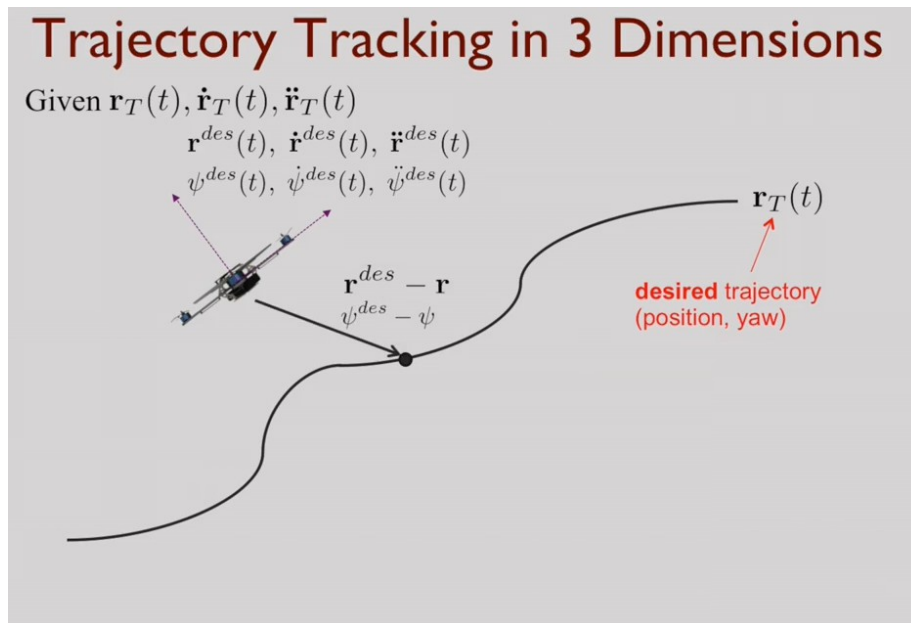
$$q = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \varphi \end{bmatrix}, \dot{q} = \begin{bmatrix} \dot{q} \\ \dot{q} \end{bmatrix}$$

The roll, pitch and yaw angles follow the usual convention: first the yaw above the z-axis, and then the roll & pitch as we've seen before. The angular velocity components in the body frame,  $p$ ,  $q$  and  $r$ , are related to the derivatives of the roll, pitch and yaw angles through this coefficient matrix:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & \sin \theta \\ \sin \theta & 0 & \cos \theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\varphi} \end{bmatrix}$$

This set of equations tells us everything we need to know about the kinematics of the vehicle.

What we'd like to do is to track arbitrarily-specified trajectories in three-dimensions.



We assume we're given a trajectory. This trajectory consists of a position vector that varies as a function of time, and a yaw angle, which also varies as a function of time:

$$r_T(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \\ \varphi(t) \end{bmatrix}$$

We want this four-dimensional vector to be differentiable, and we want to be able to obtain not only its derivative, but also its second derivative.

As before, we're interested in the difference between the desired position and the actual position. But now we're also interested in the difference between the desired yaw angle and the actual yaw angle. That gives us the error vector and its derivative:

$$e_p = r_T(t) - r$$

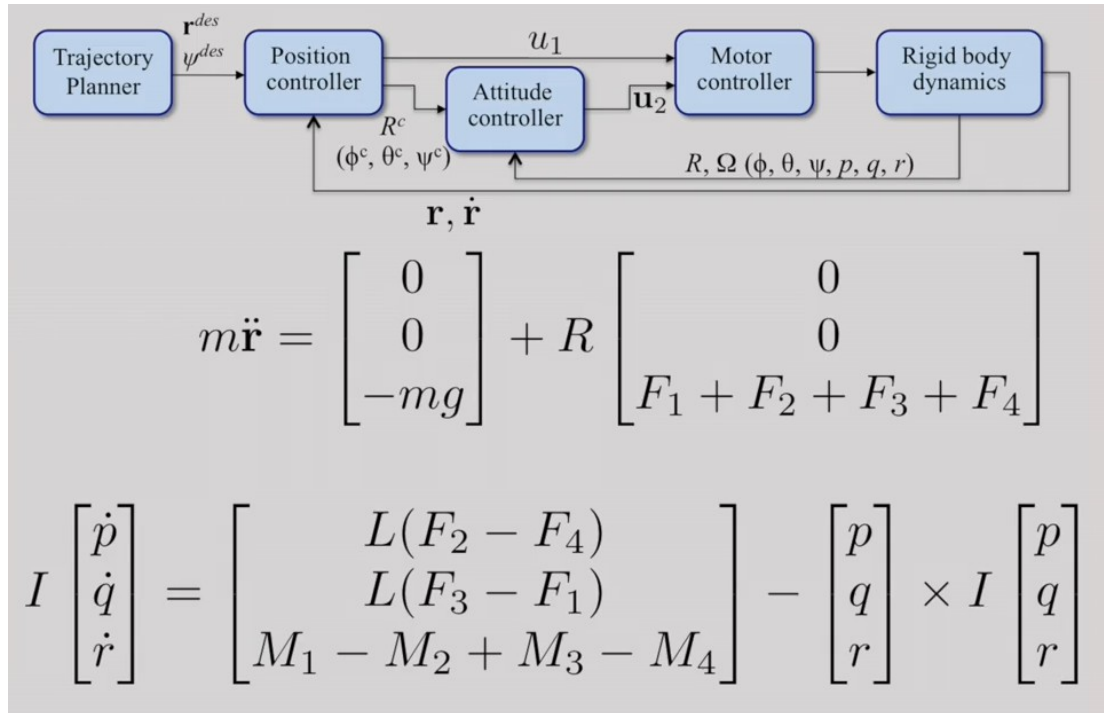
$$e_v = \dot{r}_T(t) - \dot{r}$$

And again, we want the error vector to go exponentially to zero.

$$(\dot{r}_T(t) - \ddot{r}_c) + K_d e_v + K_p e_p = 0$$

This lets us calculate the commanded acceleration,  $\ddot{r}_c$ , whether it's the 2nd-derivative of the position vector, or the 2nd-derivative of the yaw angle.

Let's take another look at the equations of motion, and the nested-feedback-loop we described for the two-dimensional quadrotor model:



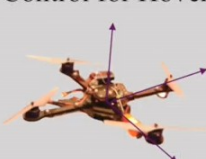
As in the planar case, we have nested feedback loops. The inner loop corresponds to attitude control, and the outer loop corresponds to position control. In the inner loop, we specify the orientation either using a rotation matrix or a series of roll, pitch, and yaw angles. We'll feedback the actual attitude and angular velocity, or the roll, pitch and yaw angles and the angular rates. From that, we calculate the input  $u_2$ .  $u_2$  is a function of the thrusts and moments that we get from the motors.

In the equations-of-motion at the bottom, we have to calculate the value of  $u_2$  based on the desired attitude.

We now turn our attention to the outer-loop, which is a position feedback loop. In this loop, we take the specified position vector & specified yaw angle from the Trajectory Planner. We compare that with the actual position and velocity, and from that we calculate  $u_1$ ,  $u_1$  is essentially the sum of all the thrust forces.

Rather than looking at the general trajectory-following problem, let's initially focus on a very specific case, the case of hovering:

Control for Hovering



$$m\ddot{\mathbf{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix}$$

$$I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

So in hovering, the robot's position and orientation are fixed. In other words, all velocities are zero. Further, the roll and pitch angles are also equal to zero. We want to consider small perturbations around the hover position. Accordingly, we'll linearise the dynamics around this current configuration.

In order for the hover position to be one of equilibrium, the input  $u_2$  has to be 0, the angular velocity components  $p$ ,  $q$ , and  $r$  are equal to 0, and their derivatives are also equal to 0. Likewise, the sum of the thrusts has to compensate for the weight of the robot, but the yaw angle can be non-zero as long as it's fixed.

Linearising around this point, means that we are assuming  $u_1$  is almost equal to  $mg$ . We assume the roll and pitch angles are close to zero, and we assume that the yaw angle is fixed, or close to fixed at a given value  $\varphi_0$ .

$$(u_1 \sim mg, \theta \sim 0, \phi \sim 0, \varphi \sim \varphi_0)$$

When we linearise the equations, we end up with these simplified equations:

$$\ddot{r}_1 = \ddot{x} = g(\theta \cos \varphi + \phi \sin \varphi)$$

$$\ddot{r}_2 = \ddot{y} = g(\theta \sin \varphi - \phi \cos \varphi)$$

We can now design the inner feedback loop, by simply specifying  $u_2$  using a proportional plus derivative controller:

$$u_2 = \begin{bmatrix} K_{p,\phi}(\phi_c - \phi) + K_{d,\phi}(p_c - p) \\ K_{p,\theta}(\theta_c - \theta) + K_{d,\theta}(q_c - q) \\ K_{p,\varphi}(\varphi_c - \varphi) + K_{d,\varphi}(r_c - r) \end{bmatrix}$$

Here we assume that we have the commanded roll, pitch, and yaw angles and their derivatives. And all we require for feedback are the actual roll, pitch, and yaw angles, and their derivatives.

Now, let's consider the outer feedback loop. If we linearise the equations-of-motion, the expressions for the first two components of acceleration can be written in the form shown below:

$$\ddot{r}_1 = \ddot{x} = g(\theta \cos \varphi + \phi \sin \varphi)$$

$$\ddot{r}_2 = \ddot{y} = g(\theta \sin \varphi - \phi \cos \varphi)$$

We can now turn our attention to the error equation describing the error in position, using these linearised equations of motion. Remember, we want this error to satisfy the second-order differential equation:

$$(\ddot{r}_{i,des} - \ddot{r}_{i,c}) + K_{d,i}(\dot{r}_{i,des} - \dot{r}_i) + K_{p,i}(r_{i,des} - r_i) = 0$$

This equation contains terms to do with the desired trajectory,  $r_{i,des}$ , and terms to do with the actual trajectory being followed,  $r_i$ . We will use this equation to calculate the commanded acceleration,  $\ddot{r}_{i,c}$ . This commanded acceleration will, in turn, tell us what thrust we need to apply with the rotors.

$$u_1 = m(g + \ddot{r}_{3,c})$$

Finally, we need to determine the commanded roll, pitch, and yaw using the linearised equations. If we know that the commanded acceleration needs to be in the X and Y direction, we can calculate the roll and pitch angles and their derivatives.

$$\phi_c = \frac{1}{g}(\ddot{r}_{1,c} \sin \varphi_{des} - \ddot{r}_{2,c} \cos \varphi_{des})$$

$$\theta_c = \frac{1}{g}(\ddot{r}_{1,c} \cos \varphi_{des} + \ddot{r}_{2,c} \sin \varphi_{des})$$

We can determine the commanded yaw angles in a similar manner.

$$\varphi_c = \varphi^{des}$$