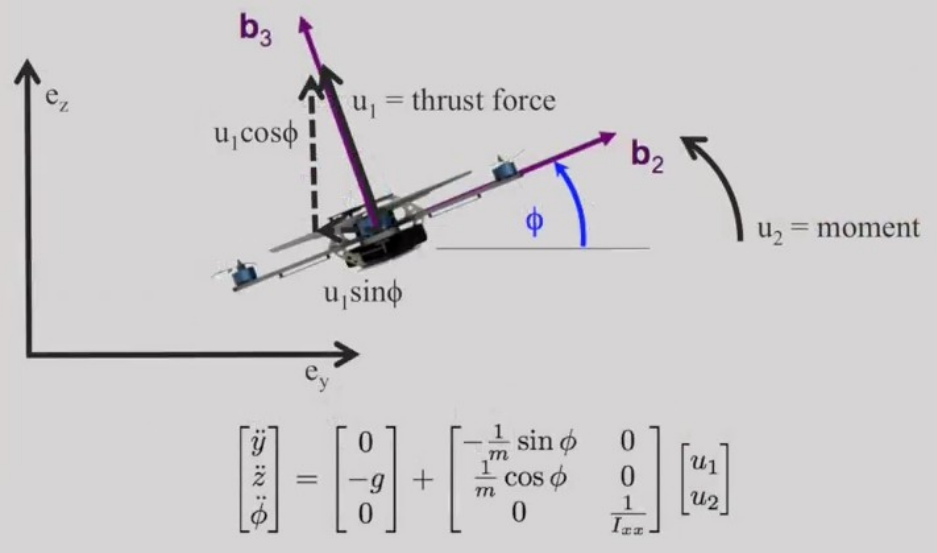
Lecture 3A.1

2-D Quadrotor Control

Now that we understand how to write the equations of motion for a quadrotor, let's turn our attention to the model for a planar quadrotor. We saw the equations of motion for the planar quadrotor earlier:



These can now be rewritten in state-space form as shown here:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y \\ z \\ \phi \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \end{bmatrix}$$

And the derivative is given by:

$$\dot{x} = \begin{bmatrix} \dot{y} \\ \dot{z} \\ \dot{\phi} \\ 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -\frac{1}{m}\sin\phi & 0 \\ \frac{1}{m}\cos\phi & 0 \\ 0 & \frac{1}{I_{xx}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

The first three elements of this derivative are velocities. The bottom three elements are accelerations. This is just a compact way of writing the equations of motion. On the right hand side, we have two inputs, $\mathbf{u_1}$ and $\mathbf{u_2}$. This vector is the input that will

drive our dynamical system. By specifying appropriate values for $\mathbf{u_1}$ and $\mathbf{u_2}$, we can change the state of the quadrotor and get it to go where we want it to go.

We can see that the equations of motion, non-linear:

$$\ddot{y} = -\frac{u_1}{m}\sin(\phi)$$

$$\ddot{z} = -g + \frac{u_1}{m}\cos(\phi)$$

$$\ddot{\phi} = -\frac{u_2}{I_{xx}}$$

The non-linearity here comes only from the fact that we have sin(ϕ) and cos(ϕ) in the equations. Let's consider these equations of motion at a hover configuration.

We define a hover configuration as:

$$y_0, z_0, \phi y_0 = 0, u_{0,1} = mg, u_{2,0} = 0$$

The hover configuration can only be in equilibrium if the thrust vector, u_1, opposes the gravity vector, so, at hover, the nominal value for this input must equal mg. Further, the nominal value of u_2 must be zero. If it were not, then the vehicle would try to accelerate.

Next we obtain a linearised version of this nonlinear model by considering how the non-linear functions sin(ϕ) and cos(ϕ) behave in close proximity to the hover configuration. When ϕ is close to 0, sin(ϕ) behaves roughly like ϕ and cos(ϕ) is roughly constant. If we replace sin(ϕ) with ϕ, and cos(ϕ) with 1, we get the linearised equations below:

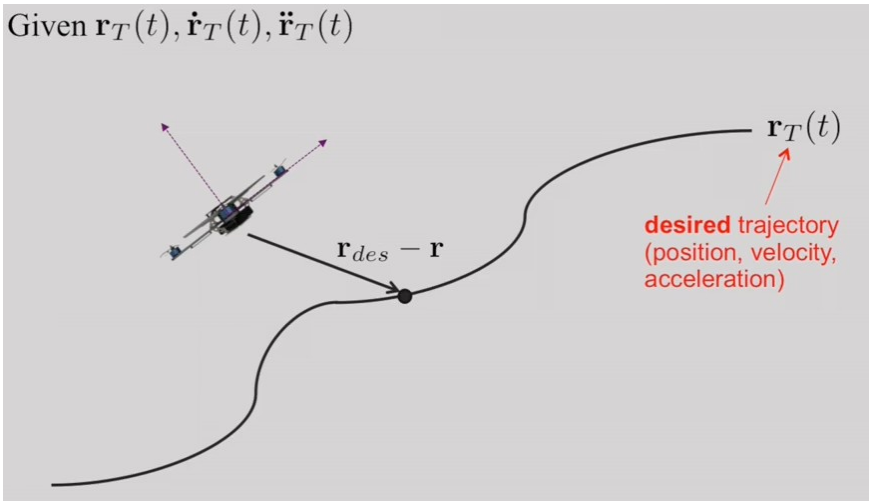
$$\ddot{y} = -\frac{u_1}{m}\phi$$

$$\ddot{z} = -g + \frac{u_1}{m}$$

$$\ddot{\phi} = -\frac{u_2}{I_{xx}}$$

We call this the linearised dynamic model.

Let's now look at the control problem. Imagine a planar quadrotor trying to follow a specified trajectory, $\mathbf{r_T}(t)$.



$\mathbf{r_T}(t)$ denotes a position vector. The fact that it's changing in time is denoted by the function $\mathbf{r_T}(t)$ as a function of t:

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

Of course, this means that both y and z are changing as functions of time. Let's also assume that this function is differentiable. That is, we can differentiate it to calculate the velocity and differentiate again to calculate the acceleration.

We're interested in taking the vector, $\mathbf{r_{des}} - \mathbf{r}$, and reducing it to 0. $\mathbf{r_{des}}$ being the point on the trajectory that we want to follow, and $\mathbf{r}$ being the robot's current point. Let's define the error vector as $\mathbf{r_{des}} - \mathbf{r}$, $\mathbf{r_{des}}$ being the same as $\mathbf{r_T}(t)$, so the position error is:

$$e_p = \mathrm{r_T(t)} - \mathrm{r}$$

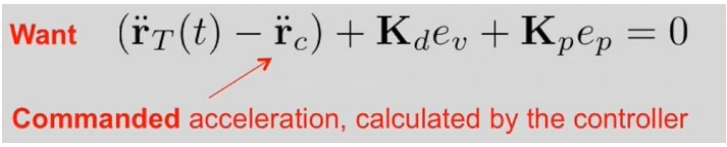
The velocity error is given by the derivative of the position error:

$$e_v = \dot{\mathrm{r}}_\mathrm{T}(\mathrm{t}) - \dot{\mathrm{r}}$$

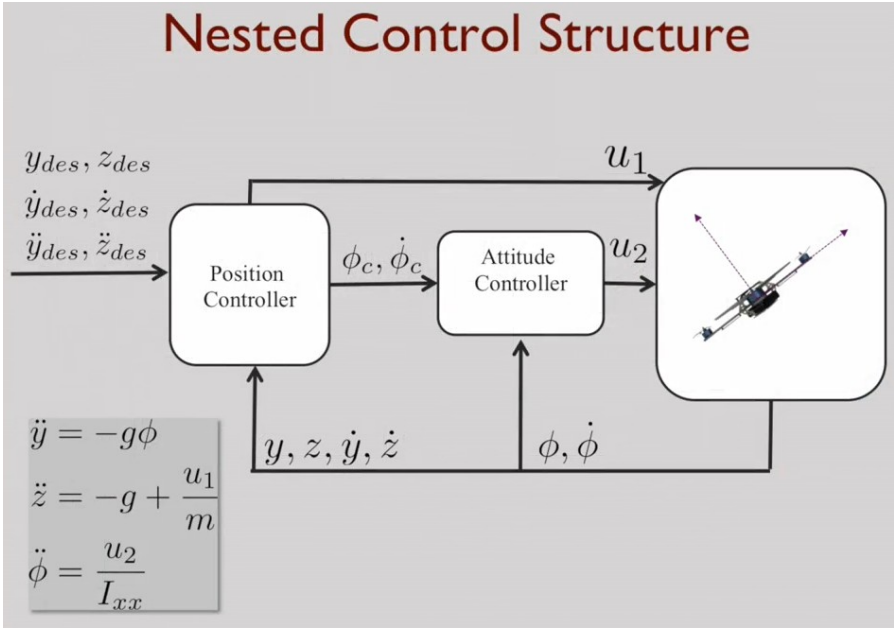
And what we really want is for this error to decay exponentially to 0:

$$\ddot{\mathrm{r}}_\mathrm{T}(\mathrm{t}) - \ddot{\mathrm{r}} + K_d e_v + K_p e_p = 0$$

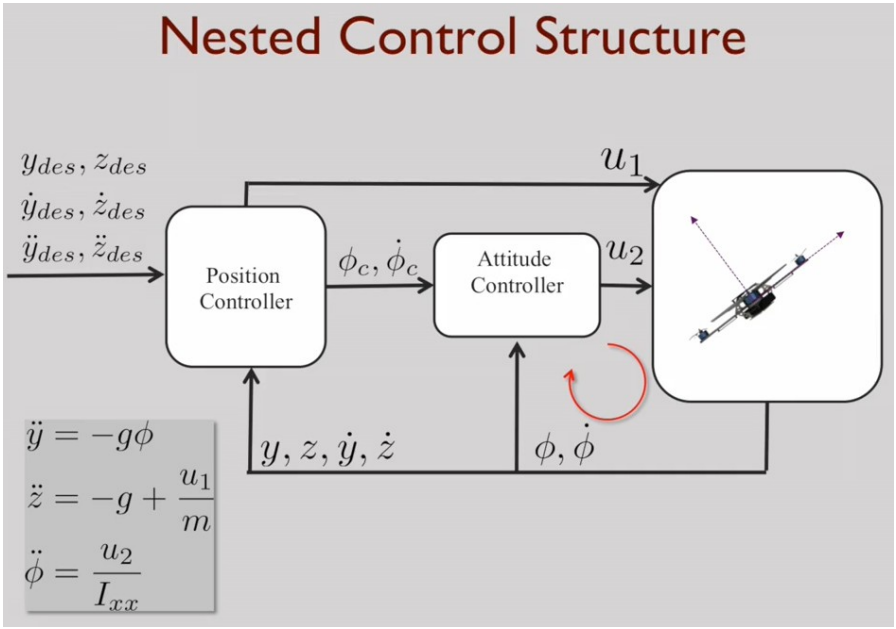
As we've seen before, if we take the error term and make it obey a second-order linear differential equation with positive coefficients, we will guarantee that the error goes exponentially to 0. So we command an acceleration which forces this error term to obey this second-order differential equation



This idea can be extended to develop a hierarchical control structure:



Before we consider the block diagram in its entirety, let's focus on the inner loop, which is the attitude controller.



Let's imagine that we're just trying to control the attitude of the robot. In other words, we're trying to control the roll angle, ϕ, to have it follow a specified trajectory. What that means is, we want to synthesize u_2, such that ϕ goes to some desired value. That desired value is shown as the reference input, ϕ_c. Of course, as before, we assume that ϕ_c is differentiable, so we can calculate the derivative of ϕ_c, and likewise, calculate the derivative of ϕ.

The equations of motion are shown on the bottom left of the figure. The last equation essentially gives the relationship between u_2 and $\ddot{\phi}$. This is a linear second-order differential equation in one variable, and it's not too hard to determine u_2 such that ϕ converges to ϕ_c.

Let's assume we can do that and move onto the next sub-problem. We want to follow a specified trajectory. Let's focus on $\mathbf{y_{des}}$, the y component of that trajectory. If we rearrange the top equation of the equations of motion, we can set the roll angle ϕ to be given by:

$$\phi_c = -\frac{\ddot{y}}{g}$$

This effectively ensures that the roll angle will track the desired acceleration in the y-direction.

In summary, the inner loop is designed so that ϕ converges to ϕ_c, and ϕ_c is obtained from the y-component of the desired trajectory. Given ϕ_c we can write an expression for u_2:

$$u_2 = K_{p,\phi}(\phi_c - \phi) + K_{d,\phi}(\dot{\phi}_c - \dot{\phi})$$

Finally, the z-component tells us how to calculate u_1. We take the middle equation, of the equations of motion and the desired trajectory, looking specifically at the z-component, and we can write u_1 such that the z-component tracks the desired z-value.

$$u_1 = m(g + \ddot{z}_{des} + K_{d,z}(\dot{z}_{des} - \dot{z}) + K_{p,z}(z_{des} - z))$$

So we now have two equations, one for u_1 and one for u_2. In fact, when we put everything together, we actually have three equations:

$$u_1 = m(g + \ddot{z}_{des} + K_{d,z}(\dot{z}_{des} - \dot{z}) + K_{p,z}(z_{des} - z))$$

$$u_2 = K_{p,\phi}(\phi_c - \phi) + K_{d,\phi}(\dot{\phi}_c - \dot{\phi})$$

$$\phi_c = -\frac{1}{g}(\ddot{y}_{des} + K_{d,y}(\dot{y}_{des} - \dot{y}) + K_{p,y}(y_{des} - y))$$

The one at the bottom tells us what ϕ_c is. From that, we get u_2, and then u_1 from that. These are the three control equations we can use to drive a quadrotor, assuming that our assumption that it's close to the operating point is valid, and assuming that it stays on the y-z plane.

The only thing left to do is to calculate the constants. These are the gains, and there are actually six of these that need to be calculated.