Design of Hovering Rocket PID Controller

for the kOS community by BriarAndRye
v 1.0 | June 28, 2015

1 INTRODUCTION

This tutorial covers the analysis of a hovering rocket for the purpose of designing a PID controller. It is intended to walk the user through the basics of designing a PID controller for a physical system. Concepts include using feed-forward terms to condition output and improve disturbance rejection, second-order system response, actuator saturation, and anti-windup techniques. Once completed, the user should be able to derive the equations governing the motion of the rocket, choose an appropriate controller type, design a feed-forward term to condition the controller output, and select gains to achieve a desirable response.

Required Concepts

This tutorial only requires concepts which should be covered by a high-school physics class, including: algebra, basic geometry/trigonometry, and $F = ma$.

2 PHYSICAL MODEL

Consider a rocket with mass, m, producing an amount of thrust, T, and tilted at an angle of α from the up vector. We are interested in how the rocket's height above the surface, z, will change over time. We know from Newton's laws that the force, F, on an object determines it's acceleration, a, ($F = ma$). Because we are specifically interested in only the vertical component we can simplify the math to $F_z = ma_z$.

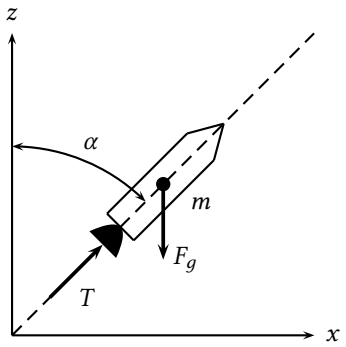

Figure 1: Forces acting on hovering rocket.

In the considered rocket we assume that it is moving slow enough that air resistance is negligible. This leaves two forces on the rocket: gravity, and thrust. The force of gravity is $F_g = mg$ where g is the local gravitational acceleration. The force due to the thrust is more difficult since the rocket is tilted. However, we know that the force in the z direction is $T_z = T\cos(\alpha)$. Combining with the force of gravity, we find the total force along the z-axis:

$$F_z = T\cos(\alpha) - mg. \tag{1}$$

Finally, we know that the acceleration of the rocket in the z direction is the second derivative of the position z. This gives us the differential equation governing the rocket's height:

$$m\frac{d^2}{dt^2}z = T\cos(\alpha) - mg \tag{2}$$

This not only tells us that we can control the rocket's height by varying the thrust, but it gives us insight into *how* we can control it. For instance, we could control the rocket's height by changing its thrust, the angle α, or both. To keep things simple, let's only use the thrust.

3 CONTROL DESCRIPTION

The goal is to control the rocket's height such that is tracks a setpoint, z^*, by modifying the thrust it produces through the throttle setting. To accomplish this we are going to use a PID controller. In order to design the controller, we will utilize the Laplace transform so that we can work in the frequency domain which only requires algebra. It may sound intimidating, but all you need to know is that derivatives are represented by s, and integrals are represented by $1/s$.

A PID controller operates on a error signal to produce a control signal. For us, the error is $z^* - z$, and the control signal is T. This results in

$$T = \left(K_p + \frac{K_i}{s} + K_d s\right)(z^* - z) \tag{3}$$

where K_p is the proportional gain, K_i is the integral gain, and K_d is the derivative gain.

P Controller

For a proportional only controller we set $K_i = K_d = 0$. This makes our thrust signal to be

$$T = K_p (z^* - z) . \tag{4}$$

If we substitute this into our differential equation and convert all derivatives to s we get

$$ms^2 z = K_p(z^* - z)\cos(\alpha) - mg \tag{5}$$

which describes the dynamics of the rocket with the controller included.

One neat feature of the frequency domain is that if we set $s = 0$, we find out what the final state of our system will be. Doing so and rearranging gives

$$K_p z = K_p z^* - \frac{mg}{\cos(\alpha)}. \tag{6}$$

Hmm, the goal of our controller was to make $z = z^*$, but this isn't happening. This is what's known as steady-state error. It's possible to get rid of steady-state error by including the integral control, but let's try something first.

What happens if we replace (4) with

$$T = \frac{K_p (z^* - z) + mg}{\cos(\alpha)}? \tag{7}$$

The $\cos(\alpha)$ and mg terms will cancel with those in (5) resulting in

$$ms^2 z = K_p(z^* - z). \tag{8}$$

Notice that our dynamics are much improved. The effects of the pitch and gravity have been compensated for. This isn't just algebraic voodoo, but we can easily measure α, m, and g for our rocket and modify the output of our control as in (7). This is known as feed-forward and is used to condition the output of PID controllers to improve its response.

Let's look at our steady-state response now by setting $s = 0$.

$$K_p z = K_p z^* \tag{9}$$

Now that's much better. According to this there will be no steady-state error in our rocket's height. Let's look at how the controller behaves on a 1 ton rocket and a $K_p = 1000$ when we change z^* by 1 meter shown in Figure 2.

Hmm, it appears that our controller results in a rocket that will oscillate up and down. Perhaps we can look at the dynamics to get some insight. If we rearrange (8) we can find the transfer function of the controlled system:

$$\frac{z}{z^*} = \frac{\frac{K_p}{m}}{s^2 + \frac{K_p}{m}}. \tag{10}$$

The transfer function defines how z will respond to changes in z^*. The bottom of the fraction can be used to determine whether the response will be stable and how fast z will converge to z^*. For a second-order response (which our controlled rocket is due to the s^2) the denominator is often expressed in a standard form of

$$s^2 + 2\zeta\omega_n s + \omega_n^2 \tag{11}$$

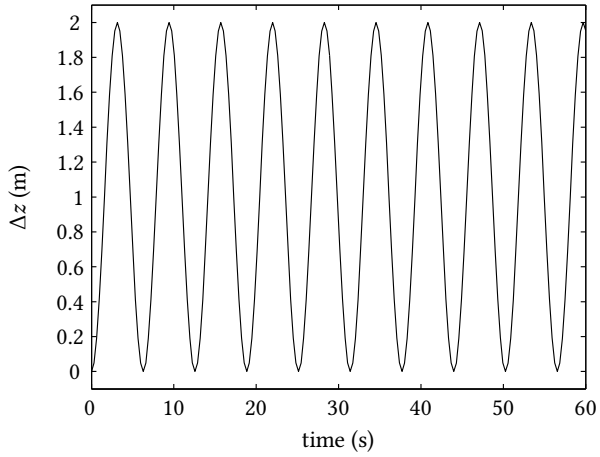

Figure 2: P controller response to a 1 m change in z^*.

where ζ is called the damping ratio, and ω_n the natural frequency.

The damping ratio ζ determines how much the response oscillates and how much overshoot occurs. ζ must be greater than or equal to 0, and typical values are often between 0.5 and 1.5. When $\zeta = 0$, the response will oscillate forever, while a value of 1 or greater means there will be no oscillation. For our rocket, $\omega_n^2 = K_p/m$, but there is no s term, which implies that $\zeta = 0$. This explains why our rocket oscillated.

The natural frequency determines how fast the response is. A rough estimate of the settling time of the response is $t_s = 5/\omega_n$ when $0.5 \leq \zeta \leq 1$. This means that z will be almost equal to z^* after $5/\omega_n$ seconds.

PD Control

Derivative control is often explained as counteracting the proportional and integral control. Given that our P controller resulted in constant oscillation, some D could help us out. With the derivative term added, our thrust value is now

$$T = \frac{(K_p + K_d s)\,(z^* - z) + mg}{\cos(\alpha)} \tag{12}$$

and this changes our system dynamics to

$$ms^2 z = (K_p + K_d s)(z^* - z). \tag{13}$$

Rearranging we find the transfer function to be

$$\frac{z}{z^*} = \frac{\frac{1}{m}(K_p + K_d s)}{s^2 + \frac{K_d}{m} s + \frac{K_p}{m}}. \tag{14}$$

From this we can see that we can set ζ using K_d and set ω_n using K_p. Because K_p and K_d can be set independently, we have complete control over the values for ζ and ω_n.

This now begs the question, what do we set ω_n and ζ to? First let's choose ζ. Our rocket overshooting its height setpoint could be a bad thing, especially if we are lowering our rocket it could overshoot into the ground or a building. For this reason let's set $\zeta = 1$ so that there is no overshoot without sacrificing response speed.

To choose ω_n, let's decide how fast we want our rocket to respond. Let's assume that most of the time we won't change our height setpoint by more than 10–20 meters at a time. It seems reasonable for a rocket to cover that distance in about 5 seconds. So let's choose a settling time of 5 seconds, which gives us an $\omega_n = 1$.

With ω_n and ζ chosen, we now can calculate K_p and K_d. Comparing the bottom of (14) and (11) we find

$$K_p = m\omega_n^2 \tag{15}$$

$$K_d = 2m\zeta\omega_n \tag{16}$$

which gives us a couple equations that we can code directly into our kOS script. Notice that the gains depend on the mass of the rocket. If we tuned the parameters manually, the rocket's response would change as its mass decreased due to the burned fuel. But because we performed this analysis, we now have expressions for K_p and K_d that we can update as the rocket's mass changes, keeping its response the same.

Let's see how our rocket responds now in Fig. 3. That's pretty good. Our rocket changes its height by 1 m in a

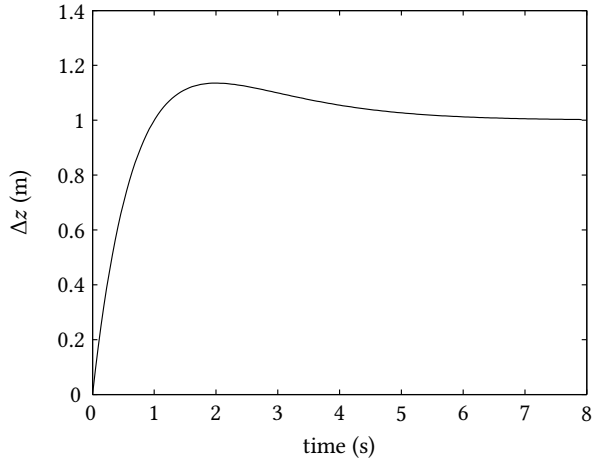

Figure 3: PD controller response to a 1 m change in z^*.

little over 5 s and exhibited no steady-state error. However, even though we set $\zeta = 1$, there was still overshoot in our response. I didn't lie to you, when $\zeta = 1$ there should be no overshoot, but if you look at (14) you'll notice that adding the D controller also added an s to the top of the fraction. Adding s terms to the top of the fraction tends to increase overshoot, and this is why we had overshoot even though we set $\zeta = 1$. If the overshoot that occurs is acceptable, we can stick with these gains. If not, it will be necessary to increase ζ.

It appears that we can't completely escape manual tuning. But, with the analysis we performed, all we have to modify are ζ and ω_n which have much more predictable effects on the response than the gains K_p and K_d.