PID Control in Frequency Domain

Dr. Mitch Pryor

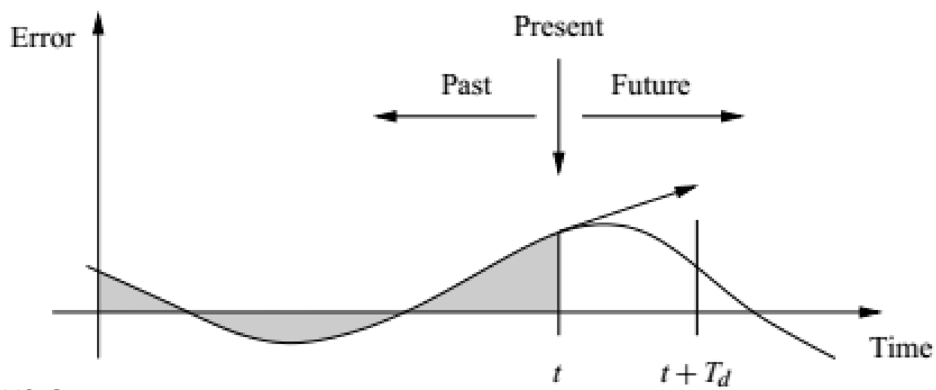
Lesson Objective

- Define the PID controller in the frequency domain
  - PID Impact on Root Locus
- Design example using PID and Root Locus
- Mechanical Analogy for PID Control
- Tuning PID Controllers

Recall PID Controller in time domain

$$u(t) = k_P e(t) + \mathrm{k}_I \int_0^t e(\tau) dt + k_D \frac{de(t)}{dt}$$

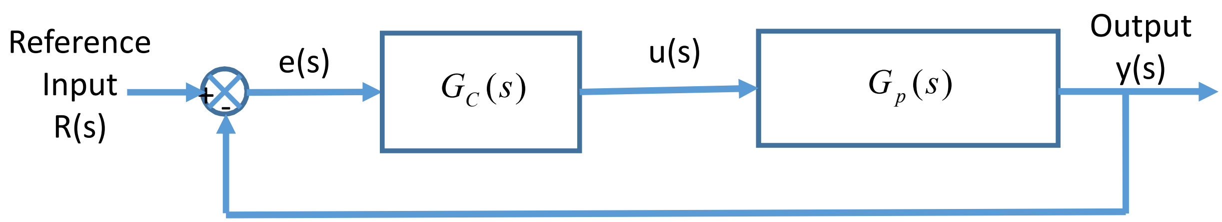
- A controller that eliminates steady-state error
- Address the “past, present, and future” error in the controller
- More than 95% of all industrial control problems are solved using a PID Controller



- Very difficult to tune.
- Cumbersome to evaluate over a range of inputs.

From frequency to time domain

$$u(t) = k_P e(t) + \mathrm{k}_I \int_0^t e(\tau)dt + k_D \frac{de(t)}{dt}$$



$$u(\mathrm{s}) = K_P e(s) + K_I \frac{1}{s} e(s) + K_D s e(s)$$

$$= \left(K_P + K_I \frac{1}{s} + K_D s \right) e(s)$$

PID Controller Summary

$$u(s) = \left(K_P + K_I \frac{1}{s} + K_D s \right) e(s)$$

- Been around (though not formalized) since 1890s
- First published in 1920's
- Today, PID controllers found in 97% of the controllers.
  - Based on a survey of 11,000 controllers across multiple industries [Desborough & Miller, 2002]
- Advantages
  - Process independent
  - Leads to feasible solution in most cases after tuning
  - Inexpensive, readily available
  - Often tuned without much experience (especially using software tools)
- Disadvantages
  - Not optimal (graduate course in Optimal Controls!)
  - Difficult to extend to MIMO and nonlinear. systems (graduate courses as well)
  - Poor tuning can lead to instability or "hunting" (oscillations about the desired operating point)
  - Integrator wind-up (Can the input device meet the demands of the controller we have designed? What happens when it doesn't?)
  - D element may amplify the effects of high frequency input or noise
- We will discuss Integrator wind-up and other "implementation issues" in an upcoming lecture(s).

What does does PID control add to RL?

$$G_c(s) = \left(K_P + K_I \frac{1}{s} + K_D s\right)\frac{s}{s}$$

$$= \frac{\left(K_P s + K_I + K_D s^2\right)}{s}$$

$$= \frac{K_D\left(s^2 + \frac{K_P}{K_D}s + \frac{K_I}{K_D}\right)}{s}$$

$$= \frac{K\left(s^2 + bs + c\right)}{s}$$

$$= \frac{K\left(s + z_1\right)\left(s + z_2\right)}{s}$$

So a PID controller adds a pole at the origin and two zeros that the designer selects.

RL for different gains...

$$u(\mathrm{s}) = \left(K_P + K_I \frac{1}{s} + K_D s \right) e(s)$$

Previously we isolated K_D.

$$
\begin{aligned}
G_c(s) &= \left(K_P + K_I \frac{1}{s} + K_D s \right) \frac{s}{s} \\
&= \frac{\left(K_P s + K_I + K_D s^2 \right)}{s} \\
&= \frac{K_D \left(s^2 + \frac{K_P}{K_D} s + \frac{K_I}{K_D} \right)}{s} \\
&= \frac{K\left(s^2 + bs + c \right)}{s} \\
&= \frac{K\left(s + z_1 \right)\left(s + z_2 \right)}{s}
\end{aligned}
$$

But we can algebraically isolate most parameters we may be interested in.

For example, if we want to look at RL with respect to the K_P:

$$
\begin{aligned}
G_c(s) &= \left(K_P + K_I \frac{1}{s} + K_D s \right) \frac{K_P}{K_P} \\
&= K_P \left(\frac{K_P}{K_P} + \frac{K_I}{K_P} \frac{1}{s} + \frac{K_D}{K_P} s \right) \\
&= K_P \left(1 + \frac{1}{T_I s} + T_D s \right)
\end{aligned}
$$

Many formulations are possible and common in the literature.

What if K_D=0?

$$u(\mathrm{s}) = \left(K_P + K_I \frac{1}{s} + K_D s\right) e(s)$$

$$G_c(s) = \left(K_P + K_I \frac{1}{s}\right)\frac{s}{s}$$

$$= \frac{(K_P s + K_I)}{s}$$

$$= \frac{K_P\left(s + \dfrac{K_I}{K_P}\right)}{s}$$

$$= \frac{K_P(s+\tau)}{s}$$

Lots of algebraic variations are possible. So you can use RL to see how a variety of parameters impact performance.

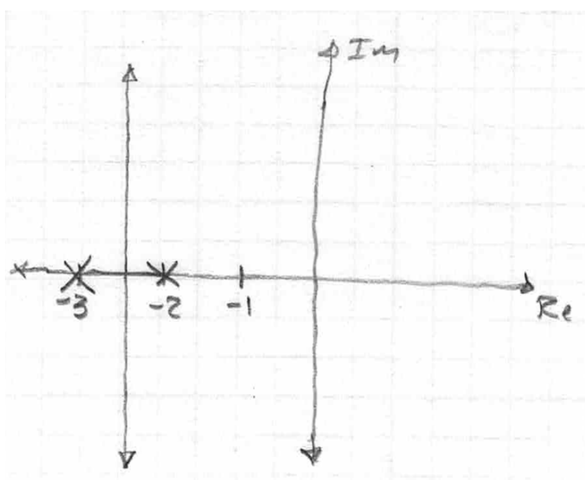
So different choices can lead to different PZ maps and, thus, Root Locus configurations.

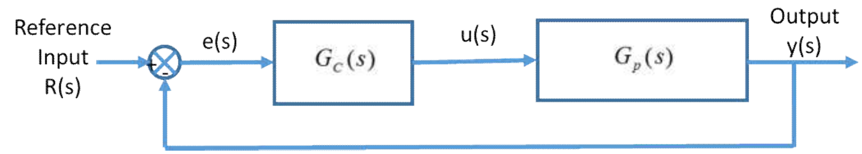
Example

Consider the following plant:

$$G_P(s) = \frac{K_{plant}}{s^2 + 5s + 6}$$

Which has the following Root Locus…





Can you get this on your own? Also, not ideal since more gain leads to increasingly higher oscillations and overshoot. So let's add a PID controller!

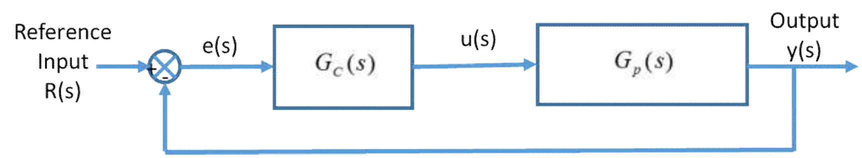
Let,

$$p = 0 \qquad z_{1,2} = -3 \pm i$$

Thus,

$$G_C(s) = \frac{K(s^2 + 6s + 10)}{s}$$

Example, cont'd



From the previous slide

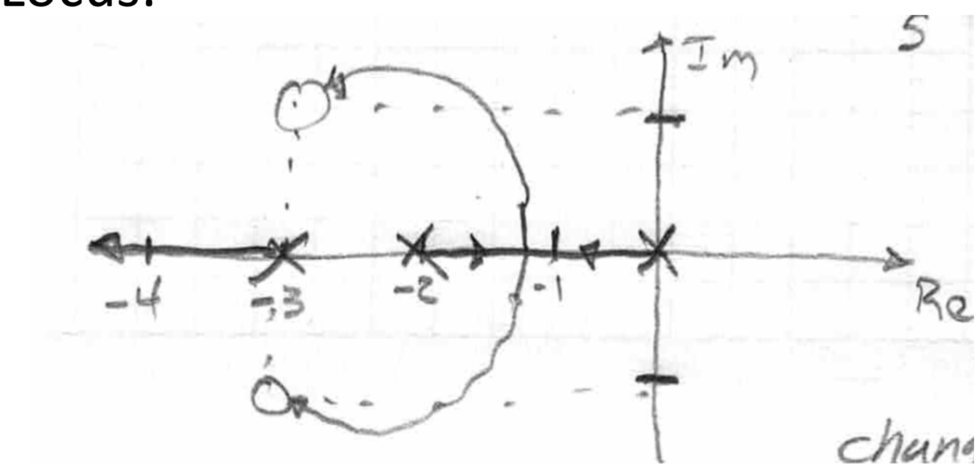
$$G_P(s) = \frac{K_{plant}}{s^2 + 5s + 6}$$

$$G_C(s) = \frac{K\left(s^2 + 6s + 10\right)}{s}$$

So our new open loop Transfer Function is

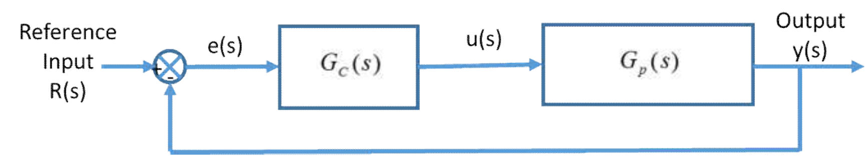
$$G_C(s) = \frac{K\left(s^2 + 6s + 10\right)}{s\left(s^2 + 5s + 6\right)}$$

Which produces the following Root Locus:



The zeros limit the oscillations at high gain. The pole at the origin means (in this case) there is only one pole at infinity (thus on real axis).

Example, cont'd



Impact on closed loop Transfer Functions

Unity feedback (K=10)

$$G_{CL}(s)=\frac{G_p}{1+G_p}$$

$$=\frac{\frac{K}{s^2+5s+6}}{1+\frac{K}{s^2+5s+6}}$$

$$=\frac{10}{s^2+5s+16}$$

PID controller feedback (K=10)

$$G_{CL}(s)=\frac{G_cG_p}{1+G_cG_p}$$

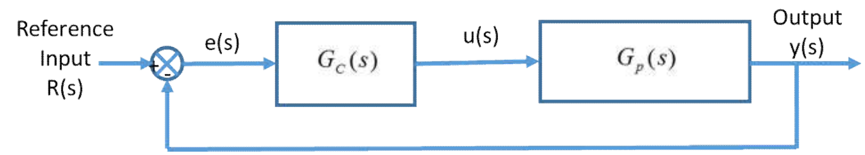
$$=\frac{\frac{K\left(s^2+6s+10\right)}{s^3+5s^2+6s}}{1+\frac{K\left(s^2+6s+10\right)}{s^3+5s^2+6s}}$$

$$=\frac{10\left(s^2+6s+10\right)}{s^3+5s^2+6s+10s^2+60s+100}$$

$$=\frac{10s^2+60s+100}{s^3+15s^2+66s+100}$$

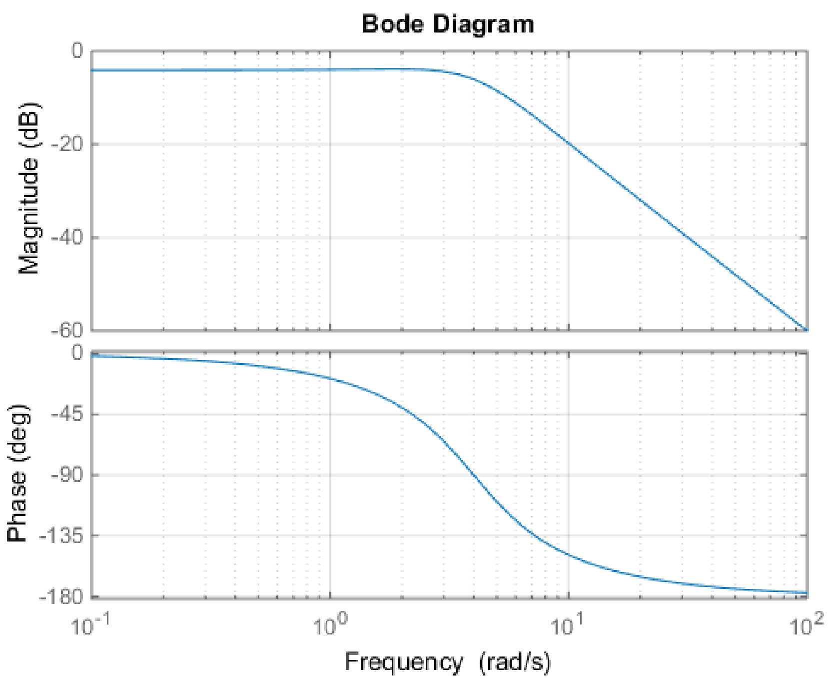
Note, there is no steady state error!

Example cont'd



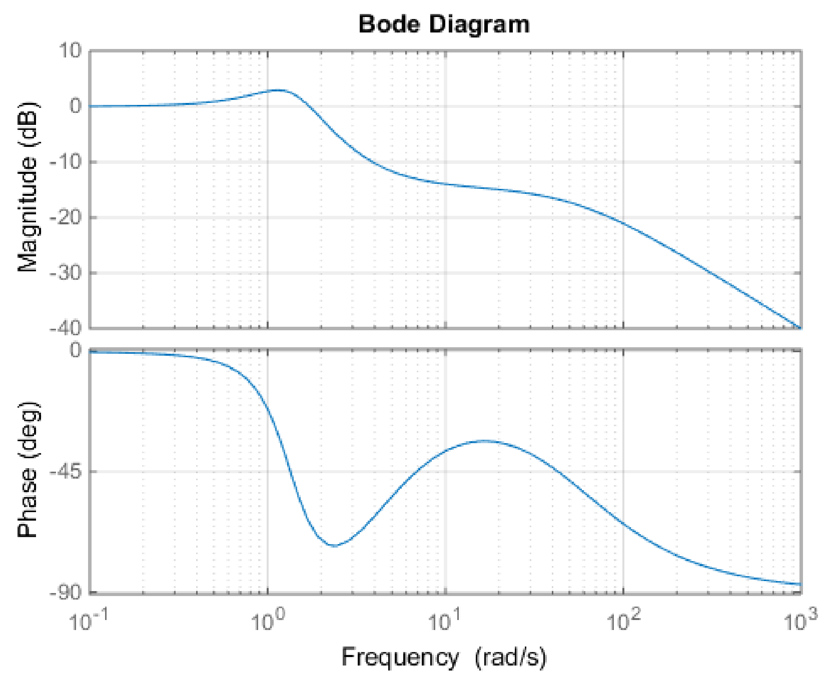
Unity feedback (K=10)

$$G_{CL}(s) = \frac{10}{s^2 + 5s + 16}$$



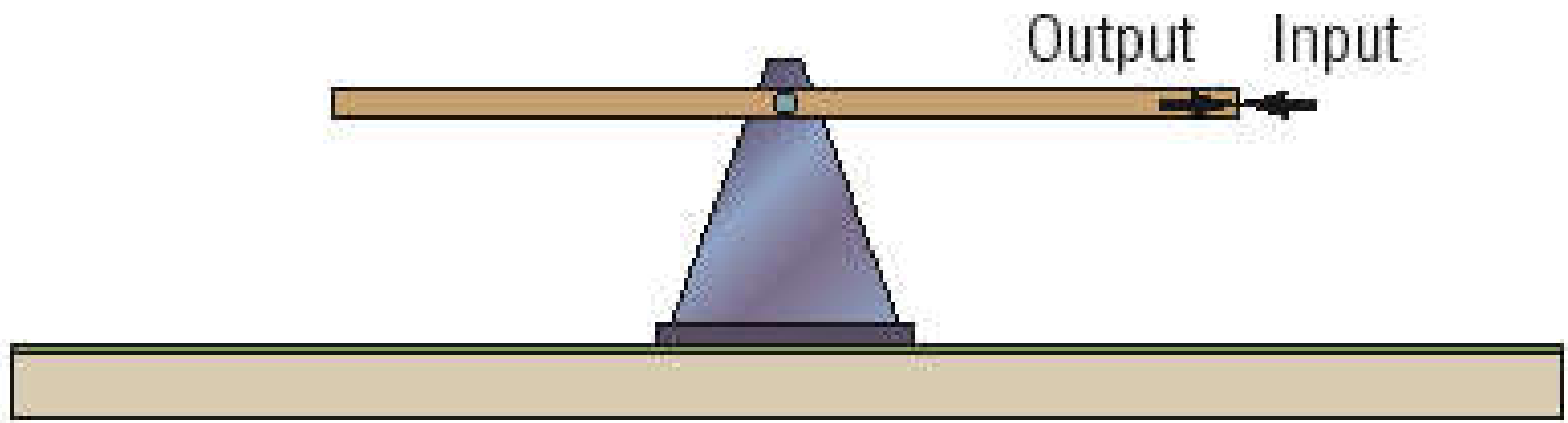
PID controller feedback (K=10)

$$G_{CL}(s) = \frac{10s^2 + 60s + 100}{s^3 + 15s^2 + 66s + 100}$$



Note the frequency scale and phase lag at high frequency.

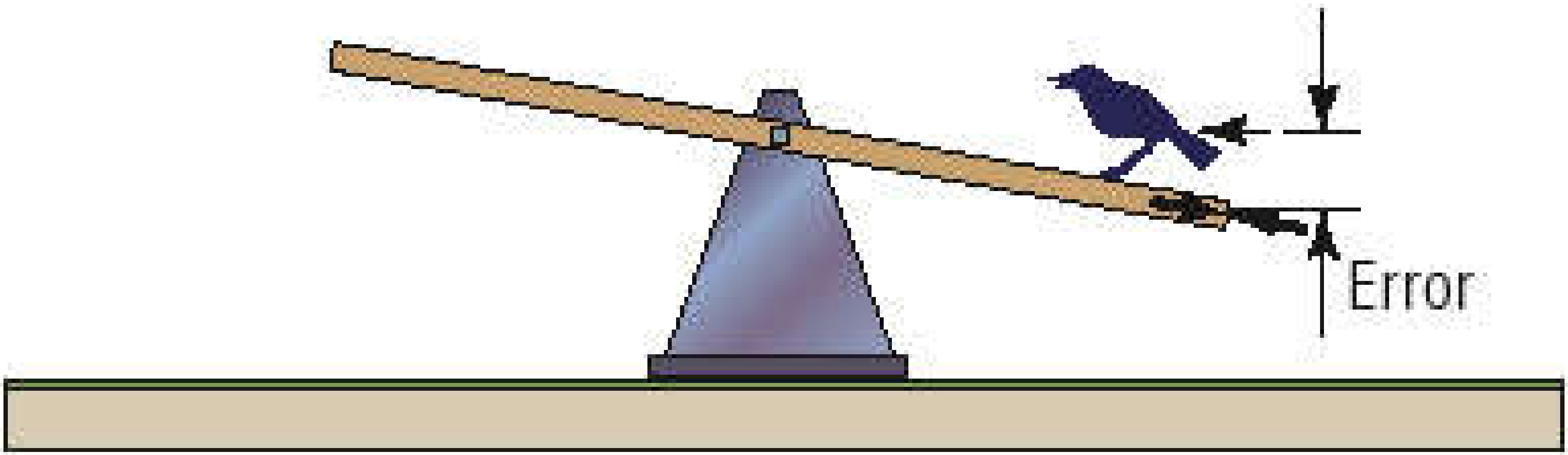
Mechanical/PID Analogy



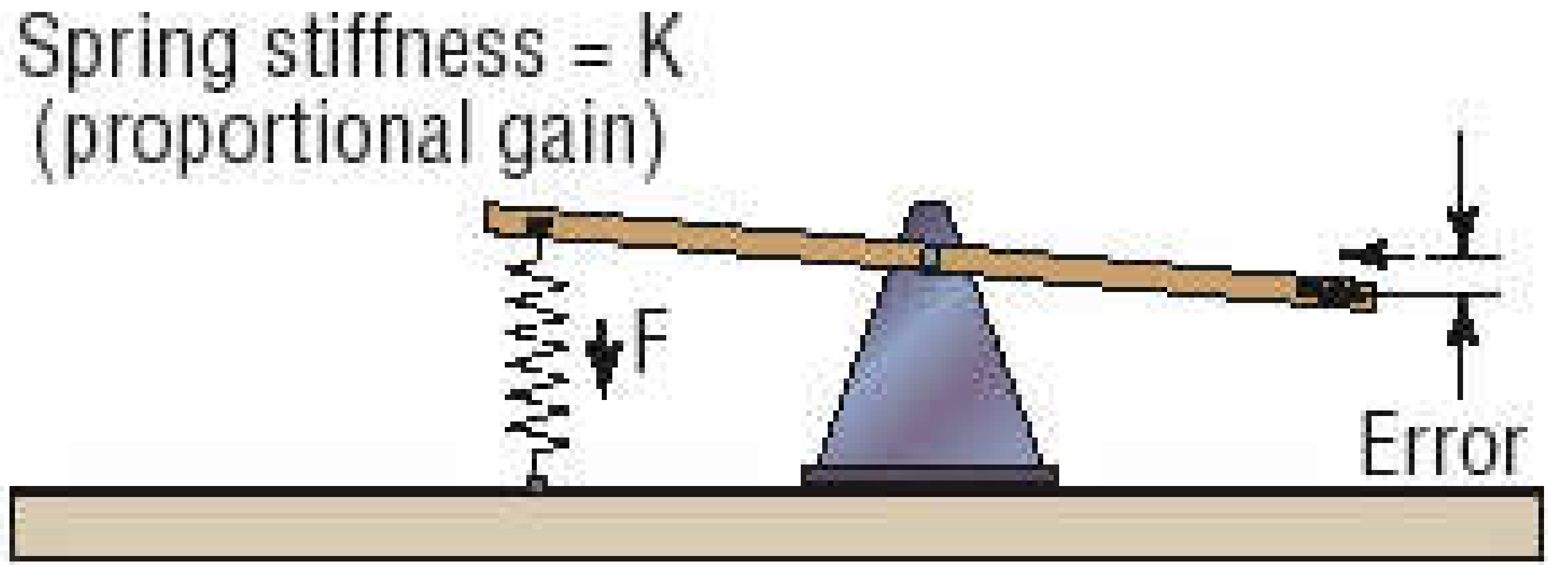
Let's start with a child's seesaw

http://hydraulicspneumatics.com/other-technologies/learn-ride-pid-seesaw

Uh Oh. A Disturbance

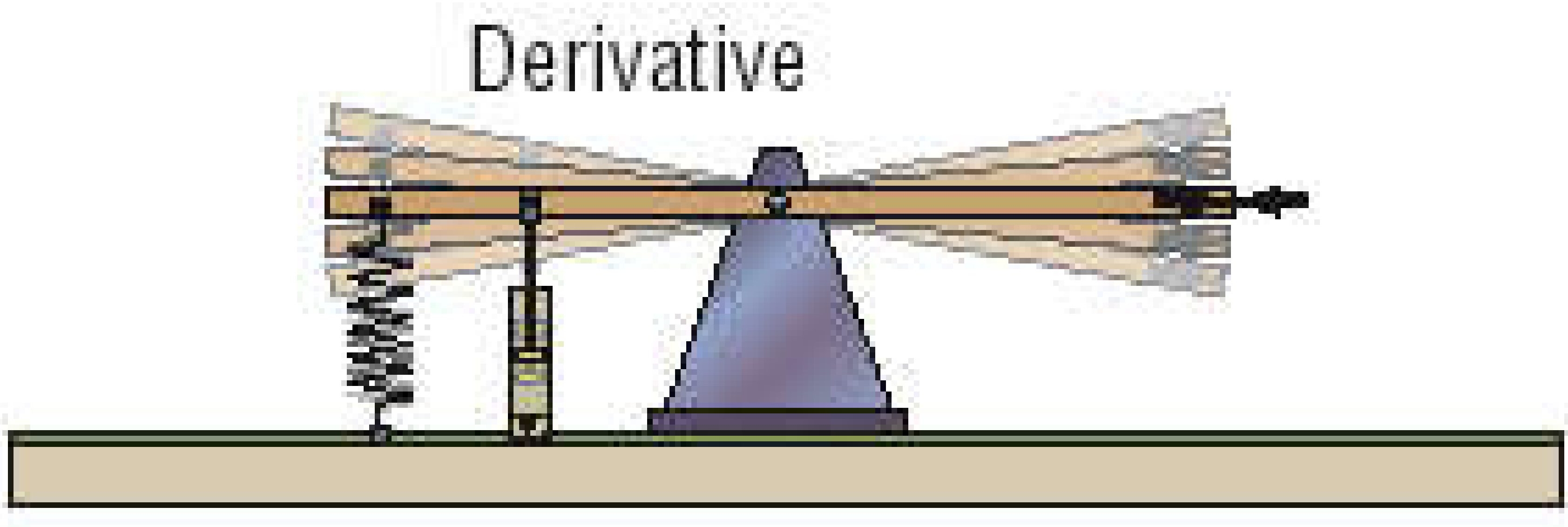


Fixing the error.



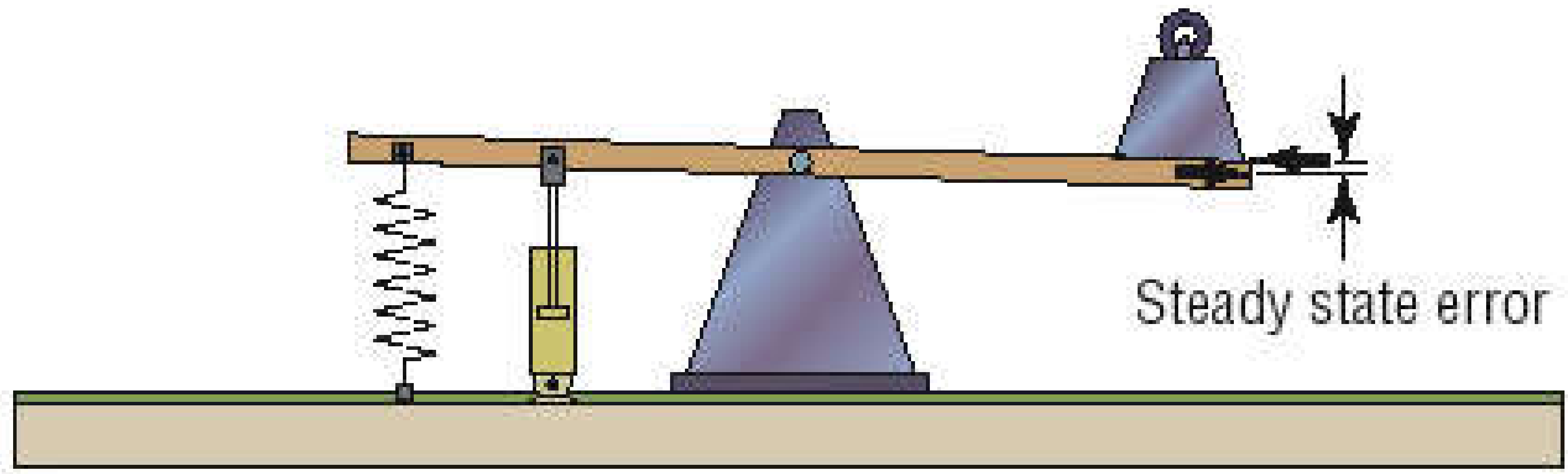
The stiffness of the spring can be thought of as the proportional gain or our control system.

Fixing the oscillation problem



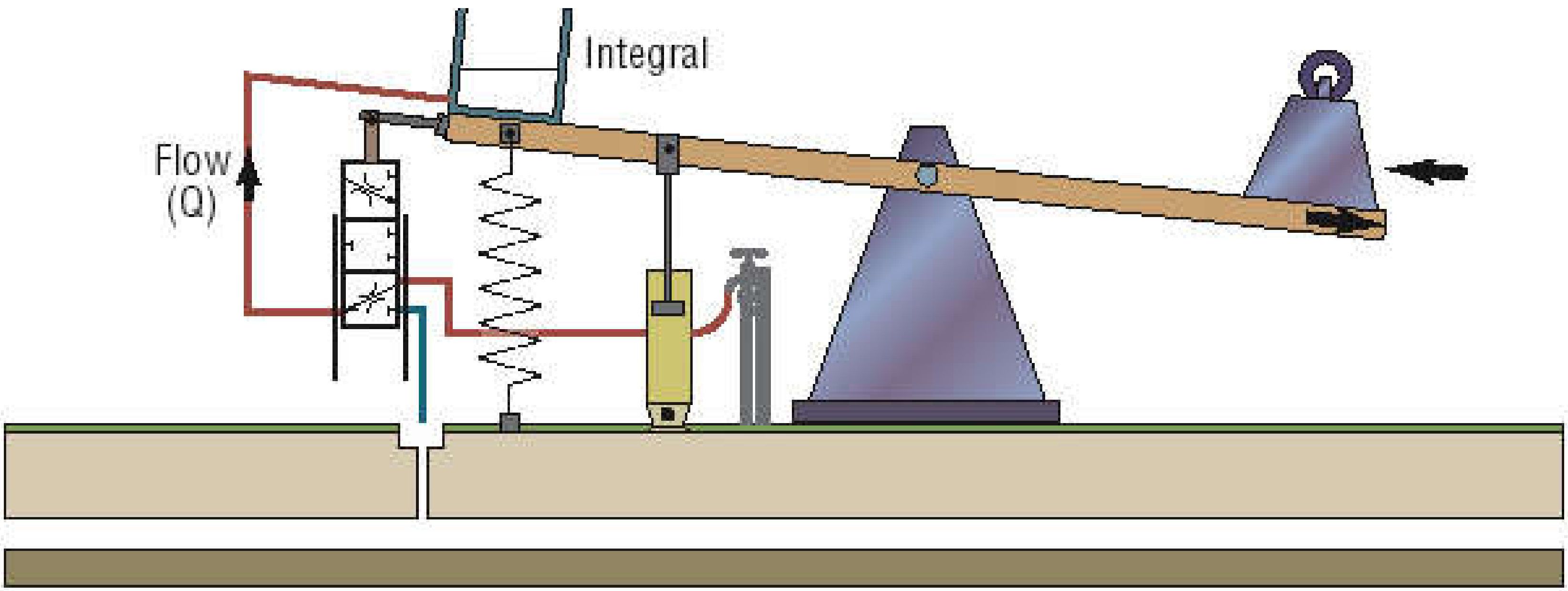
Let's use the dashpot to eliminate any unwanted oscillations and its input is proportional to the velocity (or derivative) of our error.

Fixing the offset error problem



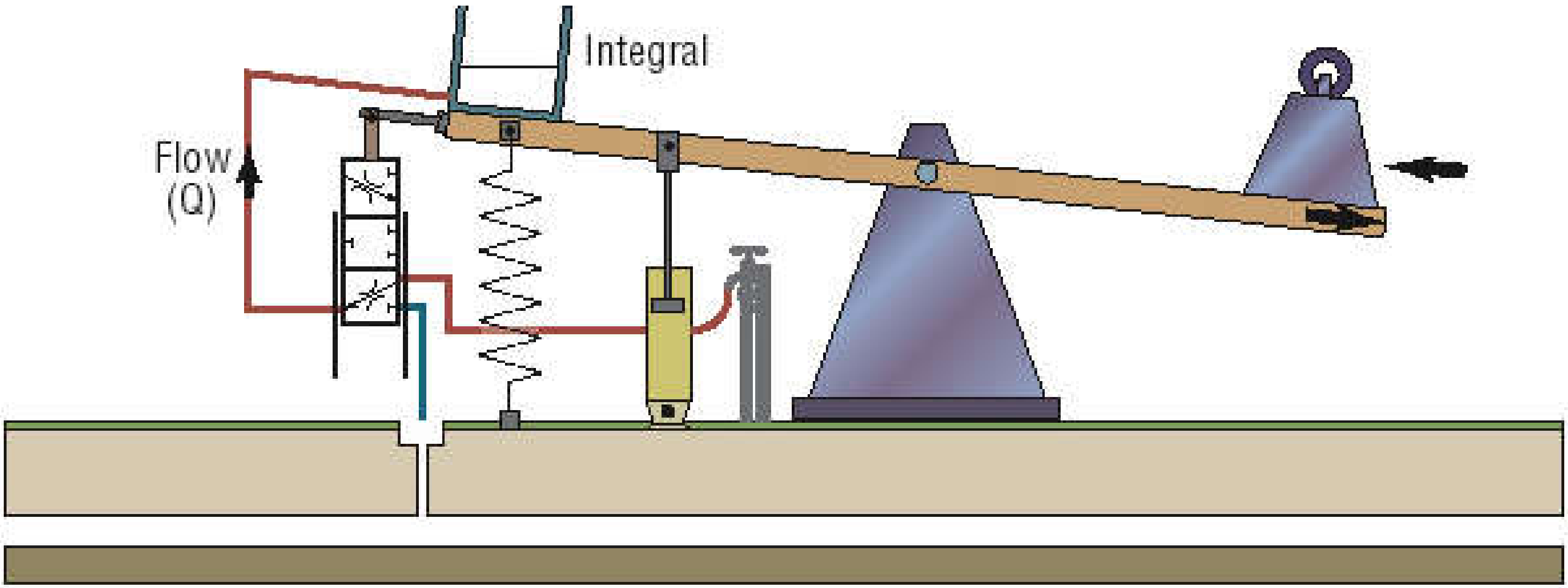
But if our disturbance is of the more permanent variety (think friction / drag in the cruise controller), then there will be a steady state error.

Fix the offset variability problem



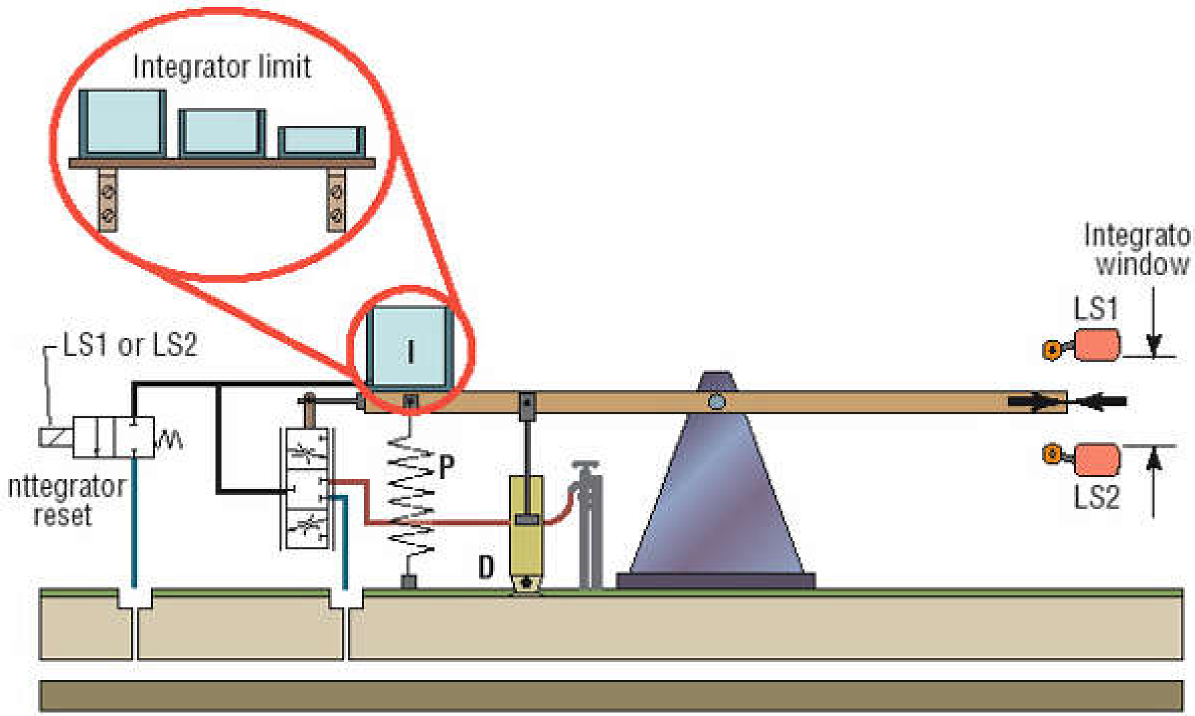
To counter act this, I add a counterweight to other side and the ability to modify the amount of weight to match the disturbance.

Fix the offset variability problem



This represents the integral terms which now accounts for both the offset of the error and the rate of change of the water (i.e. weight) in the bucket. The rate of flow is the integrator term.

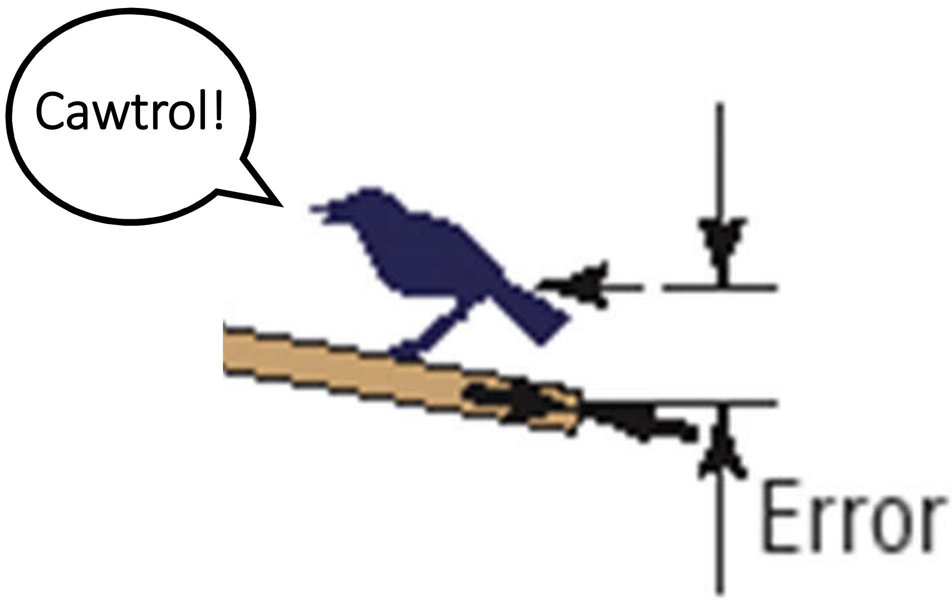
But water flow isn't instantaneous



Bucket can't be more than full or less than empty. Also, the system could fill up rapidly and then empty too slowly if the disturbance goes away. I can address these with limit switches and or a periodic dumping of the water.

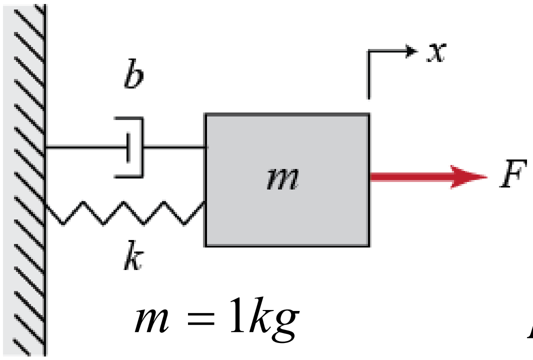
Mechanical/PID Analogy

It is all about the inevitable disturbance of that bird, and our need to address it in an automatic way.



Physical Impact of PID Gains

Given:



$$m = 1kg \qquad F = 1N$$

$$b = 10N\tfrac{s}{m} \qquad x_{desired} = 1.0$$

$$k = 20\tfrac{N}{m}$$

Find:

How various controllers (and controller gains) impact:

- Rise time
- % overshoot
- steady-state error

Solution:

The equations of motion…

$$m\ddot{x} + b\dot{x} + kx = F$$

The Laplace Transform…

$$ms^2X(s) + bsX(s) + kX(s) = F(s)$$

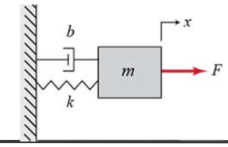
And the transfer function…

$$\mathrm{P}(s) = \frac{X(s)}{F(s)}$$

$$= \frac{1}{ms^2 + bs + k}$$

$$= \frac{1}{s^2 + 10s + 20}$$

System's open-loop step response

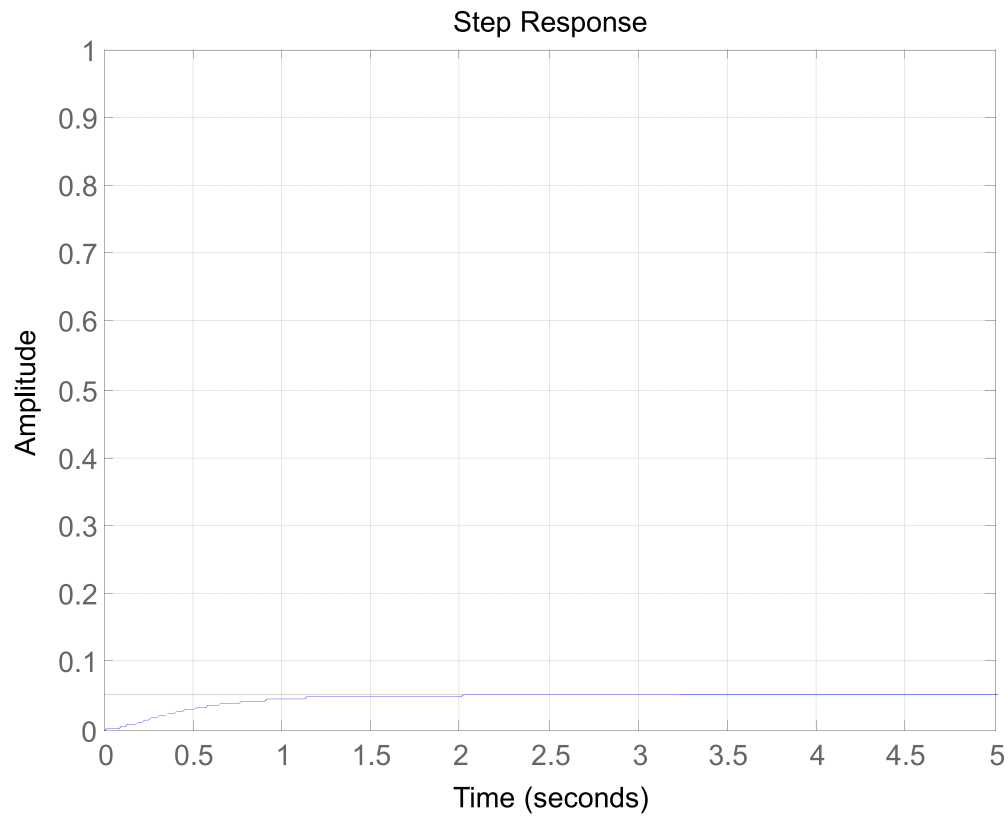


Controller: none

$$T_{ol}(s) = P(s) = \frac{1}{s^2 + 10s + 20}$$

```
%open loop response
n = [1];
d = [ 1 10 20 ];
sys = tf( n, d )

step( sys );
grid on;
axis( [ 0 5 0 1 ] )
```



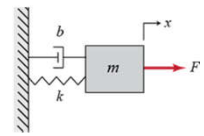
<u>Performance summary:</u>

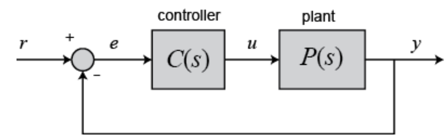
Rise time: about 1 sec.

% overshoot: none

e_{ss}: 0.95

Controller response





Controller: Proportional Controller

$$T_{cl}(s) = \frac{C(s)\,\mathrm{P}(s)}{1 + C(s)\,\mathrm{P}(s)} = \frac{K_p\,\mathrm{P}(s)}{1 + K_p\,\mathrm{P}(s)}$$

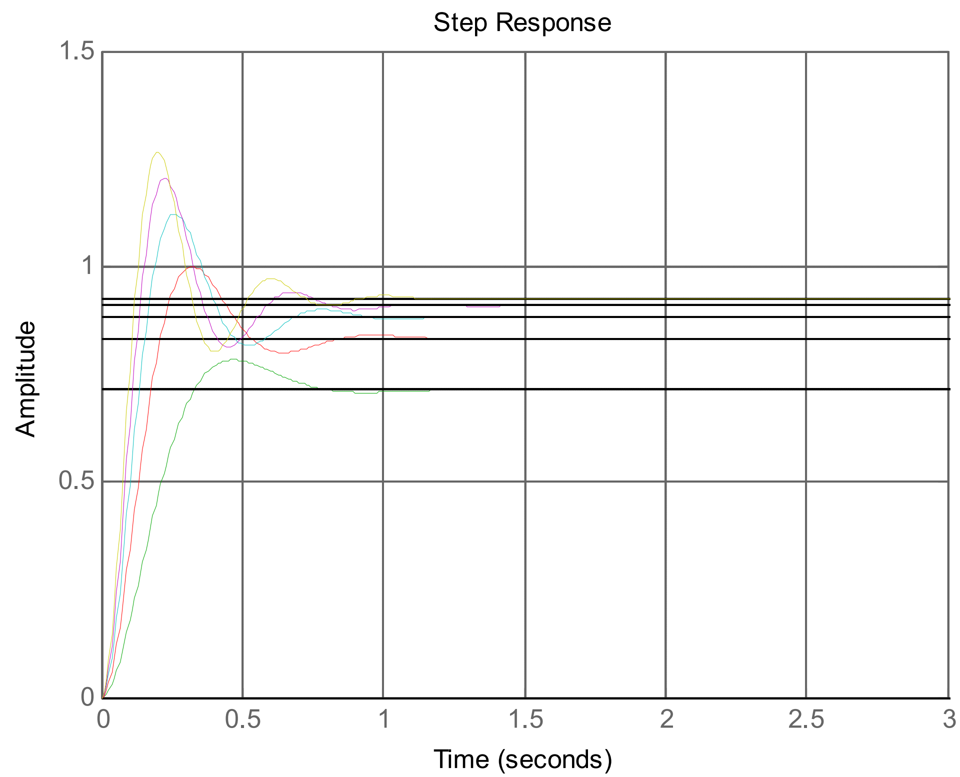
$$= \frac{K_p}{s^2 + 10s + (20 + K_p)}$$

```
Kp = [0:50:250];
for i=1:length(Kp)
    n = [Kp(i)];
    d = [ 1 10
20+Kp(i) ];
    sys = tf( n, d )
    step( sys );
    hold on;
end
```

or....

```
n = [1];
d = [ 1 10 20 ];
sys = tf( n, d );
Kp = 250;
C = pid(Kp)
T =
feedback(C*sys,1)

t = 0:0.01:3;
step(T,t,'b')
```



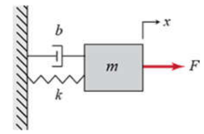
Performance summary:

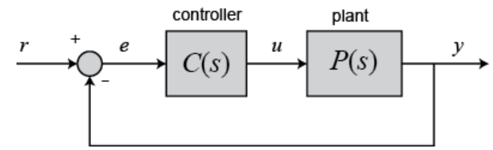
Rise time: slightly reduced

% overshoot: increased, but varies

e_{ss}: reduced, but dependent on gain

Controller response





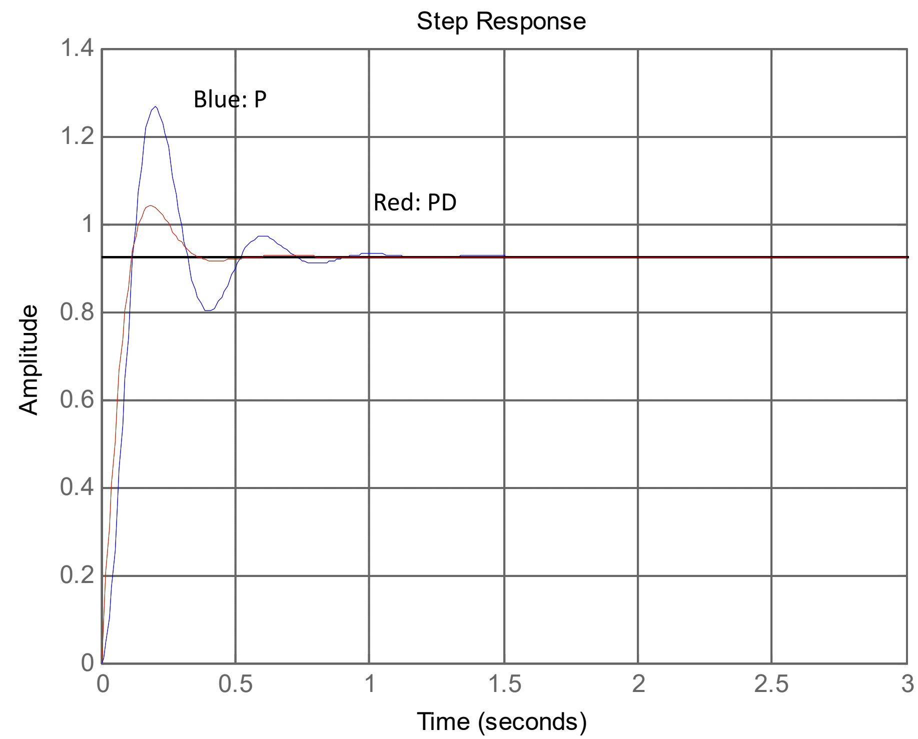
Controller: Proportional Derivative

$$T_{cl}(s) = \frac{C(s)\,\mathrm{P}(s)}{1 + C(s)\,\mathrm{P}(\mathrm{s})} = \frac{\left(K_d s + K_p\right)\mathrm{P}(s)}{1 + \left(K_d s + K_p\right)\mathrm{P}(s)}$$

$$= \frac{K_d s + K_p}{s^2 + \left(10 + K_d\right)s + (20 + K_p)}$$

```
n = [1];
d = [ 1 10 20 ];
sys = tf( n, d );
Kp = 250;
Kd = 10;
C = pid(Kp, 0, 0 )
T = feedback(C*sys,1)
figure(1)
step(T,t,'b')
hold on;
```

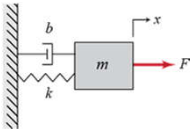
```
grid on;
C = pid(Kp, 0, Kd )
T = feedback(C*sys,1)
figure(1)
step(T,t,'r')
hold on;
```

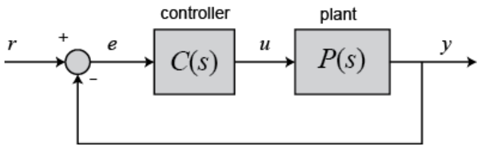


<u>Performance summary:</u>

Rise time: similar
% overshoot: decreased
e_{ss}: no improvement

Controller response



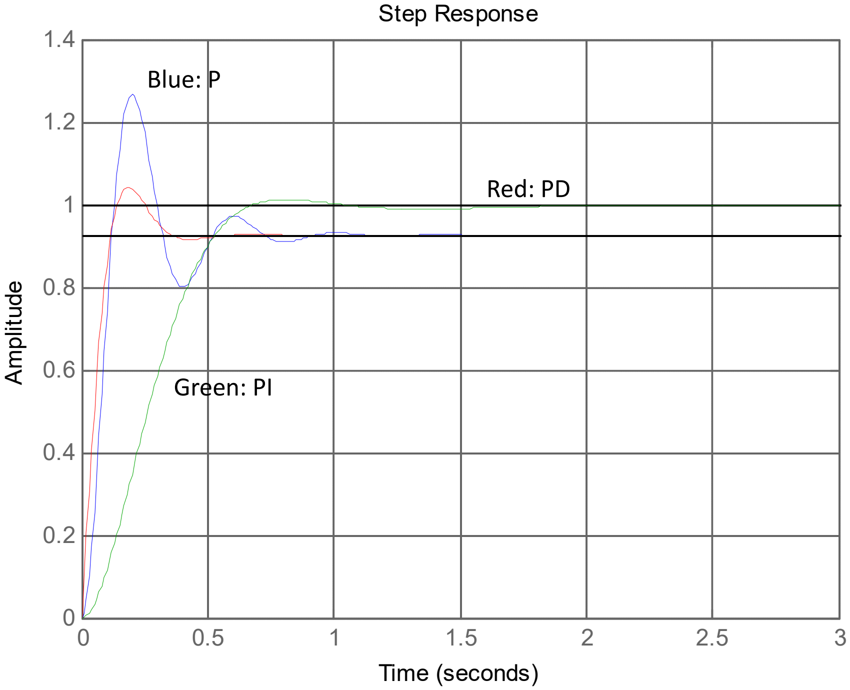


Controller: Proportional Integral

$$T_{cl}(s) = \frac{C(s)\mathrm{P}(s)}{1 + C(s)\mathrm{P(s)}} = \frac{\left(K_I + K_p s\right)\mathrm{P}(s)}{1 + \left(K_I + K_p s\right)\mathrm{P}(s)}$$

$$= \frac{K_p s + K_I}{s^3 + 10s^2 + (20 + K_p\ \mathrm{s} + K_I)}$$

```
Kp = 30;                    T = feedback(C*sys,1)
Ki = 70;                    figure(1)
grid on;                    step(T,t,'g')
C = pid(Kp, Ki, 0 )         hold on;
```



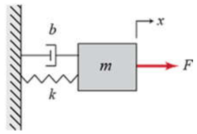
<u>Performance summary:</u>

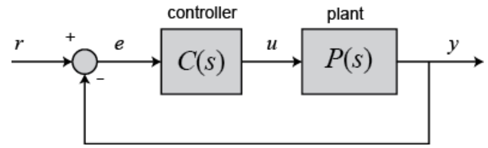
Rise time: slower

% overshoot: little or none

e_{ss}: eliminated

Controller response





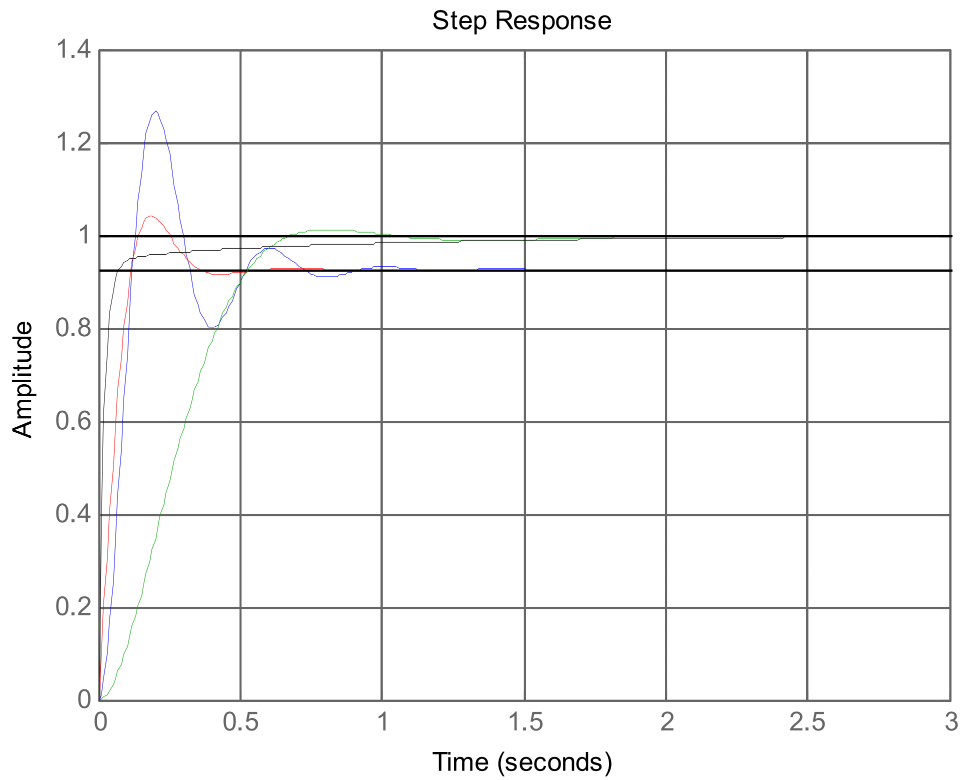
Controller: Full PID

$$T_{cl}(s) = \frac{C(s)\,\mathrm{P}(s)}{1 + C(s)\,\mathrm{P}(s)} = \frac{\left(K_I + K_p s + K_d s^2\right)\mathrm{P}(s)}{1 + \left(K_I + K_p s + K_d s^2\right)\mathrm{P}(s)}$$

$$= \frac{K_d s^2 + K_p s + K_I}{s^3 + \left(10 + K_d\right)s^2 + (20 + K_p)\,\mathrm{s} + K_I}$$

```
Kp = 350;
Ki = 300;
Kd = 50
grid on;
C = pid(Kp, Ki, Kd );
```

```
C = pid(Kp, Ki, Kd )
T = feedback(C*sys,1)
figure(1)
step(T,t,'k')
hold on;
```

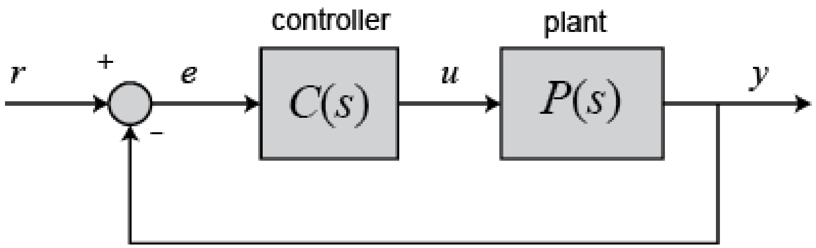


Performance summary:

Rise time: significantly reduced
% overshoot: none
e_{ss}: eliminated

Of course, we still have to pick the right gains

Summary of differences



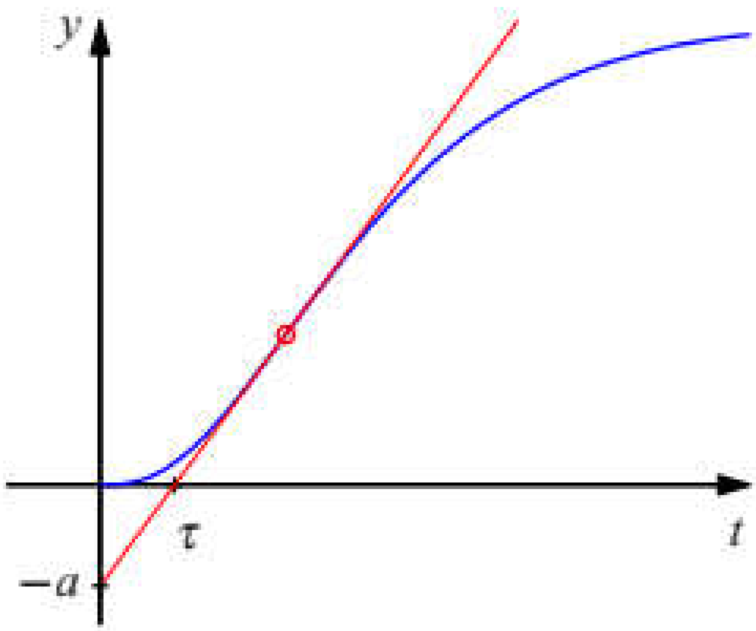
$$K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s}$$

PID Gain	Rise time	Overshoot	Settling time	Steady-state error
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Eliminate
Kd	Small Change	Decrease	Decrease	No Change

(**small print:** correlations may not always be accurate, your mileage may vary, non-refundable in Michigan, highly coupled interactions may be observed, prohibited from asserting in Oklahoma, changing one value can cause the effects of the other variables to change and/or exhibit erratic behavior, experimentation may lead to increased blood pressure, sleep deprivation, and general sense of internal rage.)

Tuning Technique: Ziegler-Nichols

$$G_c(s) = K_P\left(1+\frac{1}{T_I s}+T_D s\right)$$



Apply a unit step to a process and record the response. The response is characterized by the terms *a* and τ.

a: the *y-intercept using the slope at the steepest point of the response.*

τ*:* the *x-intercept using the slope at the steepest point of the response.*

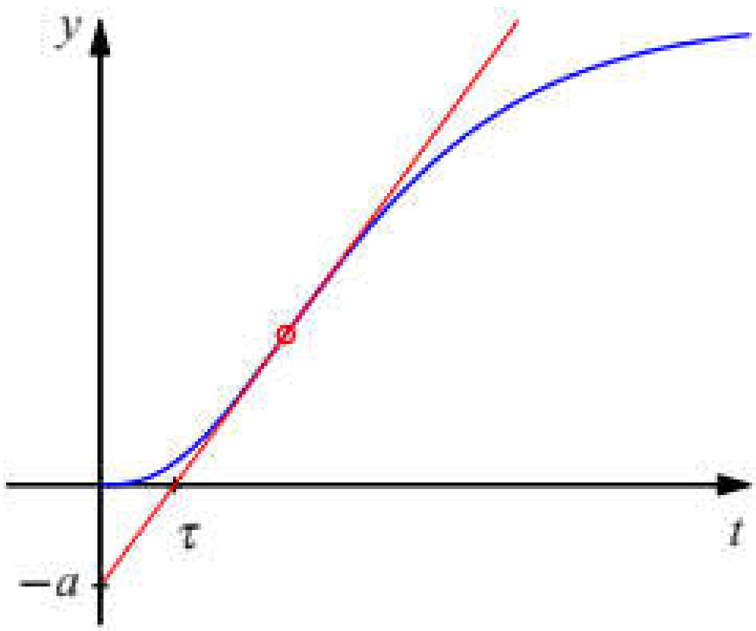
Therefore *a*/τ is the steepest slope of the response.

Type	k_p	T_i	T_d
P	1/a		
PI	0.9/a	3τ	
PID	1.2/a	2τ	0.5τ

Determined by manually tuning PID controllers and then identifying correlations to *a* and τ

Tuning Technique: Ziegler-Nichols

$$G_c(s) = K_P\left(1 + \frac{1}{T_I s} + T_D s\right)$$



The method can be improved by fitting the experimental response with the following curve.

$$P(s) = \frac{K}{1+sT}e^{-\tau s}$$

And then finding *a* using the equation:

$$a = \frac{K\tau}{T}$$

Type	k_p	T_i	T_d
P	1/a		
PI	0.9/a	3τ	
PID	1.2/a	2τ	0.5τ

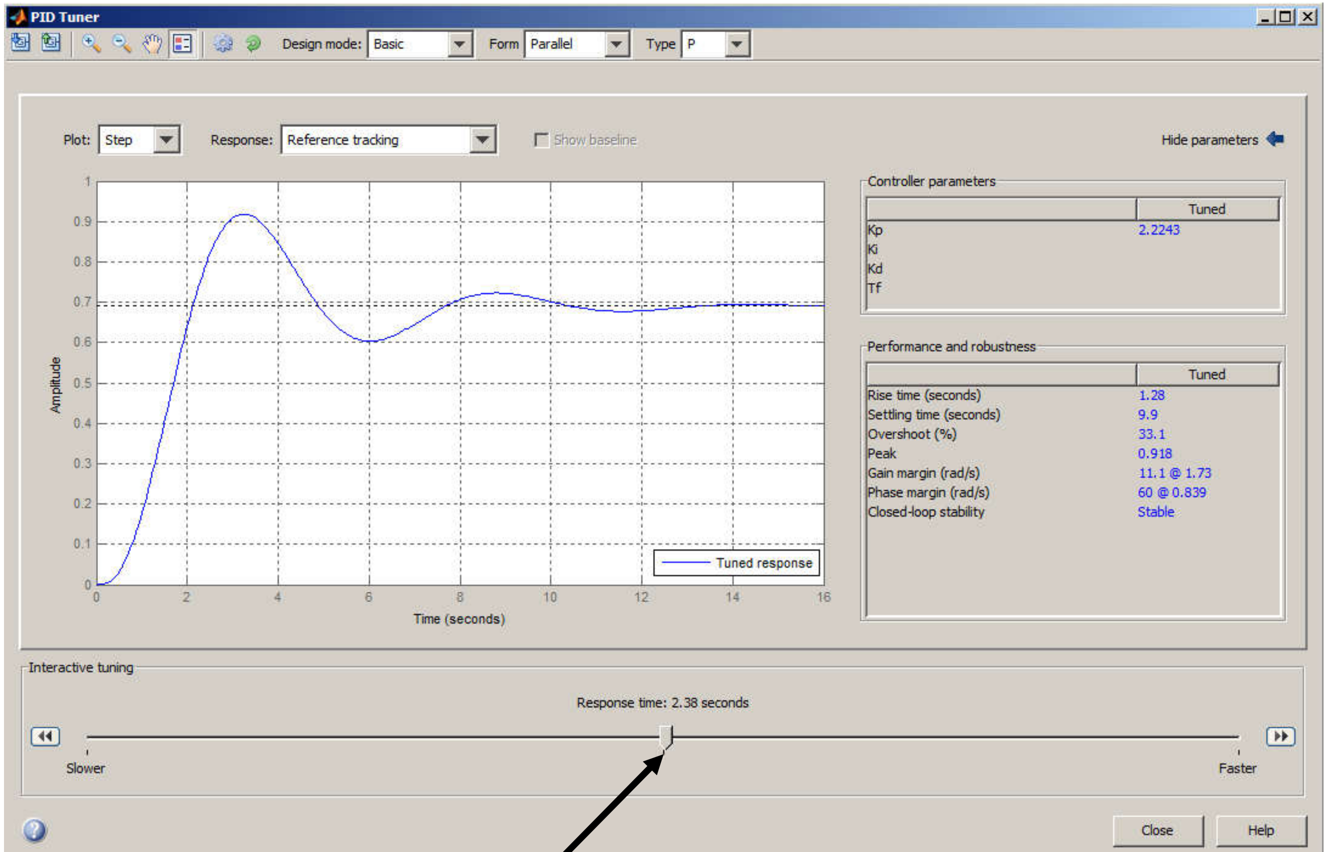
Other methods and variants of this method exist.

Controller Design MATLAB

```
>>pidtuner
```

Proportional only…

```
a = [1];
b = [ 1 3 3 1 ];
sys = tf( a, b );
pidtool( sys, 'P' );
```

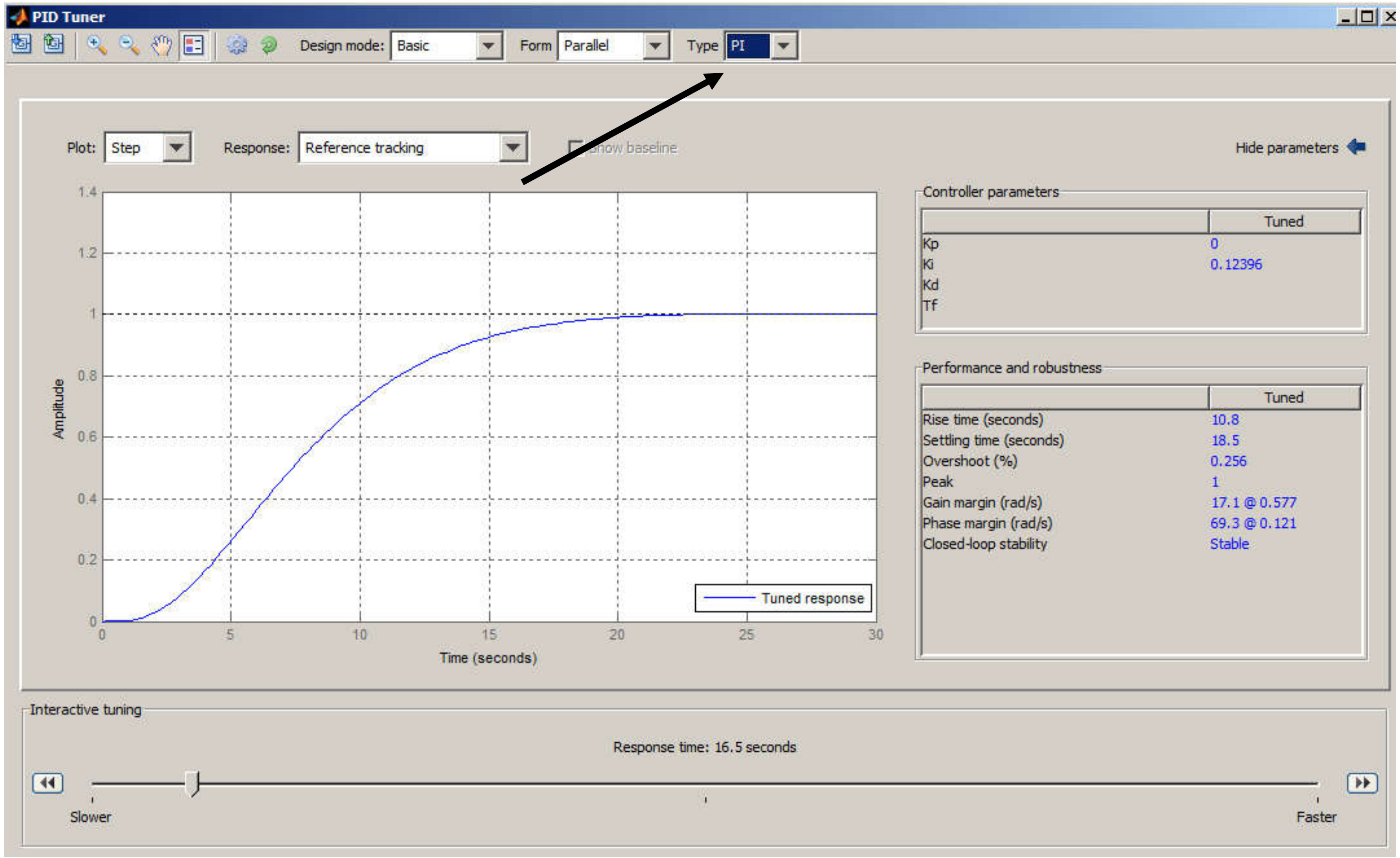


As we modify the response time, we will likely see the % overshoot change.

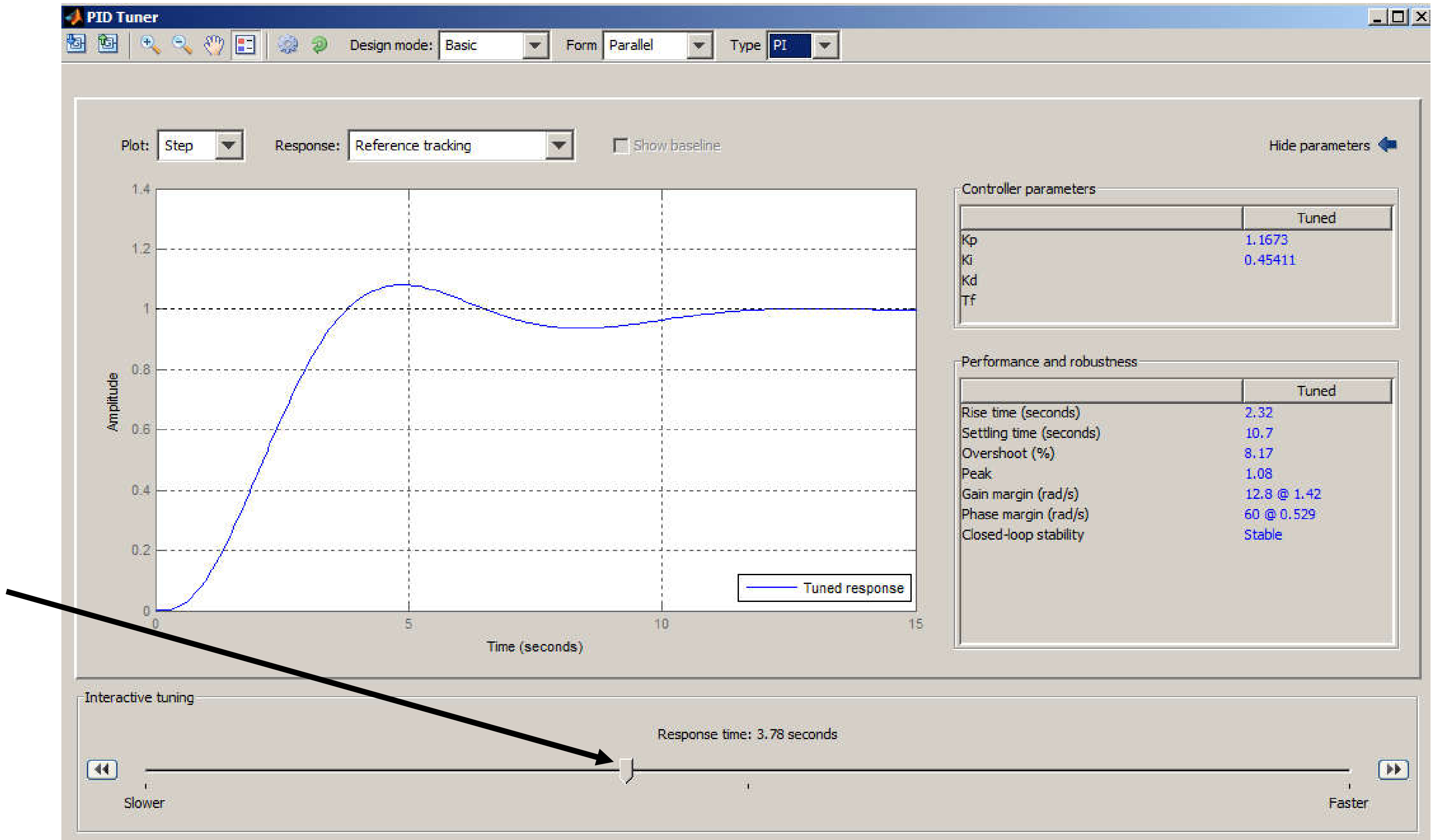
PID Tool options...

String	Type	Continuous-Time Controller Formula (parallel form)	Discrete-Time Controller Formula (parallel form, `ForwardEuler` integration method)
`'p'`	proportional only	K_p	K_p
`'i'`	integral only	$\frac{K_i}{s}$	$K_i \frac{T_s}{z-1}$
`'pi'`	proportional and integral	$K_p + \frac{K_i}{s}$	$K_p + K_i \frac{T_s}{z-1}$
`'pd'`	proportional and derivative	$K_p + K_d s$	$K_p + K_d \frac{z-1}{T_s}$
`'pdf'`	proportional and derivative with first-order filter on derivative term	$K_p + \frac{K_d s}{T_f s + 1}$	$K_p + K_d \frac{1}{T_f + \frac{T_s}{z-1}}$
`'pid'`	proportional, integral, and derivative	$K_p + \frac{K_i}{s} + K_d s$	$K_p + K_i \frac{T_s}{z-1} + K_d \frac{z-1}{T_s}$
`'pidf'`	proportional, integral, and derivative with first-order filter on derivative term	$K_p + \frac{K_i}{s} + \frac{K_d s}{T_f s + 1}$	$K_p + K_i \frac{T_s}{z-1} + K_d \frac{1}{T_f + \frac{T_s}{z-1}}$

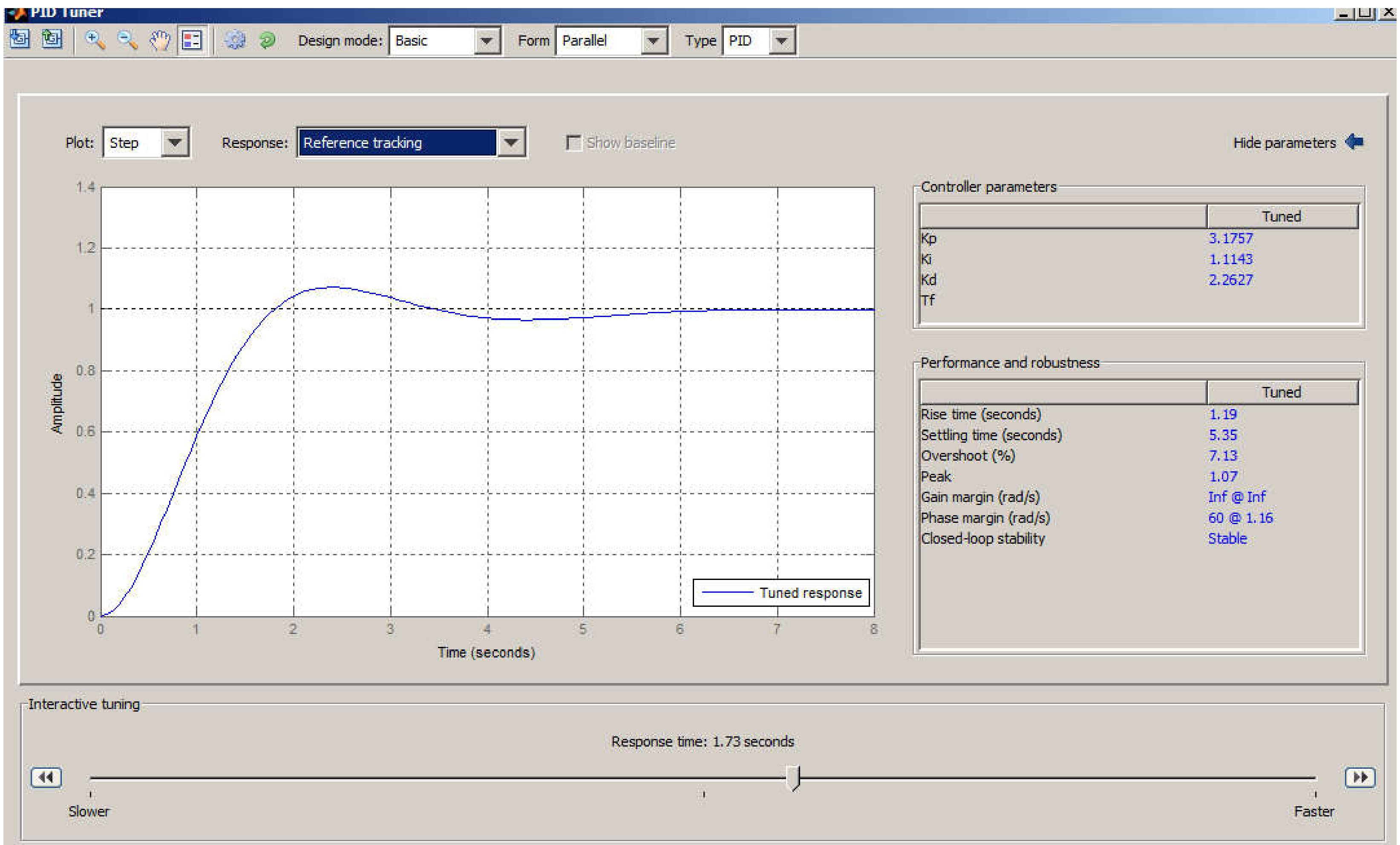
PI Controller



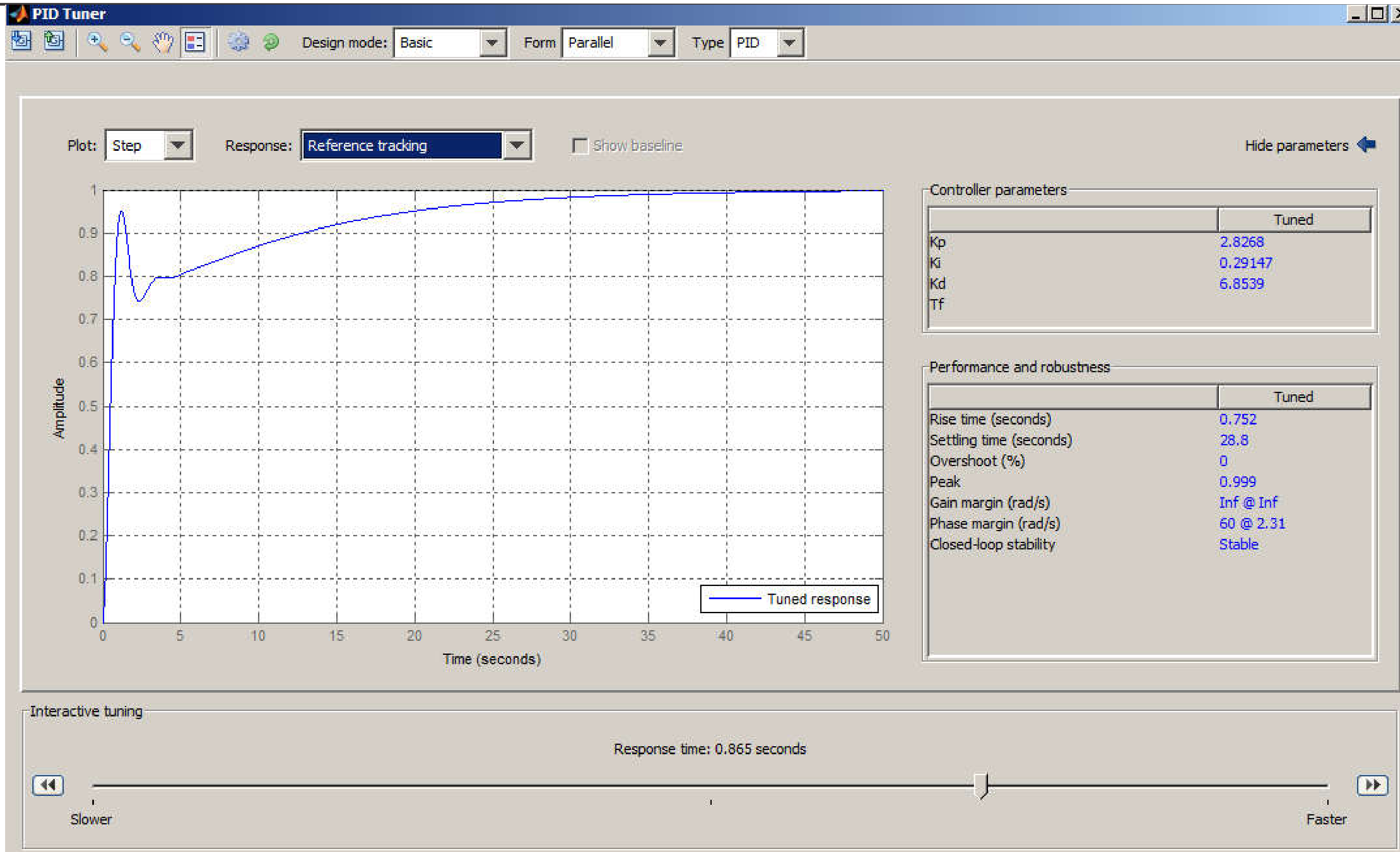
PI Controller, higher gain



PID Controller



PID Controller, high gain



Summary

- Can now add PID controller to systems defined in frequency domain
- The PID controller can be set up in a variety of different ways
  - Be careful to make sure you don’t compare apples and oranges when utilizing online resources
- Illustrated the impact of a PID controller on a simple system
- A better understanding of PID through a mechanical analogy.
- Presented some basic guidelines for PID tuning.