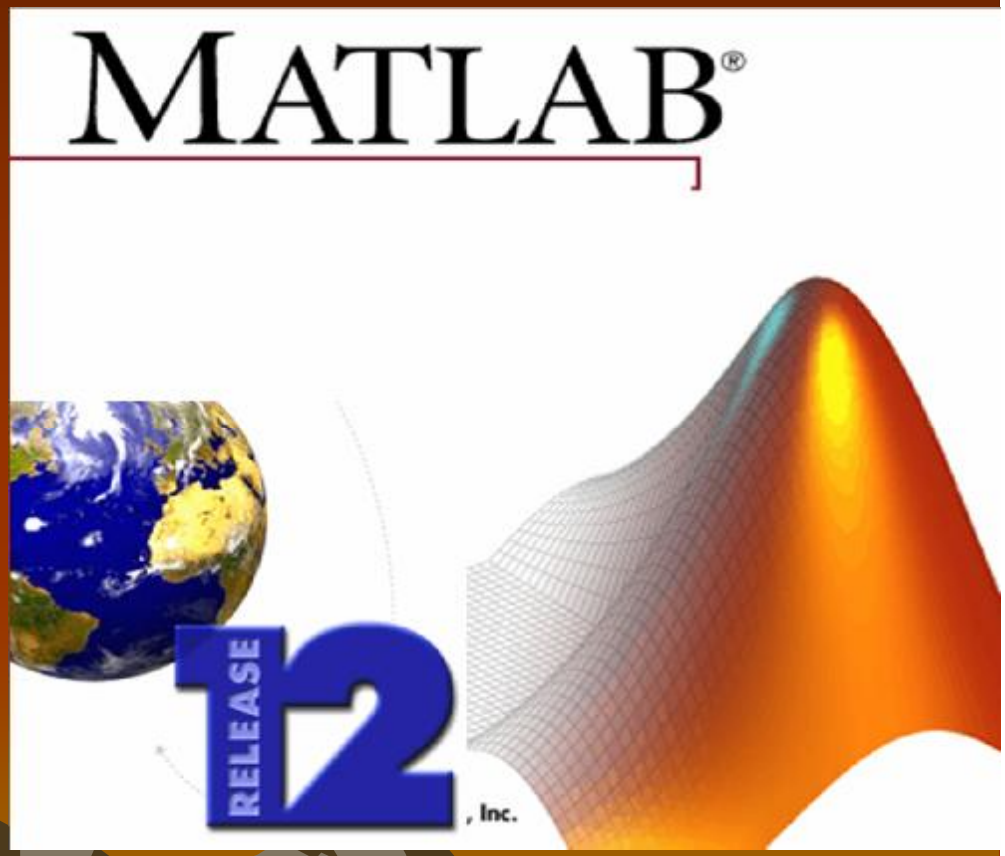
# Tuning PID controller with Introduction to

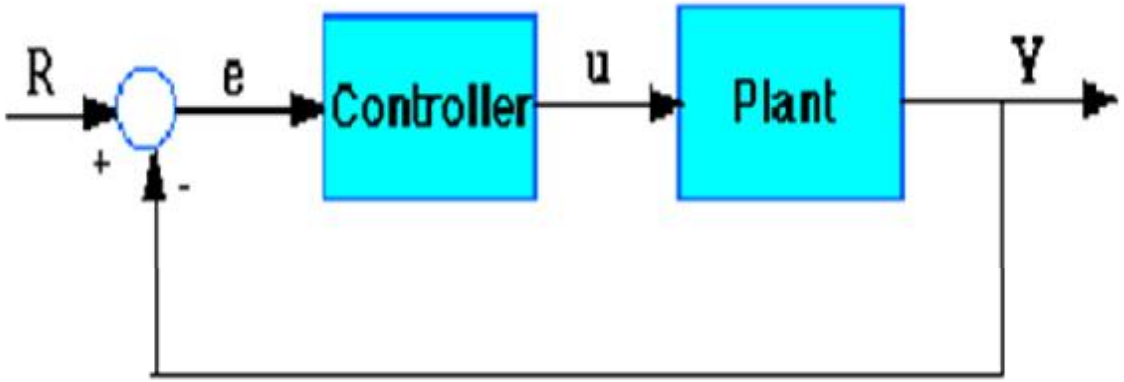



By:

**Dr. Juma Alaydi**

# Introduction

This tutorial will show you the characteristics of the each of proportional (P), the integral (I), and the derivative (D) controls, and how to use them to obtain a desired response. In this tutorial, we will consider the following unity feedback system:



# The three-term controller

The transfer function of the PID controller looks like the following:

$$K_p + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_p s + K_I}{s}$$

- Kp = Proportional gain
- KI = Integral gain
- Kd = Derivative gain

$$G_c(s) = K_p\left[1 + \frac{1}{T_i \cdot s} + T_d \cdot s\right]$$

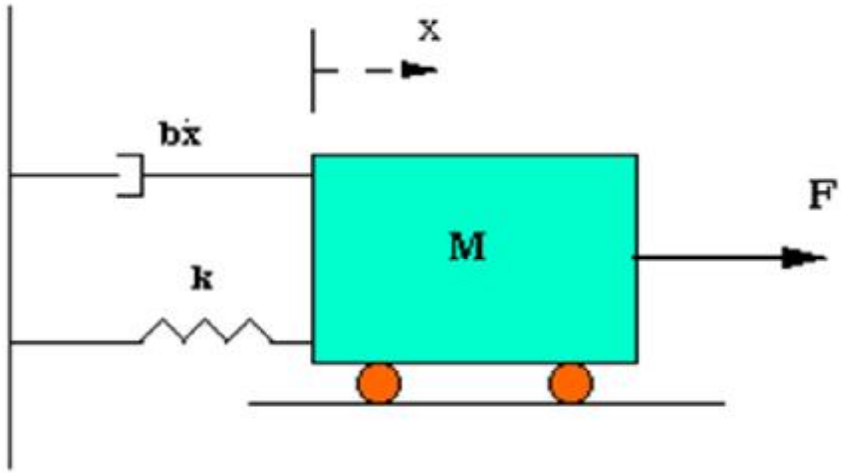
# The characteristics of P, I, and D controllers

A proportional controller (Kp) will have the effect of reducing the rise time and will reduce but never eliminate the steady-state error. An integral control (Ki) will have the effect of eliminating the steady-state error, but it may make the transient response worse. A derivative control (Kd) will have the effect of increasing the stability of the system, reducing the overshoot, and improving the transient response. Effects of each of controllers Kp, Kd, and Ki on a closed-loop system are summarized in the table shown below.

| CL RESPONSE | RISE TIME | OVERSHOOT | SETTLING TIME | S-S ERROR |
|---|---|---|---|---|
| **Kp** | Decrease | Increase | Small Change | Decrease |
| **Ki** | Decrease | Increase | Increase | Eliminate |
| **Kd** | Small Change | Decrease | Decrease | Small Change |

# Example Problem

Suppose we have a simple mass, spring, and damper problem.




The modeling equation of this system is

$$M\ddot{x} + b\dot{x} + kx = F \quad (1)$$

Taking the Laplace transform of the modeling equation (1), we get

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

The transfer function between the displacement X(s) and the input F(s) then becomes

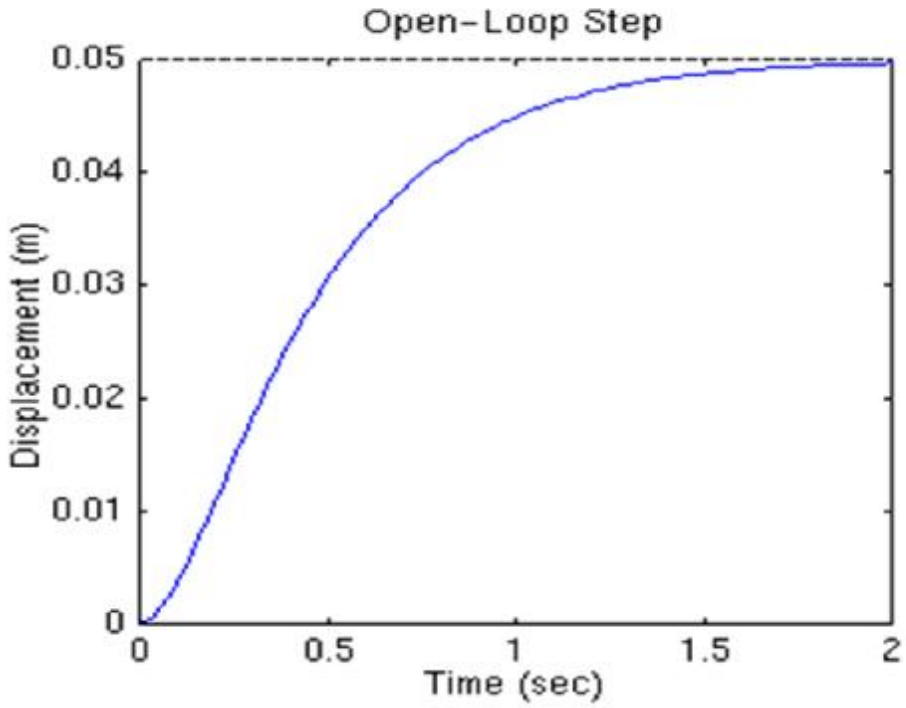
$$\frac{X(s)}{F(s)} = \frac{1}{Ms^2 + bs + k}$$

# Open-loop step response

Let's first view the open-loop step response. Create a new m-file and add in the following code:

```
num=1;
den=[1 10 20];
plant=tf(num,den);
step(plant)
```

Running this m-file in the MATLAB command window should give you the plot shown below.



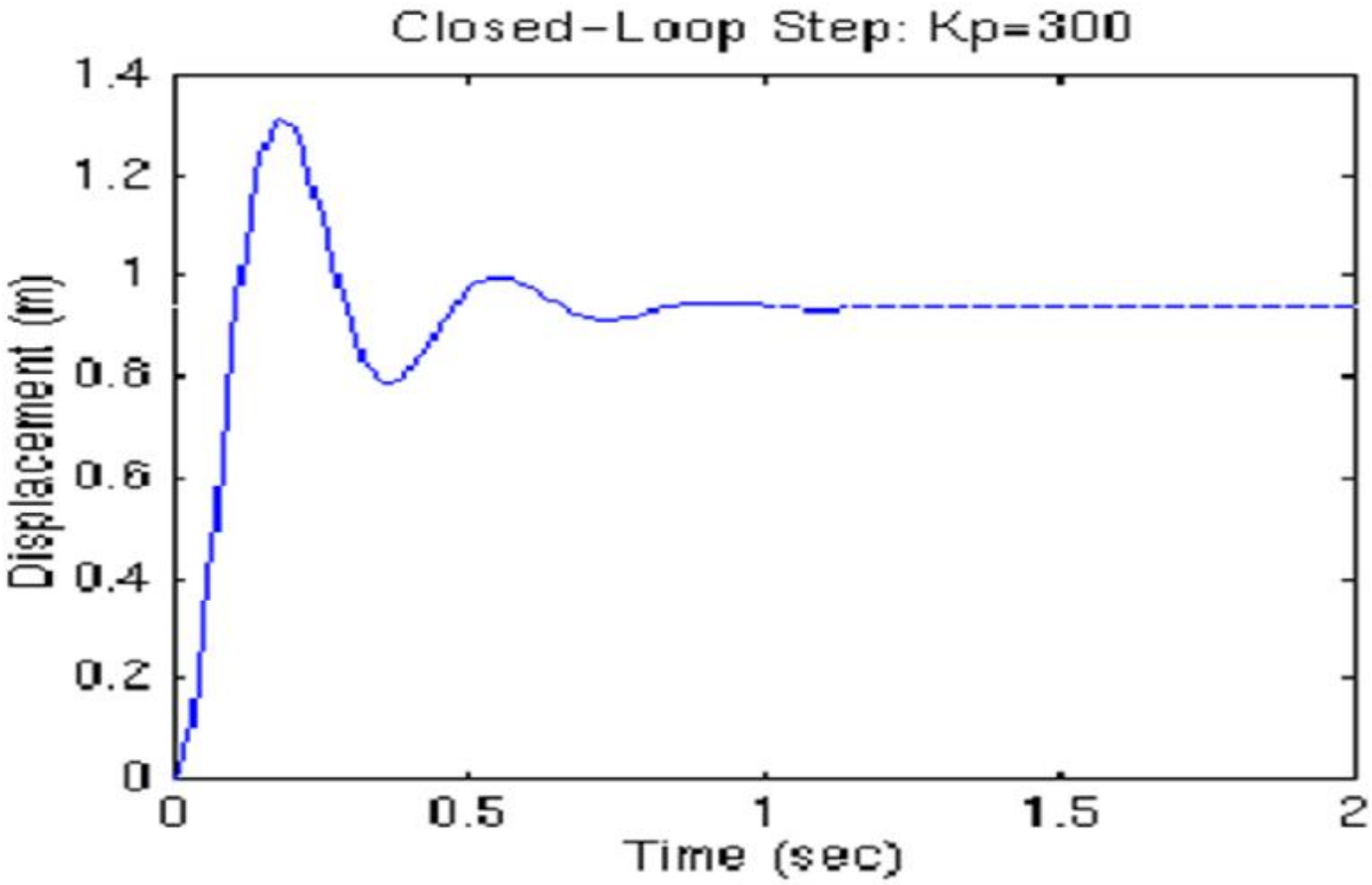
# Proportional control

From the table shown above, we see that the proportional controller (Kp) reduces the rise time, increases the overshoot, and reduces the steady-state error. The closed-loop transfer function of the above system with a proportional controller is:

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

Let the proportional gain (Kp) equal 300 and change the m-file to the following:

```
Kp=300;
contr=Kp;
sys_cl=feedback(contr*plant,1);
t=0:0.01:2;
step(sys_cl,t)
```


Closed-Loop Step: Kp=300
Displacement (m)
Time (sec)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
0.5
1
1.5
2

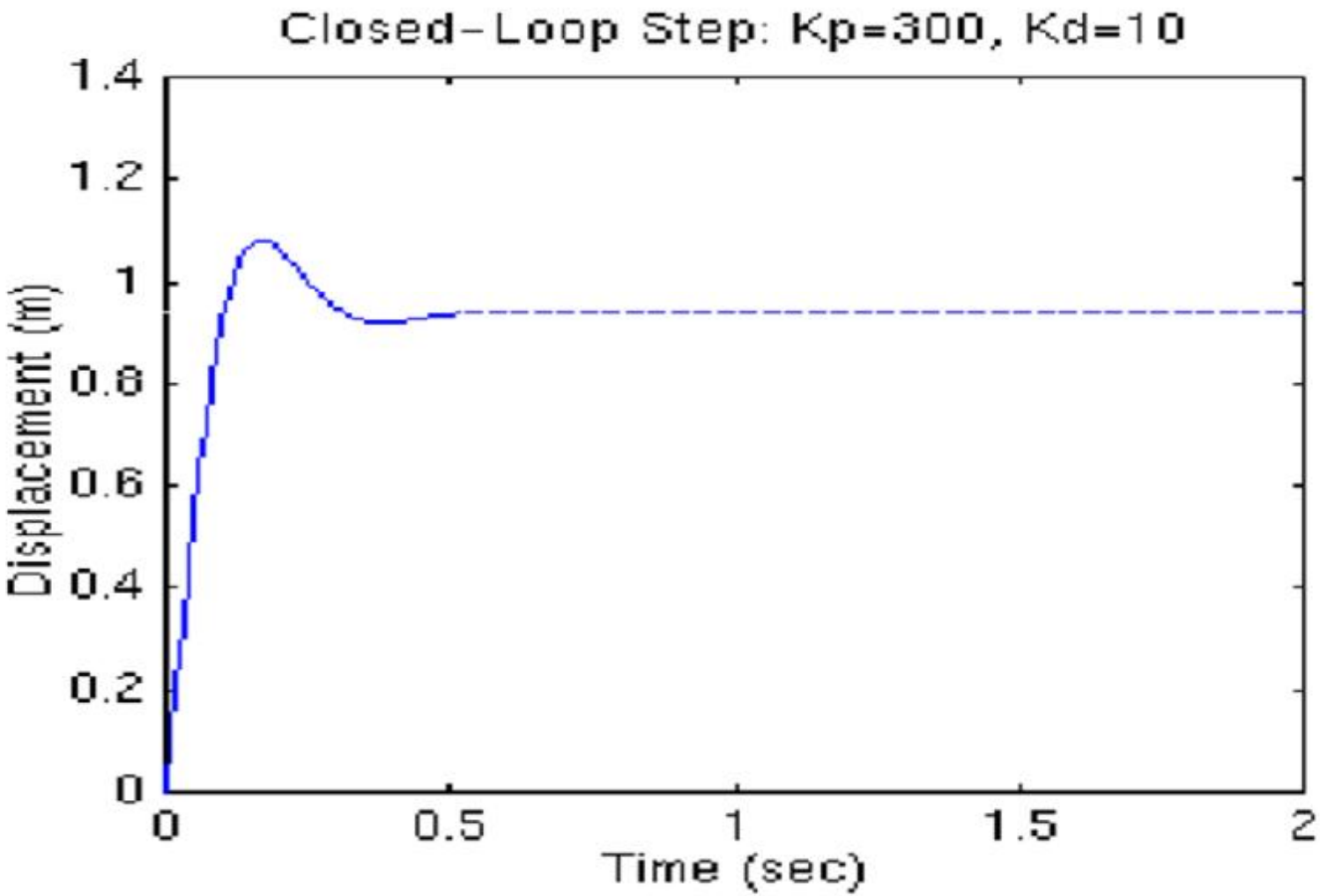
# Proportional-Derivative control

Now, let's take a look at a PD control. From the table shown above, we see that the derivative controller (Kd) reduces both the overshoot and the settling time. The closed-loop transfer function of the given system with a PD controller is:

$$\frac{X(s)}{F(s)} = \frac{K_D s + K_P}{s^2 + (10 + K_D)s + (20 + K_P)}$$

Let Kp equal 300 as before and let Kd equal 10. Enter the following commands into an m-file and run it in the MATLAB command window.

```
Kp=300;
Kd=10;
contr=tf([Kd Kp],1);
sys_cl=feedback(contr*plant,1);
t=0:0.01:2;
step(sys_cl,t)
```


Closed-Loop Step: Kp=300, Kd=10
Displacement (m)
Time (sec)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
0.5
1
1.5
2

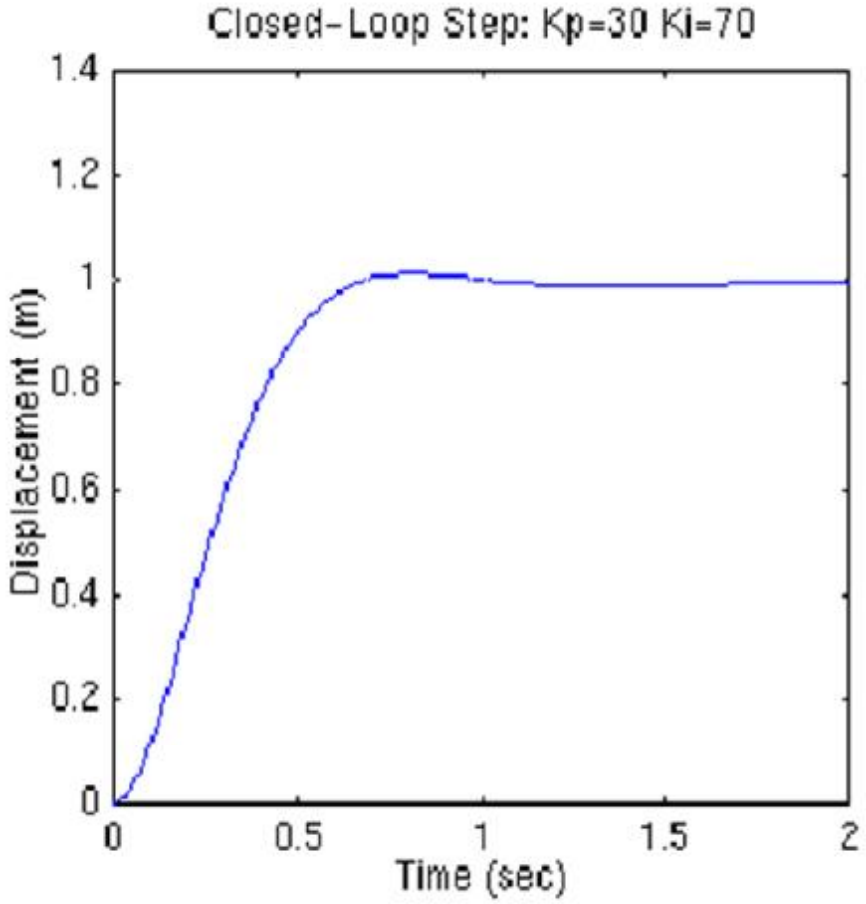
# Proportional-Integral control

Before going into a PID control, let's take a look at a PI control. From the table, we see that an integral controller (Ki) decreases the rise time, increases both the overshoot and the settling time, and eliminates the steady-state error. For the given system, the closed-loop transfer function with a PI control is:

$$\frac{X(s)}{F(s)} = \frac{K_P s + K_I}{s^3 + 10s^2 + (20 + K_P)s + K_I}$$

Let's reduce the Kp to 30, and let Ki equal 70. Create an new m-file and enter the following commands.

```
Kp=30;
Ki=70;
contr=tf([Kp Ki],[1 0]);
sys_cl=feedback(contr*plant,1);
t=0:0.01:2;
step(sys_cl,t)
```




We have reduced the proportional gain (Kp) because the integral controller also reduces the rise time and increases the overshoot as the proportional controller does (double effect). The above response shows that the integral controller eliminated the steady-state error.

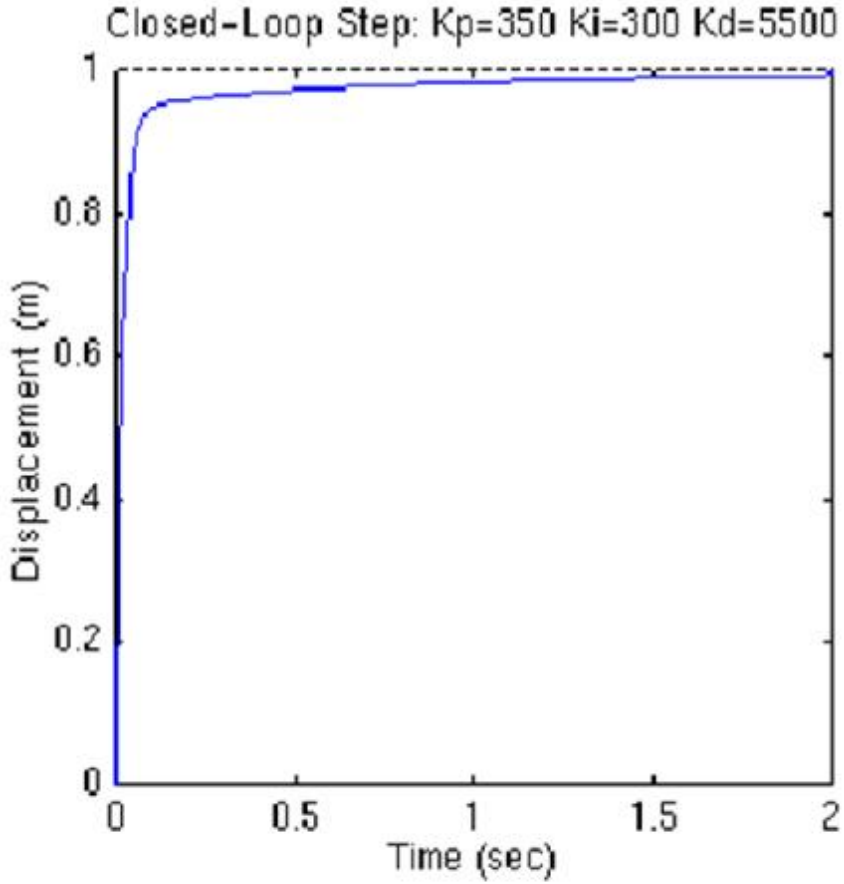
# Proportional-Integral-Derivative control

Now, let's take a look at a PID controller. The closed-loop transfer function of the given system with a PID controller is:

$$\frac{X(s)}{F(s)} = \frac{K_D s^2 + K_P s + K_I}{s^3 + (10 + K_D)s^2 + (20 + K_P)s + K_I}$$

After several trial and error runs, the gains Kp=350, Ki=300, and Kd=50 provided the desired response. To confirm, enter the following commands to an m-file and run it in the command window. You should get the following step response.

```
Kp=350;
Ki=300;
Kd=50;
contr=tf([Kd Kp Ki],[1 0]);
sys_cl=feedback(contr*plant,1);
t=0:0.01:2;
step(sys_cl,t)
```




Now, we have obtained a closed-loop system with no overshoot, fast rise time, and no steady-state error.

# General tips for designing a PID controller

When you are designing a PID controller for a given system, follow the steps shown below to obtain a desired response.

1. Obtain an open-loop response and determine what needs to be improved
2. Add a proportional control to improve the rise time
3. Add a derivative control to improve the overshoot
4. Add an integral control to eliminate the steady-state error
5. Adjust each of Kp, Ki, and Kd until you obtain a desired overall response. You can always refer to the table shown in this "PID Tutorial" page to find out which controller controls what characteristics.

**Lastly**, please keep in mind that you do not need to implement all three controllers (proportional, derivative, and integral) into a single system, if not necessary. For example, if a PI controller gives a good enough response (like the above example), then you don't need to implement a derivative controller on the system. Keep the controller as simple as possible.

# Consider the following configuration

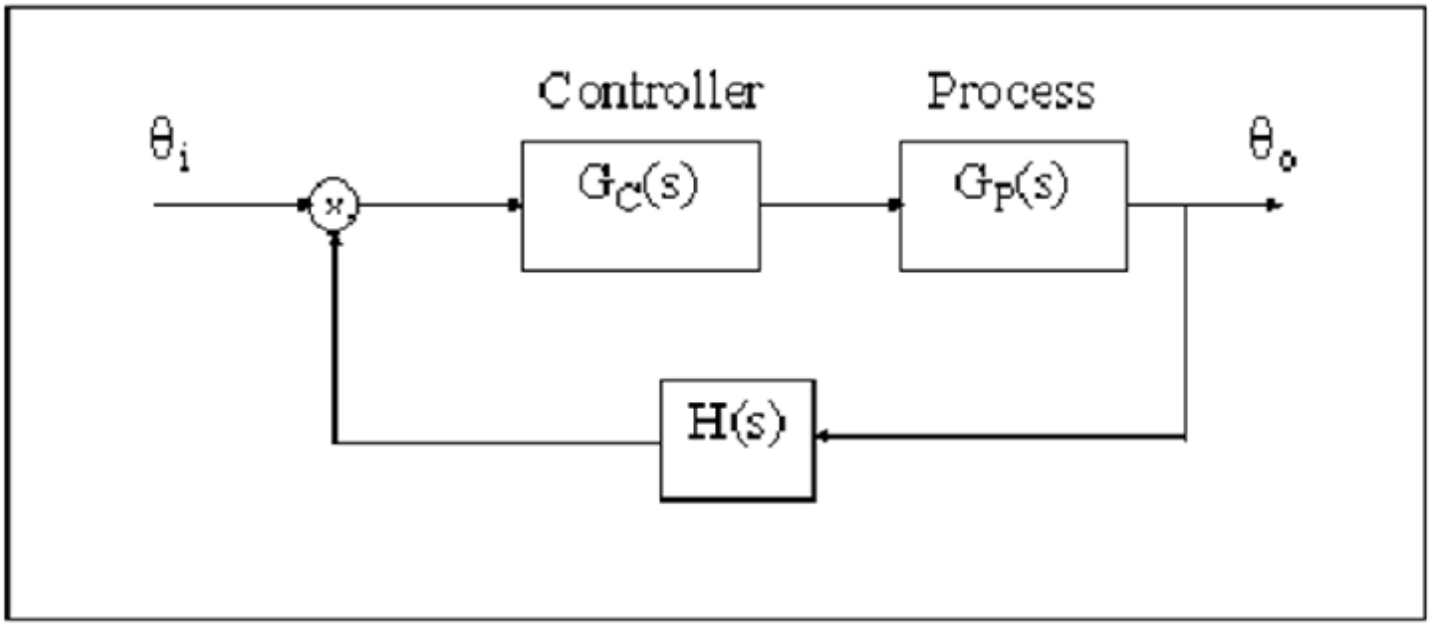



Figure 1

The design specifications are:

- Zero steady state error
- Settling time within 5 seconds
- Rise time within 2 seconds
- Only some overshoot permitted

# Tuning methods

As mentioned before, the set up procedure or tuning of a controller can be tedious. One approach is to use a technique which was developed in the 1950's but which has stood the test of time and is still used today. This is known as the Ziegler Nichols tuning method.

## Ziegler Nichols Tuning Method

The procedure is as follows:

1. Select proportional control alone
2. Increase the value of the proportional gain until the point of instability is reached (sustained oscillations), the critical value of gain, $\mathbf{K_c}$, is reached.
3. Measure the period of oscillation to obtain the critical time constant, $\mathbf{T_c}$.

| Control | $K_p$ | $T_i$ | $T_d$ |
|---|---|---|---|
| P only | **0.5** $K_c$ | | |
| PI | **0.45** $K_c$ | **0.833** $T_c$ | |
| PID tight control | **0.6** $K_c$ | **0.5** $T_c$ | **0.125** $T_c$ |
| PID some overshoot | **0.33** $K_c$ | **0.5** $T_c$ | **0.33** $T_c$ |
| PID no overshoot | **0.2** $K_c$ | **0.3** $T_c$ | **0.5** $T_c$ |

Table 1

Assume the transfer function of the road vehicle is:

$$G_p(s) = \frac{40}{2 \cdot s^3 + 10 \cdot s^2 + 82 \cdot s + 10}$$

$$H(s) = 1$$

Now form the closed loop transfer function with proportional gain K and increase the gain up to the point of instability. From this, the response $K_C$ and $T_C$ are obtained which enables the calculation of the PID parameters (Table1 above). Apply these to the closed loop transfer function.
Finally obtain the response and compare it with the design specification.

$$\frac{\theta_o}{\theta_i} = \frac{K \cdot G_p(s)}{1 + K \cdot G_p(s)}$$

$$G_p(s) = \frac{40}{2 \cdot s^3 + 10 \cdot s^2 + 82 \cdot s + 10}$$

$$\frac{\theta_o}{\theta_i} = \frac{K \cdot 40}{2 \cdot s^3 + 10 \cdot s^2 + 82 \cdot s + (10 + K \cdot 40)}$$

**Increasing the gain**

- Increase the gain by steps of one in a recommended range of 7 to 11.
- Calculate the values in the Closed Loop Transfer Function. (above equations)
- Define the transfer functions for MATLAB.
- Obtain the step responses.

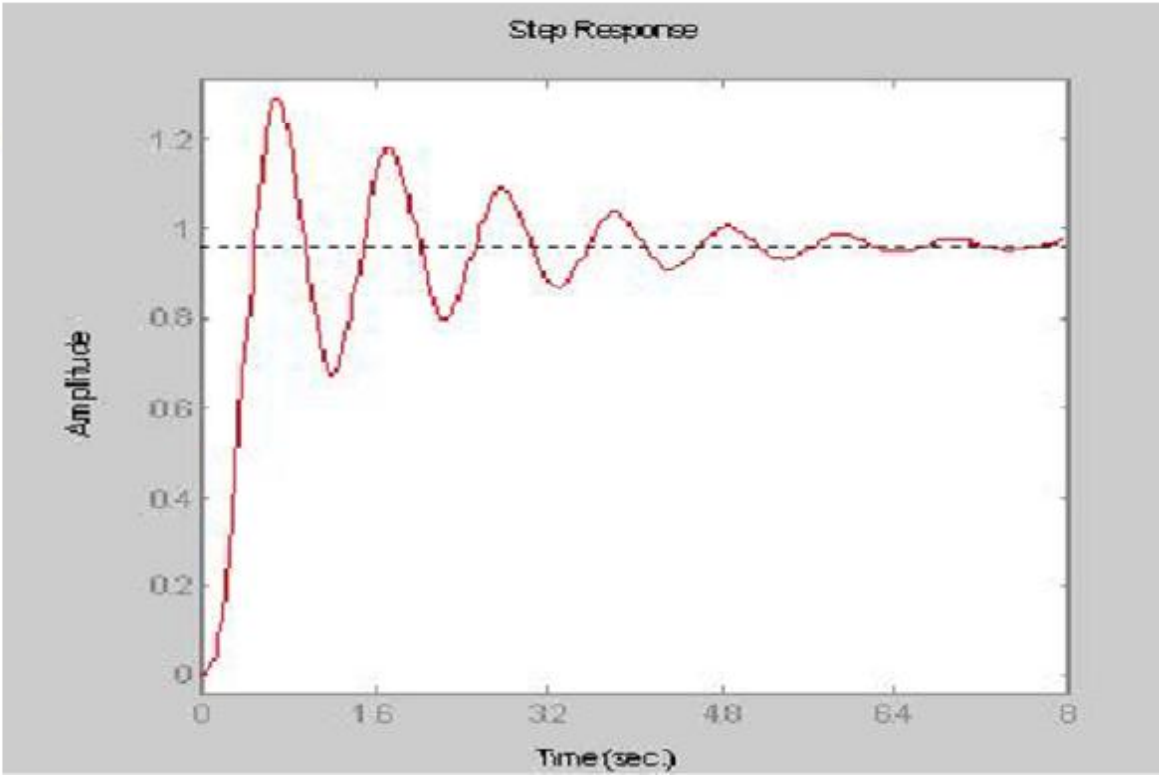
Gradually increase the gain until the point of sustained oscillations is reached (instability).

when K=7

Transfer function is:

$$\frac{280}{2 \cdot s^3 + 10 \cdot s^2 + 82 \cdot s + 290}$$

This gives the step response below.

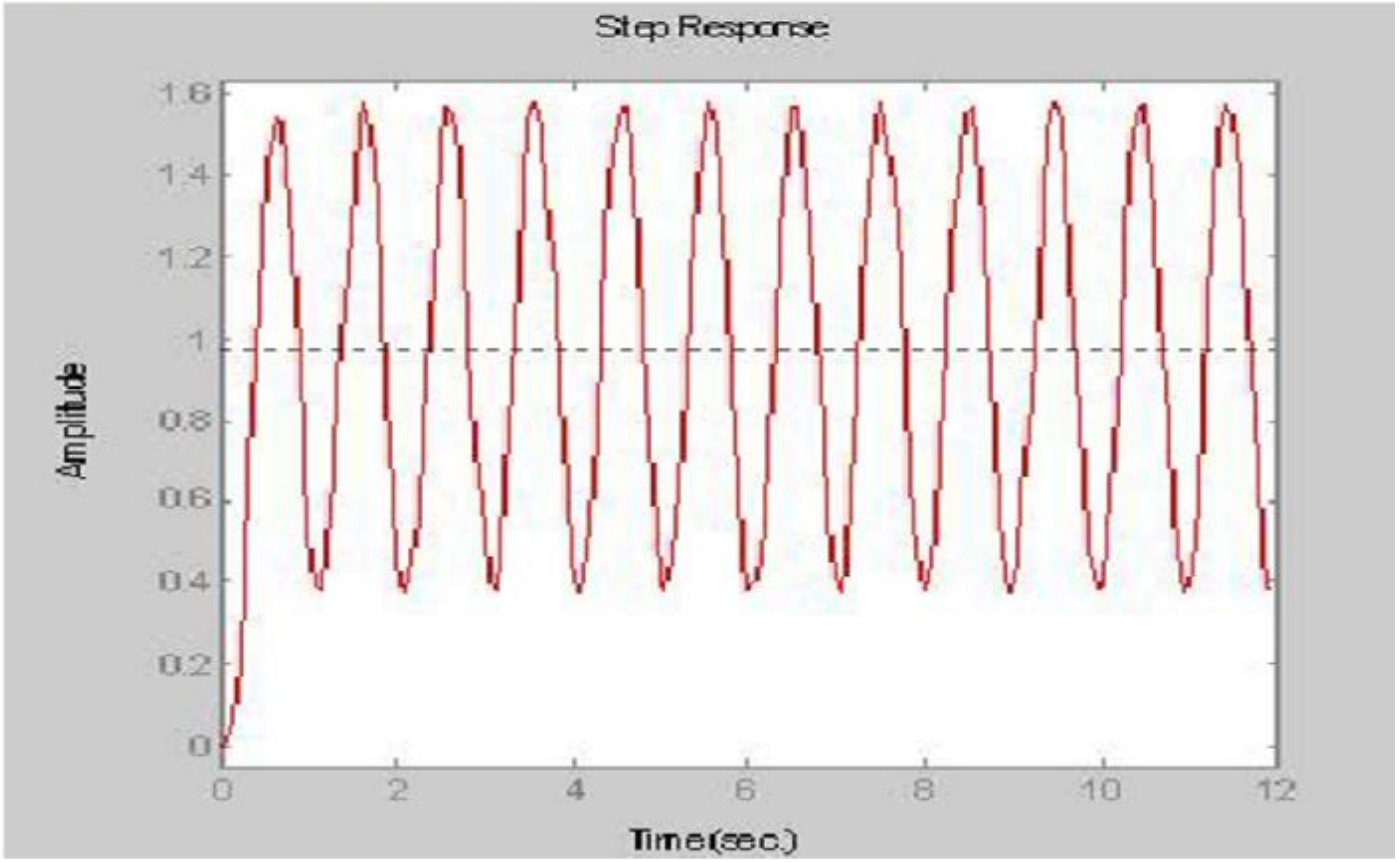


$K_C=10$

Transfer function is:

$$\frac{400}{2\cdot s^3+10\cdot s^2+82\cdot s+410}$$

This gives the step response below.



This is obviously the point of sustained oscillation.
Therefore

$K_C = 10$
$T_C = 1$ ( time of one period )

Obtain PID parameters from Table 1 (above).

$K_P = 3.3$
$T_i = 0.5$
$T_d = 0.33$

Now form the closed loop transfer function with the PID controller and the process.

Controller transfer function:

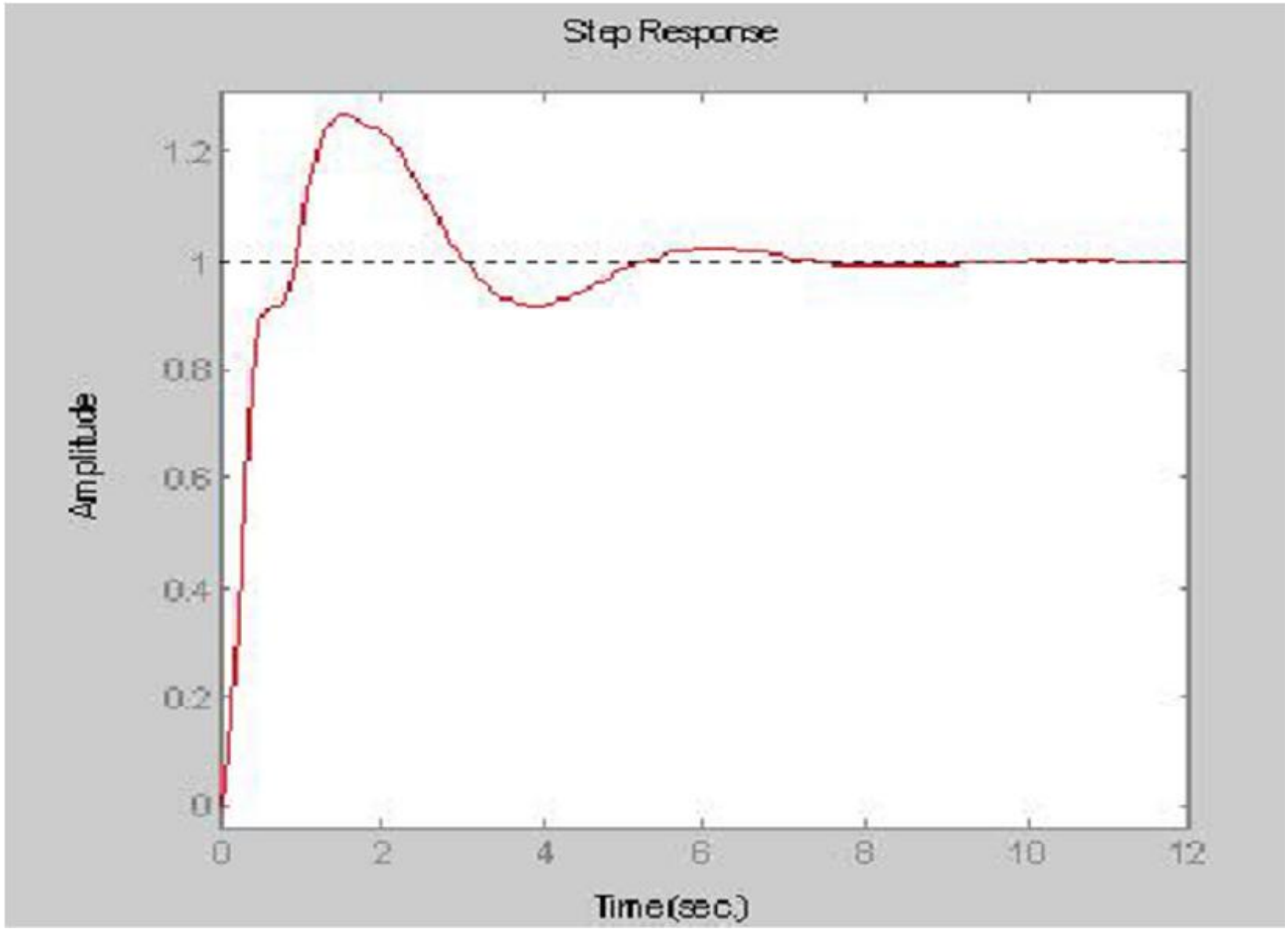
$$G_c(s) = \frac{K_p(T_i \cdot T_d \cdot s^2 + T_i s + 1)}{T_i s}$$

Process transfer function:

$$G_p(s) = \frac{40}{2 \cdot s^3 + 10 \cdot s^2 + 82 \cdot s + 10}$$

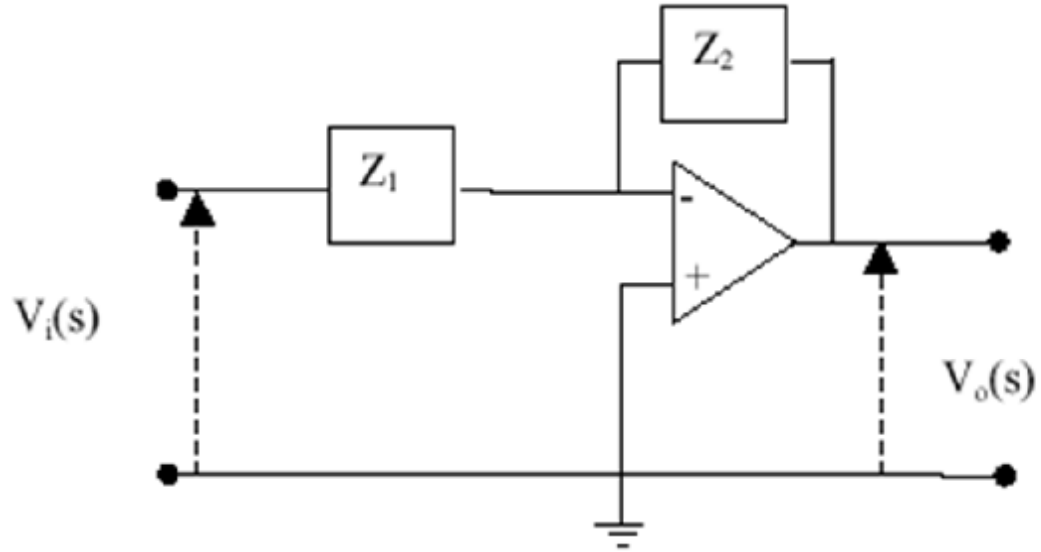
Now replace the variables, close the feedback loop and obtain the closed loop transfer function.

$$\frac{\theta_i}{\theta_o} = \frac{21.78 \cdot s^2 + 66 \cdot s + 132}{s^4 + 5 \cdot s^3 + 62.78 \cdot s^2 + 71 \cdot s + 132}$$


Step Response
Amplitude
Time (sec.)
0
0.2
0.4
0.6
0.8
1
1.2
0
2
4
6
8
10
12

**IMPLEMENTATION**

The proportional compensator is implemented with only an amplifier. The PI and PID need a compensating circuit in addition to the amplifier (which serves to supply the needed power). Suppose that the amplifier gain = 1 for the PI and PID compensators, then the compensating circuits must equal $G_c(s)$. The easiest way to model the PI and PID circuits is to first consider a generic op amp circuit

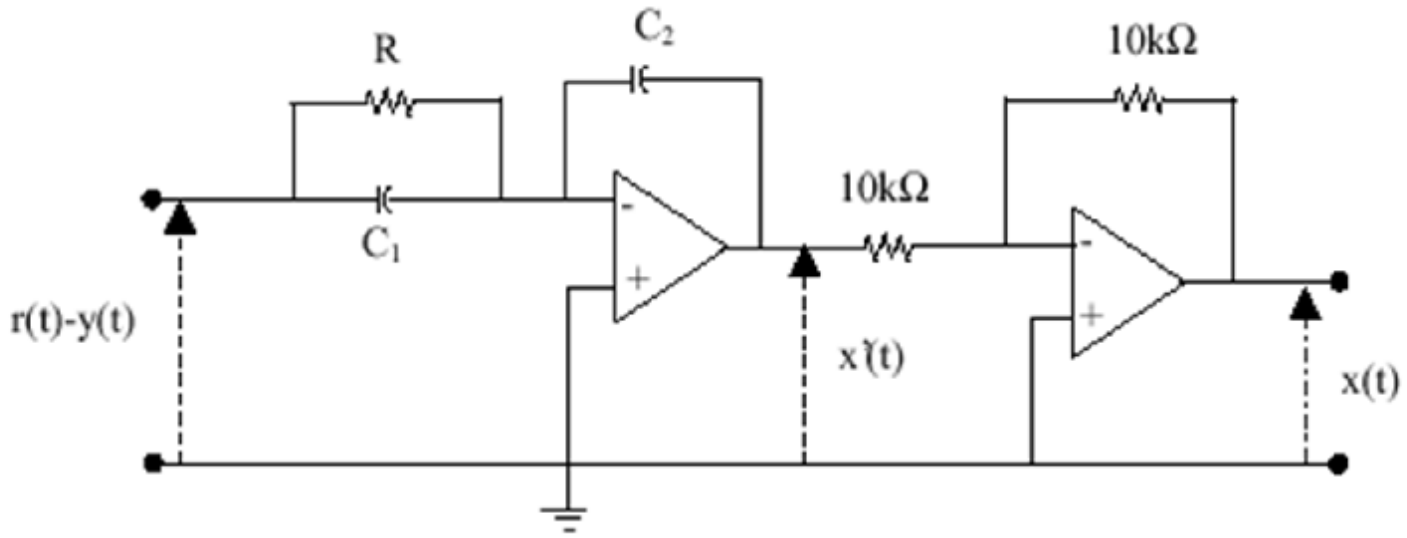



where $Z_1$ and $Z_2$ are impedances. A KVL around the leftmost loop yields: $I(s) = {V_i(s)}/{Z_1}$. The current through $Z_1$ must equal the current through $Z_2$. A KVL around the rightmost loop yields: $I(s) = {-V_o(s)}/{Z_2}$. Equating the expressions for I(s) and solving for the transfer functions yields:

$$\frac{V_o(s)}{V_i(s)} = \frac{-Z_2}{Z_1}$$

PI Circuit:

The PI circuit is composed of two cascaded generic single op amp circuits.



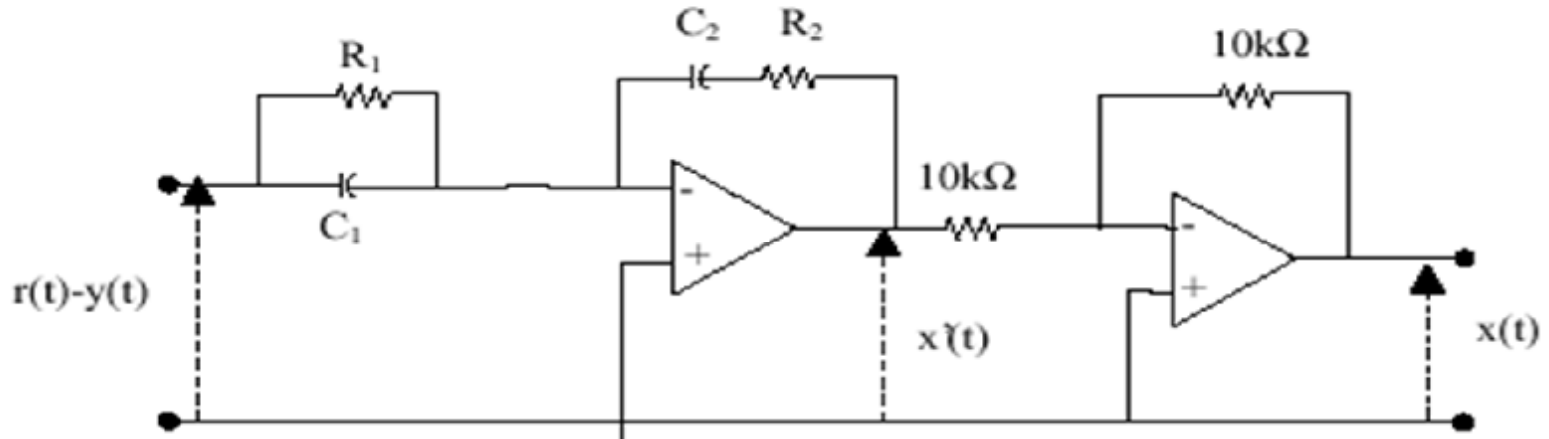

In the first stage, $Z_1 = \frac{R}{RC_1 s + 1}$ and $Z_2 = \frac{1}{C_2 s}$ so $\frac{X'(s)}{R(s) - Y(s)} = -\frac{RC_1 s + 1}{C_2 R s}$. The second stage acts like an inverting circuit where $\frac{X(s)}{X'(s)} = -1$. Combining yields:

$$G_c(s) = \frac{RC_1 s + 1}{RC_2 s}$$

Equating this expression to the desired PI controller, $G_c(s) = \frac{17(s + 0.3923)}{s} = \frac{2.549s + 1}{0.15s}$, shows that there are an infinite number of values for the parameters that give the PI controller. For example, choose R = 10kΩ, then $C_1$=255μf and $C_2$=15μf.

PID Circuit:

Again, this circuit can be treated like two cascaded single op amp circuits.




In the first stage, $Z_1 = \dfrac{R_1}{R_1C_1s+1}$ and $Z_2 = \dfrac{R_2C_2s+1}{C_2s}$ so $\dfrac{X'(s)}{R(s)-Y(s)} = -\dfrac{(R_1C_1s+1)(R_2C_2s+1)}{C_2R_1s}$.

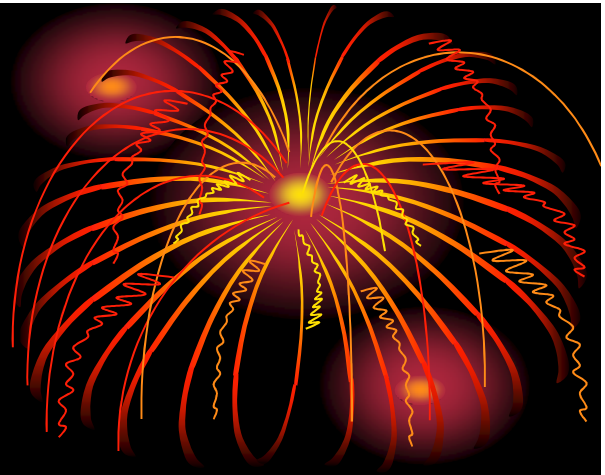
The second stage acts like an inverting circuit where $\dfrac{X(s)}{X'(s)} = -1$. Combining yields:

$$G_c(s) = \frac{(R_1C_1s+1)(R_2C_2s+1)}{C_2R_1s}$$

Equating this expression to the desired PID controller,

$$G_c(s) = \frac{5(s+0.3923)(s+3.927)}{s} = \frac{(2.55s+1)(0.255s+1)}{0.13s}$$

and solving for the parameters shows an infinite number of possibilities. Choose $R_1$=10kΩ, then $C_1$ = 255μf, $C_2$ = 13μ and $R_2$ = 19.62kΩ.



# Thank You

# Questions