ELECTRICAL ENGINEERING DEPARTMENT

Control Systems

Lab Manual

Student Name:	______________________________
Registration Number:	______________________________
Section:	______________________________

UNIVERSITY OF ENGINEERING AND TECHNOLOGY, LAHORE

Important Instructions

1. Every student should have lab manual in the lab; otherwise there will no evaluation and attendance.
2. Those students who have Laptop must bring it in the lab.
3. Every student should fill his/her own manual in the lab complete in all respects, otherwise there will be deduction of marks, and no excuse will be accepted.
4. Student should read the manual before coming to the lab.
5. Every student will have to submit simulation assignments in individual. Assignments after due date will not be accepted.
6. Waveforms should be made with pencils preferably and should be on proper scale. Plot at least two cycles of each waveform.
7. Every student must have text book in lab.
8. There will be a term project and number of simulation/hardware assignments for this lab.
9. You can use Matlab or LabView in your assignments.
10. All the material related to lab and theory will be available at http://alinspiron.weebly.com. No announcements will be made in class, keep visiting this link.

Experiment No. 6
Introduction to PID Controller

Pre-Lab Reading:

Visit this link to get idea about experiment

http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID

Objectives:

When you have completed this assignment you will:

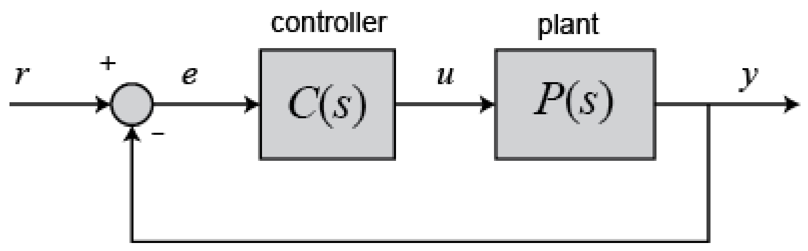
- Learn closed loop systems and improve transient and steady state responses using PID controller.

Introduction: PID Controller Design [1]

In this tutorial we will introduce a simple yet versatile feedback compensator structure, the Proportional-Integral-Derivative (PID) controller. We will discuss the effect of each of the pid parameters on the closed-loop dynamics and demonstrate how to use a PID controller to improve the system performance.

PID Overview

In this tutorial, we will consider the following unity feedback system:



The output of a PID controller, equal to the control input to the plant, in the time-domain is as follows:

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_p \frac{de}{dt} \quad (1)$$

First, let's take a look at how the PID controller works in a closed-loop system using the schematic shown above. The variable (e) represents the tracking error, the difference between the desired input value (r) and the actual output (y). This error signal (e) will be sent to the PID controller, and the controller computes both the derivative and the integral of this error signal. The control signal (u) to the plant is equal to the proportional gain (K_p) times the magnitude of the error plus the integral gain (K_i) times the integral of the error plus the derivative gain (K_d) times the derivative of the error.

This control signal (u) is sent to the plant, and the new output (y) is obtained. The new output (y) is then fed back and compared to the reference to find the new error signal (e). The controller takes this new error signal and computes its derivative and it's integral again, ad infinitum.

The transfer function of a PID controller is found by taking the Laplace transform of Eq.(1).

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

K_p = Proportional gain K_i = Integral gain K_d = Derivative gain

We can define a PID controller in MATLAB using the transfer function directly, for example:

```
Kp = 1;

Ki = 1;

Kd = 1;

s = tf('s');
C = Kp + Ki/s + Kd*s
```

```
C =

   s^2 + s + 1

  -----------

       s

 Continuous-time transfer function.
```

Alternatively, we may use MATLAB's pid controller object to generate an equivalent continuous-time controller as follows:

```
C = pid(Kp,Ki,Kd)
```

```
C =

              1

  Kp + Ki * --- + Kd * s

             s

  with Kp = 1, Ki = 1, Kd = 1

 Continuous-time PID controller in parallel form.
```

Let's convert the pid object to a transfer function to see that it yields the same result as above:

```
tf(C)
```

```
ans =

   s^2 + s + 1

  -----------

       s

 Continuous-time transfer function.
```

The Characteristics of P, I, and D Controllers

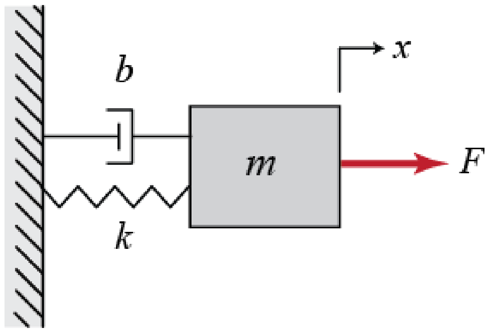
A proportional controller (K_p) will have the effect of reducing the rise time and will reduce but never eliminate the steady-state error. An integral control (K_i) will have the effect of eliminating the steady-state error for a constant or step input, but it may make the transient response slower. A derivative control (K_d) will have the effect of increasing the stability of the system, reducing the overshoot, and improving the transient response.
The effects of each of controller parameters, K_p, K_d, and K_i on a closed-loop system are summarized in the table 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	No Change

Note that these correlations may not be exactly accurate, because K_p, K_i, and K_d are dependent on each other. In fact, changing one of these variables can change the effect of the other two. For this reason, the table should only be used as a reference when you are determining the values for K_i, K_p and K_d.

Example Problem

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



The modeling equation of this system is

$$M\ddot{x} + b\dot{x} + kx = F \tag{3}$$

Taking the Laplace transform of the modeling equation, we get

$$Ms^2X(s) + bsX(s) + kX(s) = F(s) \tag{4}$$

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

$$\frac{X(s)}{F(s)} = \frac{1}{Ms^2 + bs + k} \tag{5}$$

Let

```
M = 1 kg

b = 10 N s/m

k = 20 N/m

F = 1 N
```

Plug these values into the above transfer function

$$\frac{X(s)}{F(s)} = \frac{1}{s^2 + 10s + 20} \qquad (6)$$

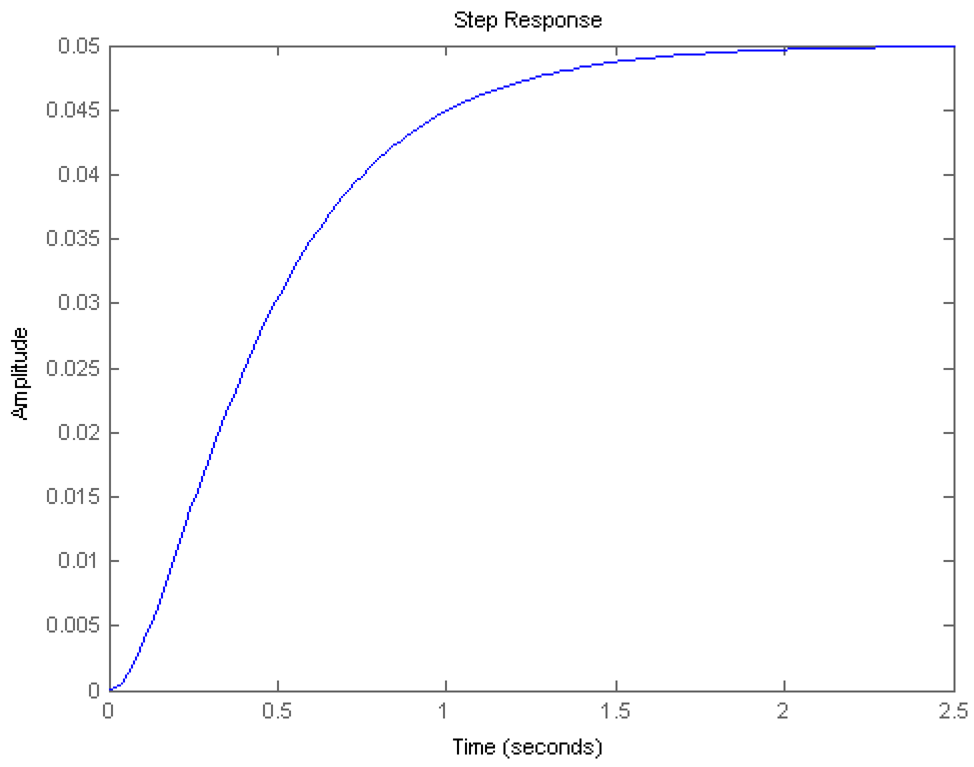
The goal of this problem is to show you how each of K_p, K_i and K_d contributes to obtain

```
Fast rise time

Minimum overshoot

No steady-state error
```

Open-Loop Step Response

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

```
s = tf('s');
P = 1/(s^2 + 10*s + 20);
step(P)
```



The DC gain of the plant transfer function is 1/20, so 0.05 is the final value of the output to an unit step input. This corresponds to the steady-state error of 0.95, quite large indeed. Furthermore, the rise time is about one second, and the settling time is about 1.5 seconds. Let's design a controller that will reduce the rise time, reduce the settling time, and eliminate the steady-state error.

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.

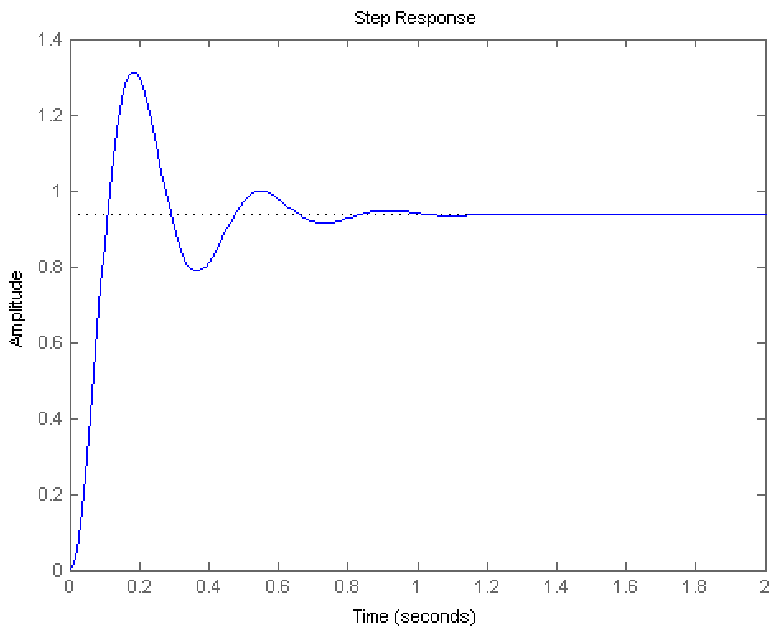
The closed-loop transfer function of the above system with a proportional controller is:

$$\frac{X(s)}{F(s)} = \frac{K_p}{s^2 + 10s + (20 + K_p)} \qquad (7)$$

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

```
Kp = 300;

C = pid(Kp)

T = feedback(C*P,1)

t = 0:0.01:2;

step(T,t)
```

```
C =

  Kp = 300

P-only controller.



T =

        300

  ----------------

  s^2 + 10 s + 320

Continuous-time transfer function.
```



The above plot shows that the proportional controller reduced both the rise time and the steady-state error, increased the overshoot, and decreased the settling time by small amount.

Proportional-Derivative Control

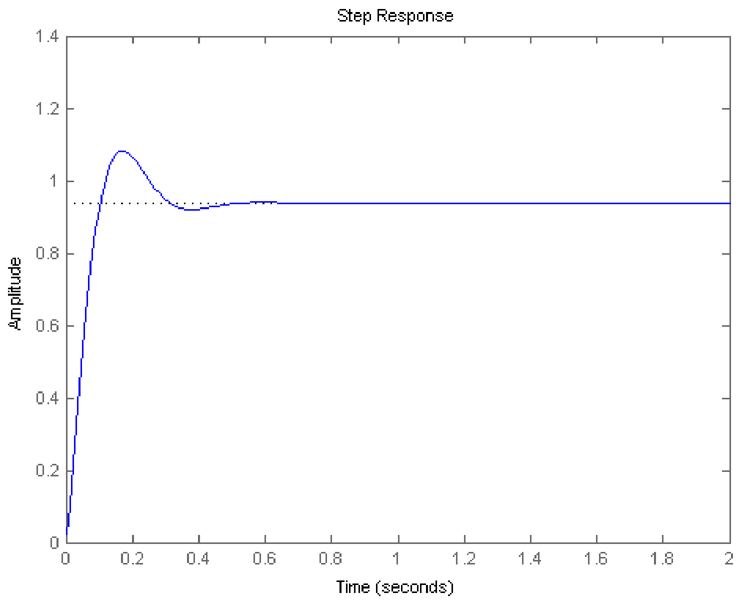
Now, let's take a look at a PD control. From the table shown above, we see that the derivative controller (Kd) reduces both the overshoot and the settling time. The closed-loop transfer function of the given system with a PD controller is:

$$\frac{X(s)}{F(s)} = \frac{K_d s + K_p}{s^2 + (10 + K_d)s + (20 + K_p)} \tag{8}$$

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

```
Kp = 300;

Kd = 10;

C = pid(Kp,0,Kd)

T = feedback(C*P,1)



t = 0:0.01:2;

step(T,t)
```

```
C =



  Kp + Kd * s



  with Kp = 300, Kd = 10

Continuous-time PD controller in parallel form.



T =

     10 s + 300

  ----------------

  s^2 + 20 s + 320

Continuous-time transfer function.
```



This plot shows that the derivative controller reduced both the overshoot and the settling time, and had a small effect on the rise time and the steady-state error.

Proportional-Integral Control

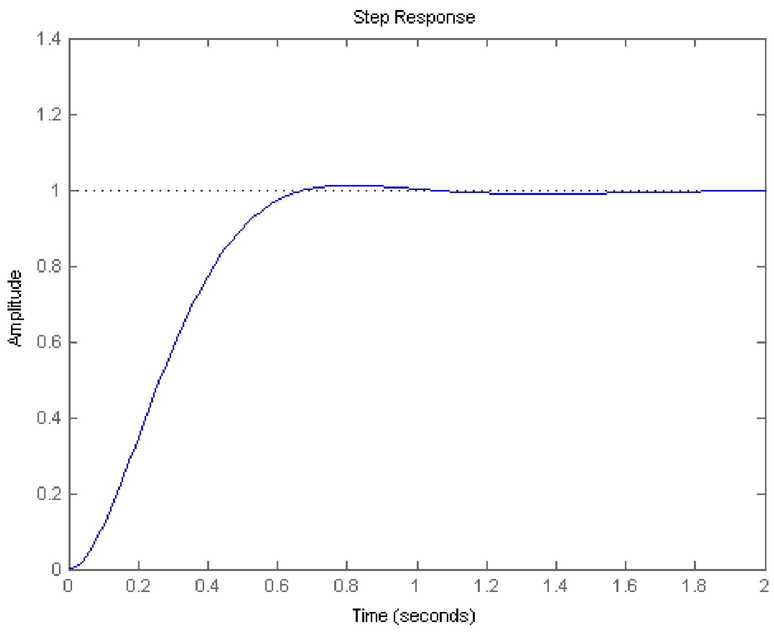
Before going into a PID control, let's take a look at a PI control. From the table, we see that an integral controller (Ki) decreases the rise time, increases both the overshoot and the settling time, and eliminates the steady-state error. For the given system, the closed-loop transfer function with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

$$\frac{X(s)}{F(s)} = \frac{K_p s + K_i}{s^3 + 10s^2 + (20 + K_p s + K_i)} \qquad (9)$$

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

```
Kp = 30;

Ki = 70;

C = pid(Kp,Ki)

T = feedback(C*P,1)



t = 0:0.01:2;

step(T,t)
```

```
C =
              1
  Kp + Ki * ---
              s

  with Kp = 30, Ki = 70

Continuous-time PI controller in parallel form.


T =

         30 s + 70
  ------------------------
  s^3 + 10 s^2 + 50 s + 70

Continuous-time transfer function.
```



Run this m-file in the MATLAB command window, and you should get the following plot. 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

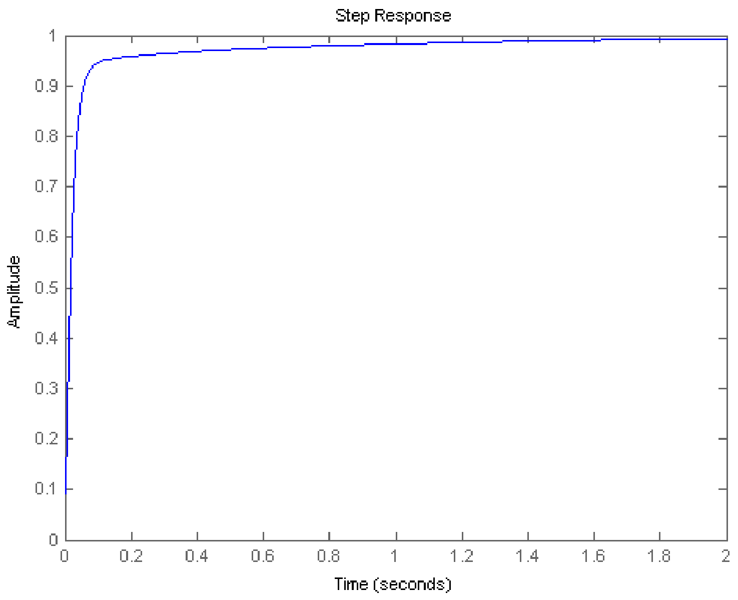
Now, let's take a look at a PID controller. The closed-loop transfer function of the given system with a PID controller is:

$$\frac{X(s)}{F(s)} = \frac{K_d s^2 + K_p s + K_i}{s^3 + (10 + K_d)s^2 + (20 + K_p)s + K_i} \qquad (10)$$

After several trial and error runs, the gains K_p = 350, K_i = 300, and K_d = 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;

C = pid(Kp,Ki,Kd)

T = feedback(C*P,1);



t = 0:0.01:2;

step(T,t)
```

```
C =

             1

  Kp + Ki * --- + Kd * s

             s

  with Kp = 350, Ki = 300, Kd = 50

Continuous-time PID controller in parallel form.
```



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.

Lab Task:

- Perform this experiment using Simulink

Reference:

[1] "Control Tutorials for MATLAB and Simulink - Introduction: PID Controller Design", Ctms.engin.umich.edu, 2016. [Online]. Available: http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID. [Accessed: 06- Mar- 2016].