Linear Control Systems

Lab Experiment 7: Introduction to PID controller

Objective: Study the three term (PID) controller and its effects on the feedback loop response. Investigate 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.

List of Equipment/Software

Following equipment/software is required:

- MATLAB
- LabVIEW

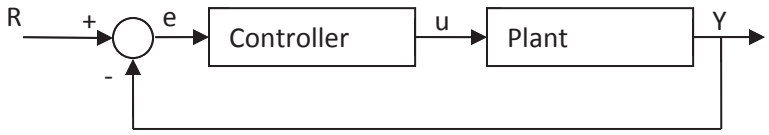
Category Soft - Experiment

Deliverables

A complete lab report including the following:

- Summarized learning outcomes.
- LabVIEW programming files (Block diagram and Front Panel)
- Controller design and parameters for each of the given exercises.

Introduction: Consider the following unity feedback system:



Plant: A system to be controlled.

Controller: Provides excitation for the plant; Designed to control the overall system behavior.

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

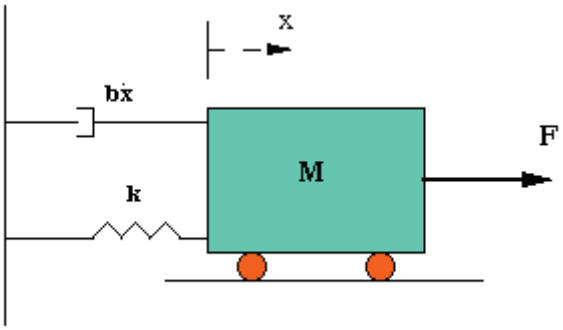
First, let's take a look at how the PID controller works in a closed-loop system using the schematic shown above. The variable (e) represents the tracking error, the difference between the desired input value (R) and the actual output (Y). This error signal (e) will be sent to the PID controller, and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

$$u = K_P e(t) + K_I \int e(t)dt + K_D \frac{de(t)}{dt}$$

This signal (u) will be sent to the plant, and the new output (Y) will be obtained. This new output (Y) will be sent back to the sensor again to find the new error signal (e). The controller takes this new error signal and computes its derivatives and its internal again. The process goes on and on.

Example Problem:

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



The modeling equation of this system is

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

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

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

Let

- M = 1kg
- b = 10 N.s/m
- k = 20 N/m
- F(s) = 1

Plug these values into the above transfer function

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

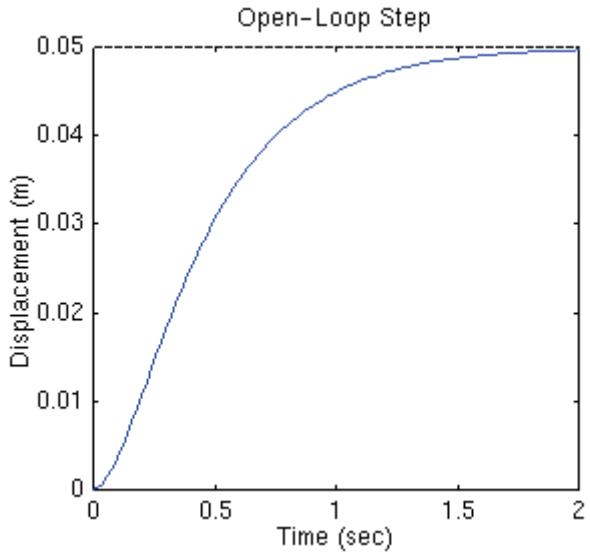
The goal of this problem is to show you how each of K_p, K_i and K_d contributes to obtain

- Fast rise time
- Minimum overshoot
- No steady-state error

Open-loop step response: Let's first view the open-loop step response.

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

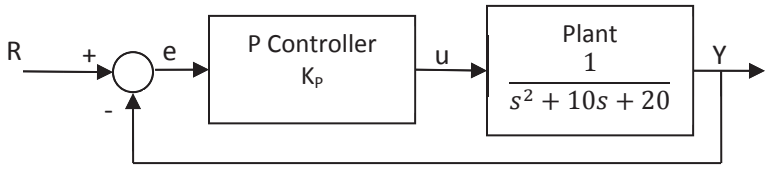
MATLAB command window should give you the plot shown below.



The DC gain of the plant transfer function is 1/20, so 0.05 is the final value of the output to a 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 eliminates the steady-state error.

Proportional control:

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

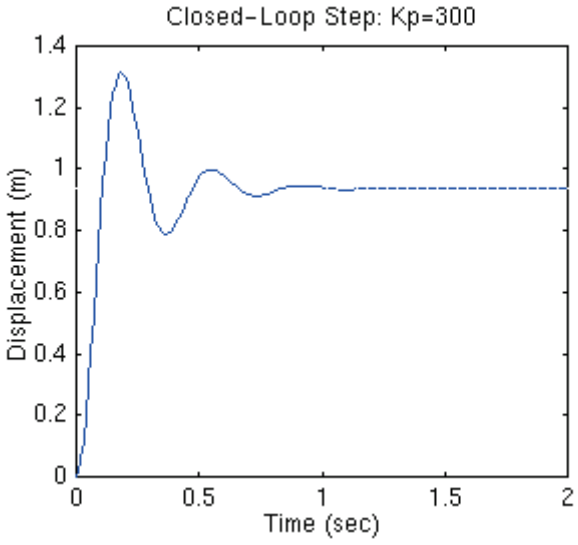


$$\frac{X(s)}{F(s)} = \frac{K_P}{s^2 + 10s + (20 + K_P)}$$

Let the proportional gain (K_P) equal 300:

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

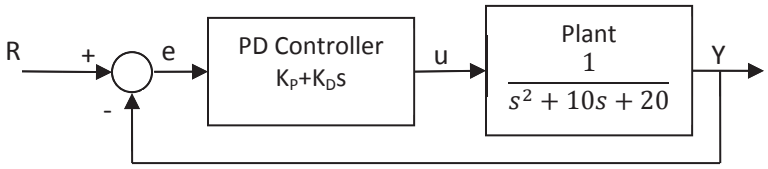
MATLAB command window should give you the following plot.



Note: The MATLAB function called feedback was used to obtain a closed-loop transfer function directly from the open-loop transfer function (instead of computing closed-loop transfer function by hand). 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:

The closed-loop transfer function of the given system with a PD controller is:

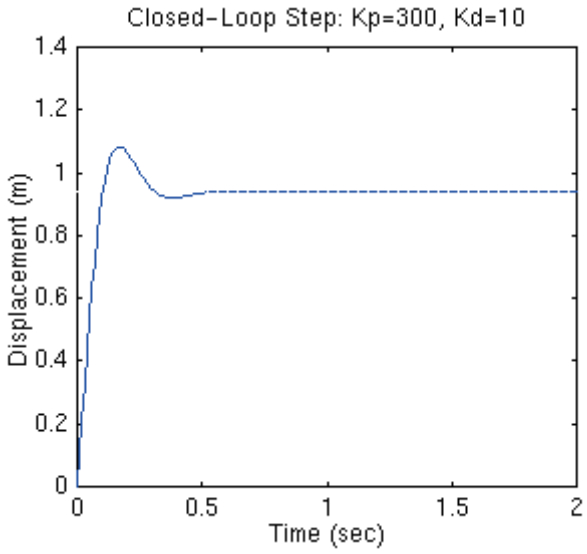


$$\frac{X(s)}{F(s)} = \frac{K_D s + K_P}{s^2 + (10 + K_D)s + (20 + K_P)}$$

Let K_P equal 300 as before and let K_D equal 10.

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

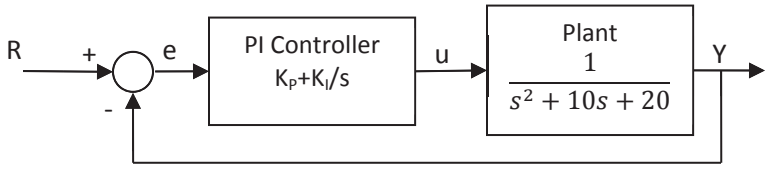
MATLAB command window should give you the following plot.



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. For the given system, the closed-loop transfer function with a PI control is:

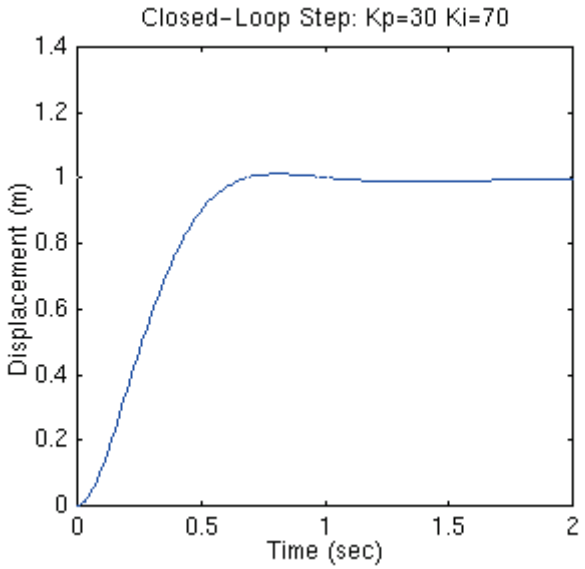


$$\frac{X(s)}{F(s)} = \frac{K_P s + K_I}{s^3 + 10s^2 + (20 + K_P)s + K_I}$$

Let's reduce the K_P to 30, and let K_I equal 70.

```
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)
```

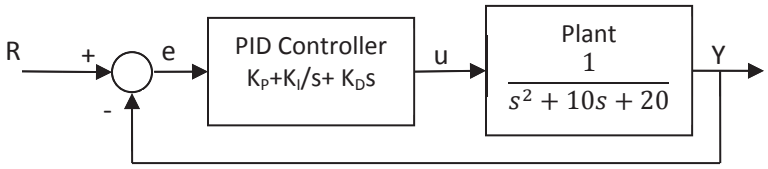
MATLAB command window gives the following plot.



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:

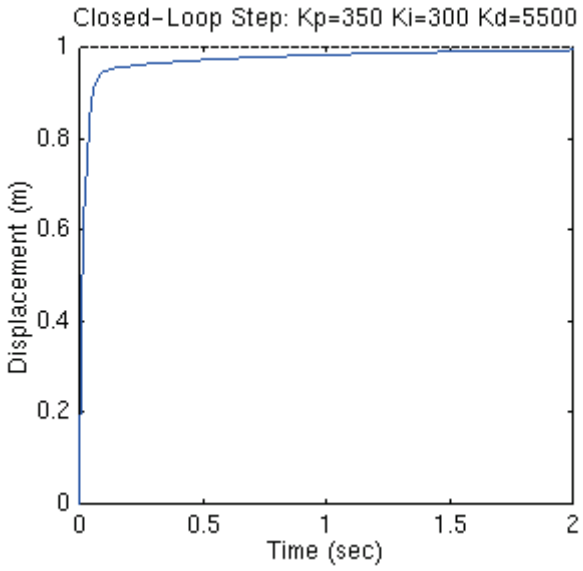
Now, let's take a look at a PID controller. The closed-loop transfer function of the given system with a PID controller is:



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

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.

The characteristics of P, I, and D controllers:

The 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 controller (K_I) will have the effect of eliminating the steady state error, but it may make the transient response worse. A derivative control (K_D) will have the effect of increasing the stability of the system, reducing the overshoot and improving the transient response.

Effect of each controller K_P, K_I and K_D on the closed-loop system are summarized below

CL Response	Rise Time	Overshoot	Settling Time	S-S Error
K_P	Decrease	Increase	Small Change	Decrease
K_I	Decrease	Increase	Increases	Eliminate
K_D	Small Change	Decreases	Decreases	Small Change

Note that these corrections may not be accurate, because K_P, K_I, and K_D are dependent of 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_P, K_I, and K_D.

Exersice:

Consider a process given below to be controlled by a PID controller,

$$G_p(s) = \frac{400}{s(s+48.5)}$$

a) Obtain the unit step response of Gp(s).
b) Try PI controllers with (Kp=2, 10, 100), and Ki=Kp/10. Investigate the unit step response in each case, compare the results and comment.
c) Let Kp=100, Ki=10, and add a derivative term with (Kd=0.1, 0.9, 2). Investigate the unit step response in each case, compare the results and comment.

Based on your results in parts b) and c) above what do you conclude as a suitable PID controller for this process and give your justification.