EXERCISE 6

CONTROL SYSTEMS USING PID CONTROLLERS

Date: ***Reg. No. :***

LAB PREREQUISITES:

Exercise 1 to 5

PREREQUISITE KNOWLEDGE:

Fundamentals of MATLAB and Simulink programming

OBJECTIVES:

The objective of this exercise is to design a PID controller for given system using MATLAB and to study the PID tuning for DC motor using MATLAB SIMULINK.

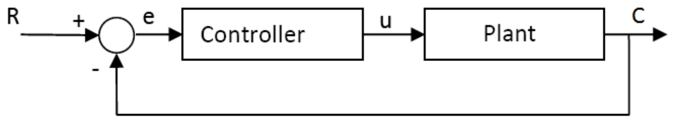
PRELAB:

Determine the constant values K_p, K_i, K_d for the PID controller for the given open loop transfer function which offers 45° at 4 rad/seconds and steady state error for unit ramp input e_{ss}=0.1.

VALUES:

Introduction to Controllers

Consider the following unity feedback system



Plant: A system to be controlled.

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

The three-term controller: The transfer function of the PID controller looks like the following:

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

K_P = Proportional gain

K_I = Integral gain K_D = Derivative gain

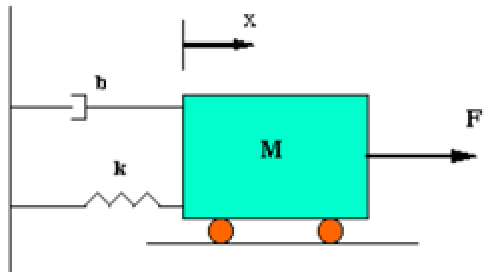
First, let's take a look at how the PID controller works in a closed-loop system using the schematic shown above. The variable (e) represents the tracking error, the difference between the desired input value (R) and the actual output (C). This error signal (e) will be sent to the PID controller, and 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 (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.

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

This signal (u) will be sent to the plant, and the new output (Y) will be obtained. This new output (C) 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 System:

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, 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) = 1N (Unit Step)

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
- No steady-state error
- Minimum overshoot

Open-loop step response:

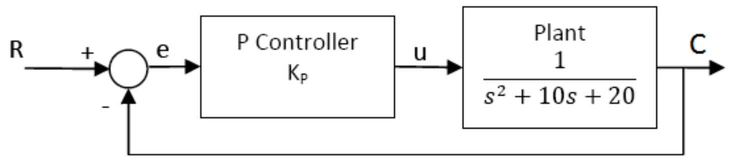
Let's first view the open-loop step response

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

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:



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

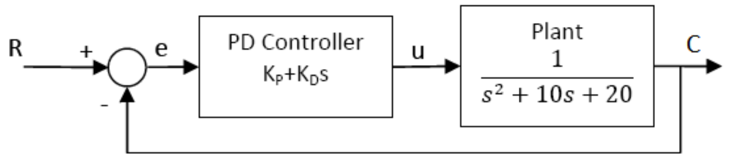
Let the proportional gain (K_P) equal 300:

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

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 plot would show that the proportional controller reduces 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:



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

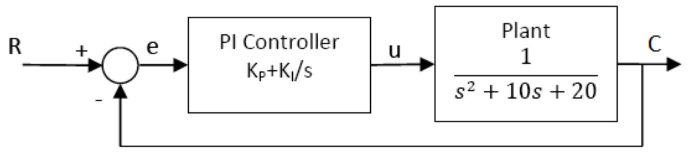
Let KP equal 300 as before and let KD equal 10.

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

The plot would show that the derivative controller reduces both the overshoot and the settling time, and has 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.

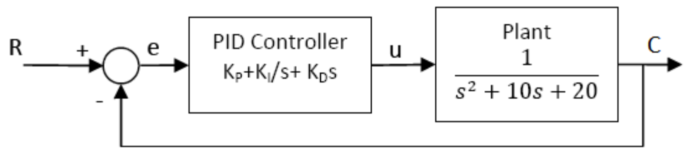
```
Kp=30; Ki=70;
contr=tf([Kp Ki],[1 0]);
sys_cl=feedback(contr*plant,1);
```

```
        t=0:0.01:2;
        step(sys_cl,t)
```

We have reduced the proportional gain (K_P) because the integral controller also reduces the rise time and increases the overshoot as the proportional controller does (double effect). The above response would also show 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 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.

```
        Kp=350;
        Ki=300;
        Kd=50;
        contr=tf([KdKp 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 KP, KI and KD on the closed-loop system are summarized in the form of a table as given 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.

PROGRAMS, OBSERVATIONS AND INFERENCES

Write the Matlab program for implementing the PID controller for the system discussed in Pre Lab exercise and also plot the step responses with and without PID controller.

PROGRAM

```
clc;
clear all;
close all;
w=logspace(0.1,2,400);
s=tf('s');
sys=(100/((s+1)*(s+2)*(s+10)))
sys1=feedback(sys,1)
figure(1);
bode(sys1,w);
figure(2);
step(sys1);
kp=input('Enter the value for kp ');
ki=input('Enter the value for ki ');
kd=input('Enter the value for kd ');
sys2=pid(kp,ki,kd);
sys3=(sys2*sys)
sys4=feedback(sys3,1)
figure(3);
bode(sys4,w);
figure(4);
step(sys4);
```

RESULTS & INFERENCES:

Evaluation Component	**Maximum Marks**	**Marks Obtained**
Pre-lab Tasks	10	
In-Lab Tasks	20	
Post-lab Tasks	10	
Bonus Tasks	10	
Signature of Faculty with Date		

(This page must be the last page of the exercise)