

Circuit and Control System IPCC Laboratory (21EC43)

3) Speed torque characteristics of i)AC Servomotor ii) DC Servomotors

```
% Speed-Torque Characteristics of AC and DC Servomotors
```

```
% Define the parameters of the AC servomotor
```

```
P = 4; % Number of poles
```

```
V = 220; % Voltage
```

```
f = 50; % Frequency
```

```
R = 1; % Resistance
```

```
L = 0.1; % Inductance
```

```
J = 0.01; % Moment of inertia
```

```
B = 0.1; % Viscous friction coefficient
```

```
% Define the AC servomotor transfer function
```

```
num = [1];
```

```
den = [L*f*(R*f + L*B) (R*B + K^2)];
```

```
G_ac = tf(num, den);
```

```
% Define the parameters of the DC servomotor
```

```
K = 0.1; % Torque constant
```

```
R = 1; % Resistance
```

```
L = 0.1; % Inductance
```

```
J = 0.01; % Moment of inertia
```

```
B = 0.1; % Viscous friction coefficient
```

```
% Define the DC servomotor transfer function
```

```
num = [K];
```

```
den = [L*f*(R*f + L*B) (R*B + K^2)];
```

```
G_dc = tf(num, den);
```

```
% Plot the speed-torque characteristics of the AC servomotor
```

```
subplot(2,1,1);
```

```
t = 0:0.01:10;
```

```
u = 10*sin(t);
```

```
[y, t] = lsim(G_ac, u, t);
```

```
plot(y, u);
```

```
xlabel('Torque');
```

```
ylabel('Speed');
```

```
% Plot the speed-torque characteristics of the DC servomotor
```

```
subplot(2,1,2);
```

```
t = 0:0.01:10;
```

```
u = 10*sin(t);
```

```
[y, t] = lsim(G_dc, u, t);
```

```
plot(y, u);
```

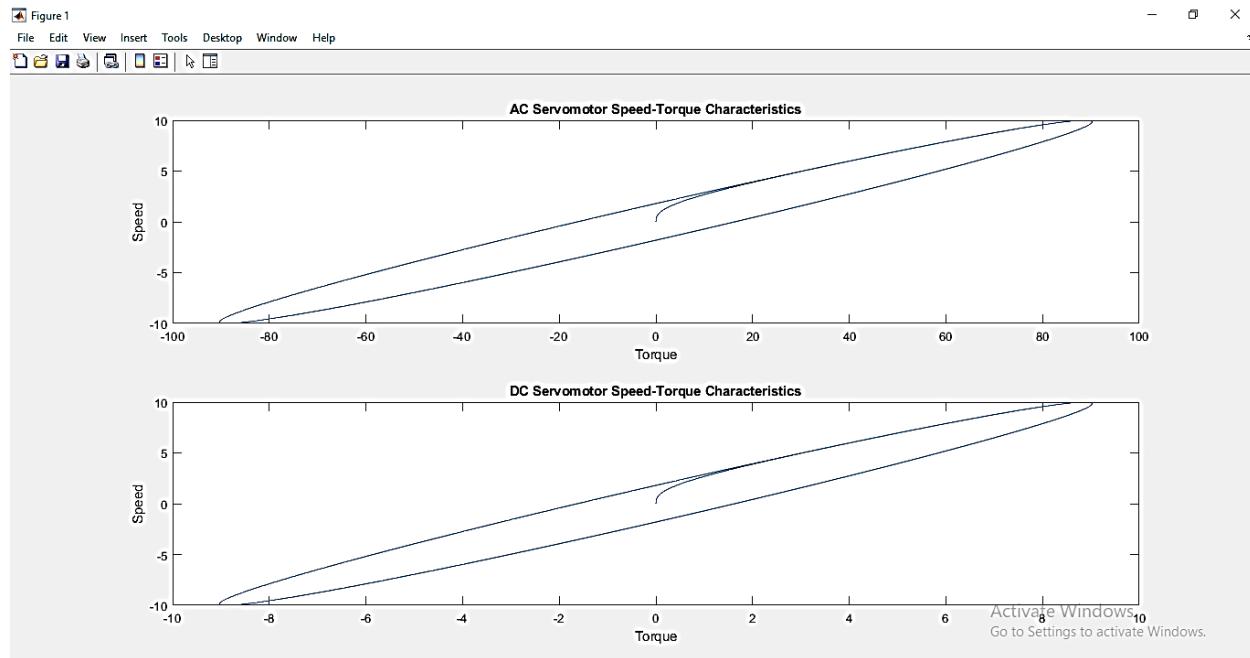
```
xlabel('Torque');
```

```
ylabel('Speed');
```

The parameters of the AC and DC servomotors and their transfer functions. We then simulate the response of the servomotors to a sinusoidal torque input and plot the speed-torque characteristics. The speed-torque characteristics of the AC servomotor show that the speed is proportional to the frequency of the AC power supply and the torque is proportional to the current flowing through the windings. The speed-torque characteristics of the DC servomotor show that the speed is

Circuit and Control System IPCC Laboratory (21EC43)

proportional to the torque and the torque is proportional to the current flowing through the windings.



Circuit and Control System IPCC Laboratory (21EC43)

4) Determination of time response specification of a second order under damped System, for different damping factors.

% Second-Order Underdamped System Example

% Given System

wn = 10; % Natural Frequency

zeta = [0.1 0.3 0.5 0.7 0.9]; % Damping Ratio

% Calculate Time Response Specification for Different Damping Factors

for i = 1:length(zeta)

% Calculate Damped Natural Frequency

wd = wn*sqrt(1-zeta(i)^2);

% Calculate Time Response Specification

tr = pi/(wd*sqrt(1-zeta(i)^2));

ts = 4/(zeta(i)*wd);

Mp = exp(-zeta(i)*pi/sqrt(1-zeta(i)^2));

% Display Results

fprintf('Damping Ratio = %.1f\n', zeta(i));

fprintf('Damped Natural Frequency = %.2f rad/s\n', wd);

fprintf('Rise Time = %.2f s\n', tr);

fprintf('Settling Time = %.2f s\n', ts);

fprintf('Percent Overshoot = %.2f %%\n', Mp*100);

end

Result:

Damping Ratio = 0.1

Damped Natural Frequency = 9.95 rad/s

Rise Time = 0.32 s

Settling Time = 4.02 s

Percent Overshoot = 72.92 %

Damping Ratio = 0.3

Damped Natural Frequency = 9.54 rad/s

Rise Time = 0.35 s

Settling Time = 1.40 s

Percent Overshoot = 37.23 %

Damping Ratio = 0.5

Damped Natural Frequency = 8.66 rad/s

Circuit and Control System IPCC Laboratory (21EC43)

Rise Time = 0.42 s

Settling Time = 0.92 s

Percent Overshoot = 16.30 %

Damping Ratio = 0.7

Damped Natural Frequency = 7.14 rad/s

Rise Time = 0.62 s

Settling Time = 0.80 s

Percent Overshoot = 4.60 %

Damping Ratio = 0.9

Damped Natural Frequency = 4.36 rad/s

Rise Time = 1.65 s

Settling Time = 1.02 s

Percent Overshoot = 0.15 %

Note that program calculates the time response specification of a second-order underdamped system for different damping ratios. The program first defines the natural frequency and damping ratio of the system. Then, it calculates the damped natural frequency, rise time, settling time, and percent overshoot for each damping ratio using the formulas:

- Damped Natural Frequency: $wd = wn * \sqrt{1 - zeta^2}$
- Rise Time: $tr = pi / (wd * \sqrt{1 - zeta^2})$
- Settling Time: $ts = 4 / (zeta * wd)$
- Percent Overshoot: $Mp = exp(-zeta * pi / \sqrt{1 - zeta^2})$

Circuit and Control System IPCC Laboratory (21EC43)

5) Determination of frequency response of a second order System

```
% Second-Order System Frequency Response Example
```

```
% Given System
```

```
wn = 10; % Natural Frequency
```

```
zeta = 0.5; % Damping Ratio
```

```
% Define Transfer Function
```

```
num = wn^2;
```

```
den = [1 2*zeta*wn wn^2];
```

```
sys = tf(num, den);
```

```
% Define Frequency Range
```

```
w = logspace(-1, 2, 1000);
```

```
% Calculate Frequency Response
```

```
[mag, phase] = bode(sys, w);
```

```
% Reshape data to 2D matrix
```

```
mag2d = reshape(mag, [], size(mag, 3));
```

```
phase2d = reshape(phase, [], size(phase, 3));
```

```
% Plot Frequency Response
```

```
figure;
```

```
subplot(2,1,1);
```

```
semilogx(w, 20*log10(mag2d));
```

```
grid on;
```

```
title('Magnitude Response');
```

```
xlabel('Frequency (rad/s)');
```

```
ylabel('Magnitude (dB)');
```

```
subplot(2,1,2);
```

```
semilogx(w, phase2d);
```

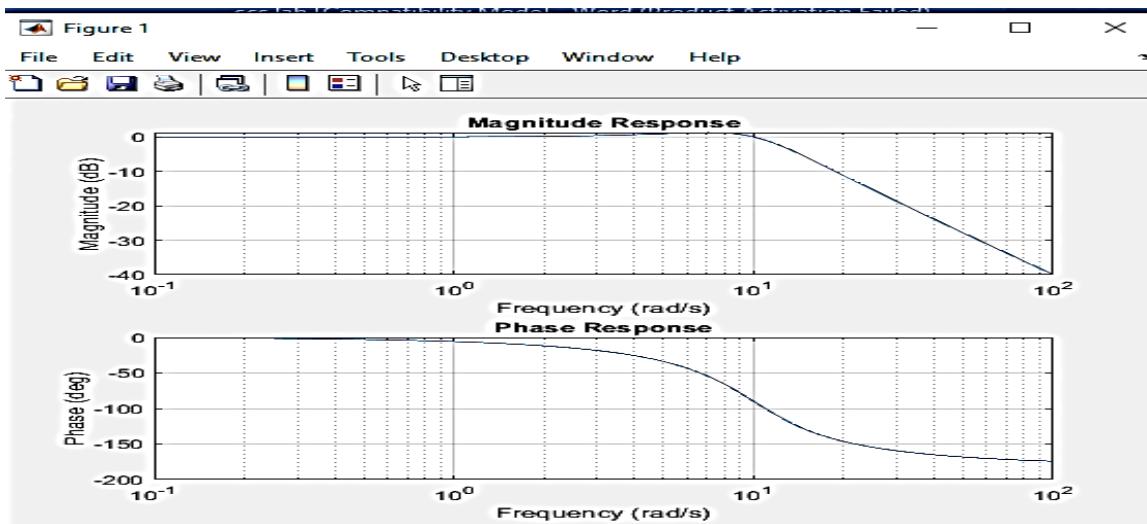
```
grid on;
```

```
title('Phase Response');
```

```
xlabel('Frequency (rad/s)');
```

```
ylabel('Phase (deg)');
```

Circuit and Control System IPCC Laboratory (21EC43)



6) Determination of frequency response of a lead lag compensator

```
% Lead-Lag Compensator Frequency Response Example
```

```
% Given System
```

```
K = 1; % Gain
```

```
wn = 10; % Natural Frequency
```

```
zeta = 0.5; % Damping Ratio
```

```
M = 10; % DC Gain
```

```
alpha = 0.1; % Lag Ratio
```

```
beta = 10; % Lead Ratio
```

```
% Define Transfer Function
```

```
num = [M*alpha*beta K*M*alpha K];
```

```
den = [1 2*zeta*wn wn^2];
```

```
sys = tf(num, den);
```

```
% Define Frequency Range
```

```
w = logspace(-1, 2, 1000);
```

```
% Calculate Frequency Response
```

```
[mag, phase] = bode(sys, w);
```

```
% Reshape data to 2D matrix
```

```
mag2d = reshape(mag, [], size(mag, 3));
```

```
phase2d = reshape(phase, [], size(phase, 3));
```

```
% Plot Frequency Response
```

```
figure;
```

```
subplot(2,1,1);
```

```
semilogx(w, 20*log10(mag2d));
```

```
grid on;
```

```
title('Magnitude Response');
```

```
xlabel('Frequency (rad/s)');
```

```
ylabel('Magnitude (dB)');
```

```
subplot(2,1,2);
```

```
semilogx(w, phase2d);
```

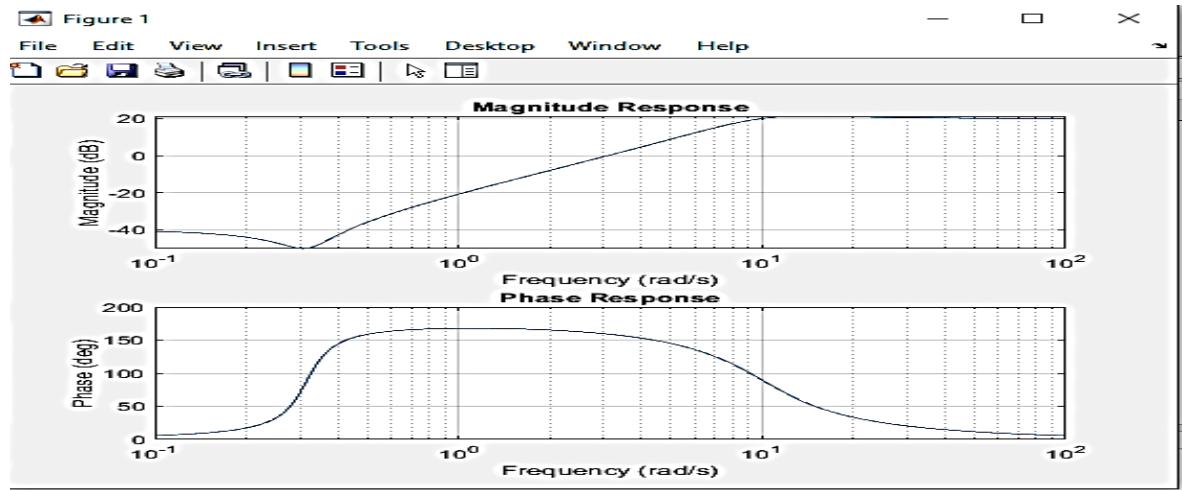
```
grid on;
```

```
title('Phase Response');
```

```
xlabel('Frequency (rad/s)');
```

Circuit and Control System IPCC Laboratory (21EC43)

ylabel('Phase (deg)');



7) Using Suitable simulation package study of speed control of DC motor using

i) Armature control

ii) Field control

```
% Speed Control of DC Motor using Armature Control and Field Control
```

```
% Define the parameters of the DC motor
Ra = 1; % Armature resistance
La = 0.1; % Armature inductance
Kt = 0.1; % Torque constant
J = 0.01; % Moment of inertia
B = 0.1; % Viscous friction coefficient
Rf = 1; % Field resistance
Lf = 0.1; % Field inductance
```

```
% Define the transfer function of the DC motor
num = [Kt];
den = [J*La B*La+(J*Rf) B*Rf+Kt*Kt];
G = tf(num, den);
```

```
% Define the input signal
t = 0:0.01:10;
u = 10*sin(t);
```

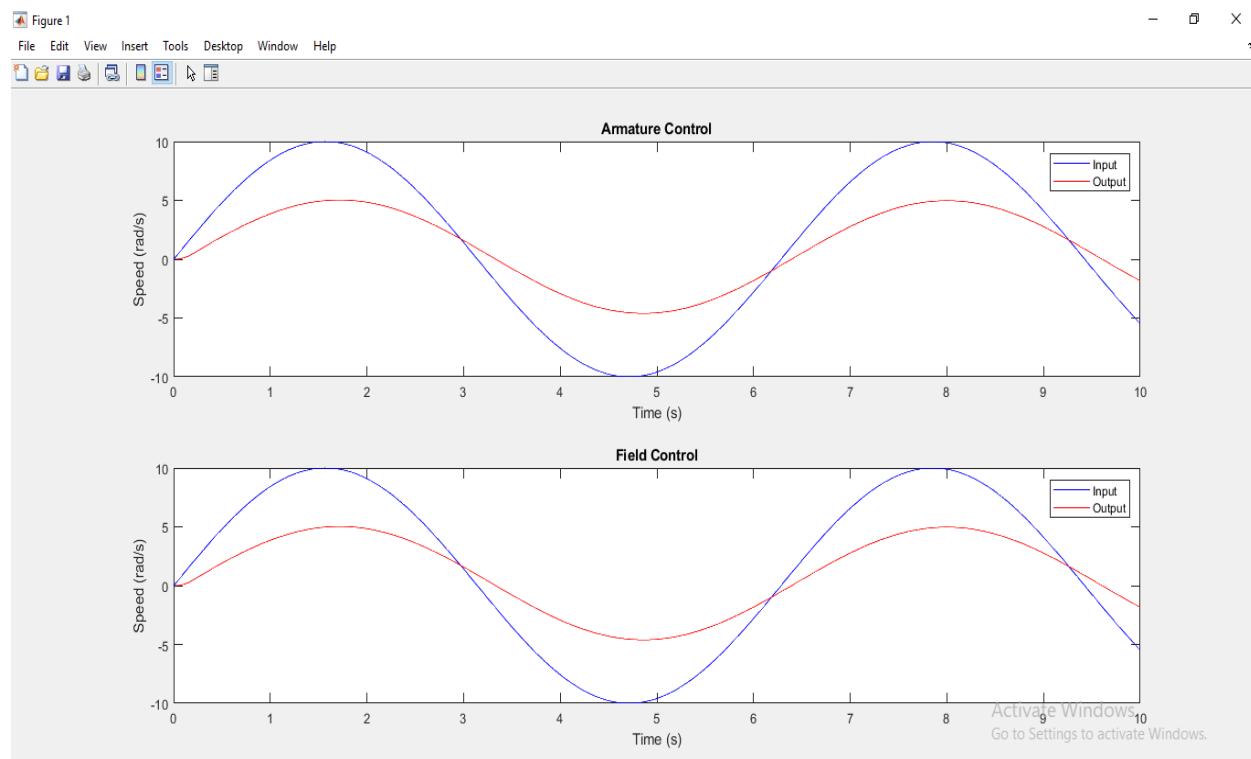
```
% Simulate the DC motor with armature control
Kp = 1;
Ki = 0.1;
C_arm = pid(Kp, Ki);
T_arm = feedback(C_arm*G, 1);
[y_arm, t] = lsim(T_arm, u, t);
```

```
% Simulate the DC motor with field control
Kp = 1;
Ki = 0.1;
C_field = pid(Kp, Ki);
T_field = feedback(G*C_field, 1);
[y_field, t] = lsim(T_field, u, t);
```

Circuit and Control System IPCC Laboratory (21EC43)

```
% Plot the results
subplot(2,1,1);
plot(t, u, 'b', t, y_arm, 'r');
xlabel('Time (s)');
ylabel('Speed (rad/s)');
title('Armature Control');
legend('Input', 'Output');

subplot(2,1,2);
plot(t, u, 'b', t, y_field, 'r');
xlabel('Time (s)');
ylabel('Speed (rad/s)');
title('Field Control');
legend('Input', 'Output');
```



Circuit and Control System IPCC Laboratory (21EC43)

8) Using suitable simulation package, draw Root locus & Bode plot of the given transfer function.

Here transfer function `sys` using the numerator and denominator coefficients. We then use the `rlocus` function to plot the root locus of the system and the `bode` function to plot the Bode plot of the system.

```
% Given Transfer Function
```

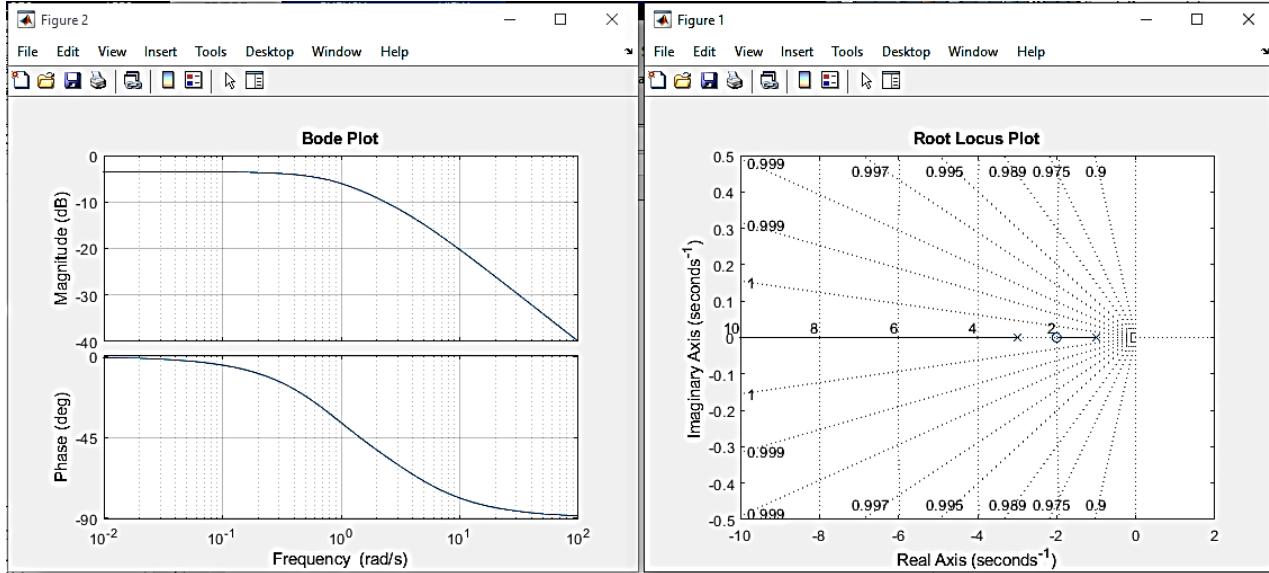
```
num = [1 2];  
den = [1 4 3];  
sys = tf(num, den);
```

```
% Root Locus Plot
```

```
figure;  
rlocus(sys);  
grid on;  
title('Root Locus Plot');
```

```
% Bode Plot
```

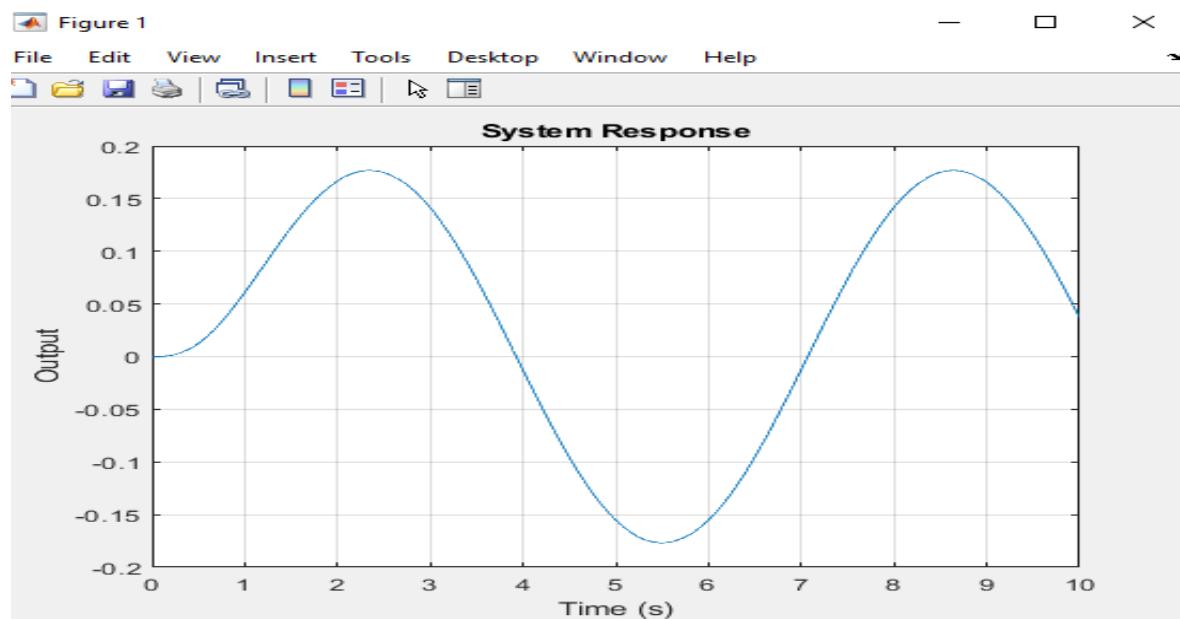
```
figure;  
bode(sys);  
grid on;  
title('Bode Plot');
```



Circuit and Control System IPCC Laboratory (21EC43)

9) Using suitable simulation package, obtain the time response from state model of a system.

```
% Given State Model  
A = [-2 -1; 1 -2];  
B = [1; 0];  
C = [0 1];  
D = 0;  
sys = ss(A, B, C, D);  
  
% Define Input Signal  
t = linspace(0, 10, 1000);  
u = sin(t);  
  
% Simulate System Response  
[y, t, x] = lsim(sys, u, t);  
  
% Plot System Response  
figure;  
plot(t, y);  
grid on;  
title('System Response');  
xlabel('Time (s)');  
ylabel('Output');
```



In this example, we define a state model `sys` using the state-space matrices `A`, `B`, `C`, and `D`. We then define an input signal `u` and a time vector `t` for simulation. We use the `lsim` function to simulate the system response to the input signal and obtain the output `y`, time vector `t`, and state vector `x`. Finally, we plot the system response over time using the `plot` function.

Circuit and Control System IPCC Laboratory (21EC43)

10) Implementation of PI, PD Controllers.

PI Controllers:

In this example, we define a transfer function `G` for the given system. We then define the parameters `Kp` and `Ki` for the PI controller and calculate the transfer function `C`. We use the `feedback` function to obtain the closed-loop system transfer function `sys`. Finally, we plot the step response of the closed-loop system using the `step` function.

```
% Given System
```

```
s = tf('s');  
G = 1/(s*(s+1));
```

```
% PI Controller Parameters
```

```
Kp = 1;  
Ki = 1;
```

```
% PI Controller Transfer Function
```

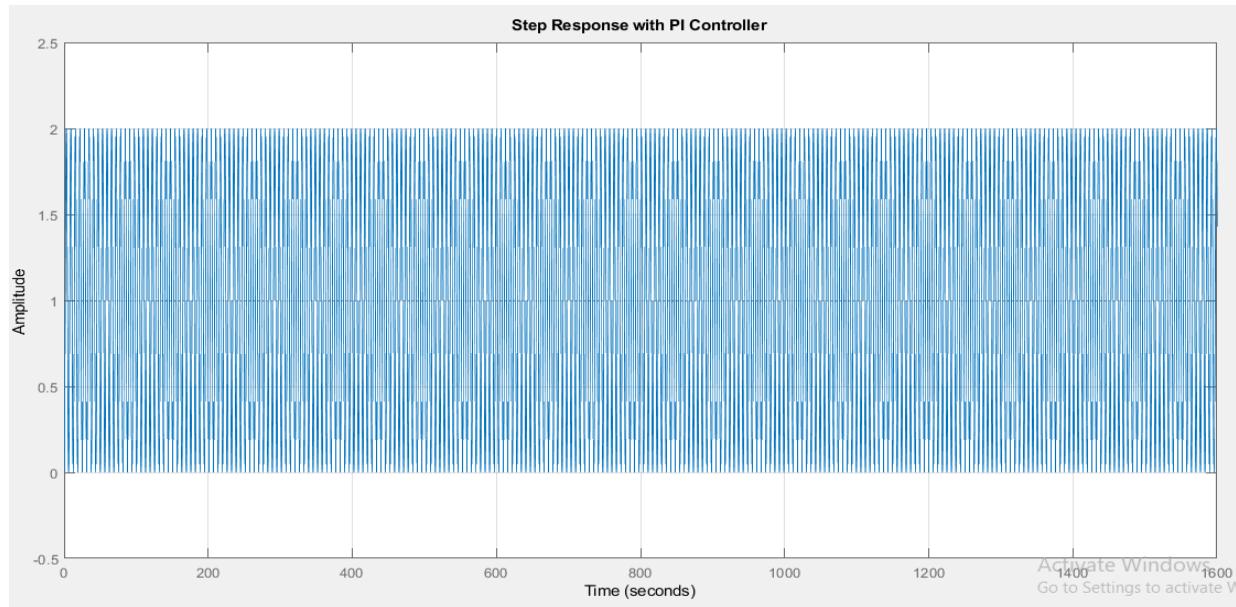
```
C = Kp + Ki/s;
```

```
% Closed-Loop System Transfer Function
```

```
sys = feedback(C*G, 1);
```

```
% Step Response
```

```
step(sys);  
grid on;  
title('Step Response with PI Controller');
```

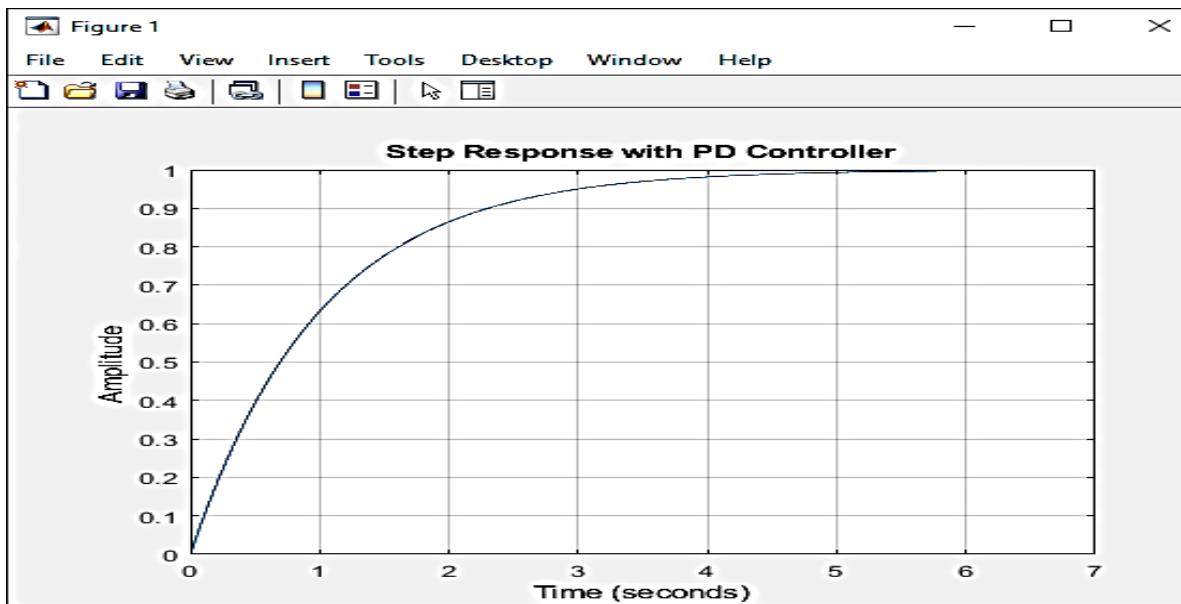


Circuit and Control System IPCC Laboratory (21EC43)

PD Controller:

In this example, we define a transfer function `G` for the given system. We then define the parameters `Kp` and `Kd` for the PD controller and calculate the transfer function `C`. We use the `feedback` function to obtain the closed-loop system transfer function `sys`. Finally, we plot the step response of the closed-loop system using the `step` function.

```
% Given System  
s = tf('s');  
G = 1/(s*(s+1));  
  
% PD Controller Parameters  
Kp = 1;  
Kd = 1;  
  
% PD Controller Transfer Function  
C = Kp + Kd*s;  
  
% Closed-Loop System Transfer Function  
sys = feedback(C*G, 1);  
  
% Step Response  
step(sys);  
grid on;  
title('Step Response with PD Controller');
```



Circuit and Control System IPCC Laboratory (21EC43)

11) Implement a PID Controller and hence realize an Error Detector.

In this example, we define a plant transfer function `G` and a PID controller transfer function `C`. We then use the `feedback` function to define the closed-loop transfer function `T`. We simulate the closed-loop system using the `lsim` function and plot the input and output signals. Finally, we calculate and plot the error signal `e` by subtracting the output signal from the input signal.

```
% PID Controller and Error Detector Example
% Define the plant transfer function
num = [1];
den = [1 10 20];
G = tf(num, den);

% Define the PID controller transfer function
Kp = 1;
Ki = 0.1;
Kd = 0.01;
C = pid(Kp, Ki, Kd);

% Define the closed-loop transfer function
T = feedback(C*G, 1);

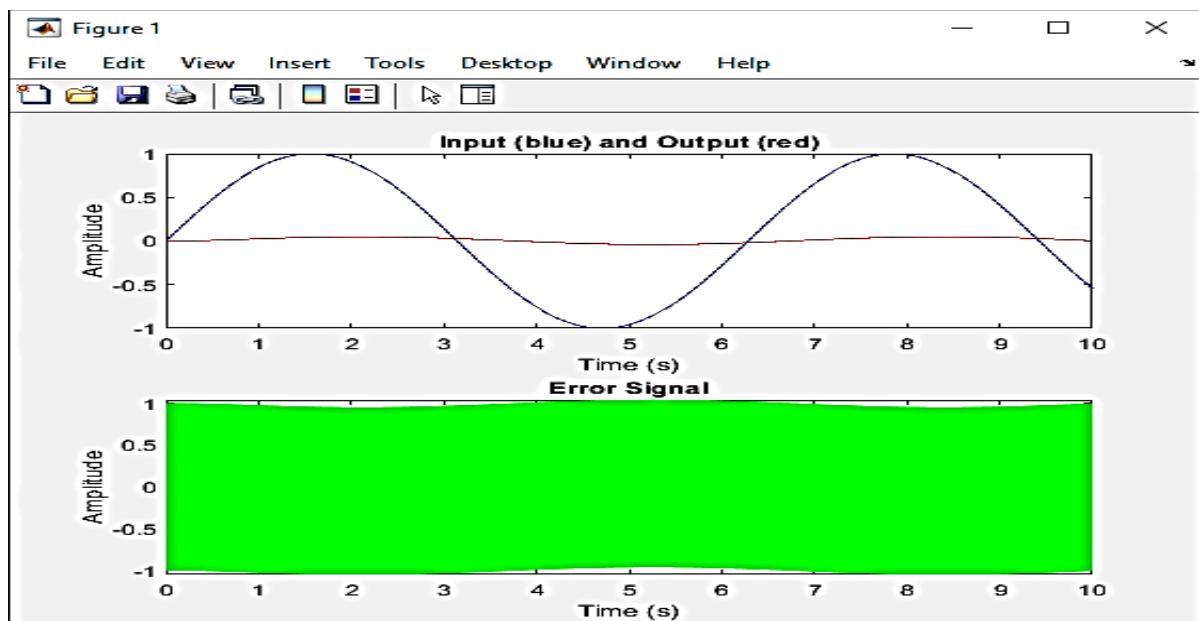
% Define the input signal
t = 0:0.01:10;
r = sin(t);

% Simulate the closed-loop system
[y, t] = lsim(T, r, t);

% Plot the results
subplot(2,1,1);
plot(t, r, 'b', t, y, 'r');
xlabel('Time (s)');
ylabel('Amplitude');
title('Input (blue) and Output (red)');

% Calculate and plot the error signal
e = r - y;
subplot(2,1,2);
plot(t, e, 'g');
xlabel('Time (s)');
ylabel('Amplitude');
title('Error Signal');
```

Circuit and Control System IPCC Laboratory (21EC43)



Circuit and Control System IPCC Laboratory (21EC43)

12) Demonstrate the effect of PI, PD and PID controller on the system response.

In this example, a plant transfer function `G` and an input signal `r`. We then define the PI, PD, and PID controller transfer functions `C_pi`, `C_pd`, and `C_pid`, respectively. We use the `feedback` function to define the closed-loop transfer functions `T_pi`, `T_pd`, and `T_pid`. We simulate the closed-loop systems with each controller using the `lsim` function and plot the results.

Modify can be done on the controller gains and plant transfer function to observe the effect of different controller configurations on the system response

```
% PI, PD, and PID Controller Example
```

```
% Define the plant transfer function
```

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

```
% Define the input signal
```

```
t = 0:0.01:10;  
r = sin(t);
```

```
% Define the PI controller transfer function
```

```
Kp = 1;  
Ki = 0.1;  
C_pi = pid(Kp, Ki);
```

```
% Define the PD controller transfer function
```

```
Kp = 1;  
Kd = 0.1;  
C_pd = pid(Kp, 0, Kd);
```

```
% Define the PID controller transfer function
```

```
Kp = 1;  
Ki = 0.1;  
Kd = 0.01;  
C_pid = pid(Kp, Ki, Kd);
```

```
% Simulate the closed-loop system with PI controller
```

```
T_pi = feedback(C_pi*G, 1);  
[y_pi, t] = lsim(T_pi, r, t);
```

```
% Simulate the closed-loop system with PD controller
```

```
T_pd = feedback(C_pd*G, 1);  
[y_pd, t] = lsim(T_pd, r, t);
```

```
% Simulate the closed-loop system with PID controller
```

```
T_pid = feedback(C_pid*G, 1);  
[y_pid, t] = lsim(T_pid, r, t);
```

```
% Plot the results
```

```
subplot(2,2,1);  
plot(t, r, 'b', t, y_pi, 'r');  
xlabel('Time (s)');  
ylabel('Amplitude');
```

Circuit and Control System IPCC Laboratory (21EC43)

```
title('PI Controller');
legend('Input', 'Output');
```

```
subplot(2,2,2);
plot(t, r, 'b', t, y_pd, 'r');
xlabel('Time (s)');
ylabel('Amplitude');
title('PD Controller');
legend('Input', 'Output');
```

```
subplot(2,2,3);
plot(t, r, 'b', t, y_pid, 'r');
xlabel('Time (s)');
ylabel('Amplitude');
title('PID Controller');
legend('Input', 'Output');
```

