# Advanced Control System

## Term project


**Submitted By;**
Soham Karak
200102107,ECE
Group 16

18 April 2023


**Course Instructor :**
Dr. Parijat Bhowmick

## Code for part 1:

```
%%                    Advanced Control System Term Project
%                              Submitted by:
%                          Soham Karak, 200102107
%                           Group 16, Question 1
%-------------------------------------------------------------------------
%% Program to design the compensator for a given system (Cacscaded Lag
compensator)
% Design a Compensator for a unity feedback system with
% open loop transfer function G(s)= K/(s(s+1)(0.5s +1)) to satisfy the
% following specifications:
% (i) velocity error constant K_v=5,
% (ii) Phase margin =40 degrees.
% (iii)Gain Margin = 10dB


% Clear the workspace and the command window

close all
clear all
clc

% Specifying the following requirement variables
PM_desired = 40;
GM_desired = 10;
Kv=1;


%Designing the uncompensated sytem plant Tf
disp(['Uncompensated system Transfer function']);
% Calculate the value of gain K from steady state value e_ss and use it
as the uncompensaated system tf gain.
num=[5];
den=[0.5 1.5 1 0];
G=tf(num,den)

% Plotting the Bode plot of the gain adjusted uncompensated system
[gm, pm, wgc, wpc] = margin(G)
figure(1);
```

```
bode(G)

title({"Uncompensated system with gain adjusted
Kv=5sec^-1",sprintf('Phase Margin = %0.4f deg, Gain Margin = %0.4f
dB',pm, gm)})

grid on;


% Finding the desired value of phase by adding a error tolerance value.

% The frequency at which the desired phase value was obtained(wValue),
is marked

% and the corresponding value of magnitude(mag_dB) for the frequency is
obtained.

% We then design a lag compensator such that the compensator brings down

% the magintude plot by mag_db such that the phase crossover frequency

% becomes wvalue. The compensator poles and zeros are wpc/10 and

% wpc/10*beta


required_PM=-180+PM_desired+10

[mag, phase,w] = bode(G);

phase_dB = squeeze(phase);

wValue = interp1(phase_dB, w, required_PM, 'spline');

disp(['The frequency value corresponding to phase margin = '
num2str(required_PM) ' degrees is ' num2str(wValue) ' rad/s']);

mag_dB = 20*log10(squeeze(mag));

w_at = interp1(w, mag_dB, wValue, 'spline');

disp(['The gain at frequency = ' num2str(wValue) ' degrees is '
num2str(w_at) ' dB']);


% Lag compensator design

% The pole is palced at 0.1 wgc so that the phase shift added by the lag

% compensator would be negligible


beta =10^(w_at/20)

zero=wValue/10;

pole=zero/beta;

T1=10/wValue

T2=beta*T1

disp(['Zero of the lead compensator is at s = ' num2str(zero)]);

disp(['Pole of the lead compensator is at s = ' num2str(pole)]);


%% Design of 1st Lag compensator
```

```
%Creating the lag compensator transfer function and the overall closed
loop
%tf to see whether we have satisfied the design requirements
disp(['Compensator Transfer function']);
Gc=tf([T1 1],[T2 1])
[gm, pm, wgc, wpc] = margin(Gc);
figure;
bode(Gc);
title({'Compensator',sprintf('Phase Margin = %0.4f deg, Gain Margin =
%0.4f dB', pm, gm)});
grid on;
final_TF1 = G*Gc

% Plotting the Bode plot of the Partially-compensated system

[gm, pm, wgc, wpc] = margin(final_TF1)
figure(2);
bode(final_TF1)
margin(final_TF1)
title({'Compensated System',sprintf('Phase Margin = %0.4f deg, Gain
Margin = %0.4f dB', pm, gm)});
grid on;
[mag, phase,w] = bode(final_TF1);

%We observe that although the steady state and phase margin requirement
is
%fulfilled the gain margin is still inadequate so we plan to cascade
%another lag compensator to add teh remaining gain shift required
%% Design of the 2nd Lag Compensator
% We have choosen that the lag compensator would shift the mag plot by
% another 5 dbs so that the gain margin requirement wouyld be
satisfied .
%For that puprose we have taken the addition gain added by the lag
compensator to be 8.5 dB's at wgc/10 or the zero  of teh first
compensator as well as any frequency there after

beta  = 10^(8/20)
wz = wgc/10
t = 1/wz
num = [t 1]
den = [beta*t 1]
```

```
% Design of 2nd lag compenstaor tf
Gc1 = tf(num,den)
bode(Gc1)
margin(Gc1)
[gm, pm, wgc, wpc] = margin(Gc1)
%% Final Transfer function
final_TF = G*Gc*Gc1

% Plotting the Bode plot of the compensated system
[gm, pm, wgc, wpc] = margin(final_TF)

figure(3);
bode(final_TF)
margin(final_TF)
title({'Compensated System',sprintf('Phase Margin = %0.4f deg, Gain
Margin = %0.4f dB', pm, gm)});
grid on;
[mag, phase,w] = bode(final_TF);
%% Step Response

cl_G=feedback(G,1);
% Closed loop TF of compensated system
cl_Final_TF=feedback(final_TF,1);

figure(4);
hold on;
subplot(211);step(cl_G);title({'Step response of uncompensated
system(closed loop system)'});
grid on;
subplot(212);step(cl_Final_TF);title({'Step response of compensated
system(closed loop system)'});
grid on;
hold off;
%% Ramp response

% the ramp signal is created for a duration of 0-10secs
t=0:0.1:10;
alpha=1;
ramp=alpha*t;
```

```
[y1,t]=lsim(cl_G,ramp,t);
[y2,t]=lsim(cl_Final_TF,ramp,t);

figure(5);

subplot(211);
hold on
plot(t,t)
plot(t,y1);title({'Ramp response of uncompensated system(closed loop
system)'});
legend('Ramp signal','Uncompensated system')
grid on;

subplot(212);
hold on
plot(t,t)
plot(t,y2);title({'Ramp response of compensated system(closed loop
system)'});
legend('Ramp signal','compensated system')
grid on;
hold off;
```

# Output(Matlab Command window):

Uncompensated system Transfer function

```
G =        5
  ---------------------
  0.5 s^3 + 1.5 s^2 + s        Continuous-time transfer function.

gm  = 0.6000
pm =-12.9919
wgc = 1.4142


wpc =1.802
required_PM =-130
The frequency value corresponding to phase margin = -130 degrees is
0.49176 rad/s
The gain at frequency = 0.49176 degrees is 18.9487 dB
beta =8.8600
T1 =20.3351
T2 =180.1700
```

```
Zero of the lead compensator is at s = 0.049176
Pole of the lead compensator is at s = 0.0055503
Compensator Transfer function

Gc =
  20.34 s + 1
  -----------
  180.2 s + 1
Continuous-time transfer function.
final_TF1 =
            101.7 s + 5
  ------------------------------------
  90.09 s^4 + 270.8 s^3 + 181.7 s^2 + s

Continuous-time transfer function.
gm =4.9674
pm =44.8157
wgc =1.3671
wpc =0.4937
beta =2.5119
wz  =0.1367
t =7.3145
Gc1 =
  7.315 s + 1
  -----------
  18.37 s + 1
 Continuous-time transfer function.
gm =Inf
pm =  -180
wgc =NaN
wpc =0
final_TF =

          743.7 s^2 + 138.2 s + 5
  --------------------------------------------
  1655 s^5 + 5065 s^4 + 3609 s^3 + 200 s^2 + s

Continuous-time transfer function.
gm =10.8234
pm = 42.5711
wgc =1.2737
wpc = 0.2465
```

# Code for part 2:

```
%%                        Advanced Control Systems
%                            Submitted By:
%                     Soham Karak, 200102107,Grp 16
% Question 2:Design a PID controller for the syetem: 1/s(s+1)(s+5),
obtain unit step response
% with MATLAB/ Simulink, and compare the responses of systems.
% Clearing the workspace and command window

close all
clc
clear all

% Defining the plant Tf
num = [1];  % numerator coefficients
den = [1 6 5 0];  % denominator coefficients
G = tf(num, den);

% controlSystemDesigner(G)
%% Marginal stability point and Zeigler nichols parameters

% % Plot the root locus
figure(1)
rlocus(G)
%Obtain the Kcritical value
gm = margin(G)
K  = gm;
num = [1];  % numerator coefficients
den = [1 6 5 30];  % denominator coefficients
G_ = tf(num, den);
step(G_,5)
% Obtain the period of sustained oscillations to get Pcr= 2.81
%% Zeigler Nichols PID tuning

G_Kp = tf(1,[1 6 5 K])
% step(G_Kp,10)
% hold on
Kc = 30;
Pc = 2.81

%Min overshoot PID model

Kp = 0.33*Kc
Ti = (0.5*Pc)
Td = (0.333*Pc)

Ki = Kp/Ti
Kd = Kp*Td
% Transfer Function of PID model
C = tf([Kd Kp Ki],[1 0])

%% PID Transfer Function obtained from root locus pole placement

s = tf('s');
C_updated = (4*(1 + 1.65*s)*(1 + 1.65*s))/s
pid(C_updated)

 %            1
 % Kp + Ki * --- + Kd * s
 %            s
```

```
 % with Kp = 13.2, Ki = 4, Kd = 10.9

%% Comparision of the step responses of both the PID controllers


G_final = feedback(C*G,1)
G_updated = feedback(C_updated*G,1)

% Transient time parametres of the PID models
S = stepinfo(G_final)
S1= stepinfo(G_updated)


% Plot the step responses
figure(2)
hold on
step(G_final)
grid on
hold on
step(G_updated)
grid on
hold on
step(feedback(G,1))
legend('Ziegler Nichols method','Root Locus method','Original closed
loop response')
hold off
```

MATLAB

# Output(Matlab Command window):

gm = 30
G_Kp =

***1***
***----------------------***
***s^3 + 6 s^2 + 5 s + 30***

Pc = 2.8100
Kp = 9.9000
Ti =1.4050
Td = 0.9357

Ki =7.0463
Kd = 9.2637
C =

***9.264 s^2 + 9.9 s + 7.046***
***-------------------------***
***s***

C_updated =

***10.89 s^2 + 13.2 s + 4***
***----------------------***

```
          s

ans =
                 1
  Kp + Ki * --- + Kd * s
                 s
  with Kp = 13.2, Ki = 4, Kd = 10.9
```

Continuous-time PID controller in parallel form.

```
G_final =
        9.264 s^2 + 9.9 s + 7.046
  ---------------------------------------
  s^4 + 6 s^3 + 14.26 s^2 + 9.9 s + 7.046
 G_updated =
        10.89 s^2 + 13.2 s + 4
  ------------------------------------
  s^4 + 6 s^3 + 15.89 s^2 + 13.2 s + 4
```

Continuous-time transfer function.

```
S =    RiseTime: 0.7087
    TransientTime: 11.2144
     SettlingTime: 11.2144
      SettlingMin: 0.9077
      SettlingMax: 1.2720
        Overshoot: 27.1958
       Undershoot: 0
             Peak: 1.2720
         PeakTime: 2.3204


S1 =  RiseTime: 0.6128
    TransientTime: 6.1982
     SettlingTime: 6.1982
      SettlingMin: 0.9234
      SettlingMax: 1.1524
        Overshoot: 15.2429
       Undershoot: 0
             Peak: 1.1524
         PeakTime: 1.5867
```