EE 324 Control Systems Lab

Problem Sheet 7

Mayur Ware | 19D070070

October 3, 2021

Question 1

Given open loop transfer function G(s) is,

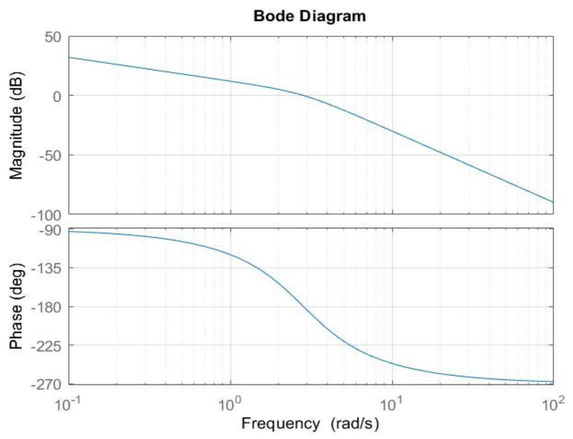
$$G(s) = \frac{1}{s(s^2+4s+8)}$$

Part (a) :
K (gain) for which the closed-loop characteristic equation has gain margin and phase margin equal to zero. Closed loop transfer function TF(s) will be,

$$TF(s) = \frac{KG(s)}{1+KG(s)}$$

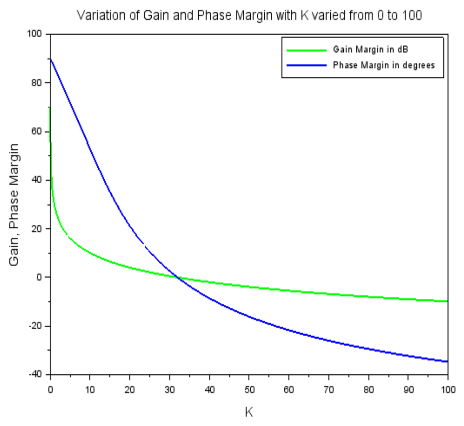
$$C\_Eqn = KG(s) = \frac{K}{s(s^2+4s+8)}$$

Using `SciLab`, the value of gain K for which the closed loop characteristic equation has gain and phase margin equal to zero is K = 32



```
s = poly(0, 's');
G = 1/(s*(s^2+4*s+8));
//Part a
K = 0:0.01:100;
K_required = 0;
Gain_margins = [];
Phase_margins = [];
for i=1:size(K, 2)
    k = K(i);
    c_eqn = k*G;        //Defined characteristic equation
    C_Eqn = syslin('c', c_eqn);
    Gain_m = g_margin(C_Eqn);
    Phase_m = p_margin(C_Eqn);
    Gain_margins = [Gain_margins, Gain_m];
    Phase_margins = [Phase_margins, Phase_m];
    if abs(Gain_m) <= 1e-5 && abs(Phase_m) <= 1e-5 then
        K_required = k;
    end
end
plot(K, Gain_margins, 'g-');
plot(K(2:size(K, 2)), Phase_margins, 'b-');
xlabel("K");
ylabel("Gain, Phase Margin");
title("Variation of Gain and Phase Margin with K varied from 0 to 100");
legend(["Gain Margin in dB", "Phase Margin in degrees"]);
K_required
G_new = K_required*G;
show_margins(G_new);
```

Part (b) :
The variation of both gain and phase margin of the closed loop characteristic equation with K is



As we can notice in the graph, both Phase Margin and Phase Margin are monotonically decreasing functions. When Gain Margin is zero, and at the same point Phase Margin is also zero. So, there is no value of K for which Gain Margin is non-zero where Phase Margin is zero.
We can also verify this using SciLab, when we change the condition to `abs(Gain_m - Phase_m) <= 1e-5`, it also gives the same value of K.

```
if abs(Gain_m     Phase_m) <= 1e-5 then
    K_required = k;
end
```

Part (c) :
In Part (a), K = 32, zeros of 1+32*G(s) will be the poles of closed loop system. Zeros of this equation come out to be -4.0, 7.216D-16 + 2.8284271i and 7.216D-16 - 2.8284271i as two poles are in Right-Half Plane, system is unstable.

```
s = poly(0, 's');
G = 1/(s*(s^2+4*s+8));
```

```
c_eqn = K_required*G;
C_Eqn = syslin('c', c_eqn);
[Zeros, Poles, Gain] = tf2zp(C_Eqn);
Poles
```

Question 2

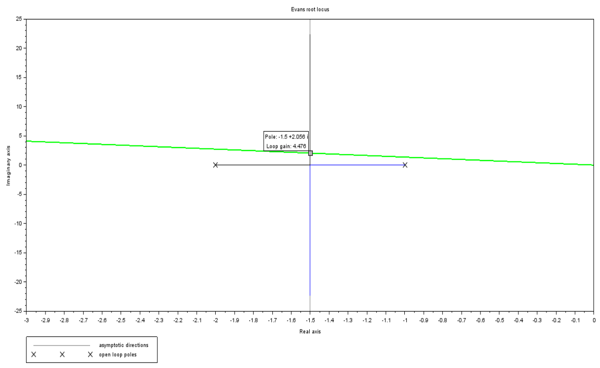
Design a Lag-compensator that has a ratio of zero-magnitude to pole-magnitude of 20.
Given open loop transfer function G(s) is,

$$G(s) = \frac{1}{(s^2 + 3s + 2)}$$

Part (a) :
For a constant Percentage Overshoot (%OS), it would be a straight line. Slope of this line will be,

$$-\frac{\sqrt{1-\zeta^2}}{\zeta} = -\frac{\pi}{ln(10)} = -1.364$$

Intersection of this line with the root locus will give us our desired point.
On plotting Locus of Closed Loop Poles using `SciLab`, We get K = 4.476



Lag Compensator

```
s = poly(0, 's');
Slope = %pi/log(10);
G = syslin('c', 1/(s^2+3*s+2));
plot(t, csim('step', t, G), 'g-');
xgrid(0);
xlabel("t");
ylabel("Step Response");
title("Step Response of G(s)");
X = -3:0.01:0;
Y = -Slope*X;
plot(X, Y, 'g-', 'LineWidth', 2);
evans(G, 500);
xlabel("Real axis");
ylabel("Imaginary axis");
title("Evans root locus");
K = 4.476;
```

Part (b) :

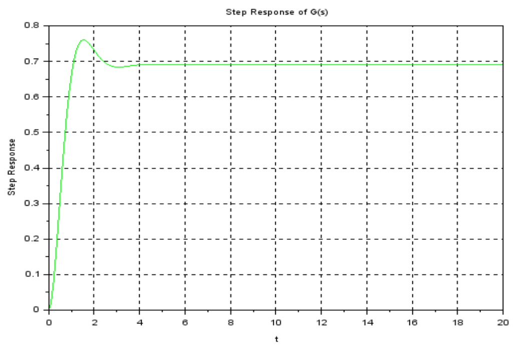
Steady state error e(∞) without Lag-compensator is given by

$$e(\infty) = \frac{1}{1 + K * G(0)} = 0.308$$

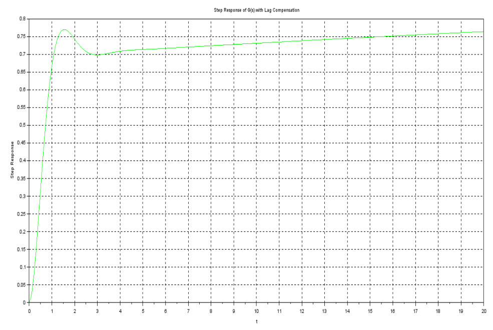
Steady state error e(∞) with Lag-compensator will be

$$e(\infty) = \frac{1}{1 + K * G(0) * Ratio} = \frac{1}{1 + K * G(0) * 20} = 0.021$$

```
K = 4.476;
SSE = 1/(1+K*0.5)
Pole = 0.0001;
Zero = Ratio*Pole;
G_LagComp = syslin('c', (s+Zero)/((s+Pole)*(s^2+3*s+2)));
plot(t, csim('step', t, G), 'g—');
xgrid(0);
xlabel("t");
ylabel("Step Response");
title("Step Response of G(s) with Lag Compensation");
SSE_LagComp = 1/(1+(K*20*0.5));
```

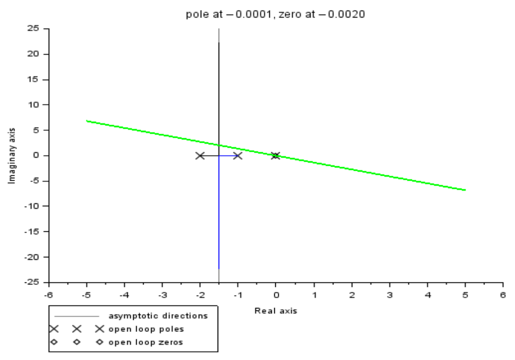


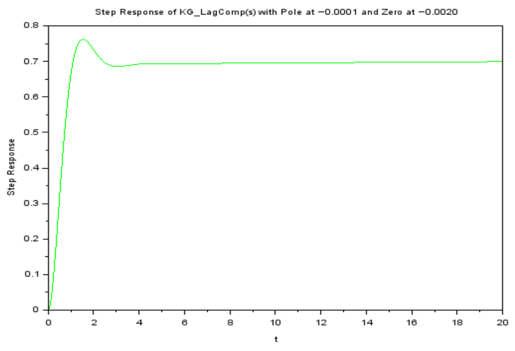
Without Lag-compensator



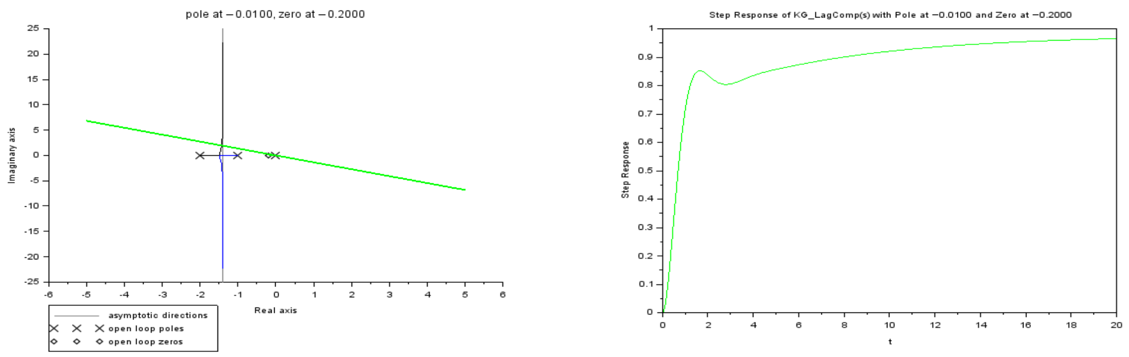
With Lag-compensator

Part (c) :
Change the location of the pole-zero pair (with say 5 different pole-zero locations) to see the degrading effect on the planned %OS. I chose [0.0001, 0.01, 1, 5, 10] as the pole locations.
As the pole moves farther away from origin, time taken to reach the given %OS increases and hence, the transient response lasts longer.
Root locus for all these cases are plotted using `SciLab`,

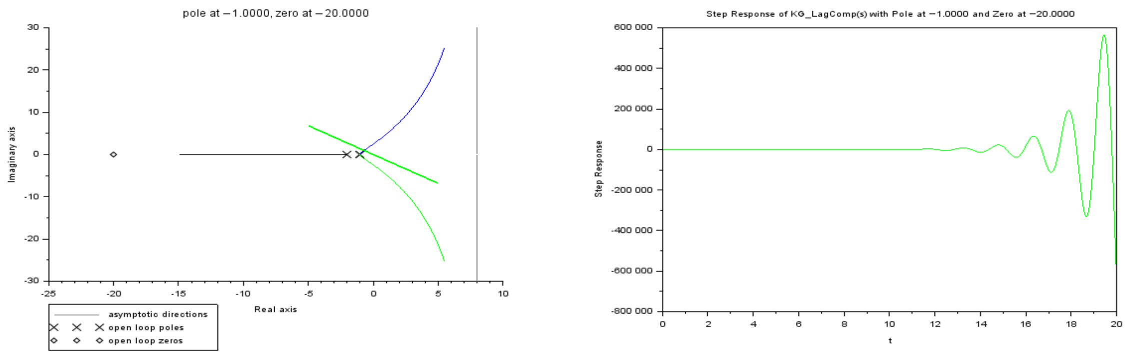




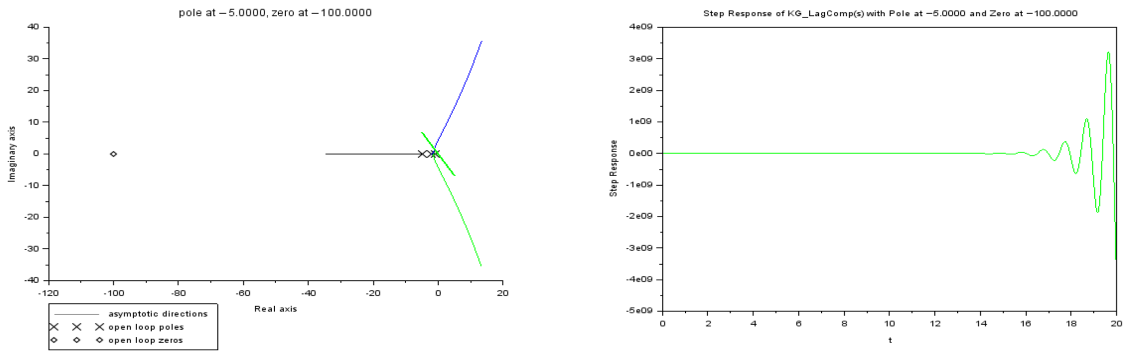
Pole = -0.0001 | Zero = -0.002



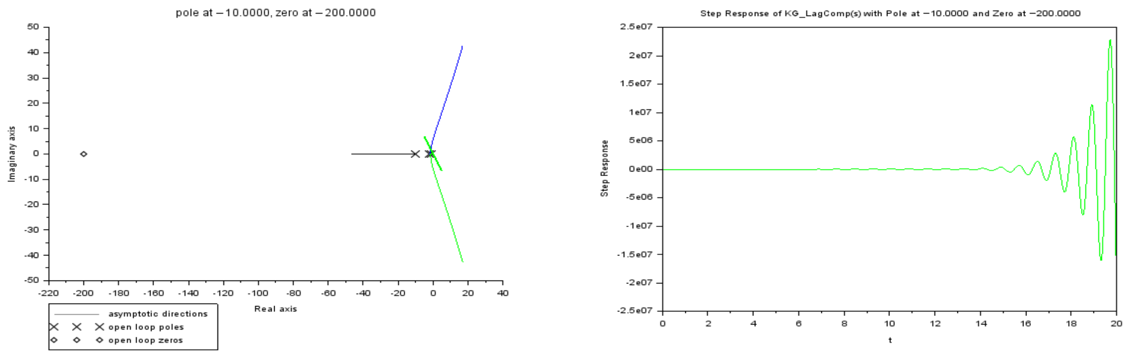
Pole = -0.01 | Zero = -0.2



Pole = -1 | Zero = -20



Pole = -5 | Zero = -100



Pole = -10 | Zero = -200

```
s = poly(0, 's');
```

```
Ratio = 20;
Poles = [0.0001, 0.01, 1, 5, 10];
Slope = %pi/log(10);
X = -5:0.01:5;
Y = -Slope*X;
for i=1:size(Poles, 2)
    pole = Poles(i);
    Zero = pole*Ratio;
    G_LagComp = syslin('c', (s+Zero)/((s+Pole)*(s^2+3*s+2)));
    scf();
    evans(G_LagComp, 500);
    plot(X, Y, 'g-', 'LineWidth', 2);
    title(sprintf("pole at  % .4f, zero at  % .4f", pole, Zero));
    xlabel("Real axis");
    ylabel("Imaginary axis");
end
```

Question 3

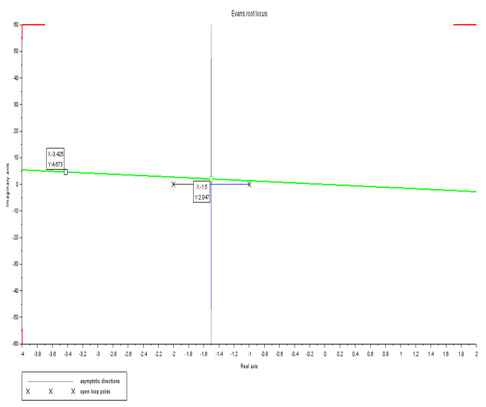
Design a Lead-compensator for G(s) of Q2 to have 2% settling time made half of the case for Q2, and %OS 10.

Part (a) :

Using the root locus method, we get the point on root locus which satisfies both these properties as,

$$P = 3.425 + j4.673$$

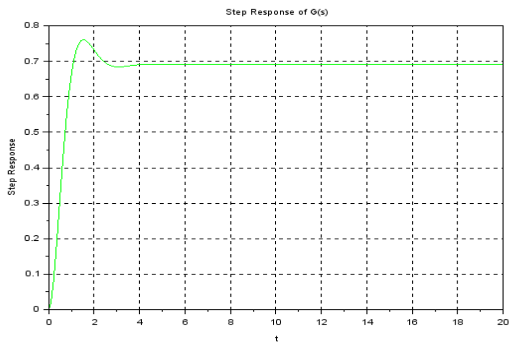
For the lead compensator, we should choose a zero suitably left and the pole should be more left. I chose Zero to be at -5. We find Pole using the angle conditions which comes out to be at -12.60.



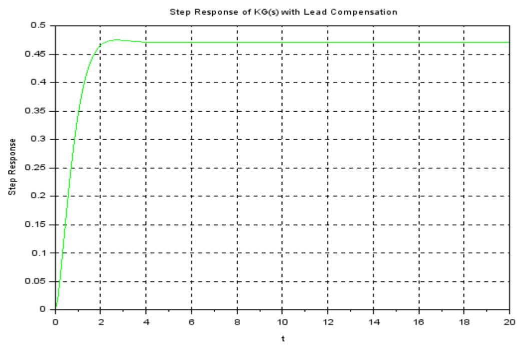
Lead Compensator

The Transfer Function comes out to be,

$$G\_LeadComp = \frac{22.38 + 4.476s}{s^3 + 15.60s^2 + 39.80s + 25.20}$$



Without Lead-compensator



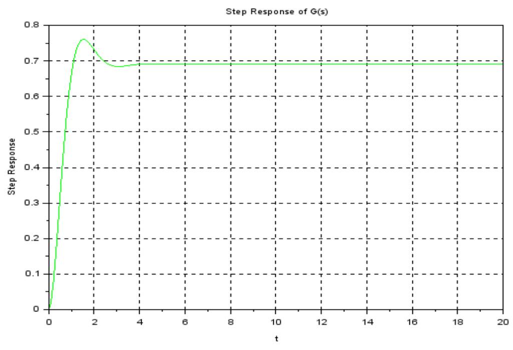
With Lead-compensator

```
s = poly(0, 's');
K = 4.476
G = syslin('c', K/(s^2+3*s+2));
t_settle = 2.3357;
Slope = %pi/log(10);
X = -4:0.001:2;
Y = -Slope*X;
evans(G, 500);
plot(X, Y, 'g-', 'LineWidth', 2);
t_settle_new = t_settle/2;
new_Real = -4/t_settle_new;
new_Im = -Slope*new_Real;
plot(new_Real, new_Im, 'go', 'LineWidth', 2);
//Assuming the zero of the lead compensator is at   5
Z_c = 5;
[z, p, g] = tf2zp(G);
if z ~= [] then
    Zeroes = zeros(size(z, 1), 2);
end
if p ~= [] then
    Poles = zeros(size(p, 1), 2);
end
for i=1:size(p, 1)
    Pi = p(i);
    Pi = [real(Pi), imag(Pi)];
    Poles(i, :) = Pi;
end
for i=1:size(z, 1)
    Zi = z(i);
    Zi = [real(Zi), imag(Zi)];
    Zeroes(i, :) = Zi;
end
Angles = 0;
for i=1:size(Poles, 1)
    A = [new_Real, new_Im]-Poles(i, :);
    Angles = Angles - atan(A(2), A(1));
end
Angles = Angles + atan(new_Im, Z_c + new_Real);
Angle_new_Pole = Angles - %pi + 2*%pi;
P_c = (new_Im/tan(Angle_new_Pole)) - new_Real;
```

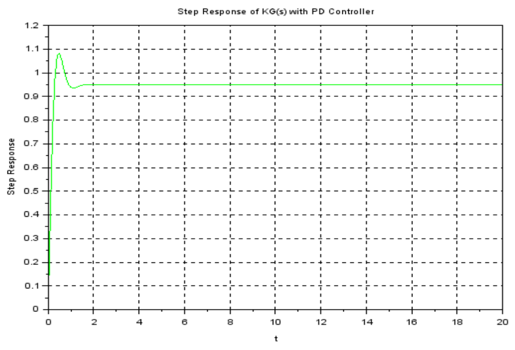
```
G_LeadComp = syslin('c', ((s+Z_c)*K)/((s^2+3*s+2)*(s + P_c)));
```

Part (b) :
For a PD Controller satisfying the given conditions, Zero comes out to be at -8.199.
The Transfer Function comes out to be,

$$G\_PD = \frac{36.7006 + 4.476s}{s^2 + 3s + 2}$$

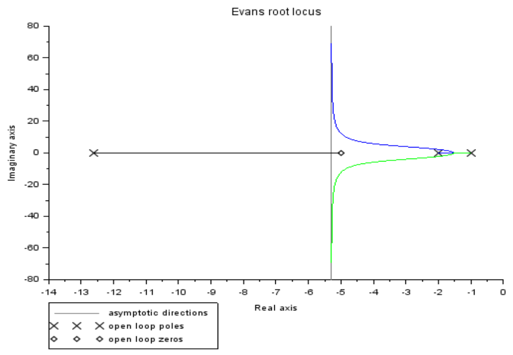


Without PD Controller

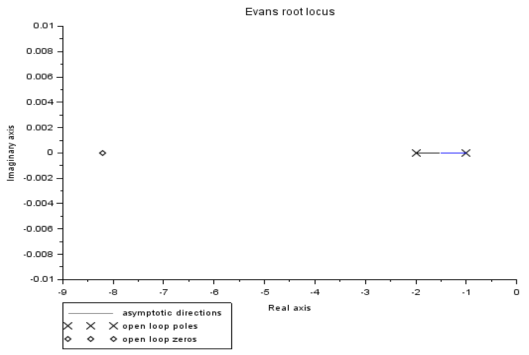


With PD Controller

```
Angles = 0;
for i=1:size(Poles, 1)
    A = [new_Real, new_Im] - Poles(i, :);
    Angles = Angles - atan(A(2), A(1));
end
Angle_new_Zero = -Angles + %pi;
Z_c = (new_Im/tan(Angle_new_Zero)) - new_Real;
G_pd = syslin('c', K*(s+Z_c)/(s^2+3*s+2));
```



Lead Compensator



PD Controller

References

1) https://help.scilab.org/