EE 324 Control Systems Lab

Problem Sheet 6

Mayur Ware | 19D070070

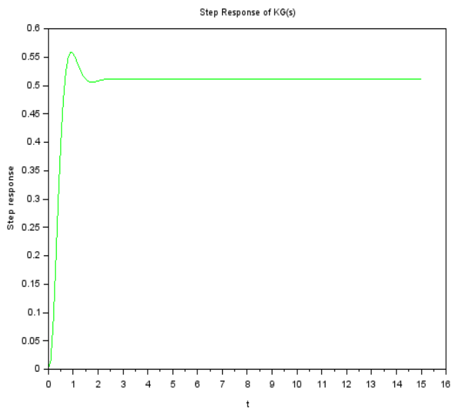
September 27, 2021

Question 1

Proportional (P) controller with gain K using the root-locus method
Given open loop transfer function G(s) is,

$$G(s) = \frac{1}{(s+3)(s+4)(s+12)}$$

Part (a) :
To obtain a steady state error of 0.489 on applying step input
Step response of G(s) is,



The equation of steady state error is,

$$e(\infty) = \frac{1}{1+KG(0)} = \frac{1}{1+K*144}$$

By solving this, we get the Gain (K) as,

$$K = 150.478$$

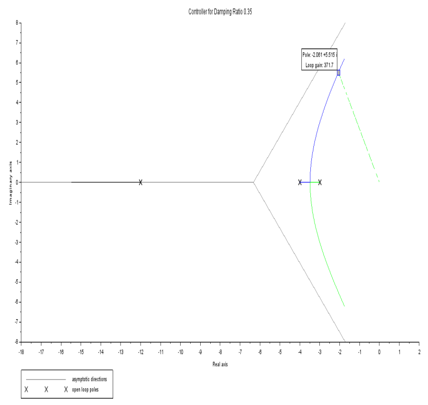
Locus of Closed Loop Poles using `SciLab` is,

```
s = poly(0,'s');
K = 150.478;
G = 1/((s+3)*(s+4)*(s+12));
S = syslin('c', K*G/(1+K*G));
t = 0:0.1:15
plot(t, csim('step', t, S),'g-');
xlabel("t"); ylabel("Step response");
title("Step Response of KG(s)");
```

Part (b) :
To attain a damping ratio of 0.35.
Percentage Overshoot (%OS) would be a straight line. Slope of this line will be,

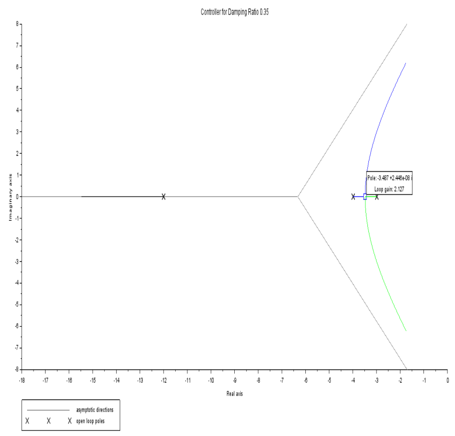
$$tan(\alpha) = -\frac{\sqrt{1-\zeta^2}}{\zeta} = 2.676$$

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 = 371.7



```
s = poly(0,'s');
G = syslin('c', 1/((s+3)*(s+4)*(s+12)));
evans(G, 500);
X = -2.3:0.00001:0;
Damping_Ratio = sqrt(1-0.35^2)/0.35;
Y = -Damping_Ratio.*X;
plot(X, Y, 'g-');
evans(G, 500);
title("Controller for Damping Ratio 0.35");
```

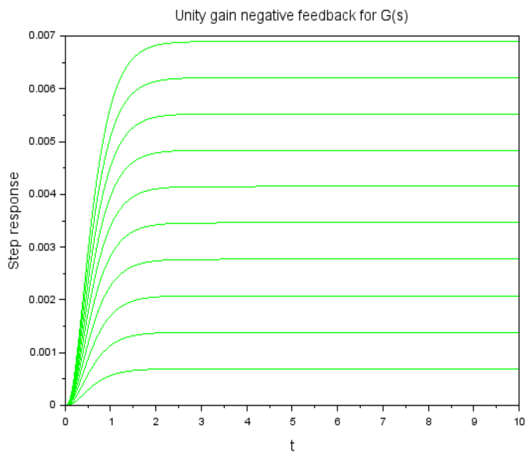
Part (c) :
Gain value at the break away point.
On plotting Locus of Closed Loop Poles using `SciLab`, we get K = 2.127



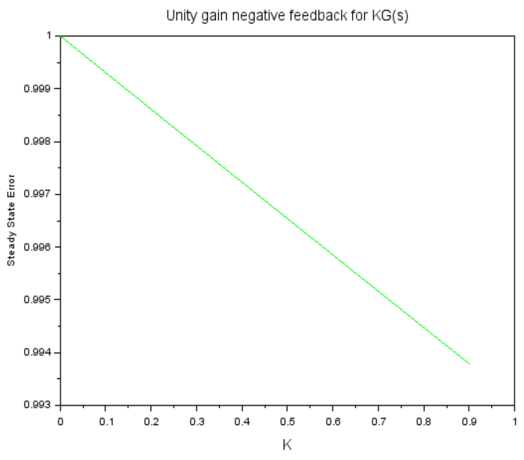
Part (d) :
Gain (K) is varied in a small range from 0 to 1.
Observations :
1) Steady state error e(∞) is given by

$$e(\infty) = \frac{1}{1 + K * 144}$$

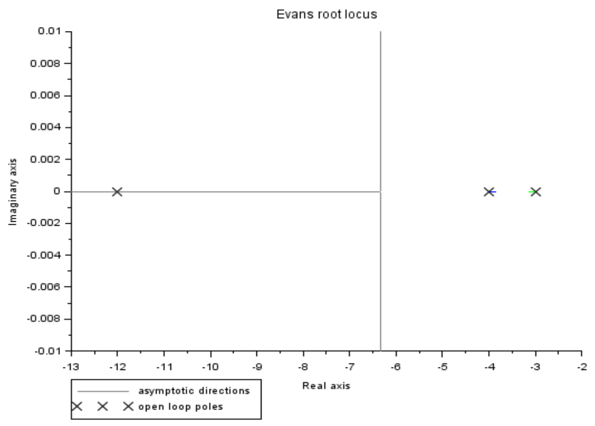
For given range of values for K, this seems like a straight line.
2) For given K, the closed loop poles lie on the real axis and are between the dominant open-loop poles.
3) By looking at the graph, we can conclude that as the value of K is increased, the rise time and settling time decreases



Step responses as K is increased from 0 to 1

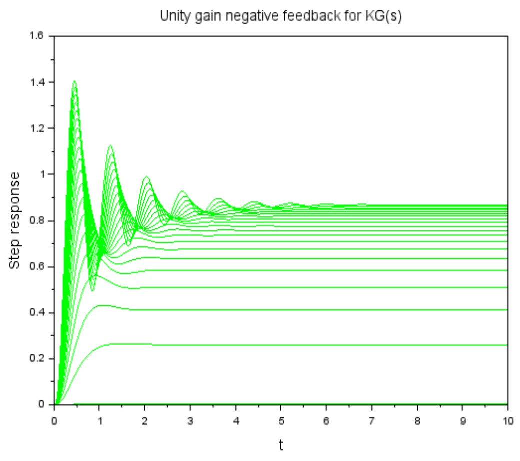


Steady State Error as K is increased from 0 to 1

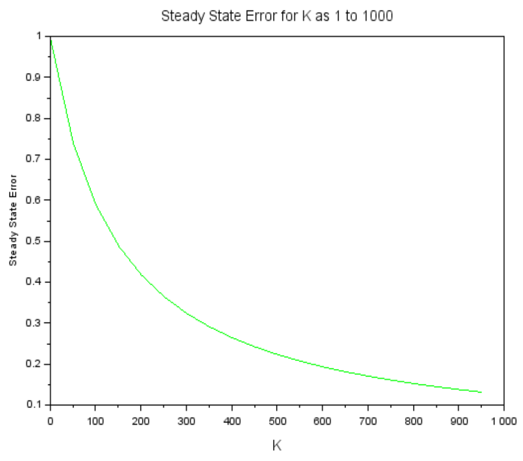


```
s = poly(0,'s');
K = %eps:0.1:1.01;
t = 0:0.01:10;
errors = [];
for i=1:size(K, 2)
    k = K(i);
    A = k/((s+3)*(s+4)*(s+12));
    G = syslin('c', A);
    T = G/.syslin('c', 1, 1);
    plot(t, csim('step', t, T),'g-');
    steady_error = 1/(1 + k/(3*4*12));
    errors = [errors, steady_error];
end
xlabel("t");
ylabel("Step response");
title("Unity gain negative feedback for KG(s)");
scf();
evans(syslin('c', 1/((s+3)*(s+4)*(s+12))), 25);
scf();
plot(K, errors, 'g—');
xlabel("K");
ylabel("Steady State Error");
title("Unity gain negative feedback for KG(s)");
```

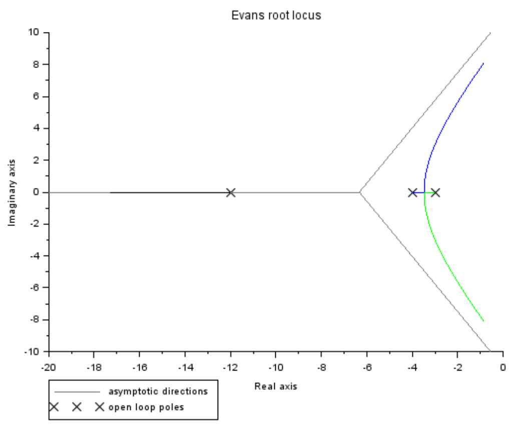
Part (e) :
Gain (K) is varied in a large range from 1 to 1000.
Observations :
1) Steady state error e(∞) decreases in the form of a hyperbola as values of K are very large.
2) The closed-loop poles lie on the real axis for smaller values of K and then break-away and more towards the imaginary axis.
3) As K increases, the response takes more time to settle as it undergoes more oscillations. This is because the closed loop poles move towards the imaginary axis.

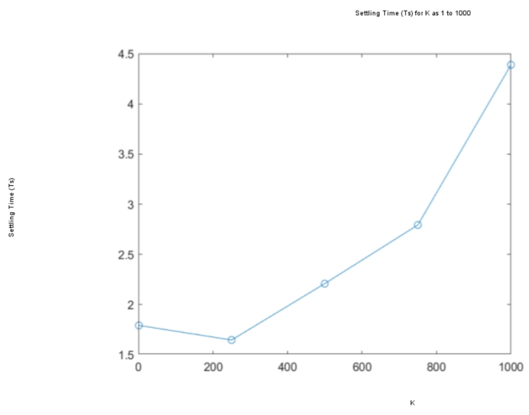


Step responses as K is increased from 1 to 1000



Steady State Error as K is increased from 1 to 1000





```
s = poly(0,'s')
K = 1:50:1000;
t = 0:0.01:10;
errors = [];
time_set = [];
for i=1:size(K, 2)
    k = K(i);
    A = k/((s+3)*(s+4)*(s+12));
    G = syslin('c', A);
    T = G /. syslin('c', 1, 1);
    Sys = csim('step', t, T)
    steady_value = Sys(size(Sys,2));
    //plot(t, csim('step', t, T), 'g-');
    time_s=();
    for i=1:size(Sys, 2)
        if(Sys(i)-(0.98*steady_value)<= 1e-5);
        time_s($+1) = t(i);
        break;
    end
    time_settle = time_s(size(time_s));
    time_set = [time_set, time_settle];
    steady_error = 1 / (1 + k / (3 * 4 * 12));
    errors = [errors, steady_error];
end
xlabel("t");
ylabel("Step response");
title("Unity gain negative feedback for KG(s)");
scf();
evans(syslin('c', 1/((s+3)*(s+4)*(s+12))), 1000);
scf();
plot(K, errors, 'g-');
xlabel("K");
ylabel("Steady State Error");
title("Steady State Error for K as 1 to 1000");
scf();
plot(K, time_set, 'g');
xlabel("K");
ylabel("Settling Time (Ts)");
title("Settling Time (Ts) for K as 1 to 1000");
```

Question 2

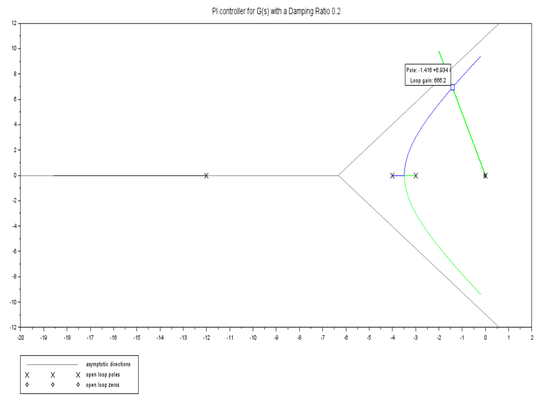
Proportional-Integral (PI) controller with transfer function (K(s+z)/s) using the root-locus method
Given open loop transfer function G(s) is,

$$G(s) = \frac{1}{(s+3)(s+4)(s+12)}$$

Part (a) :
To attain a damping ratio of 0.2.
Percentage Overshoot (%OS) would be a straight line. Slope of this line will be,

$$tan(\alpha) = -\frac{\sqrt{1-\zeta^2}}{\zeta} = 4.898$$

Intersection of this line with the root locus will give us our desired point.
Also, z = 0.01
On plotting Locus of Closed Loop Poles using `SciLab`, we get K = 666.2

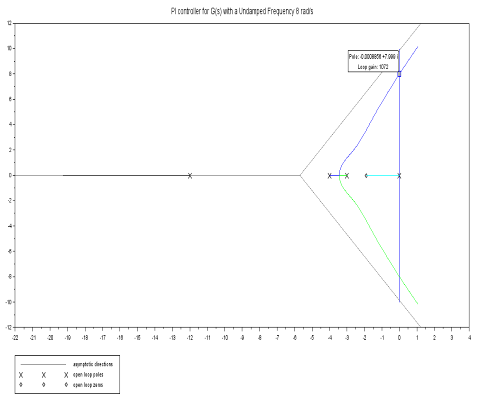


```
s = poly(0, 's');
G1 = 1/((s+3)*(s+4)*(s+12));
G = syslin('c', G1);
Zeta = 0.2;
Slope = sqrt(1-Zeta^2)/Zeta;
X = -2:0.001:0;
Y = -Slope.*X;
Z = 0.01;
P = syslin('c', G1*(s+Z)/s);
plot(X, Y, 'g-', 'LineWidth', 2);
evans(P, 1500);
title("PI controller for G(s) with a Damping Ratio 0.2");
```

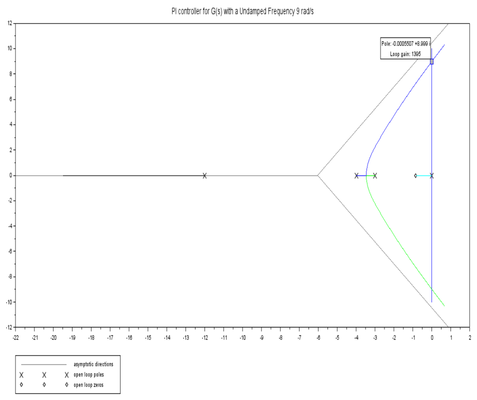
Part (b) :
Obtain undamped natural frequencies of 8 and 9 rad/s
At undamped natural frequency, root locus intersects the imaginary axis.
For G(s) = Num(s)/Den(s), system must satisfy the equation K*Num(s) + Den(s) = 0

$$\implies K(s+z) + s(s+3)(s+4)(s+12) = 0$$

By substituting s = 8j and 9j, we get (K,z) as (1072, 1.91) and (1395, 0.871) respectively.



$\omega_n = 8rad/sec$

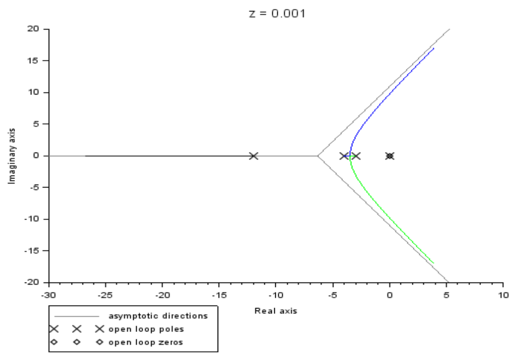


$\omega_n = 9rad/sec$

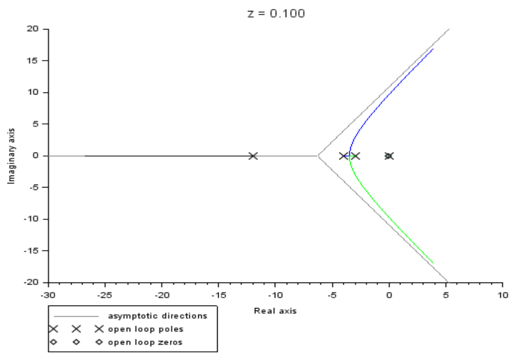
```
s = poly(0, 's');
G1 = 1/((s+3)*(s+4)*(s+12));
Zeta = 0.2;
Z = 1.91;
P = syslin('c', G1*(s+Z)/s);
Y = -10:1:10;
X = zeros(1,21);
plot(X,Y);
plot(X1, Y1, 'g--', 'LineWidth', 2);
evans(P, 2000);
title(["PI controller for G(s) with a Undamped Frequency 8 rad/s"], 'fontsize', 3);
scf();
Z = 0.871;
P = syslin('c', G1*(s+Z)/s);
Y = -10:1:10;
X = zeros(1,21);
plot(X,Y);
plot(X2, Y2, 'g--', 'LineWidth', 2);
evans(P, 2000);
title(["PI controller for G(s) with a Undamped Frequency 9 rad/s"], 'fontsize', 3);
```

Part (c) :
Vary the value of 'z' and observe its effect on the root locus of the system

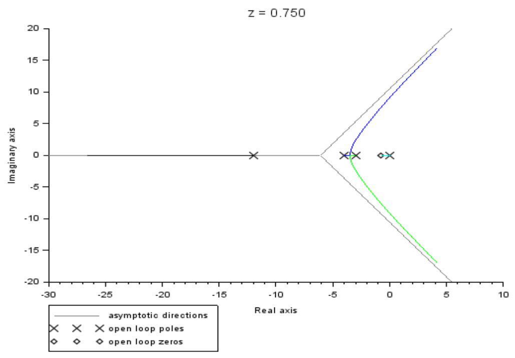
$$G(s) = \frac{s+z}{s(s+3)(s+4)(s+12)}$$



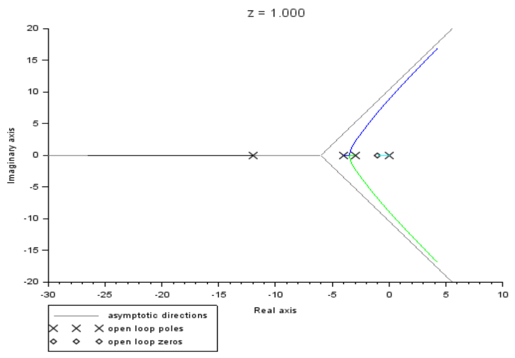
z = 0.001



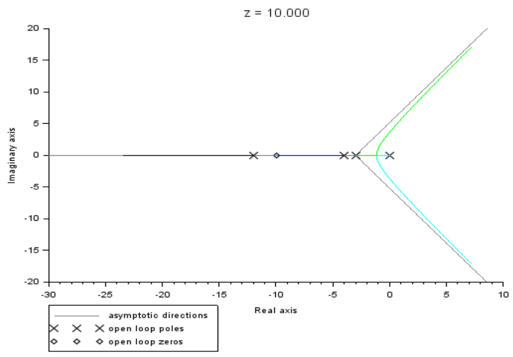
z = 0.1



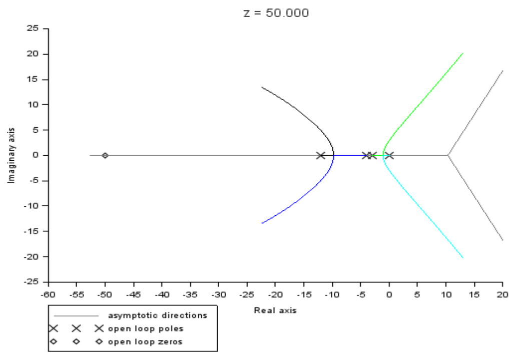
z = 0.75



z = 1



z = 5



z = 50

```
Z = [0.001, 0.1, 0.75, 1, 5, 10, 50];
for i=1:size(Z, 2)
    z = Z(i);
    G1 = 1/((s+3)*(s+4)*(s+12));
    P = syslin('c', G1*(s+z)/s);
    scf();
    evans(P, 1000);
    title([sprintf("z = %0.3f", z)], 'fontsize', 3)
end
```

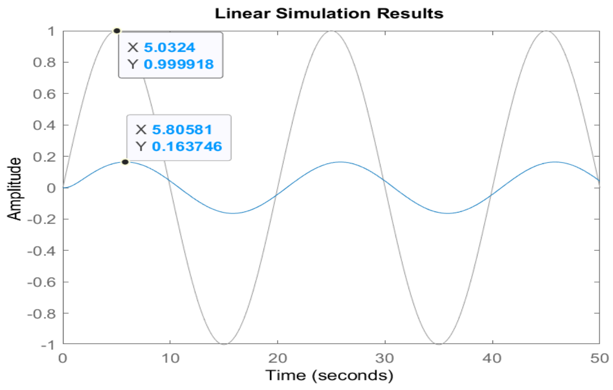
Part (d) :
No, it is not possible to change the pole locations of a system using a PI controller without changing the damping ratio (ζ). This is because a PI controller essentially increases the type of the systems so that the steady state error of the system goes to zero. This can also be observed from the above graphs. As the zero moves farther away from the origin, the root locus changes drastically (Eg. for z=5 and z=50).

Question 3

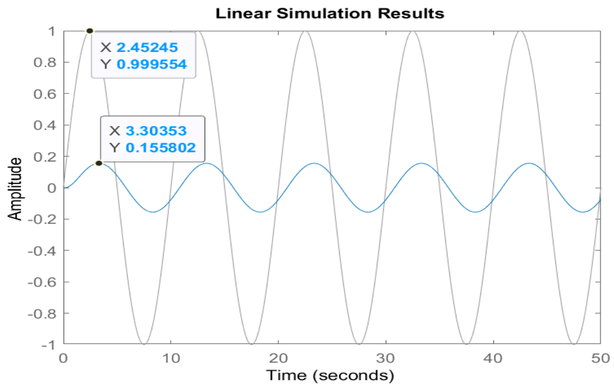
Part (a) :
Plot input and output sinusoid of varying frequency (choose 5 different frequencies) and check for a stable transfer function (I have used `MatLab`) and check that the ratio of the amplitude of output to input is |G(jω)| and the phase difference is the angle of G(jω), with omega equal to these 5 frequencies.

$$G(s) = \frac{1}{(s^2 + 5s + 6)}$$

$$Gain = \frac{Peak_1}{Peak_2} \quad | \quad Phaseshift = (Peak_1 - Peak_2) * 2 * \pi * f$$



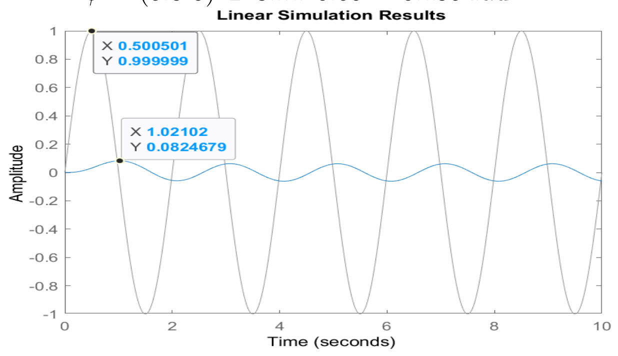
f = 0.05Hz
Gain = 0.163



f = 0.1Hz
Gain = 0.155

ϕ = (5.8-5)*2*3.14*0.05 = 0.486 rad

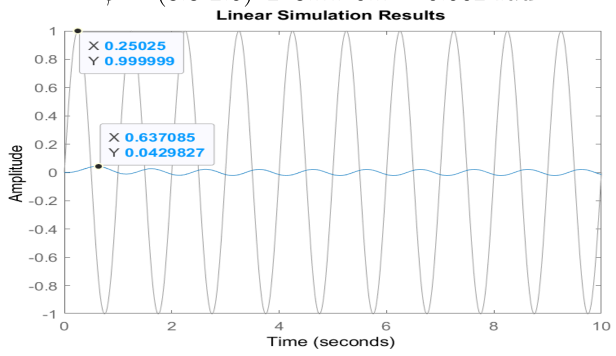
ϕ = (3.3-2.5)*2*3.14*0.1 = 0.502 rad



f = 0.5Hz

Gain = 0.082

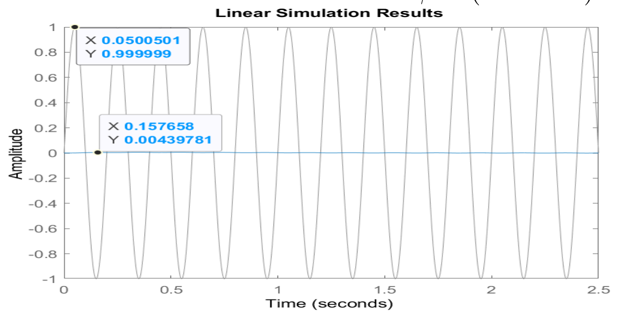
ϕ = (1.02-0.5)*2*3.14*0.5 = 1.632 rad



f = 1Hz

Gain = 0.043

ϕ = (0.63-0.25)*2*3.14*1 = 2.386 rad



f = 5Hz

Gain = 0.0043

ϕ = (0.157-0.05)*2*3.14*5 = 3.359 rad

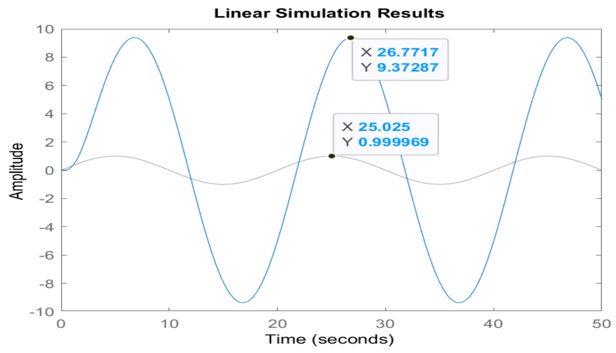
```
s = tf('s')
G = 1/(s^2+5*s+6)
t = linspace(0,50,1000)

Input = sin(2*pi*0.05*t);
figure();
lsim(G, Input, t)
Input = sin(2*pi*0.1*t);
figure();
lsim(G, Input, t)
t = linspace(0,10,1000)
Input = sin(2*pi*0.5*t);
figure();
lsim(G, Input, t)
t = linspace(0,10,1000)
Input = sin(2*pi*1*t);
figure();
lsim(G, Input, t)
t = linspace(0,2.5,1000)
Input = sin(2*pi*5*t);
figure();
lsim(G, Input, t)
```

Part (b) :

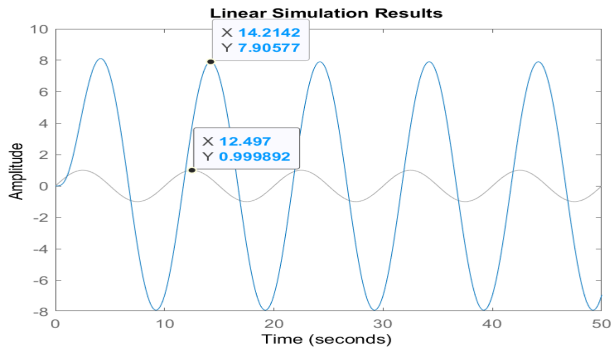
The desired relation (between phase difference and angle of G(j) is for frequency measured in **`rad/s`** **Part (c) :**

$$G(s) = \frac{60}{(s^3 + 6s^2 + 11s + 6)}$$

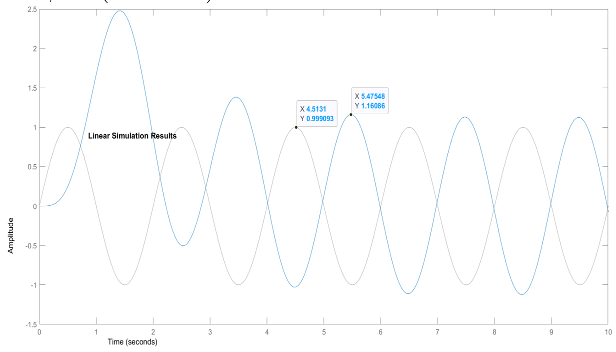
$$Gain = \frac{Peak_1}{Peak_2} \mid Phaseshift = (Peak_1 - Peak_2) * 2 * \pi * f$$



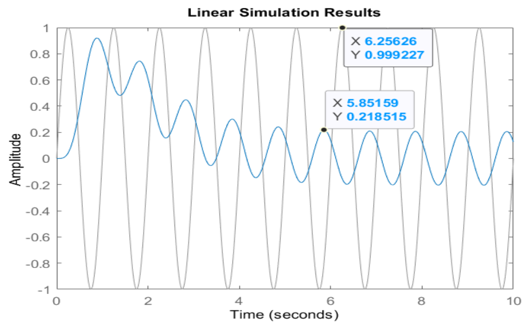
f = 0.05Hz
Gain = 9.372
ϕ = (26.77-25)*2*3.14*0.05 = 0.555 rad



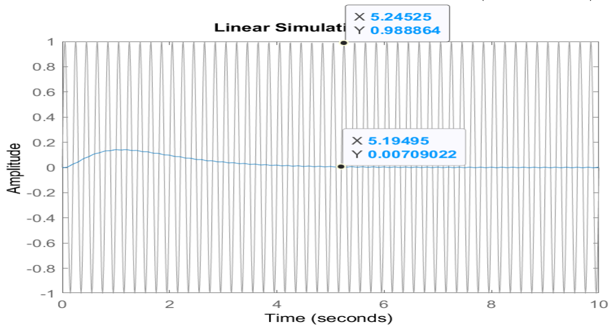
f = 0.1Hz
Gain = 7.905
ϕ = (14.21-12.5)*2*3.14*0.1 = 1.073 rad



f = 0.5Hz
Gain = 1.160
ϕ = (5.47-4.51)*2*3.14*0.5 = 3.093 rad



f = 1Hz
Gain = 0.218
ϕ = (6.25-5.85)*2*3.14*1 = 3.752 rad



f = 5Hz
Gain = 0.007
ϕ = (5.24-5.16)*2*3.14*5 = 4.418 rad

The phase angle between input and output is π at f = 0.52 Hz (using trial and error method)
Numerator 60 will have it's effect only in Gain but not in phase as it depends mainly on poles and their locations which depend only on denominator.

References

1) https://www.mathworks.com/
2) https://help.scilab.org/