Question 1)

Question 1) b) RL plot

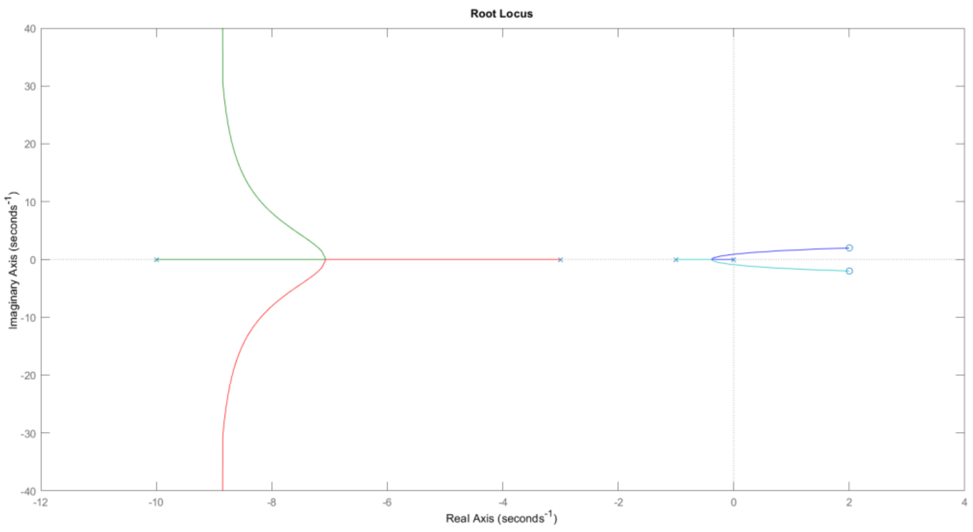


Figure 1.1: Root Locus Plot generated using rlocusplot in MATLAB

We can also see that the asymptotes meet at -9 (centroid) and at angles of 90 and 270 degrees respectively

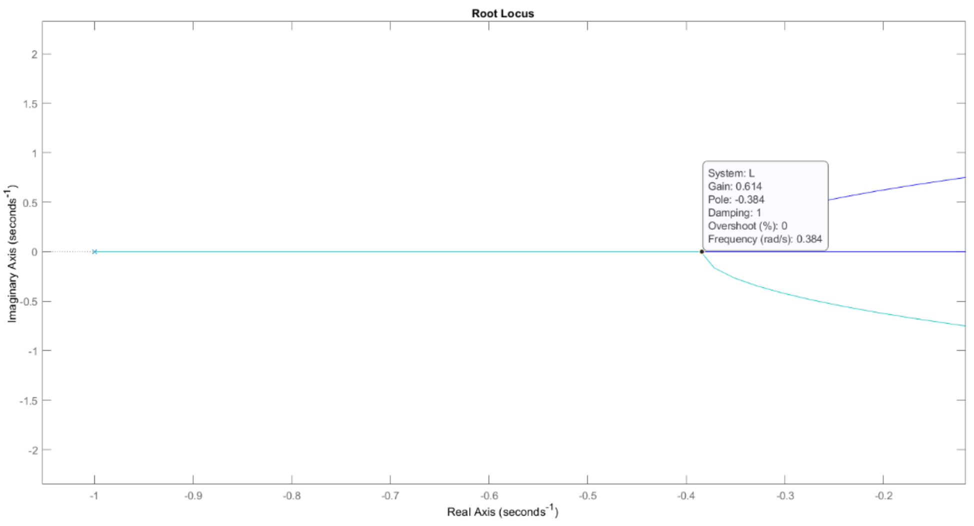


Figure 1.2 a): Verifying break away point-1

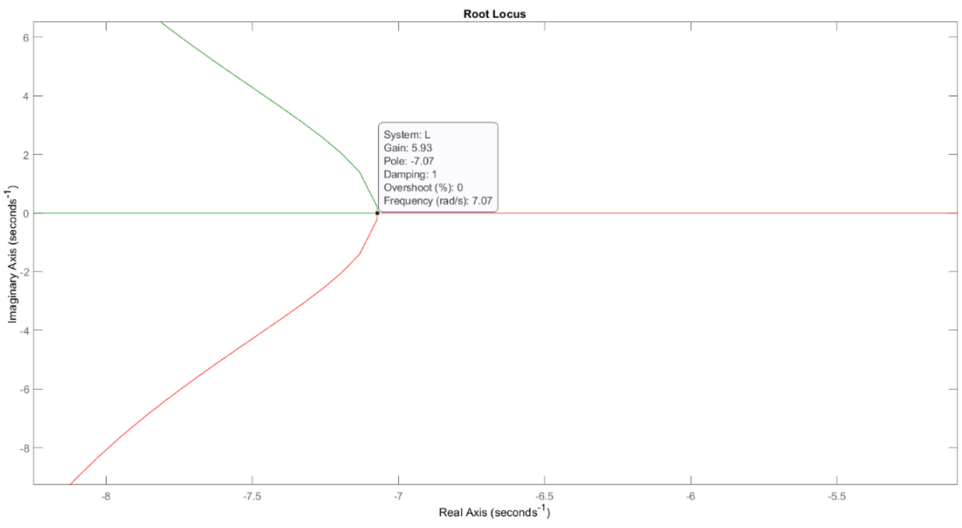


Figure 1.2 b): Verifying break away point-2

Question 1) c) Ultimate Gain verification

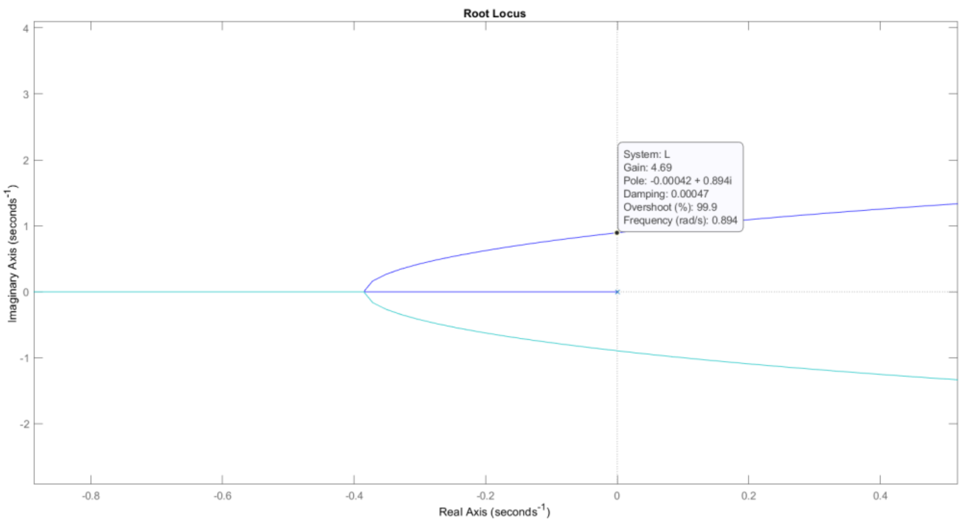


Figure 1.3: Finding the cross-over point.

From the above plot we can see that at approximately the cross-over point, the gain is approximately 4.69. This is very close to the analytically computed value of 4.696 hence verifying it.

Question 1) d) Minimising settling time

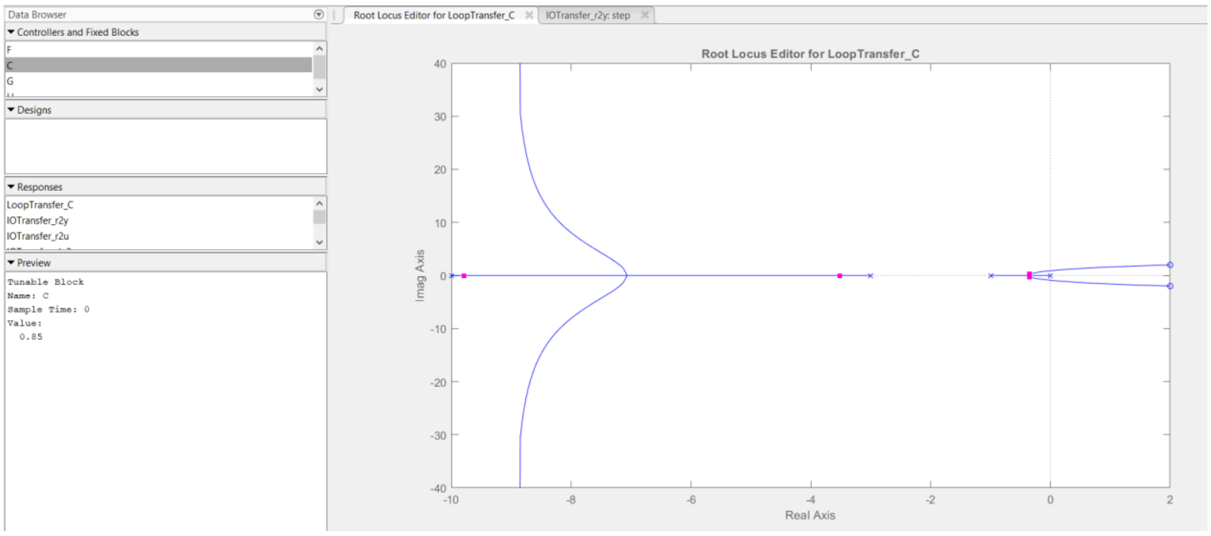
We know that $K_C > 0$, and $K_C < K_{C,U} = 4.696$.

So the K_C value which gives the minimum settling time must be between these 2 values. Employing the function '**stepinfo**' which gives the settling time (apart from an assortment of other information about the step response), we evaluate settling time for all K_C values between 0 and 4.69 (with 2 decimal place accuracy) and find the K_C which gives the minimum value.

The required **K$_C$ = 0.85** and it has a settling time of 9.0798 units.

This is further verified using rltool. We need to get the dominant pole away from origin as far as possible to quicken the dynamics of the system. (And that the same time all poles should be in LHP)

This is achieved in this configuration:



```
Tunable Block
Name: C
Sample Time: 0
Value:
  0.85
```

Figure 1.4: RL plot using rltool at minimum settling time

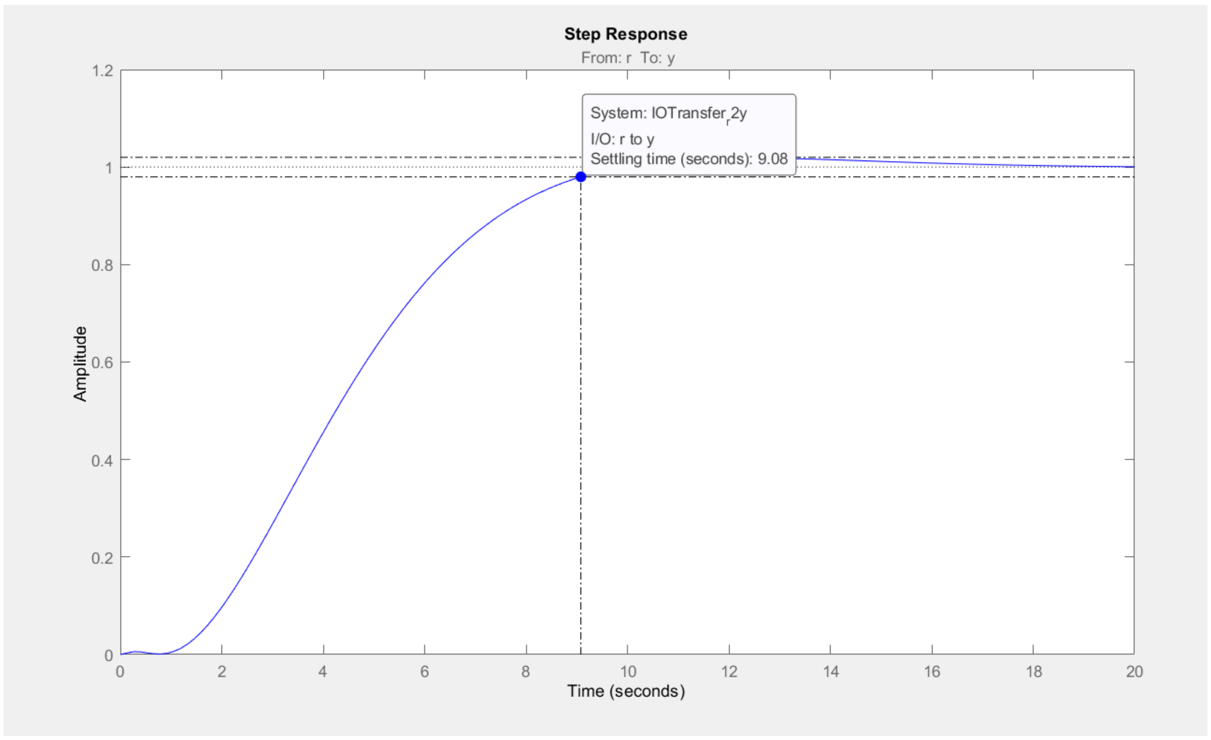


Figure 1.5: Settling time for the optimal Kc.

Question 1) e) Ultimate gain for K$_I$

Given that the equation for solving s = jw is complicated, it is much easier to use the root-locus plot generated by MATLAB.

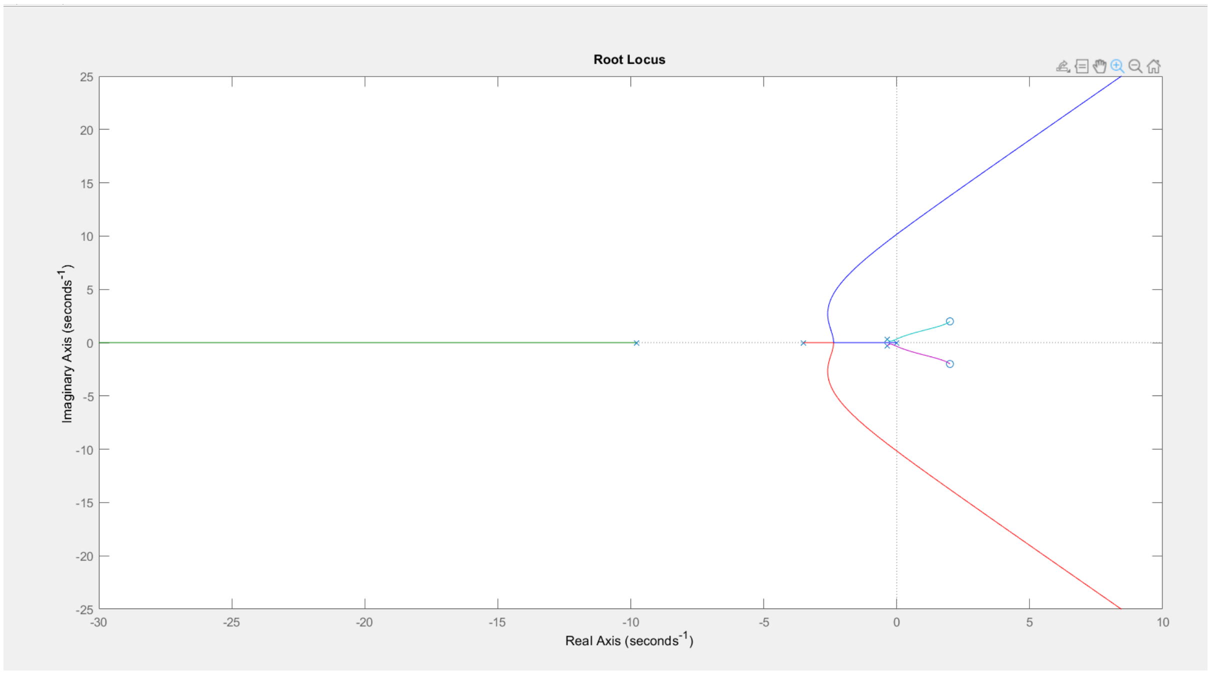
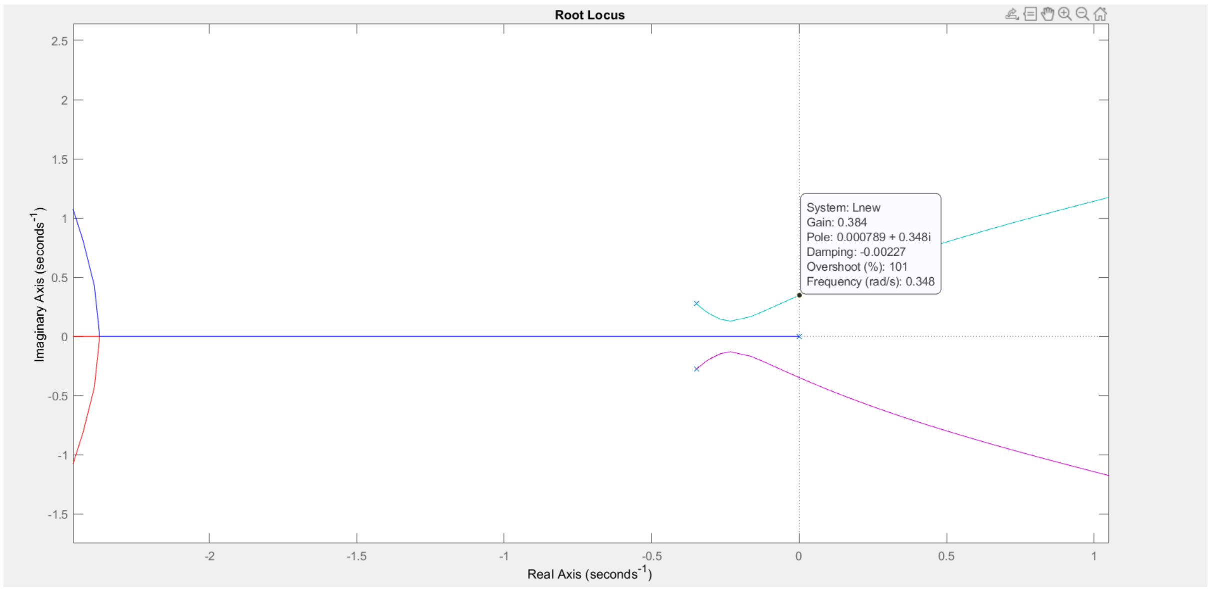


Figure 1.6 a): The required root locus plot



System: Lnew
Gain: 0.384
Pole: 0.000789 + 0.348i
Damping: -0.00227
Overshoot (%): 101
Frequency (rad/s): 0.348

Figure 1.6 b): Finding the cross-over point

$K_{I,ultimate}$ = 0.384.

Note: there is a another cross-over point at 1.540*10^3. But if K_I>0.38 there will be RHP poles (although there will be some LHP poles also). So the ultimate gain is the gain at the first cross-over point.

Question 2)

Question 2) a) P-Control design for given GM

We can use the Bode stability criteria and obtain the Gain margin for $K_C = 1$. So L is simply equal to G_P as given below.

```
L =
 
                     2 s + 8
  exp(-2*s) * ----------------
              10 s^2 + 7 s + 1
 
Continuous-time transfer function.
```

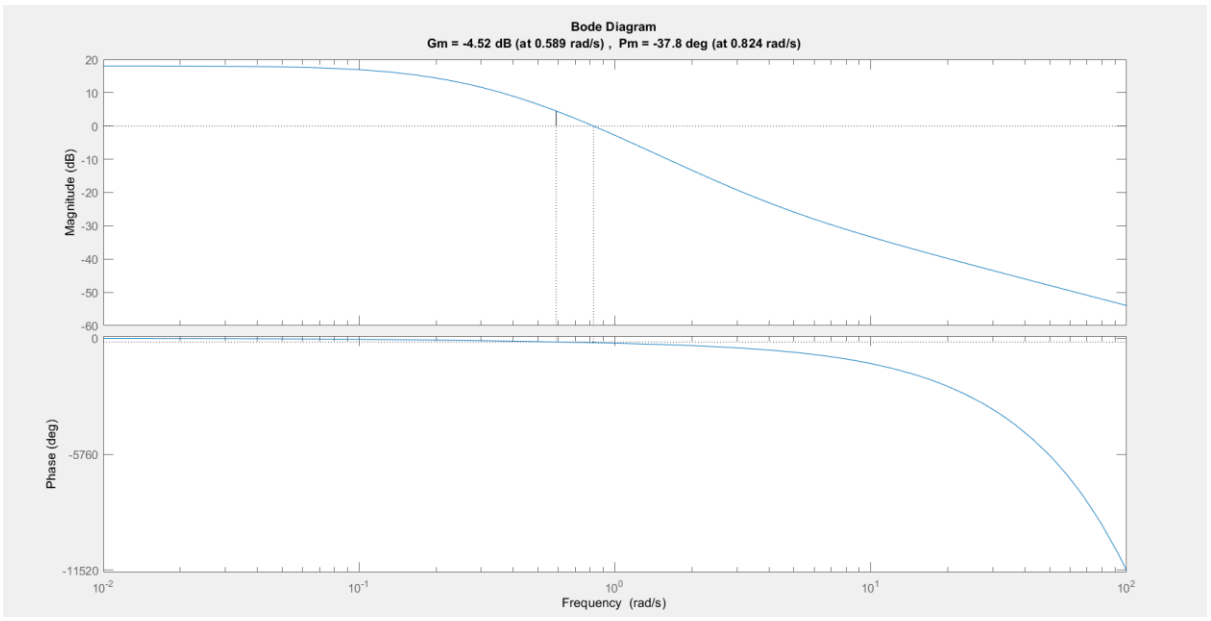


Figure 2.1: Bode Diagram (using 'margin' function).

G_M = -4.52 dB

So using this gain margin and the definition of gain margin,

$G_M = \log(\frac{1}{|L_G|})$ and $L_G = G_C G_P = K_C * G_P$

We can get the required K_C as

$$K_C = 10^{\frac{(G_M - G_{M,reqd})}{20}}$$

where $G_{M,required}$ = 8.2 dB

K_C was found to be **0.2311**. The corresponding phase margin = **73.815 degrees**

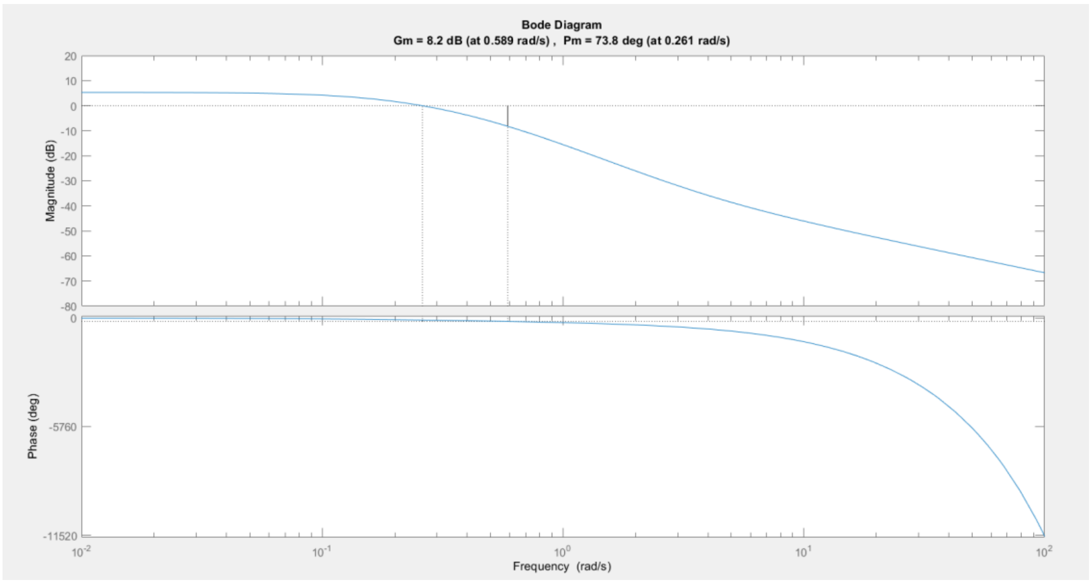


Figure 2.2: Margins of the required P-Controller

Question 2) b) Delay uncertainty

Value of delay uncertainty = Phase Margin/gain cross-over frequency

(since, Phase is directly proportional to –w*Delay)

Gain cross-over frequency = 0.2607 rad/s

=> **Delay Uncertainity = 4.942 units**

Question 2) c) PI Controller

Unlike part a), changing τ_I changes both magnitude and phase responses of the system.

So one option is to solve for gain cross over frequency and the controller parameter τ_I simultaneously by imposing

$$|L(j\omega)| = 1 \tag{1}$$

$$phase\big(L(j\omega)\big) = 60 \tag{2}$$

Instead, I used the function 'margin' to compute the phase margin for the closed loop system given a τ_I. This results in simply solving for tau such that phase is 60.

```
%% function that gives 60-PM for a given tau
function P  = func(tauI,L,Kc)
    s = tf('s');
    [~,PM,~,~] = margin(L*Kc*(1+1/tauI/s));
    P = 60 - PM;
end
```

PI controller parameter τ_I was found to be **17.8327** units

The Gain margin was found be **7.46 dB**

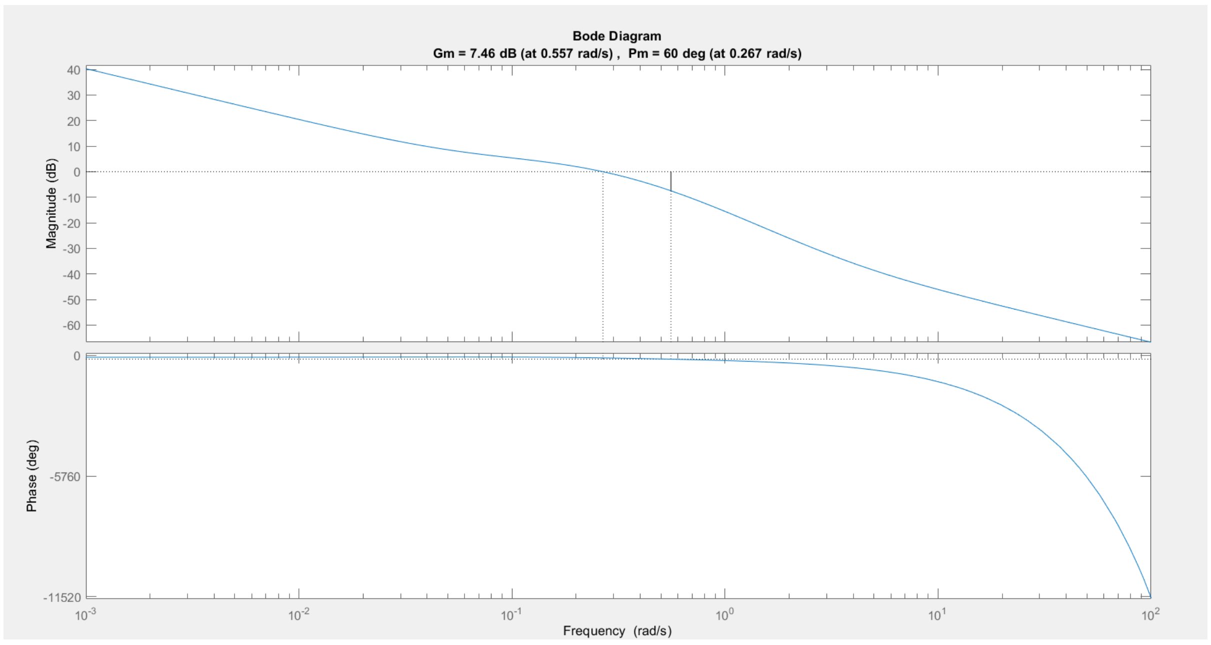


Figure 2.3: Bode diagram for the closed loop system with the PI controller

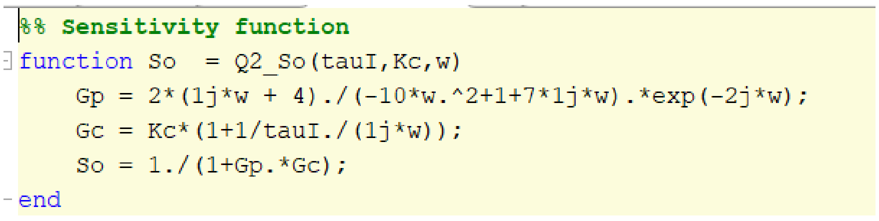
Question 2) d) Verifying Bode's Senstitivity Integral

$$\int_0^\infty \log|S_O(j\omega)|d\omega = 0$$

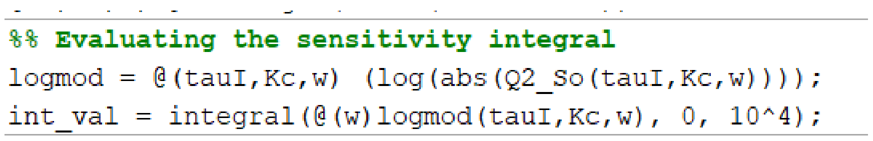
where $S_O(j\omega) = \frac{1}{1+Gp(j\omega)Gc(j\omega)}$

The above integral is verified using the following script:

1. PI Controller



```
%% Sensitivity function
function So  = Q2_So(tauI,Kc,w)
    Gp = 2*(1j*w + 4)./(-10*w.^2+1+7*1j*w).*exp(-2j*w);
    Gc = Kc*(1+1/tauI./(1j*w));
    So = 1./(1+Gp.*Gc);
end
```



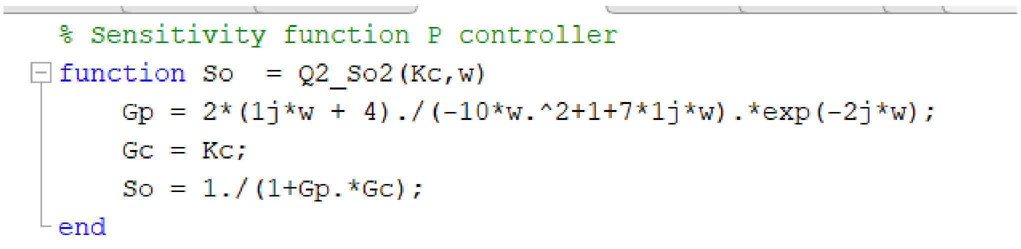
```
%% Evaluating the sensitivity integral
logmod = @(tauI,Kc,w)  (log(abs(Q2_So(tauI,Kc,w))));
int_val = integral(@(w)logmod(tauI,Kc,w), 0, 10^4);
```

(Since it is numerical integration, I replaced infinity with a high value instead. Higher the value, closer it gets to zero)

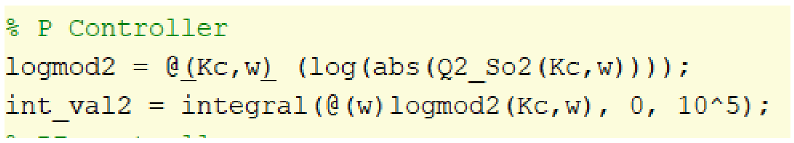
The integral turns out to be, int_val =   -1.8790e-06 which is sufficiently close to 0!

Thus, the Bode's sensitivity integral holds for the PI controller!

2. P Controller



```
% Sensitivity function P controller
function So  = Q2_So2(Kc,w)
    Gp = 2*(1j*w + 4)./(-10*w.^2+1+7*1j*w).*exp(-2j*w);
    Gc = Kc;
    So = 1./(1+Gp.*Gc);
end
```



```
% P Controller
logmod2 = @(Kc,w) (log(abs(Q2_So2(Kc,w))));
int_val2 = integral(@(w)logmod2(Kc,w), 0, 10^5);
```

The integral turns out to be, int_val =   -2.3075e-07which is sufficiently close to 0!

Thus, the Bode's sensitivity integral holds for P-Controller also!