AAE 364: Control Systems Analysis

HW8: Controller Design and Root Locus Anaylsis

Dr. Sun

School of Aeronautical & Astronautical Engineering

Purdue University

Tomoki Koike

Friday March 27th 2020

B–6–12. Plot root-locus diagrams for the nonminimum-phase systems shown in Figures 6–102(a) and (b), respectively.

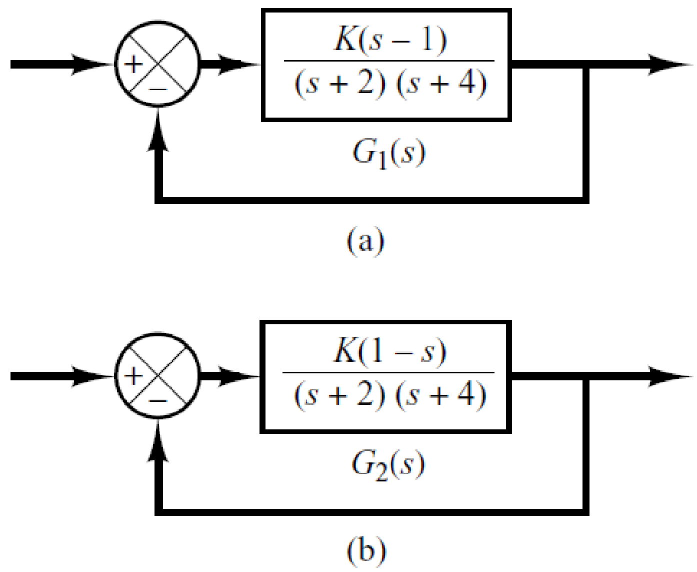


Figure 6–102 (a) and (b) Nonminimum-phase systems.

(a)

This a feedback system. Then the CE becomes

$$CE := 1 + K\frac{(s-1)}{(s+2)(s+4)} \qquad \text{where} \quad L(s) = \frac{s-1}{(s+2)(s+4)}$$

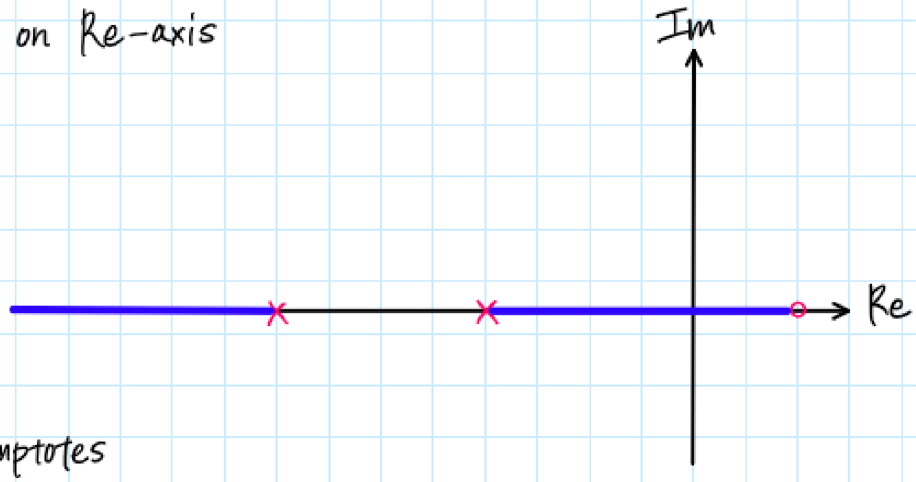
<i> Poles and Zeros

Poles: $(s+2)(s+4) = 0 \Rightarrow s = -2, -4 \qquad \therefore n = 2$

zeros: $s - 1 = 0 \Longrightarrow s = 1 \qquad \therefore m = 1$

<ii> Symmetry TRUE

<iii> RL on Re-axis



<iv> Asymptotes

$$\theta_a = \frac{180^\circ + 360^\circ l}{n - m} = \frac{180^\circ + 360^\circ l}{2 - 1} = 180^\circ + 360^\circ l \quad (l = 0)$$

$$\theta_a = 180^\circ$$

and

$$\sigma_a = \frac{\sum p_i - \sum z_i}{n - m} = \frac{(-2) + (-4) - 1}{2 - 1} = -7$$

<V> Break-in/away points

$$\frac{d}{ds}\left[-\frac{1}{L(s)}\right] = 0 \Rightarrow \frac{d}{ds}\left[-\frac{(s+2)(s+4)}{s-1}\right] = 0$$

$$\frac{-s^2+2s+14}{(s-1)^2} = 0$$

$$\begin{cases} \hat{s}_1 = 4.8730 \\ \hat{s}_2 = -2.8730 \end{cases} \longrightarrow$$ do not reside in $s \in [-2, 1]$, $s \in (-\infty, -4]$

$\hat{s}_1$ & $\hat{s}_2$ are NOT break-in/away points

<vi> Angle of departure

since we know that all poles and zeros are on the Re-axis departure/arrival angles are unecessary

<vii> Intersection of RL with Im-axis

$$1 + \hat{k}L(j\hat{\omega}) = 0$$

$$1 + \hat{k} \cdot \frac{j\hat{\omega} - 1}{(j\hat{\omega}+2)(j\hat{\omega}+4)} = 0$$

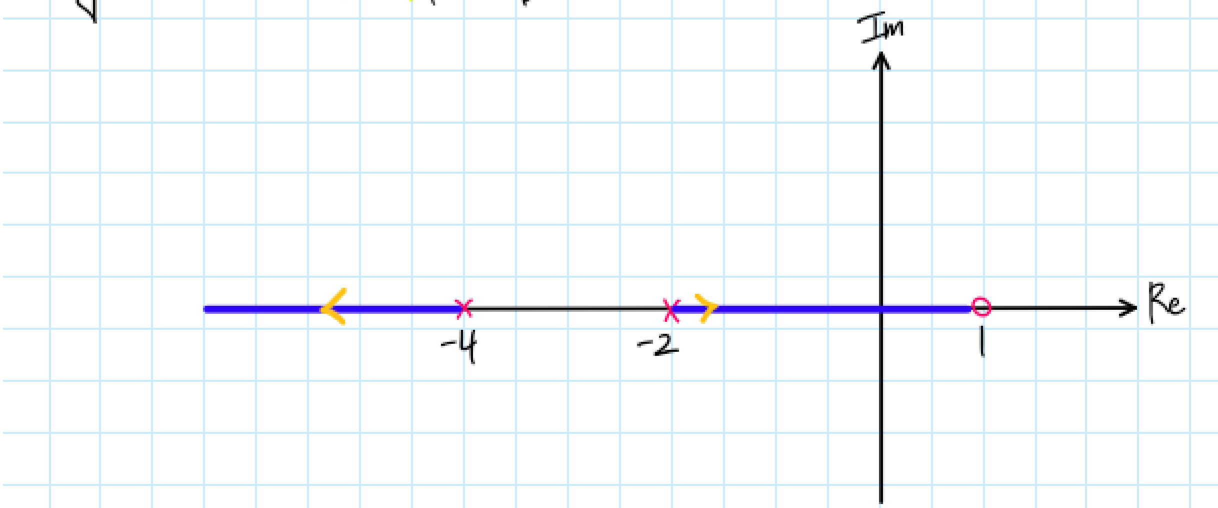
solving this we get Re : $-\hat{k} = \hat{\omega}^2 - 8$

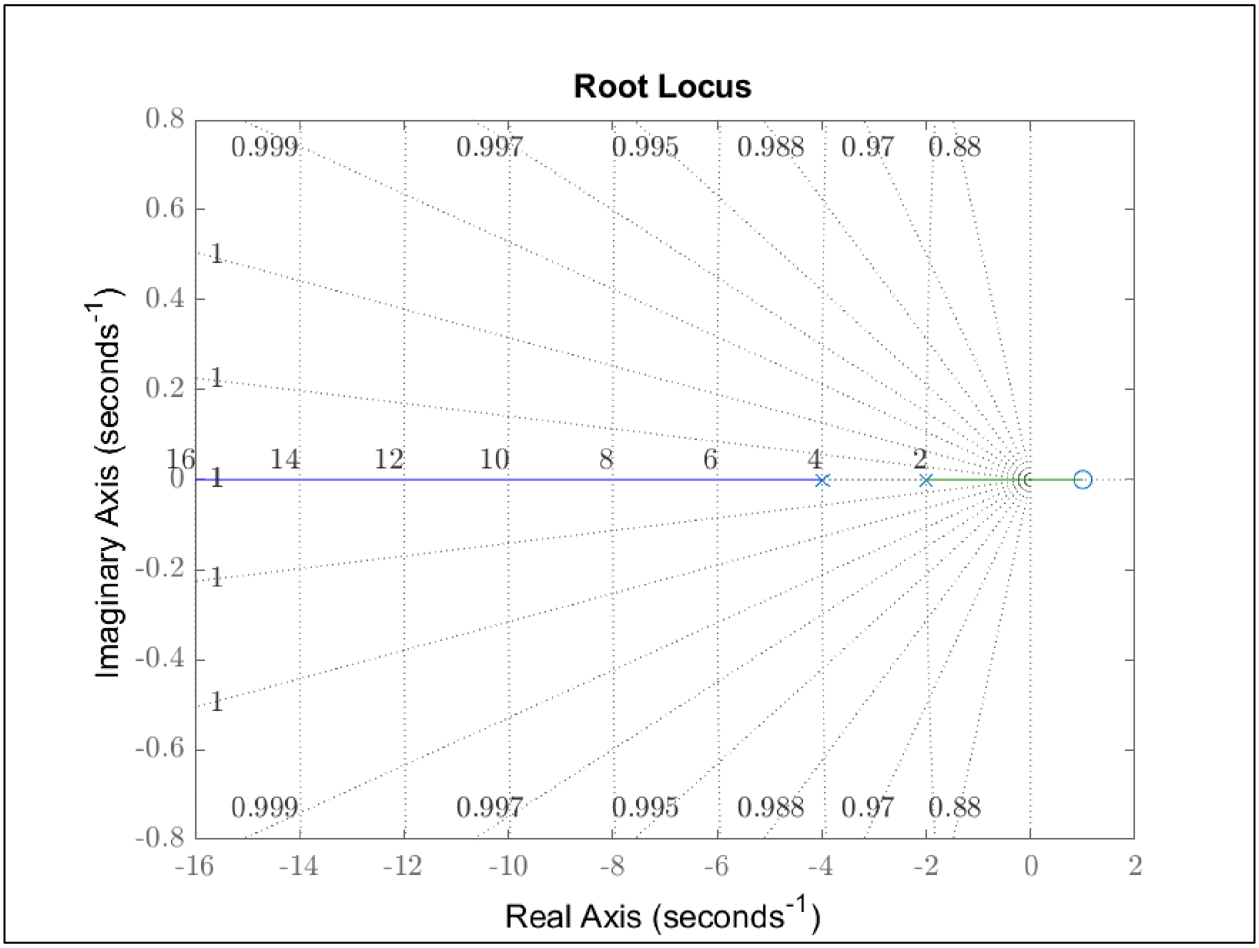
Im : $\hat{k}\hat{\omega} = -6\hat{\omega}$

$(*\hat{k} > 0)$

$$\hat{k} = \begin{bmatrix} -6 \\ -6 \\ 8 \end{bmatrix}, \quad \hat{\omega} = \begin{bmatrix} -3.7417 \\ 3.7417 \\ 0 \end{bmatrix} \longrightarrow$$ but there is NO intersection with Im-axis

from <i> ~ <vii> RL becomes





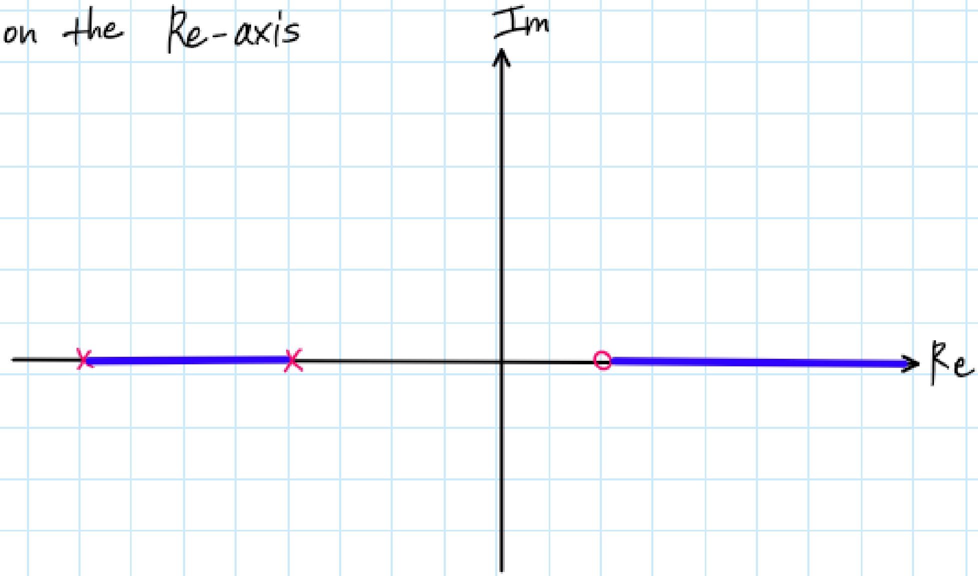
(b) This becomes a positive feedback

$$CE := 1 - k\frac{(s-1)}{(s+2)(s+4)} \quad \text{where} \quad L(s) = \frac{s-1}{(s+2)(s+4)}$$

<i> Poles and zeros same as (a)

<ii> Symmetry TRUE

<iii> RL on the Re-axis



<iv> Asymptotes

$$\theta_a = \frac{360^\circ l}{n-m} = 360^\circ l \qquad l = 0$$

and

$$\sigma_a = -\frac{\Sigma P_i - \Sigma Z_i}{n-m} = 7$$

<v> Break-in/away points same as (a)

$$\hat{s}_1 = 4.8730$$
$$\hat{s}_2 = -2.8730$$

$\longrightarrow$ resides on the blue domain on the figure above. So both are break-in/away points

<vi> Angle of departure

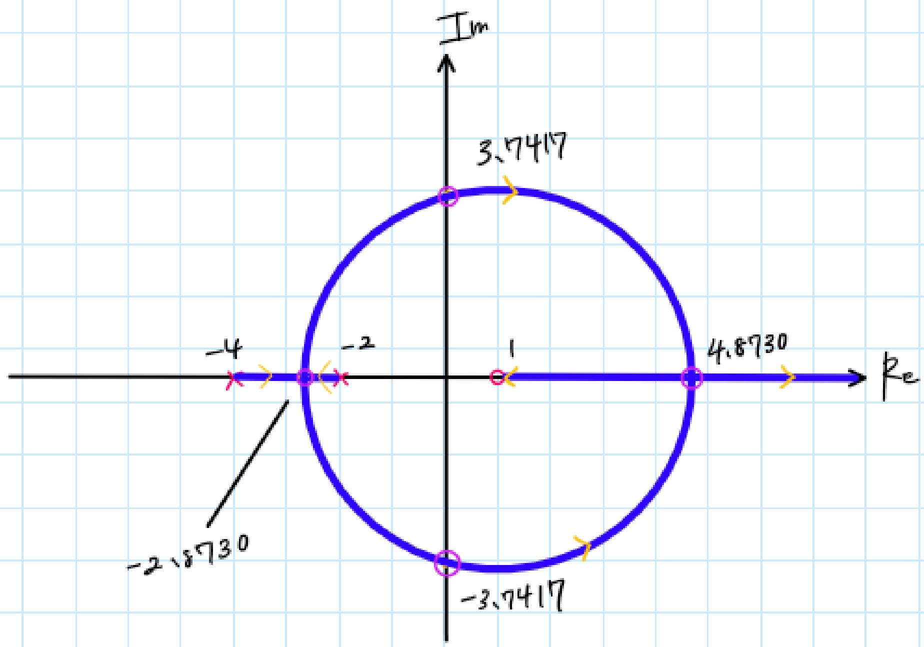
unecessary for the same reason as (a)

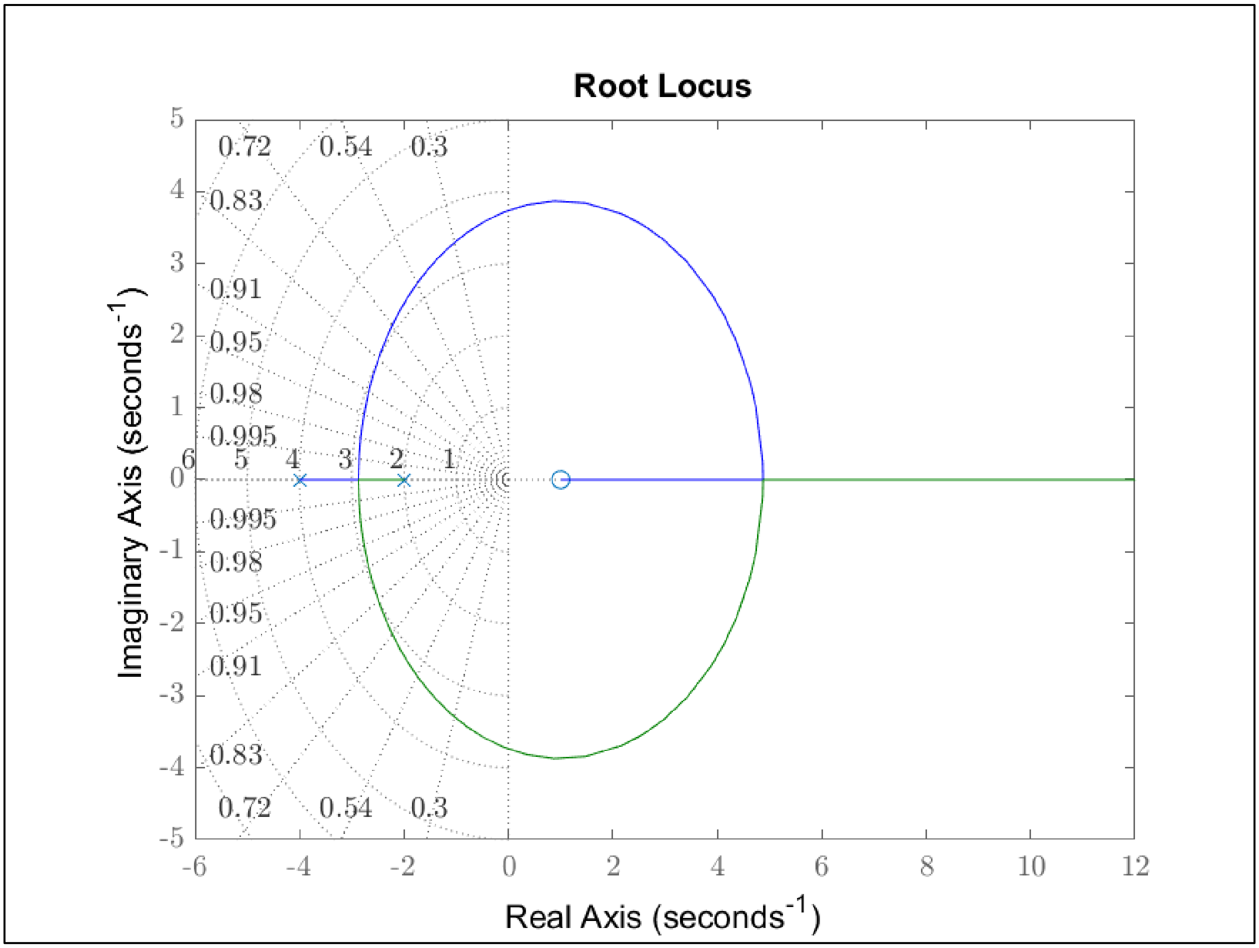
<vii> Intersection of RL with Im-axis

→ Same as (a)

$$\hat{k} = \begin{bmatrix} -6 \\ -6 \\ 8 \end{bmatrix}, \quad \hat{\omega} = \begin{bmatrix} -3.7417 \\ 3.7417 \\ 0 \end{bmatrix} \quad (^{*}\hat{k} < 0)$$

from <i> ~ <vii> RL becomes (non minimum phase system)




Root Locus
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})
0.72
0.54
0.3
0.83
0.91
0.95
0.98
0.995
6
5
4
3
2
1
0.995
0.98
0.95
0.91
0.83
0.72
0.54
0.3
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8
10
12

B–6–14. Consider the system shown in Figure 6–104. Plot the root loci for the system. Determine the value of K such that the damping ratio ζ of the dominant closed-loop poles is 0.5. Then determine all closed-loop poles. Plot the unit-step response curve with MATLAB.

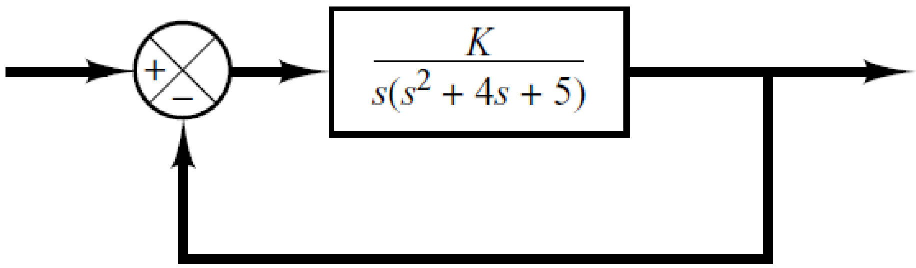


Figure 6–104 Control system.

(a) find the RL

Since this is a negative feedback

$$CE := 1 + k\frac{1}{s(s^2+4s+5)} \quad \text{where} \quad L(s) = \frac{1}{s(s^2+4s+5)}$$

<i> Poles and zeros

Poles: $s(s^2+4s+5)=0 \implies s = 0, -2\pm j \implies n = 3$

zeros: none $\implies m = 0$

<ii> Symmetry **TRUE**

<iii> RL on Re-axis

<iv> Asymptotes

$$\theta_a = \frac{180^\circ + 360^\circ \ell}{n-m} = 60^\circ + 120^\circ \ell \qquad \ell = 0, 1, 2$$

$$\theta_a = 60^\circ, 180^\circ, 300^\circ$$

and

$$\sigma_a = \frac{\Sigma P_i - \Sigma Z_i}{n-m} = \frac{0 + (-2+i) + (-2-i)}{3} = -\frac{4}{3}$$

<v> Break-in/away points

$$\frac{d}{ds}\left(-\frac{1}{L(s)}\right) = -\frac{d}{ds}\left(s(s^2+4s+5)\right) = 0$$

$$-3s^2 - 8s - 5 = 0$$

$$\Rightarrow \hat{s}_1 = -1.6667 \ , \ \hat{s}_2 = -1$$

<vi> Angle of departure

defining a hypothetical point s^d in the proximity of $-2+j$

$$\angle L(s^d) = -180° = 0 - \arg(-2+j-0) - \arg(-2+j-(-2-j)) - \theta_d$$

$$\Rightarrow \theta_d = -63.4349°$$

<vii> Intersection of RL with Im-axis

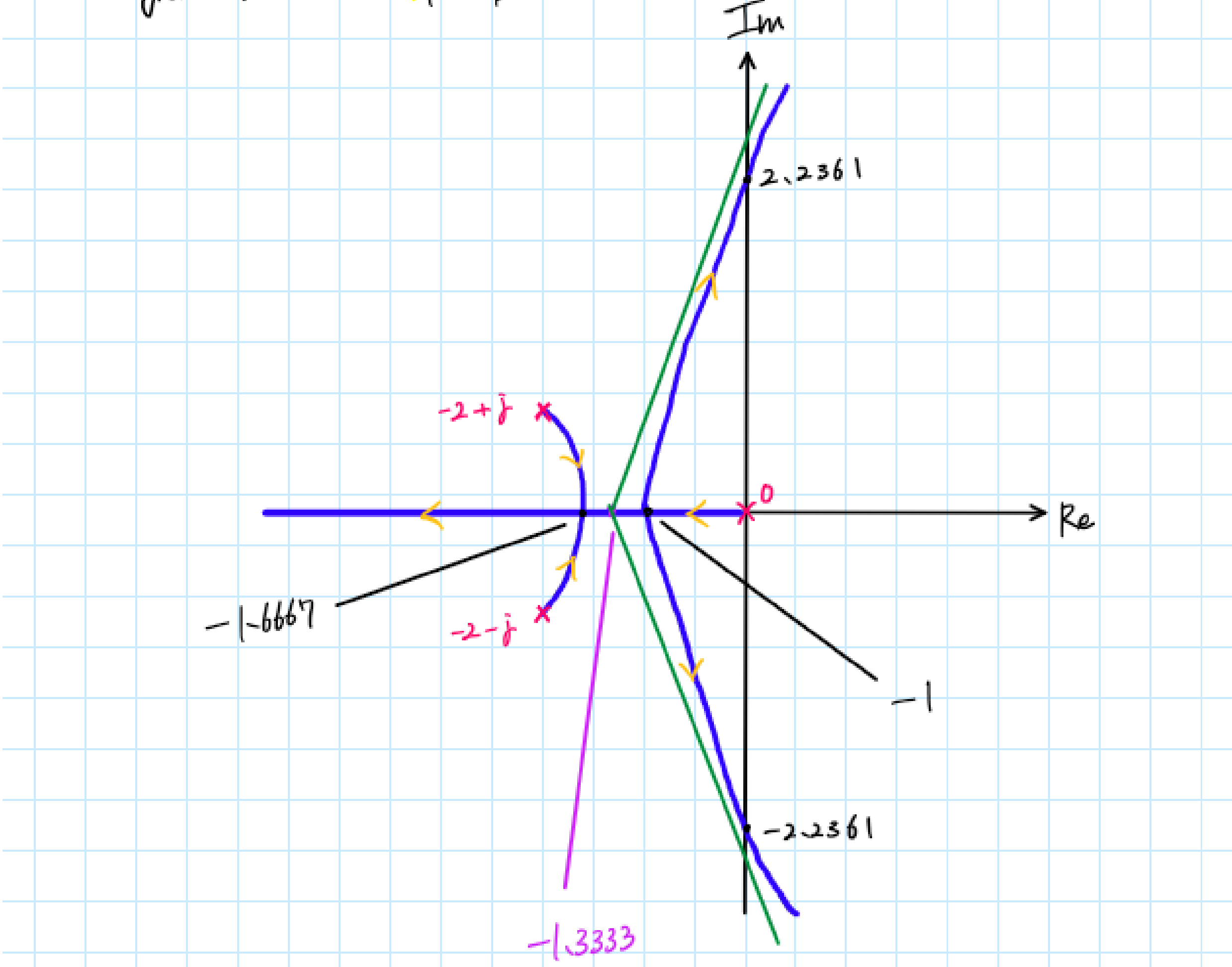
$$1 + \hat{k} L(j\hat{\omega}) = 0 \Rightarrow 1 + \hat{k} \frac{1}{j\hat{\omega}(-\hat{\omega}^2 + 4j\hat{\omega} + 5)} = 0$$

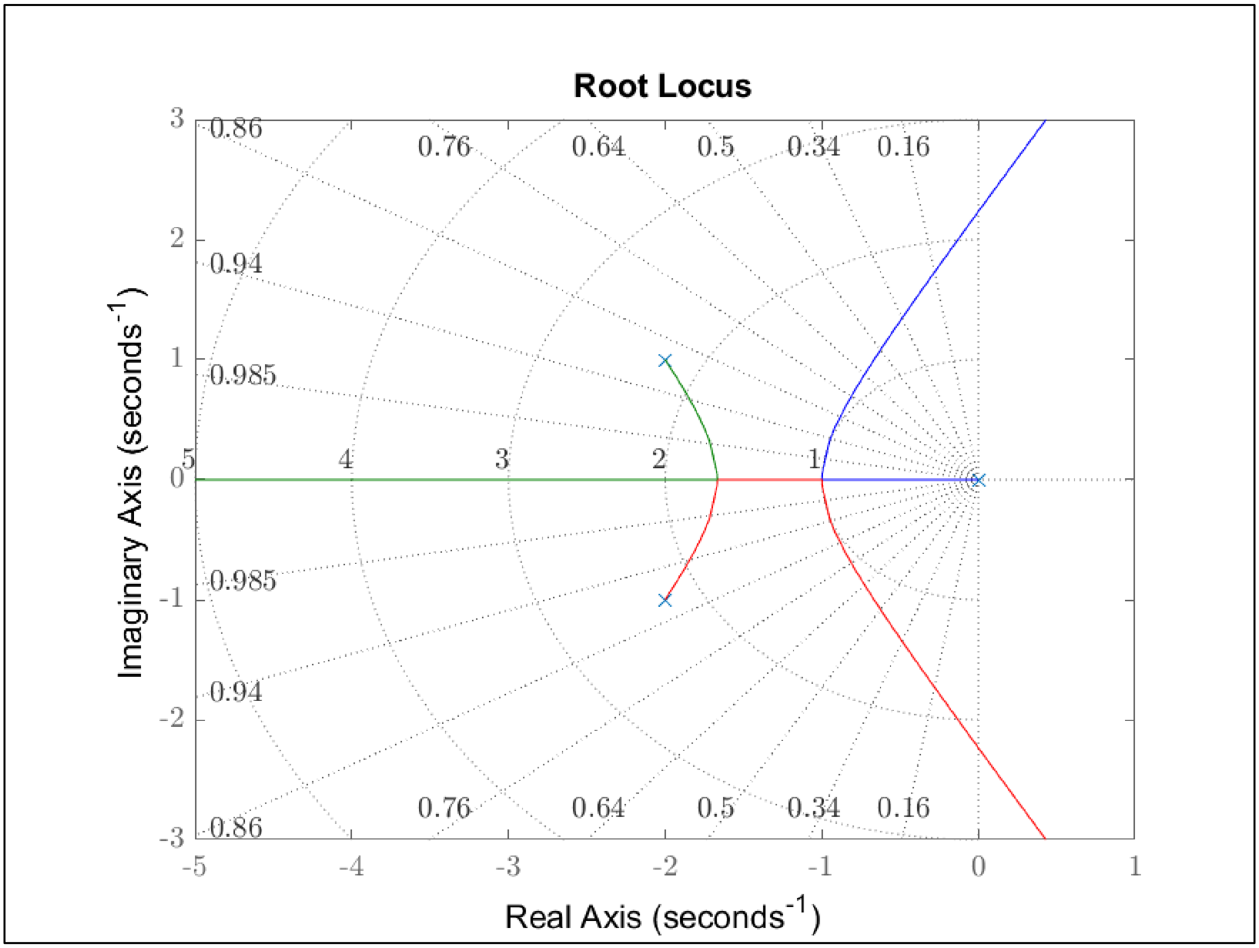
$$\Rightarrow \hat{k} = j\hat{\omega}^3 + 4\hat{\omega}^2 - 5j\hat{\omega}$$

$$\Rightarrow \begin{aligned} Im &: \quad 0 = \hat{\omega}^3 - 5\hat{\omega} \\ Re &: \quad \hat{k} = 4\hat{\omega}^2 \end{aligned}$$

$$\hat{k} = \begin{bmatrix} 0 \\ 20 \\ 20 \end{bmatrix} \qquad \hat{\omega} = \begin{bmatrix} 0 \\ -2.2361 \\ 2.2361 \end{bmatrix}$$

from <i> ~ <vii> RL becomes




Root Locus
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})
0.86
0.76
0.64
0.5
0.34
0.16
0.94
0.985
0.985
0.94
0.76
0.64
0.5
0.34
0.16
0.86
5
4
3
2
1
3
2
1
0
-1
-2
-3
-5
-4
-3
-2
-1
0
1

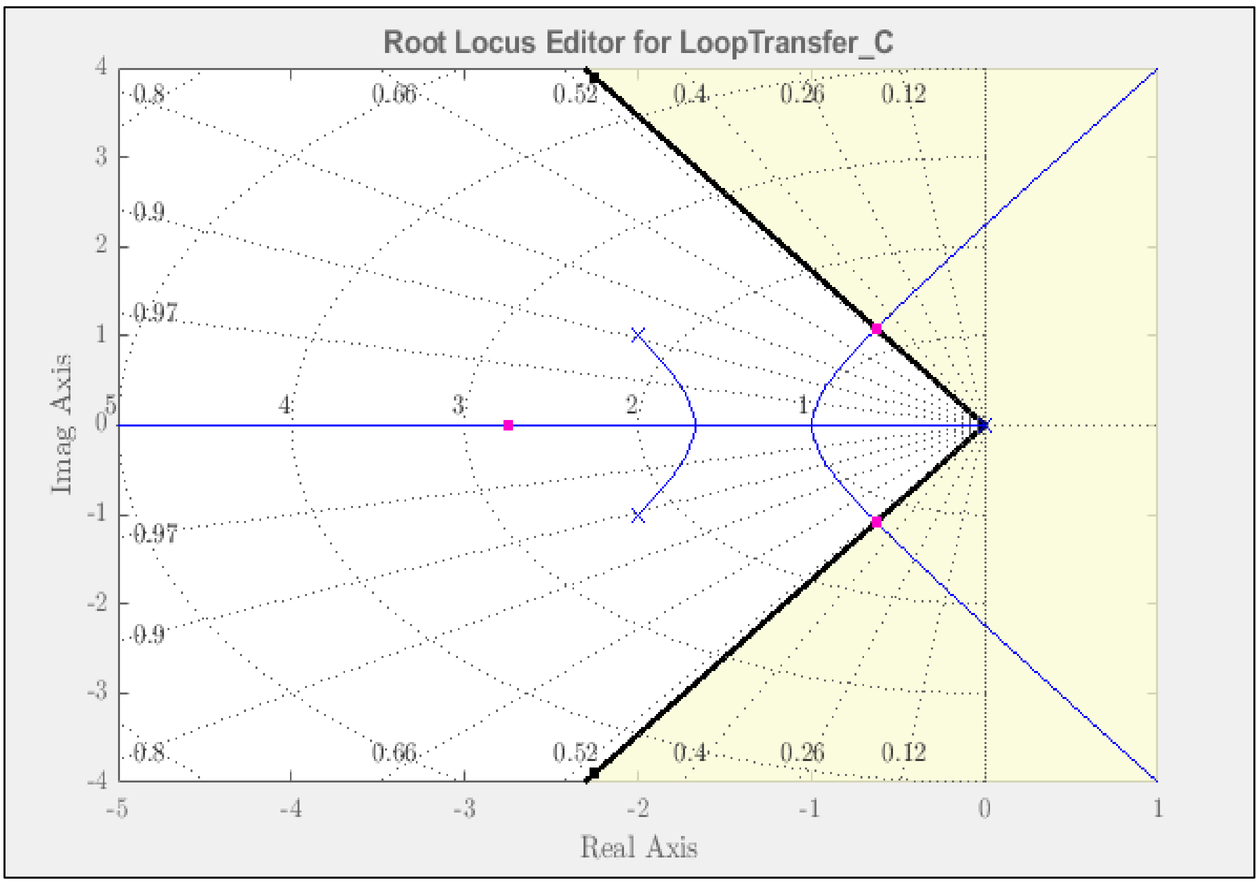
(b) Find value of K

if the damping ratio $\zeta = 0.5$ then $\Rightarrow \cos\theta = 0.5 \Rightarrow \theta = \pm 60°$

now from this we want to find the location of the pole for this ζ on the RL. To do this we will use MATLAB's Control System Designer

→ call >> controlSystemDesigner(tf(num, den));

→ drag the complex poles to where it satisfies $\zeta = \cos\theta = 0.5$



Tomoki Koike AAE 364

→ for this the pole is $p = -0.625 \pm 1.08j$
→ then find the gains by

calling rlocfind( ) or looking at the gains corresponding to the design you have in your control system designer

now we obtain $K = 4.2852$

Analytically we can obtain K by

→ define a pole $p = -\sigma + \omega j$
→ this pole must satisfy the two

(1) $1 + KL(p) = 0$

(2) $\omega = (\tan 60°) \cdot \sigma = \sqrt{3}\sigma$

→ solving this we get

$$1 + K\frac{1}{(-\sigma+\omega j)\left((-\sigma+\omega j)^2 + 4(-\sigma+\omega j)+5\right)} = 0$$

$$K = -\omega^3 j + 3\omega^2\sigma - 4\omega^2 + 3\omega\sigma^2 j - 8\omega\sigma j + 5\omega j - \sigma^3 + 4\sigma^2 - 5\sigma$$

$$\Rightarrow K = 8\sigma^3 - 8\sigma^2 - 5\sigma + 5\sqrt{3}\sigma j - 8\sqrt{3}\sigma^2 j$$

Re: $K = -8\sigma^3 + 8\sigma^2 + 5\sigma$

Im: $0 = 8\sqrt{3}\sigma^2 - 5\sqrt{3}\sigma$

$$\therefore \sigma = \begin{bmatrix} 0 \\ 0.6250 \end{bmatrix} \quad K = \begin{bmatrix} 0 \\ 4.2969 \end{bmatrix}$$

$$\omega = \begin{bmatrix} 0 \\ 1.0825 \end{bmatrix}$$

$$\boxed{\begin{aligned} K_{comp} &= 4.2852 \\ K_{TH} &= 4.2969 \end{aligned}} \rightarrow$$

OLTF (open loop)

$$\frac{4.2969}{s(s^2+4s+5)}$$

The CLTF becomes

$$\frac{G(s)}{1+G(s)} = \frac{4.2969}{s^3+4s^2+5s+4.2969}$$

because this is a unit-step response

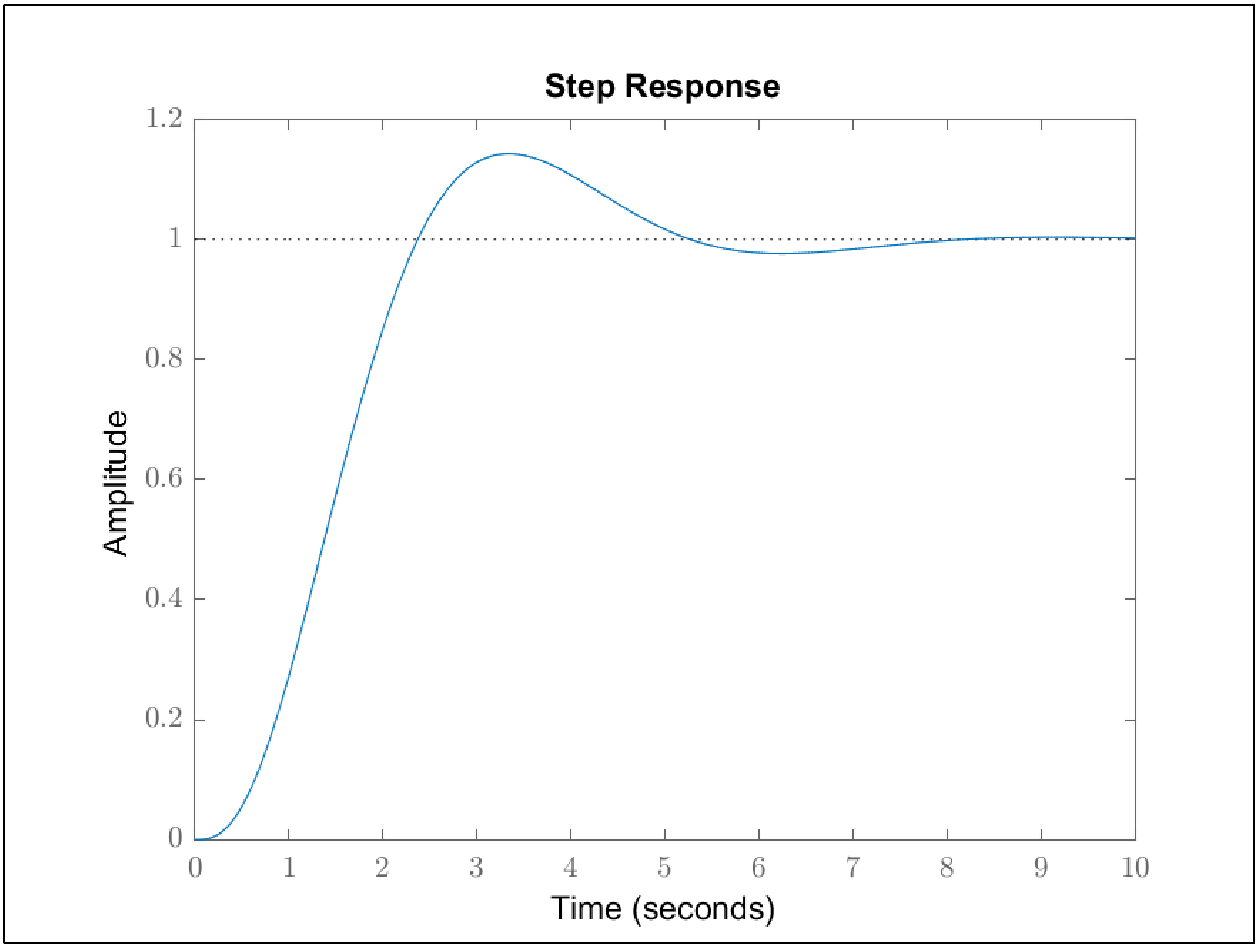
input $R(s) = \frac{1}{s}$

$$\Longrightarrow \quad Y(s) = \frac{4.2969}{s(s^3+4s^2+5s+4.2969)}$$

$$\Downarrow$$

$$\mathcal{L}^{-1}[Y(s)] = 1 - \frac{66}{91}e^{-\frac{5}{8}t}\left[\cos\left(\frac{5\sqrt{3}}{8}t\right) + \frac{8\sqrt{3}}{9}\sin\left(\frac{5\sqrt{3}}{8}t\right)\right] - \frac{25}{91}e^{-\frac{11}{4}t}$$

$$\Downarrow$$


Step Response
Amplitude
Time (seconds)
0
0.2
0.4
0.6
0.8
1
1.2
0
1
2
3
4
5
6
7
8
9
10

B–6–15. Determine the values of K, T_1, and T_2 of the system shown in Figure 6–105 so that the dominant closed-loop poles have the damping ratio $\zeta = 0.5$ and the undamped natural frequency $\omega_n = 3$ rad/sec.

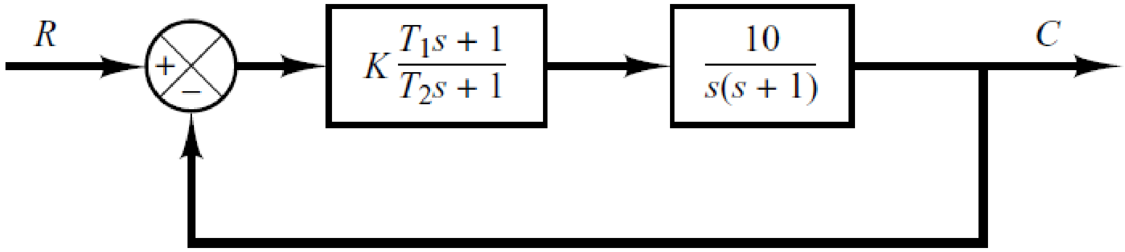


Figure 6–105 Control system.

$$OLTF := G_c G = \frac{10K(T_1 s+1)}{s(s+1)(T_2 s+1)}$$

since this is a negative feedback loop

$$CE := 1 + G_c G = 1 + \frac{10K(T_1 s+1)}{s(s+1)(T_2 s+1)} = 0$$

$$\Rightarrow \quad s(s+1)(T_2 s+1) + 10K(T_1 s+1) = 0$$

$$(s^2+s)(T_2 s+1) + 10KT_1 s + 10K = 0$$

$$T_2 s^3 + (T_2+1)s^2 + (10KT_1+1)s + 10K = 0$$

if $\zeta = 0.5 = \cos\theta$ and $\omega_n = 3$ rad/s
the complex pole locations are

$$p = -\omega_n \zeta \pm j\omega_n\sqrt{1-\zeta^2}$$

$$p = -1.50 \pm 2.5981j \qquad CLTF \Rightarrow \frac{10}{s^2+s+10}$$

this the desired pole. from this find the angle

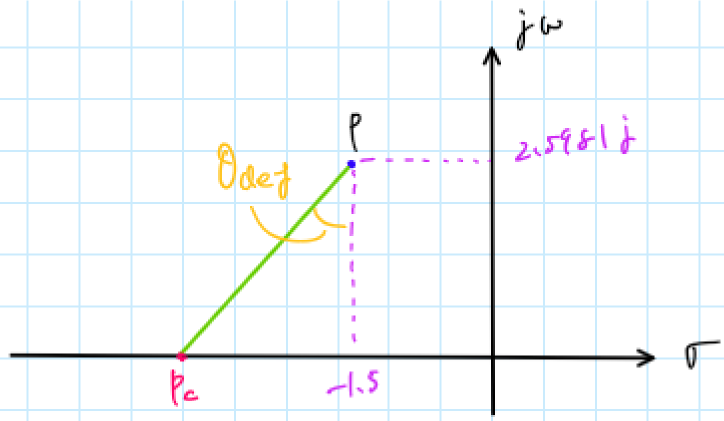
$$\theta_{def} = 180 - \arg(p) - \arg(p-(-1))$$

$$= -40.8934$$

choose the zero of the compensator to be

$$z_c = -1.5$$

now if $G_c = \dfrac{s-z_c}{s-p_c}$ (* p_c and z_c are real)



$$p_c = 1.5 + 2.5981 \times \tan(\theta_{deg}) = -3.750$$

thus,

$$G_c = k_c \frac{s+1.5}{s+3.75}$$

the value k_c can be determined by magnitude condition

$$k_c \left| \frac{s+1.5}{s+3.75} \cdot \frac{10}{s(s+1)} \right|_{s=p} = 1$$

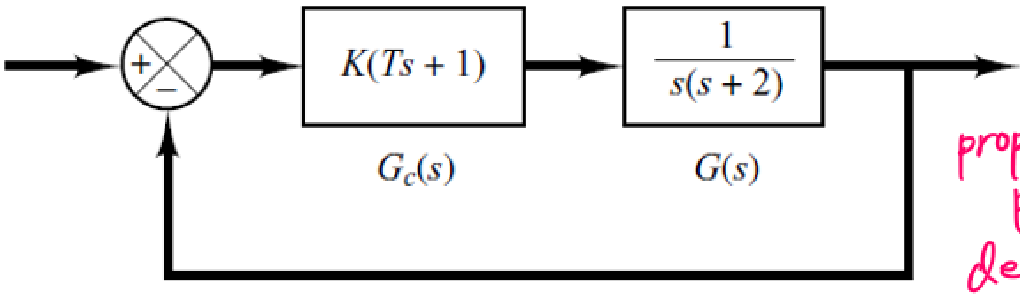
$$k_c = 1.0500$$

Thus,

$$k_c \frac{s+1.5}{s+3.75} = K \frac{T_1 s + 1}{T_2 s + 1} = \frac{KT_1}{T_2} \frac{\left(s + \frac{1}{T_1}\right)}{\left(s + \frac{1}{T_2}\right)}$$

$$\begin{cases} \frac{1}{T_1} = 1.5 \\ \frac{1}{T_2} = 3.75 \\ \frac{KT_1}{T_2} = k_c \end{cases} \Rightarrow \begin{cases} T_1 = 0.6667 \\ T_2 = 0.2667 \\ K = 0.4200 \end{cases}$$

B–6–16. Consider the control system shown in Figure 6–106. Determine the gain K and time constant T of the controller $G_c(s)$ such that the closed-loop poles are located at $s = -2 \pm j2$.



proportional
plus
derivative

Figure 6–106 Control system.

Similar approach to B-6-15

if $s = -2 \pm j2$ is the desired pole
the angle deficiency θ_{def} is calculated by the following method

say $p = -2 + j2$

$$\theta_{def} = 180 - \arg(p) - \arg(p - (-2))$$
$$= -45 \text{ deg}$$

this means that

$s = -\frac{1}{T}$ must contribute to 45 deg

which means if the zero for the compensator is z_c

$$45 \text{ deg} = \arg(p - z_c)$$

$$\Rightarrow z_c = -4 \quad \Rightarrow -\frac{1}{T} = -4$$

Hence, we have

$$K(Ts+1) = KT\left(s + \frac{1}{T}\right)$$
$$= KT(s+4)$$

T is

$$T = \frac{1}{4} = 0.25$$

now from magnitude condition K is

$$K\left|\frac{1}{4}(s+4)\,\frac{1}{s(s+2)}\right|_{s=p}=1$$

$$\Rightarrow \quad \frac{K}{8} = 1$$

$$K = 8$$

$$\boxed{K=8\ ,\quad T=0.25}$$

B–6–18. Consider the system shown in Figure 6–108. Design a compensator such that the dominant closed-loop poles are located at $s = -1 \pm j1$.

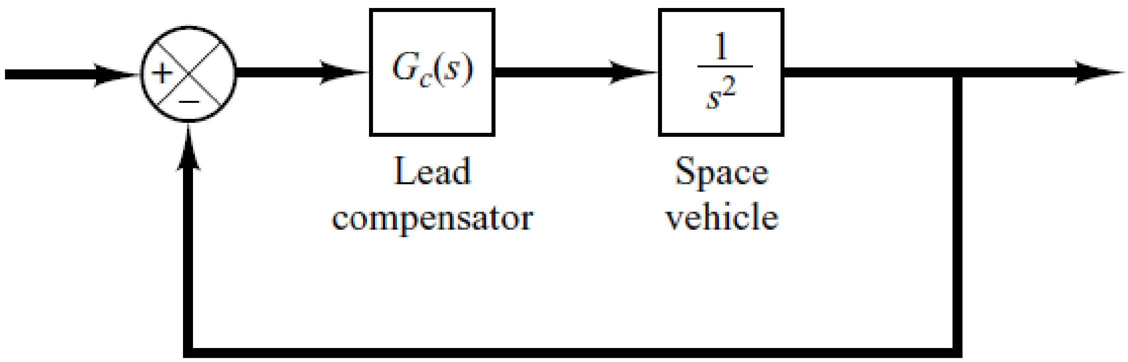


Figure 6–108 Control system.

<i> Attempt #1

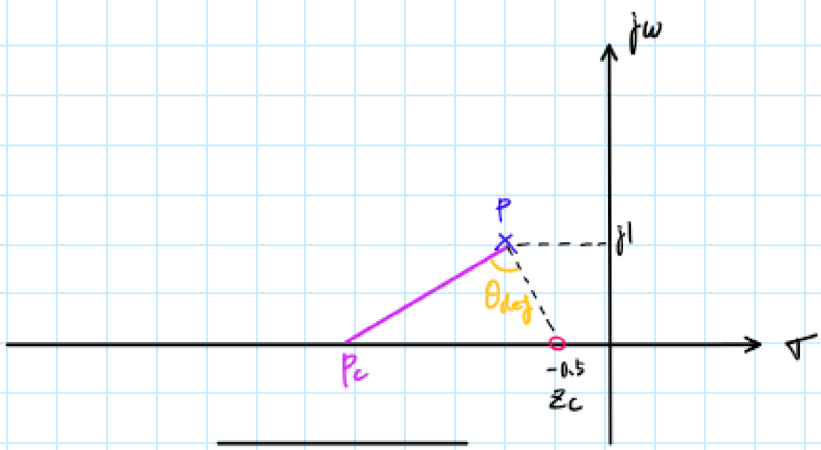
Assume the compensator $G_c(s)$ to be in the form of

$$G_c(s) = K_c\left(\frac{s+\frac{1}{T}}{s+\frac{1}{aT}}\right) \qquad (0<a<1)$$

we are given that the desired pole is @ $s=p=-1\pm j$
now, the angle deficiency will be

$$\theta_{def} = 180° - \arg(p-0) - \arg(p-0)$$
$$= -90°$$

arbitrarily choose the zero of $G_c(s)$, $z_c = -0.5$. Then the pole, p_c can be determined



$$\Longrightarrow p_c = -0.5 - \sqrt{\|p+0.5\|^2 + \|p-p_c\|^2}$$

$$\because z_c = 0.5$$

$$\Longrightarrow p_c = -3$$

therefore $G_c(s) = K_c\frac{s+0.5}{s+3}$
K_c can be determined from magnitude condition

$$K_c\left\|\frac{s+0.5}{s+3}\cdot\frac{1}{s^2}\right\|_{s=p} = 1$$

$$K_c \left\| \frac{-1+j+0.5}{-1+j+3} \frac{1}{(-1+j)^2} \right\| = 1$$

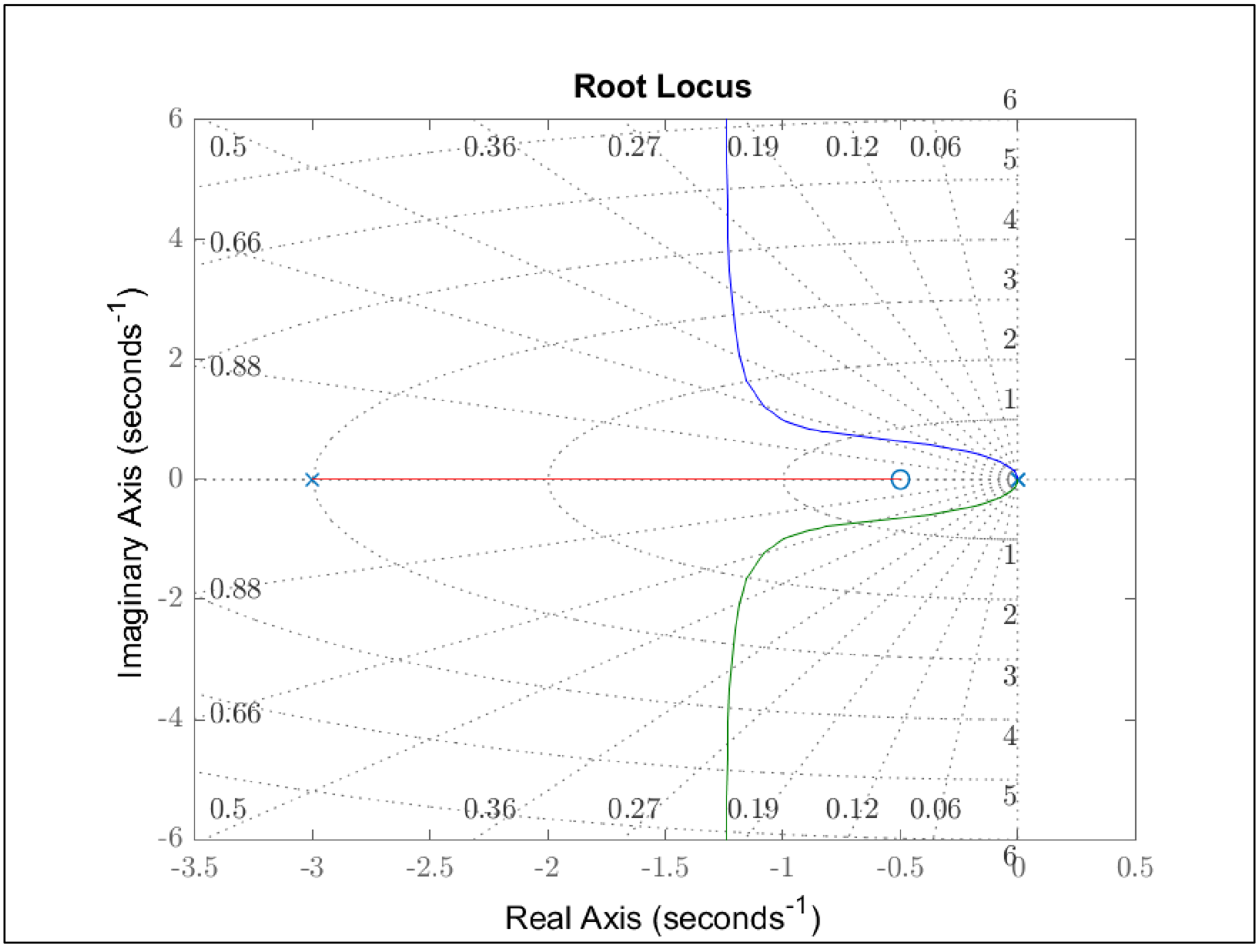
$$\therefore K_c = 4$$

thus, $G_c(s) = 4\dfrac{s+0.5}{s+3}$

and

$$G_c(s)G(s) = \frac{4(s+0.5)}{s^2(s+3)}$$

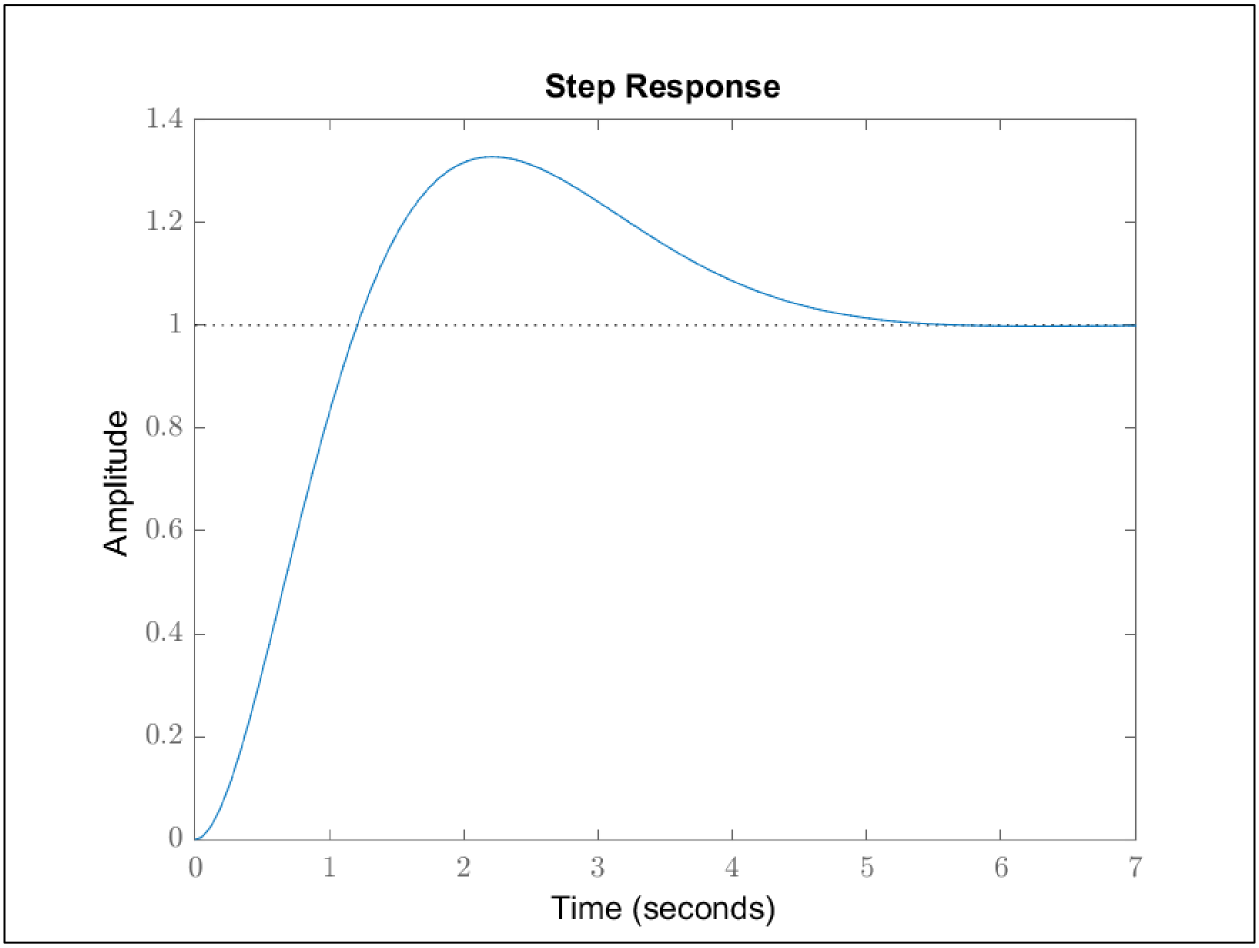
the RL for this becomes



and the step response becomes

$$\frac{C(s)}{R(s)} = \frac{G_c G}{1 + G_c G} = \frac{4s+2}{s^3+3s^2+4s+2}$$

$$\Rightarrow \quad C(s) = \frac{4s+2}{s(s^3+3s^2+4s+2)}$$



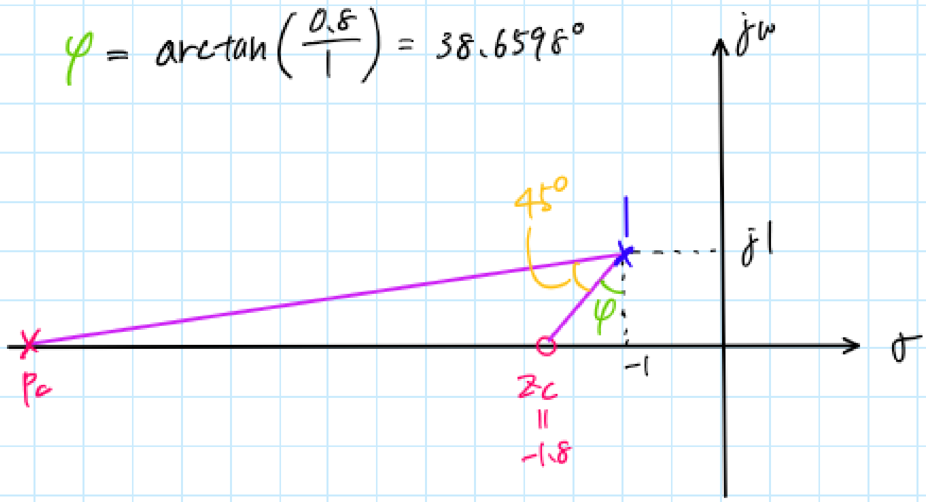
The overshoot is unacceptable so we go to the second attempt

<ii> Attempt #2

We will use two lead networks using half the necessary lead angle $\frac{90°}{2} = 45°$

next arbitrarily select $z_c = -1.8$

$$\varphi = \arctan\left(\frac{0.8}{1}\right) = 38.6598°$$



$$\Rightarrow p_c = -1 - \tan(45° + \varphi)$$

$$= -10$$

therefore the lead compensator (2 lead networks)

$$G_c(s) = K_c\left(\frac{s - z_c}{s - p_c}\right)^2 = K_c\left(\frac{s+1.8}{s+10}\right)^2$$

K_c can be determined from magnitude condition

$$K_c \left\| \left(\frac{s+1.8}{s+10}\right)^2 \cdot \frac{1}{s^2} \right\| = 1$$

$$K_c \left\| \left(\frac{-1+j+1.8}{-1+j+10}\right)^2 \frac{1}{(-1+j)^2} \right\| = 1$$

$$\therefore \; K_c = 100$$

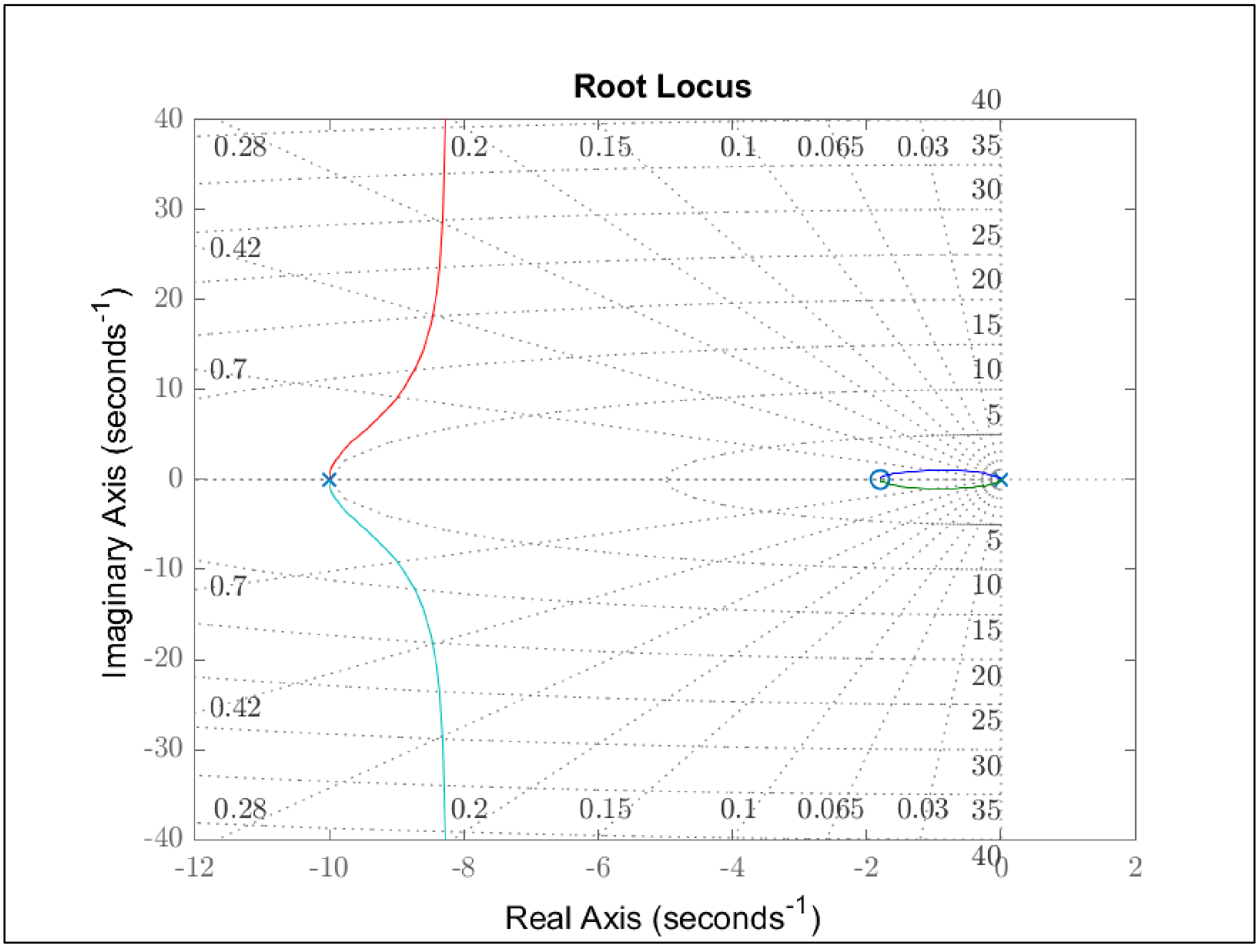
thus,

$$G_c(s) = 100\left(\frac{s+1.8}{s+10}\right)^2$$

then

$$G_c(s)G(s) = \frac{100}{s^2}\left(\frac{s+1.8}{s+10}\right)^2$$

the RL becomes



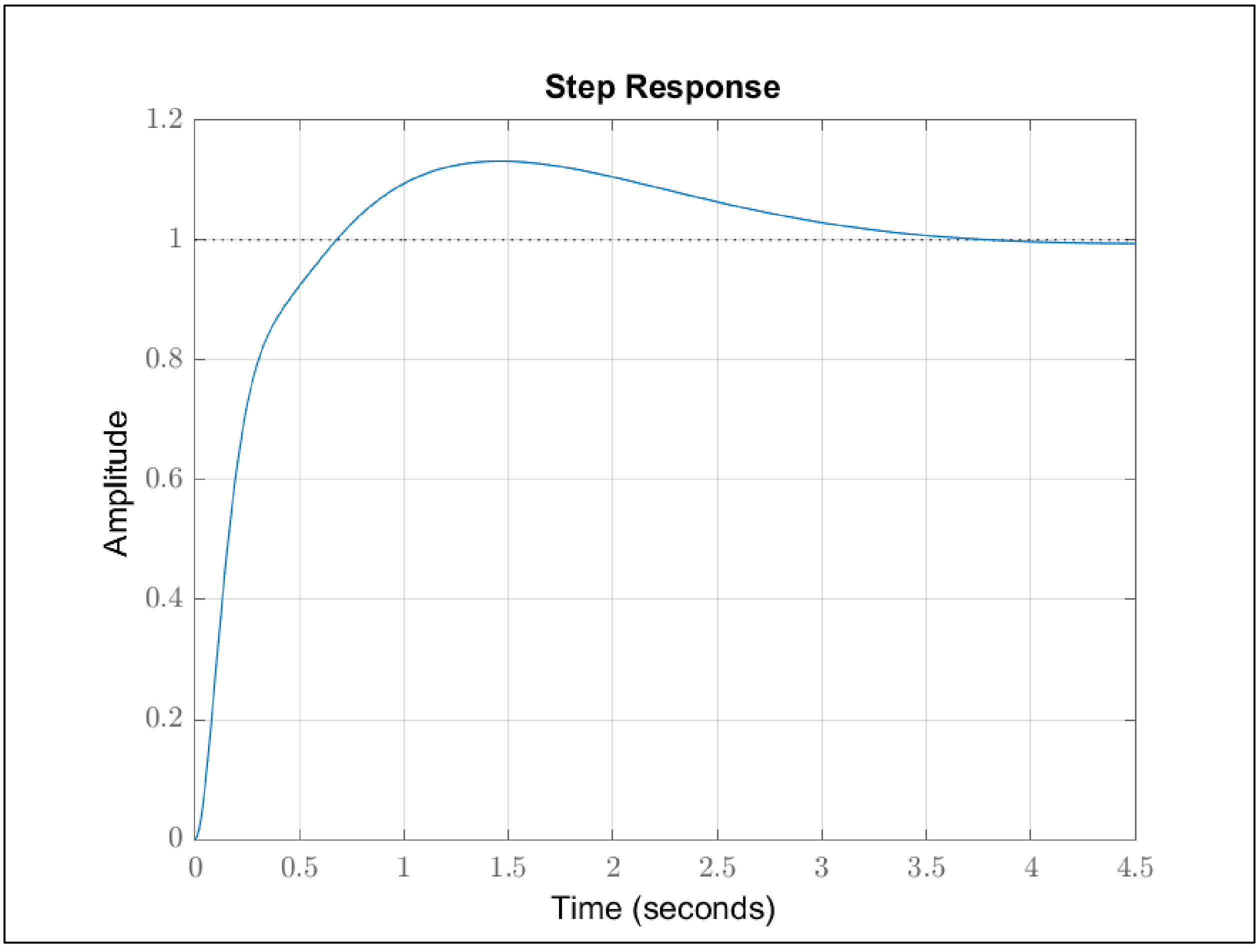
the CLTF becomes

$$G_1 = \frac{G_cG}{1+G_cG} = \frac{100s^2+360s+324}{s^4+20s^3+200s^2+360s+324}$$

then

$$C(s) = G_1(s)R(s) = \frac{100s^2+360s+324}{s(s^4+20s^3+200s^2+360s+324)}$$

Hence the step response becomes



The overshoot has been improved significantly and this is acceptable

$$\therefore \quad \boxed{G_c = 100\left(\frac{s+1.8}{s+10}\right)^2}$$

Problem 2

The following figure shows the coordinate axes and forces acting on the aircraft in the longitudinal plane of motion. Assuming that the aircraft is cruising at constant velocity and altitude.

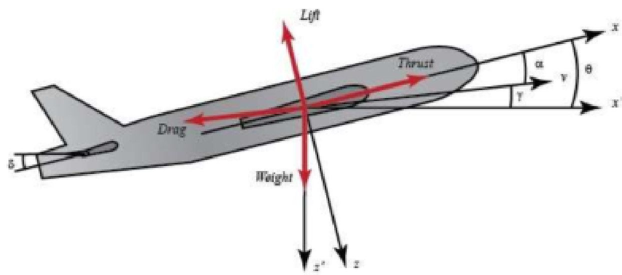


Figure 1: Forces acting on an aircraft in the Longitudinal plane.

Consider the unity-feedback system in Figure 2 with the plant $G(s)$ representing the aircraft shown in Figure 1. Sketch the root locus of the unity-feedback system, with $K(s) = K$, as K varies from 0 to ∞ (as accurately as you can) with the following $G(s)$:

1. $G(s)$ representing the aircraft pitch angle response output to the elevator deflection input:

$$G(s) = \frac{\Theta(s)}{\Delta(s)} = \frac{1.1057s + 0.1900}{s^3 + 0.7385s^2 + 0.8008s}$$

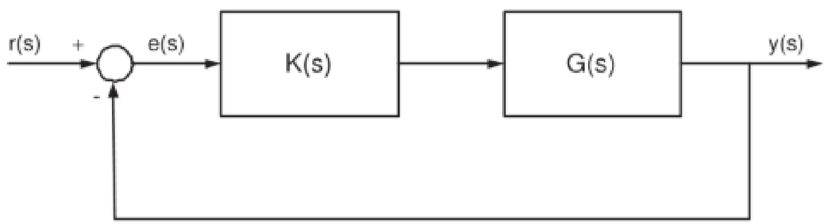


Figure 2: Unity-Feedback System with controller $K(s)$ and plant $G(s)$.

2. $G(s)$ representing the aircraft altitude response output to the elevator deflection input:

$$G(s) = \frac{H(s)}{\Delta(s)} = \frac{1.1057s - 0.1900}{s^5 + 17.95s^4 + 123.3s^3 + 366.3s^2 + 112.2s}$$

(1) when $G(s) = \dfrac{\Theta(s)}{\Delta(s)} = \dfrac{1.1057s + 0.1900}{s^3 + 0.7385s^2 + 0.8008s}$

since this is a negative feed back system

$$CE := 1 + KG(s) = 1 + K\frac{1.1057s + 0.1900}{s^3 + 0.7385s^2 + 0.8008s} = 0$$

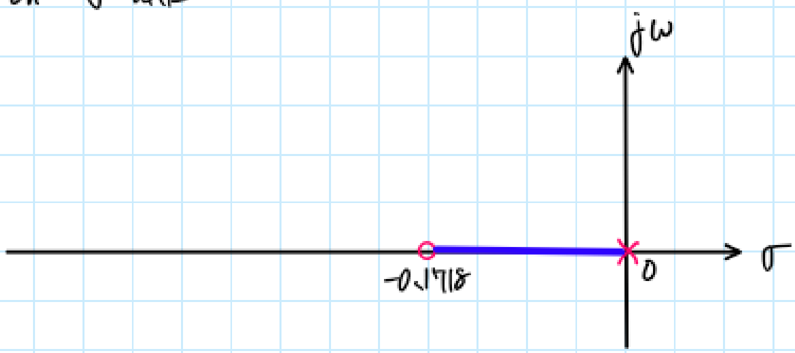
<i> Poles and Zeros

Poles : $s^3 + 0.7385s^2 + 0.8008s = 0 \Rightarrow s = 0, -0.3693 \pm 0.8151j \Rightarrow n = 3$

Zeros : $1.1057s + 0.1900 = 0 \Rightarrow s = -0.1718 \Rightarrow m = 1$

<ii> Symmetry TRUE

<iii> RL on σ-axis



<iv> Asymptotes

$$\theta_a = \frac{180° + 360° \ell}{n - m} = 90° + 180° \ell \qquad (\ell = 0, 1)$$

$$\theta_a = 90°,\ 270°$$

and

$$\sigma_a = \frac{\Sigma p_i - \Sigma z_j}{n - m}$$

$$= \frac{0 + (-0.3693 + 0.8151j) + (-0.3693 - 0.8151j) - (-0.1718)}{2}$$

$$= -0.2833$$

<v> Break-in/away points

$$\frac{d}{ds}\left(-\frac{1}{G(s)}\right) = -\frac{d}{ds}\left(\frac{s^3 + 0.7385s^2 + 0.8008s}{1.1057s + 0.1900}\right) = 0$$

$$\Rightarrow \frac{-2.2114s^3 - 1.3866s^2 - 0.2806s - 0.1522}{1.2226s^2 + 0.4202s + 0.0361} = 0$$

the real answer is

$\times \hat{s}_1 = -0.6052 \rightarrow$ However this not in the domain of RL on σ-axis

⟨vi⟩ Departure/Arrival angles

take a hypothetical point in the proximity of a complex pole

$$s^d = -0.3693 + 0.8151j$$

$$\angle G(s^d) = -180^\circ = \arg\left(-0.3693+0.8151j - (-0.1718)\right) - \theta_d - \arg(-0.3693+0.8151j - 0) - \arg(-0.3693+0.8151j - (-0.3693-0.8151j))$$

$$\Longrightarrow \theta_d = 79.2439^\circ$$

⟨vii⟩ Intersection with $j\omega$-axis

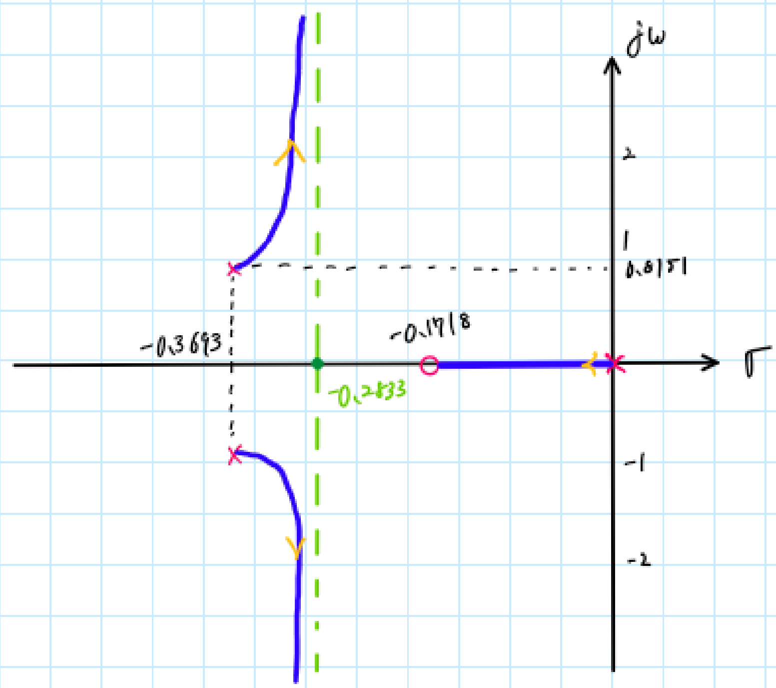
$$1 + \hat{k}G(j\hat{\omega}) = 1 + \hat{k}\frac{1.1057(j\hat{\omega}) + 0.1900}{-j\hat{\omega}^3 - 0.7385\hat{\omega}^2 + 0.8008j\hat{\omega}} = 0$$

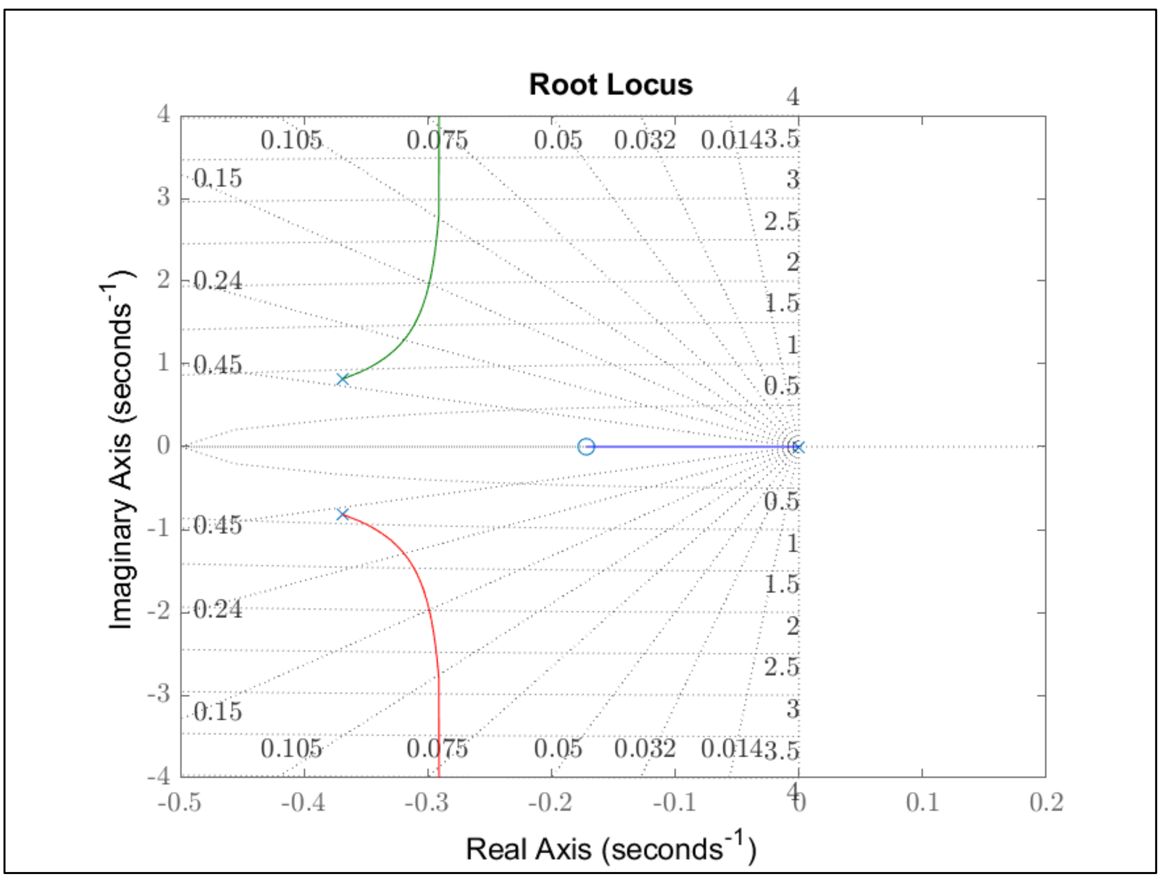
solving this we get

$$\begin{cases} \sigma: & \dfrac{19\hat{k}}{100} = \dfrac{1477}{2000}\hat{\omega}^2 \\ j\omega: & \dfrac{11057}{10000}\hat{k}\hat{\omega} = \hat{\omega}^3 - \dfrac{1001}{1250}\hat{\omega} \end{cases}$$

$$\hat{k} = \begin{bmatrix} -0.9439 \\ -0.9439 \\ 0 \end{bmatrix} \quad \hat{\omega} = \begin{bmatrix} -0.4928j \\ 0.4928j \\ 0 \end{bmatrix}$$ NONE

therefore RL becomes





(2) when

$$G(s) = \frac{H(s)}{\Delta(s)} = \frac{1.1057s - 0.1900}{s^5 + 17.95s^4 + 123.3s^3 + 366.3s^2 + 112.2s}$$

since this is a negative feedback system

$$CE := 1 + kG(s) = 1 + k\frac{1.1057s - 0.1900}{s^5 + 17.95s^4 + 123.3s^3 + 366.3s^2 + 112.2s}$$

<i> Poles and Zeros

Poles : $s^5 + 17.95s^4 + 123.3s^3 + 366.3s^2 + 112.2s = 0$

$\Rightarrow$ $P_1 = 0$ $P_4 = -4.7533 + 4.2012j$

$P_2 = -8.0992$ $P_5 = -4.7533 - 4.2012j$

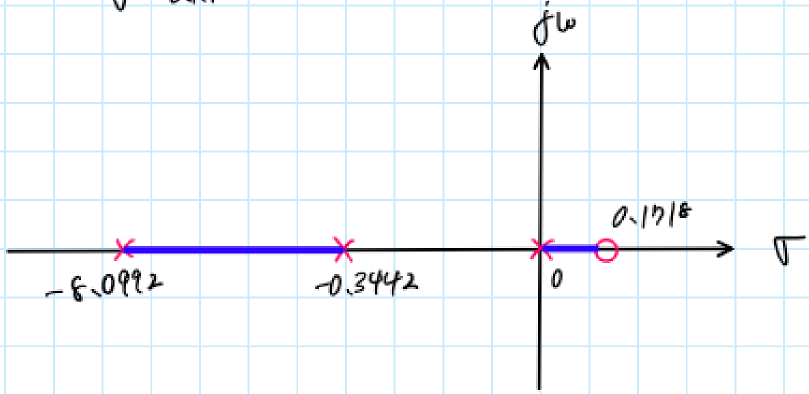
$P_3 = -0.3442$ $\Rightarrow n = 5$

Zeros : $1.1057s - 0.1900 = 0$

$\Rightarrow z_1 = 0.1718 \Rightarrow m = 1$

<ii> Symmetry TRUE

<iii> RL on σ-axis



<iv> Asymptotes

$$\theta_a = \frac{180° + 360°\ell}{n-m} = 45° + 90°\ell \quad (\ell = 0, 1, 2, 3)$$

$$\theta_a = 45°, 135°, 225°, 315°$$

and

$$\sigma_a = \frac{\Sigma P_i - \Sigma z_i}{n-m}$$

$$= \frac{1}{4}\left[0 + (-8.0992) + (-0.3442) + (-4.7533 + 4.2012j) + (-4.7533 - 4.2012j) - (0.1718)\right]$$

$$= -4.5305$$

<V> Break-in/away point

$$\frac{d}{ds}\left(-\frac{1}{G(s)}\right) = -\frac{d}{ds}\left(\frac{s^5 + 17.95s^4 + 123.3s^3 + 366.3s^2 + 112.2s}{1.1057s - 0.1900}\right) = 0$$

$$\Rightarrow \frac{-4.4228s^5 - 58.5919s^4 - 259.0236s^3 - 334.7319s^2 + 139.1940s + 21.3180}{1.2226s^2 - 0.4202s + 0.0361} = 0$$

$$\Rightarrow \hat{s}_1 = -3.2759 \text{ ok!}$$
$$\hat{s}_2 = 0.4189 \times$$
$$\hat{s}_3 = -0.1211 \times$$

<VI> Departure/arrival angles

take a hypothetical point in the proximity of p_4

$$s^d = -4.7533 + 4.2012j$$

$$\angle G(s^d) = -180° = \arg(s^d - z_1) - \arg(s^d - p_1) - \arg(s^d - p_2) - \arg(s^d - p_3) - \theta_d - \arg(s^d - p_5)$$

$$\Rightarrow \theta_d = -96.8418°$$

<VII> Intersection with $j\omega$-axis

$$1 + \hat{k}G(j\hat{\omega})$$
$$= 1 + \hat{k}\frac{1.1057(j\hat{\omega}) - 0.1900}{(j\hat{\omega})^5 + 17.95(j\hat{\omega})^4 + 123.3(j\hat{\omega})^3 + 366.3(j\hat{\omega})^2 + 112.2j\hat{\omega}} = 0$$

$$\Rightarrow \hat{k}(1.1057j\hat{\omega} - 0.1900)$$

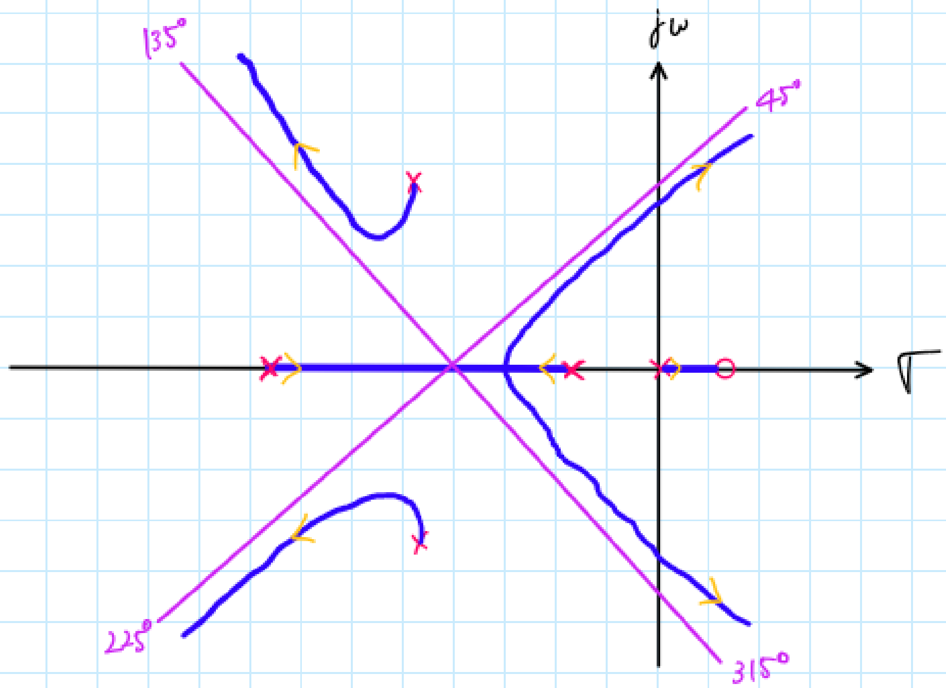
$$= -\left(j\hat{\omega}^5 + 17.95\hat{\omega}^4 - 123.3 j\hat{\omega}^3 - 366.3\hat{\omega}^2 + 112.2 j\hat{\omega}\right)$$

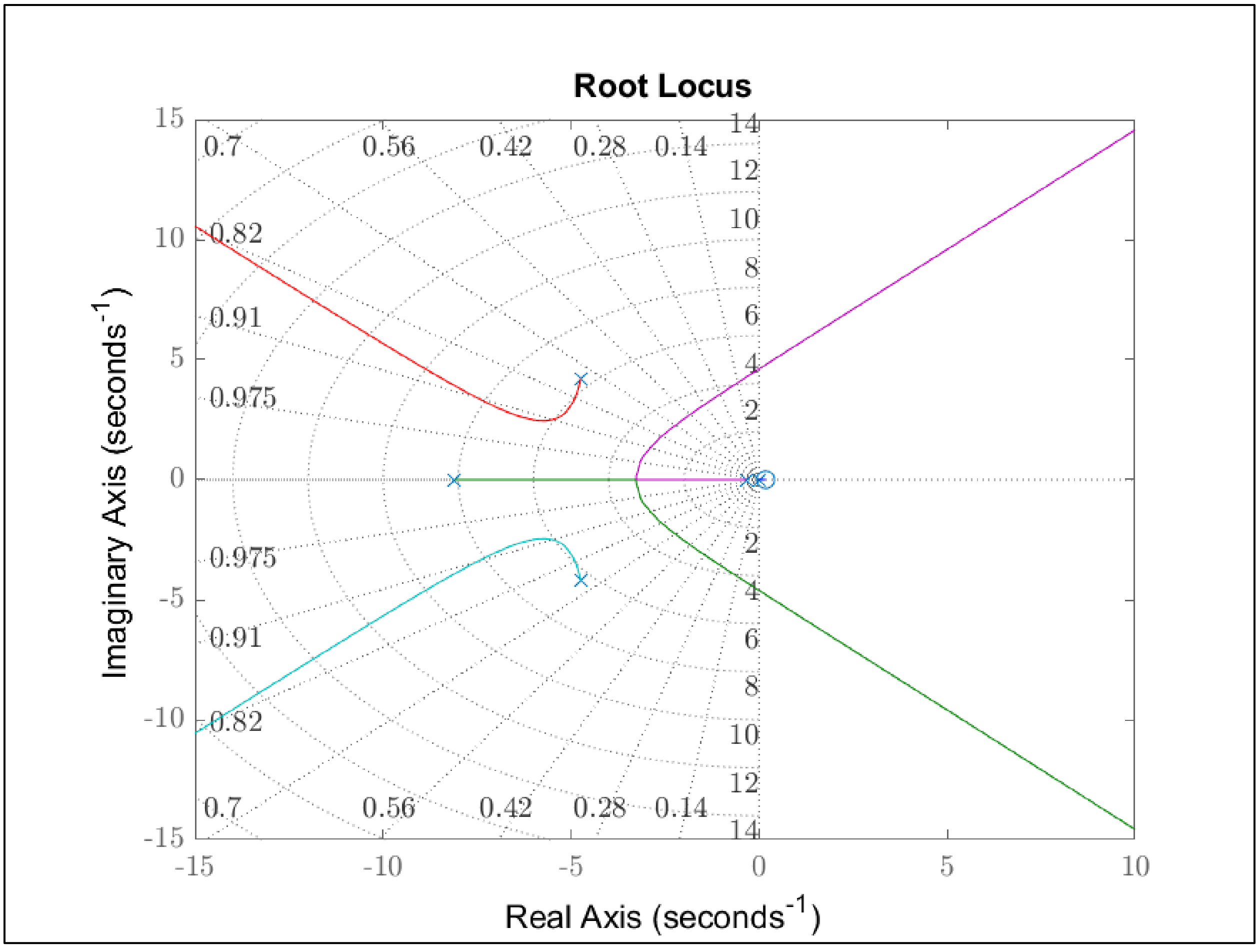
Solving this

$$\begin{cases} \sigma: & -\dfrac{19\hat{k}}{100} = \dfrac{3663}{100}\hat{\omega}^2 - \dfrac{359}{20}\hat{\omega}^4 \\ j\omega: & \dfrac{11057}{10000}\hat{k}\hat{\omega} = -\hat{\omega}^5 + \dfrac{1233}{10}\hat{\omega}^3 - \dfrac{561}{5}\hat{\omega} \end{cases}$$

$$\hat{k} = \begin{bmatrix} \boxed{\begin{matrix} 1.8658 \\ 1.8658 \end{matrix}} \\ -0.0959 \\ -0.0959 \\ 0 \end{bmatrix} \quad \hat{\omega} = \begin{bmatrix} \boxed{\begin{matrix} -4.6187 \\ 4.6187 \end{matrix}} \\ -0.2233 \\ 0.2233 \\ 0 \end{bmatrix} \quad (*\hat{k} > 0)$$

Hence, from <i> ~ <vii> RL becomes





Root Locus
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})
0.7
0.56
0.42
0.28
0.14
0.82
0.91
0.975
15
10
5
0
-5
-10
-15
-15
-10
-5
0
5
10
14
12
10
8
6
4
2

Appendix

AAE364 HW8 MATLAB CODE

```
clear all; close all; clc;
fdir = 'C:\Users\Tomo\Desktop\studies\2020-
Spring\AAE364\matlab\matlab_output\hw8';
set(groot, 'defaulttextinterpreter',"latex");
set(groot, 'defaultAxesTickLabelInterpreter',"latex");
set(groot, 'defaultLegendInterpreter',"latex");
```

B-6-12

```
% (a)
num = [1 -1];
den = conv([1 2],[1 4]);
[poles,zrs,angs,sigma,bi_pt,T_P,T_Z,k,w,fig1] =
rootLocus_stepBystep_negFeedback(num,den)
saveas(fig1, fullfile(fdir,'RL_B-6-12_a.png'))
```

```
% (b)
num = -[1 -1];
den = conv([1 2],[1 4]);
[poles,zrs,angs,sigma,bi_pt,T_P,T_Z,k,w,fig1] =
rootLocus_stepBystep_posFeedback(num,den)
saveas(fig1, fullfile(fdir,'RL_B-6-12_b.png'))
```

B-6-14

```
num = [0 1];
den = conv([1 0],[1 4 5]);
[poles,zrs,angs,sigma,bi_pt,T_P,T_Z,k,w,fig1] =
rootLocus_stepBystep_negFeedback(num,den)
saveas(fig1, fullfile(fdir,'RL_B-6-14.png'))
```

```
% Step Response


% Find gain, K when zeta = 0.5
% Analytical
syms K x w s
assume(K,'real');
assume(x,'real');
```

```
assume(w,'real');
p = -x + w*1j
RHS = (s*(s^2 + 4*s + 5))
RHS = subs(RHS,s,p)
RHS = expand(RHS)
RHS = subs(RHS,w,tand(60)*x)
eqn1 = K == -real(RHS)
eqn2 = 0 == -imag(RHS)
res = solve([eqn1 eqn2],[x K]);
sigma = double(res.x)
w = tand(60)*sigma
K_th = double(res.K)
K_th = nonzeros(K_th)
```

```
% Verify (computational)
G = tf(num,den);
% controlSystemDesigner(G)  % from this we find that the pole we need is
                            % the following variable opz to satisfy zeta = 0.5
opz = [-0.625+1.08i, -0.625-1.08i];
[K_comp,pls] = rlocfind(G,opz)
```

```
% Plot step response
[numCL, denCL] = cloop((K_th)*num, den)
L_inv_expr = return_inverseLaplace_expression(numCL,conv(denCL,[1 0]))
```

```
fig2 = figure("Renderer","painters");
step(numCL,denCL);
grid on; grid minor; box on;
saveas(fig2,fullfile(fdir,"STEP_RES_B-6-14.png"));
```

B-6-15

```
wn = 3;  % natural frequency [rad/s]
zeta = 0.5;  % damping ratio
p = -wn*zeta + 1j*wn*sqrt(1-zeta^2)


% deficiency angle
theta_def = pi + angle(10/p/(p + 1))
theta_deg_deg = rad2deg(theta_def)


% find the pole and zero for compensator
pc = real(p) - tan(2*pi - theta_def)*imag(p)
```

```
zc = real(p)


% find Kc
syms Kc
eqn = Kc*abs((p+1.5)/(p+3.75) * 10/(p*(p+1))) == 1;
K_c = double(solve(eqn,Kc))


% T1 T2 and K
T1 = 1/-zc
T2 = 1/-pc
K = K_c*T2/T1
```

B-6-16

```
% angle deficiency
p = -2+2j;
theta_def = deg2rad(180) - angle(p) - angle(p + 2)
theta_def_deg = rad2deg(theta_def)


% find zero
syms zc
assume(zc,'real')
eqn = -theta_def == angle(p - zc)
Z_c = double(solve(eqn,zc))


syms K
eqn = K*0.25*abs((p+4)/p/(p+2)) == 1
K_p = double(solve(eqn,K))
```

B-6-18

```
% First attempt
% angle deficiency
p = -1+1j;
theta_def = pi - angle(p - 0) - angle(p - 0)
theta_def_deg = rad2deg(theta_def)


% find pc
syms pc
eqn = pc == -0.5 - sqrt((abs(p + 0.5))^2 + (abs(p - pc))^2);
P_c = double(solve(eqn,pc))
```

```
% find Kc
syms Kc
eqn = Kc*abs((p + 0.5)/(p + 3)/p^2) == 1;
Kc = double(solve(eqn,Kc))


% Plot RL
num = Kc*[1 0.5];
den = conv([1 0 0],[1 3]);
fig1 = figure("Renderer","painters");
rlocus(tf(num,den))
sgrid
saveas(fig1, fullfile(fdir,'RL_B-6-18.png'))
```

```
% Step response
[numCL, denCL] = cloop(num,den)
fig2 = figure("Renderer","painters");
step(numCL,denCL);
grid on; grid minor; box on;
saveas(fig2,fullfile(fdir,"STEP_RES_B-6-18.png"));
```

```
% Second attempt
zc = -1.8;
phi = atan(0.8)
phi_deg = rad2deg(phi)
pc = -1 - tan(-theta_def/2 + phi)


% find Kc
syms Kc
eqn = Kc*abs(((p - zc)/(p - pc))^2/p^2) == 1;
Kc = double(solve(eqn,Kc))
```

```
% Plot RL
fig1 = figure("Renderer","painters");
num = Kc*conv([1 -zc],[1 -zc])
den = conv([1 0 0],[1 -pc]);
den = conv(den,[1 -pc])
rlocus(tf(num,den));
sgrid
saveas(fig1, fullfile(fdir,"RL_B-6-18_2.png"));
```

```
% Step response
[numCL, denCL] = cloop(num,den)
```

```
fig2 = figure("Renderer","painters");
step(numCL,denCL);
grid on; grid minor; box on;
saveas(fig2,fullfile(fdir,"STEP_RES_B-6-18_2.png"));
```

P2

```
% (1)
num = [1.1057 0.1900];
den = [1 0.7385 0.8008 0];
[poles,zrs,angs,sigma,bi_pt,T_P,T_Z,k,w,fig1] =
rootLocus_stepBystep_negFeedback(num,den)
saveas(fig1, fullfile(fdir,'RL_P2_1.png'))
```

```
% (2)
num = [1.1057 -0.1900];
den = [1 17.95 123.3 366.3 112.2 0];
[poles,zrs,angs,sigma,bi_pt,T_P,T_Z,k,w,fig1] =
rootLocus_stepBystep_negFeedback(num,den)
saveas(fig1, fullfile(fdir,'RL_P2_2.png'))
```

```
function [poles,zrs,angs,sigma,bi_pt,T_P,T_Z,k,w,fig1] =
rootLocus_stepBystep_posFeedback(num,den)
    %{
       NAME:    ROOTLOCUS_STEPBYSTEP_NEGFEEDBACK
       AUTHOR:  TOMOKI KOIKE
       INPUTS:  (1) num:   THE NUMERATOR OF THE TRANSFER FUNCTION
                (2) den:   THE DENOMINATOR OF THE TRANSFER FUNCTION
       OUTPUTS: (1) poles: POLES OF THE TRANSFER FUNCTION
                (2) zrs:   ZEROS OF THE TRANSFER FUNCTION
                (3) angs:  ANGLES OF THE ASYMPTOTES
                (4) sigma: INTERSECTION OF THE ASYMPTOTES
                (5) bi_pt: BREAK-IN/AWAY POINT
                (6) T_P:   TABLE WITH EACH POLE AND THEIR
                           DEPARTURE OR ARRIVAL ANGLES
                (6) T_Z:   TABLE WITH EACH ZERO AND THEIR
                           DEPARTURE OR ARRIVAL ANGLES
                (7) k:     VALUE K_HAT FOR INTERSECTION WITH IM AXIS
                (8) w:     INTERSECTION POINT WITH THE IM AXIS
                (9) fig1:  THE FIGURE WITH THE ROOT LOCUS PLOT
       DESCRIPTION: CONDUCTS THE 7 STEP PROCEDURE OF THE ROOT LOCUS
       ANALYSIS AND DISPLAYS THE RESULTS AS WELL AS THE PLOT FOR A POSITIVE
       FEEDBACK LOOP
    %}
```

```
    % STEP1 - POLES & ZEROS
    poles = roots(den);
    zrs = roots(num);

    % STEP2 - SYMMETRY (*TAKEN FOR GRANTED)

    % STEP3 - ROOT LOCUS ON REAL AXIS (*OMMITTED)

    % STEP4 - ASYMPTOTES
    [angs,sigma] = RL_asymptote_posFeedback(zrs,poles);

    % STEP5 - BREAK-IN/AWAY POINTS
    bi_pt = break_in_away_pt(num,den);

    % STEP6 - ANGLE OF DEPARTURE
    [T_P, T_Z] = departure_arrival_angle_calc_posFeedback(zrs, poles);

    % STEP7 - INTERSECTION WITH IMAGINARY AXIS
    [k,w] = intersection_IM_axis(num,den);

    % DEFINE THE TRANSFER FUNCTION
    L = tf(num, den);
    % PLOTTING THE ROOT LOCUS
    fig1 = figure(1);
    rlocus(L)
    sgrid
end
```

```
function [poles,zrs,angs,sigma,bi_pt,T_P,T_Z,k,w,fig1] =
rootLocus_stepBystep_negFeedback(num,den)
    %{
       NAME:    ROOTLOCUS_STEPBYSTEP_NEGFEEDBACK
       AUTHOR:  TOMOKI KOIKE
       INPUTS:  (1) num:   THE NUMERATOR OF THE TRANSFER FUNCTION
                (2) den:   THE DENOMINATOR OF THE TRANSFER FUNCTION
       OUTPUTS: (1) poles: POLES OF THE TRANSFER FUNCTION
                (2) zrs:   ZEROS OF THE TRANSFER FUNCTION
                (3) angs:  ANGLES OF THE ASYMPTOTES
                (4) sigma: INTERSECTION OF THE ASYMPTOTES
                (5) bi_pt: BREAK-IN/AWAY POINT
                (6) T_P:   TABLE WITH EACH POLE AND THEIR
                           DEPARTURE OR ARRIVAL ANGLES
                (6) T_Z:   TABLE WITH EACH ZERO AND THEIR
                           DEPARTURE OR ARRIVAL ANGLES
                (7) k:     VALUE K_HAT FOR INTERSECTION WITH IM AXIS
                (8) w:     INTERSECTION POINT WITH THE IM AXIS
                (9) fig1:  THE FIGURE WITH THE ROOT LOCUS PLOT
```

```
        DESCRIPTION: CONDUCTS THE 7 STEP PROCEDURE OF THE ROOT LOCUS
        ANALYSIS AND DISPLAYS THE RESULTS AS WELL AS THE PLOT FOR A NEGATIVE
        FEEDBACK LOOP
    %}


    % STEP1 - POLES & ZEROS
    poles = roots(den);
    zrs = roots(num);

    % STEP2 - SYMMETRY (*TAKEN FOR GRANTED)

    % STEP3 - ROOT LOCUS ON REAL AXIS (*OMMITTED)

    % STEP4 - ASYMPTOTES
    [angs,sigma] = RL_asymptote(zrs,poles);

    % STEP5 - BREAK-IN/AWAY POINTS
    bi_pt = break_in_away_pt(num,den);

    % STEP6 - ANGLE OF DEPARTURE
    [T_P, T_Z] = departure_arrival_angle_calc(zrs, poles);

    % STEP7 - INTERSECTION WITH IMAGINARY AXIS
    [k,w] = intersection_IM_axis(num,den);

    % DEFINE THE TRANSFER FUNCTION
    L = tf(num, den);
    % PLOTTING THE ROOT LOCUS
    fig1 = figure(1);
    rlocus(L)
    sgrid
end
```

```
function [angs, sigma] = RL_asymptote_posFeedback(zrs, poles)
    n = length(poles);
    m = length(zrs);
    angs = zeros([1,n-m]);
    for i = 0:(n-m)-1
        angs(i+1) = (360*i)/(n - m);
    end
    sigma = -(sum(poles) - sum(zrs))/(n - m);
end
```

```
function [angs, sigma] = RL_asymptote(zrs, poles)
    n = length(poles);
    m = length(zrs);
    angs = zeros([1,n-m]);
    for i = 0:(n-m)-1
        angs(i+1) = (180 + 360*i)/(n - m);
    end
    sigma = (sum(poles) - sum(zrs))/(n - m);
end
```

```
function [K, W] = intersection_IM_axis(num, den)
    syms k w
    n = length(den);
    m = length(num);
    f1 = 0; f2 = 0; p1 = 0; p2 = 0;

    % RHS (denominator)
    % when the largest order of s is even
    if rem(n,2) == 1
        % powers to the even numbers (real)
        for i = 1:2:n
            if rem(n-i,4) == 0
                f1 = f1 + den(i)*w^(n-i);
            else
                f1 = f1 + den(i)*w^(n-i)*(-1);
            end
        end
        % powers to the odd numbers (imaginary)
        for i = 2:2:n-1
            if rem(n-i,4) == 1
                f2 = f2 + den(i)*w^(n-i);
            else
                f2 = f2 + den(i)*w^(n-i)*(-1);
            end
        end
    % when the largest order of s is odd
    elseif rem(n,2) == 0
        % powers to the even numbers (real)
        for i = 2:2:n
            if rem(n-i,4) == 0
                f1 = f1 + den(i)*w^(n-i);
            else
                f1 = f1 + den(i)*w^(n-i)*(-1);
            end
        end
        % powers to the odd numbers (imaginary)
        for i = 1:2:n-1
            if rem(n-i,4) == 1
```

```
                f2 = f2 + den(i)*w^(n-i);
            else
                f2 = f2 + den(i)*w^(n-i)*(-1);
            end
        end
    end

    % LHS
    % when the largest order of s is even
    if rem(m,2) == 1
        % powers to the even numbers (real)
        for i = 1:2:m
               if rem(m-i,4) == 0
                   p1 = p1 + num(i)*w^(m-i);
               else
                   p1 = p1 + num(i)*w^(m-i)*(-1);
               end
        end
        % powers to the odd numbers (imaginary)
        for i = 2:2:m-1
               if rem(m-i,4) == 1
                   p2 = p2 + num(i)*w^(m-i);
               else
                   p2 = p2 + num(i)*w^(m-i)*(-1);
               end
        end
    % when the largest order of s is odd
    elseif rem(m,2) == 0
        % powers to the even numbers (real)
        for i = 2:2:m
            if rem(m-i,4) == 0
                p1 = p1 + num(i)*w^(m-i);
            else
                p1 = p1 + num(i)*w^(m-i)*(-1);
            end
        end
        % powers to the odd numbers (imaginary)
        for i = 1:2:m-1
            if rem(m-i,4) == 1
                p2 = p2 + num(i)*w^(m-i);
            else
                p2 = p2 + num(i)*w^(m-i)*(-1);
            end
        end
    end

    % Solving the system equations
    Re = k*p1 == -f1
    Im = k*p2 == -f2
    a = vpasolve([Re Im], [k w]);
```

```
    K = double(a.k);
    W = double(a.w);
end
```

```
function [table_P, table_Z] = departure_arrival_angle_calc_posFeedback(zrs, poles)
    %{
       NAME:    ROOTLOCUS_STEPBYSTEP
       AUTHOR:  TOMOKI KOIKE
       INPUTS:  (1) zrs:   THE ZEROS OF THE TRANSFER FUNCTION
                (2) poles: THE POLES OF THE TRANSFER FUNCTION
       OUTPUTS: (1) theta: THE DEPARTURE ANGLES AND ARRIVAL ANGLES FOR THE
                           TRANSFER FUNCTION
       DESCRIPTION: CALCULATES ALL THE DEPARTURE ANGLES AND ARRIVALS ANGLES
       FOR THE PROVIDED ZEROS AND POLES OF A TRANSFER FUNCTION FOR POSITIVE
       FEEDBACK LOOP
    %}


    % PREALLOCATE ARRAY THETA TO STORE ALL ANGLES FOR THE POLES
    theta_P = zeros([1,(length(poles))]);

    % ANGLE FOR A FICTICIOUS POINT CLOSE TO EACH POLE
    for n = 1:length(poles)
        obj = poles(n);
        % ANGLES FROM THE ZERO POINT TO A THE CURRENT POINT
        if not(isempty(zrs))
            for i = 1:length(zrs)
                theta_P(n) = theta_P(n) + angle(obj - zrs(i));
            end
        end
        % ANGLE FROM ANOTHER POLE TO THE CURRENT POLE
        for i = 1:length(poles)
            theta_P(n) = theta_P(n) - angle(obj - poles(i));
        end
        % THE ANGLE BECOMES
        theta_P(n) = theta_P(n) + deg2rad(0);  % [rad]
    end

    % CREATING TABLE
    rad_P = reshape(theta_P,[length(theta_P),1]);
    deg_P = rad2deg(rad_P);
    table_P = table(reshape(poles,[length(poles),1]),rad_P,deg_P);
    table_P.Properties.VariableNames = {'POLES','RADIUS','DEGREES'};


    if not(isempty(zrs))
```

```
        % PREALLOCATE ARRAY THETA TO STORE ALL ANGLES FOR THE POLES
        theta_Z = zeros([1,(length(zrs))]);
        % ANGLE FOR A FICTICIOUS POINT CLOSE TO EACH ZERO
        for n = 1:length(zeros)
            obj = zrs(n);
            % ANGLES FROM THE ZERO POINT TO A THE CURRENT POINT
            if not(isempty(zrs))
                for i = 1:length(zrs)
                    theta_Z(n) = theta_Z(n) + angle(obj - zrs(i));
                end
            end
            % ANGLE FROM A POLE TO THE CURRENT ZERO POINT
            for i = 1:length(poles)
                theta_Z(n) = theta_Z(n) - angle(obj - poles(i));
            end
            % THE ANGLE BECOMES
            theta_Z(n) = -deg2rad(0) - theta_Z(n);  % [rad]
        end

        % CREATING TABLE
        rad_Z = reshape(theta_Z,[length(theta_Z),1]);
        deg_Z = rad2deg(rad_Z);
        table_Z = table(reshape(zrs,[length(zrs),1]),rad_Z,deg_Z);
        table_Z.Properties.VariableNames = {'ZEROS','ANGLES','DEGREES'};
    else
        table_Z = [];
    end
end
```

```
function [table_P, table_Z] = departure_arrival_angle_calc(zrs, poles)
    %{
       NAME:    ROOTLOCUS_STEPBYSTEP
       AUTHOR:  TOMOKI KOIKE
       INPUTS:  (1) zrs:   THE ZEROS OF THE TRANSFER FUNCTION
                (2) poles: THE POLES OF THE TRANSFER FUNCTION
       OUTPUTS: (1) theta: THE DEPARTURE ANGLES AND ARRIVAL ANGLES FOR THE
                           TRANSFER FUNCTION
       DESCRIPTION: CALCULATES ALL THE DEPARTURE ANGLES AND ARRIVALS ANGLES
       FOR THE PROVIDED ZEROS AND POLES OF A TRANSFER FUNCTION FOR NEGATIVE
       FEEDBACK LOOP
    %}
```

```
    % PREALLOCATE ARRAY THETA TO STORE ALL ANGLES FOR THE POLES
    theta_P = zeros([1,(length(poles))]);

    % ANGLE FOR A FICTICIOUS POINT CLOSE TO EACH POLE
    for n = 1:length(poles)
        obj = poles(n);
        % ANGLES FROM THE ZERO POINT TO A THE CURRENT POINT
        if not(isempty(zrs))
            for i = 1:length(zrs)
                theta_P(n) = theta_P(n) + angle(obj - zrs(i));
            end
        end
        % ANGLE FROM ANOTHER POLE TO THE CURRENT POLE
        for i = 1:length(poles)
            theta_P(n) = theta_P(n) - angle(obj - poles(i));
        end
        % THE ANGLE BECOMES
        theta_P(n) = theta_P(n) + deg2rad(180);  % [rad]
    end

    % CREATING TABLE
    rad_P = reshape(theta_P,[length(theta_P),1]);
    deg_P = rad2deg(rad_P);
    table_P = table(reshape(poles,[length(poles),1]),rad_P,deg_P);
    table_P.Properties.VariableNames = {'POLES','RADIUS','DEGREES'};


    if not(isempty(zrs))
        % PREALLOCATE ARRAY THETA TO STORE ALL ANGLES FOR THE POLES
        theta_Z = zeros([1,(length(zrs))]);
        % ANGLE FOR A FICTICIOUS POINT CLOSE TO EACH ZERO
        for n = 1:length(zeros)
            obj = zrs(n);
            % ANGLES FROM THE ZERO POINT TO A THE CURRENT POINT
            if not(isempty(zrs))
                for i = 1:length(zrs)
                    theta_Z(n) = theta_Z(n) + angle(obj - zrs(i));
                end
            end
            % ANGLE FROM A POLE TO THE CURRENT ZERO POINT
            for i = 1:length(poles)
                theta_Z(n) = theta_Z(n) - angle(obj - poles(i));
            end
            % THE ANGLE BECOMES
            theta_Z(n) = -deg2rad(180) - theta_Z(n);  % [rad]
        end

        % CREATING TABLE
        rad_Z = reshape(theta_Z,[length(theta_Z),1]);
        deg_Z = rad2deg(rad_Z);
```

```
        table_Z = table(reshape(zrs,[length(zrs),1]),rad_Z,deg_Z);
        table_Z.Properties.VariableNames = {'ZEROS','ANGLES','DEGREES'};
    else
        table_Z = [];
    end
end
```

```
function rts = break_in_away_pt(num,den)
    [q, d] = polyder(-den,num)
    rts = roots(q);
    rts = rts(rts==real(rts));
end
```