AAE 364: Controls System Analysis

HW9: Controller Design & Root Locus

Dr. Sun

School of Aeronautical and Astronautical

Purdue University

Tomoki Koike

Friday April 3, 2020

B–6–19. Referring to the system shown in Figure 6–109, design a compensator such that the static velocity error constant K_v is 20 sec^{-1} without appreciably changing the original location $\left(s = -2 \pm j2\sqrt{3}\right)$ of a pair of the complex-conjugate closed-loop poles.

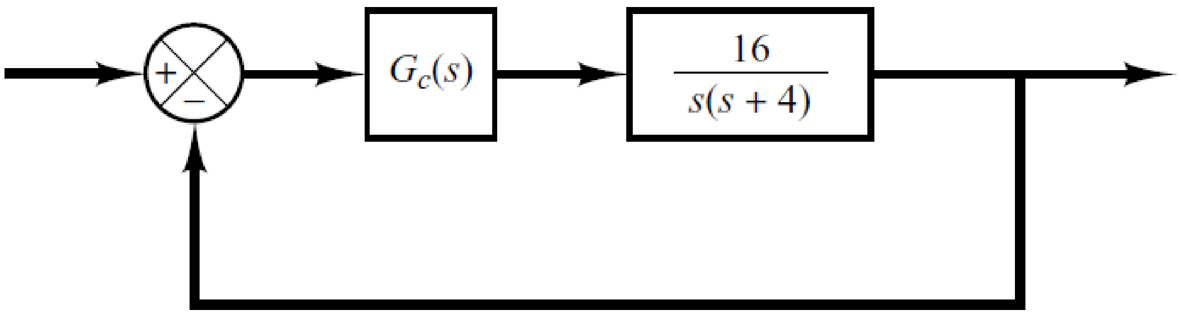
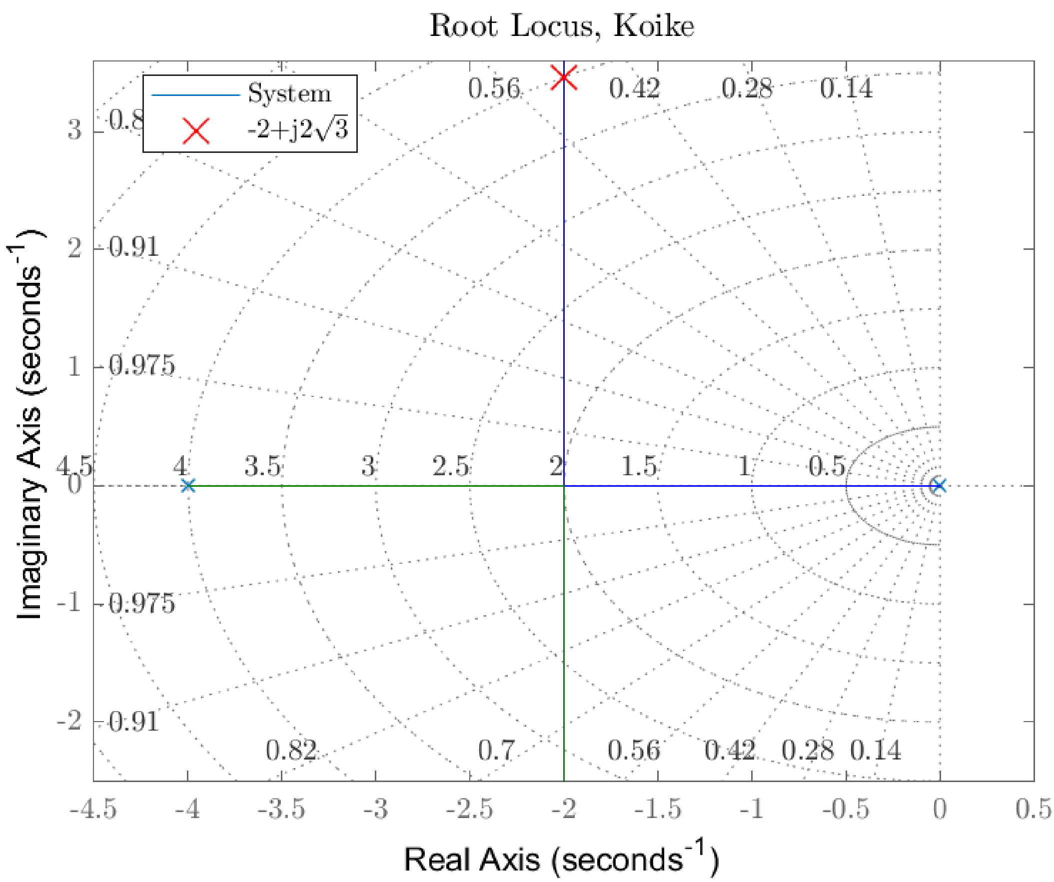


Figure 6–109
Control system.



First find the angle deficiency, ψ

The desired pole, $s_d = -2 + j\,2\sqrt{3}$

since, $G(s) = \dfrac{16}{s(s+4)}$

then,

$$\arg[G(s_d)] = \cancel{\arg(16)}^{\,0} - \arg(s_d) - \arg(s_d+4)$$

$$\Rightarrow \psi = -180^\circ + \arg(-2+j2\sqrt{3}) + \arg(2+j2\sqrt{3})$$

$$\psi = 0$$

Since, $\psi = 0$, we need a P-controller

$$G_c(s) = K_c$$

and

$$K_v = \lim_{s\to 0} s\,G_c(s)G(s) = \lim_{s\to 0} K_c \frac{16}{s+4}$$

$$\Longrightarrow \quad 20 = \frac{16}{4}K_c$$

$$K_c = 5$$

$$\boxed{G_c(s) = 5}$$

B–6–21. Consider the control system shown in Figure 6–111. Design a compensator such that the dominant closed-loop poles are located at $s = -2 \pm j2\sqrt{3}$ and the static velocity error constant K_v is 50 sec^{-1}.

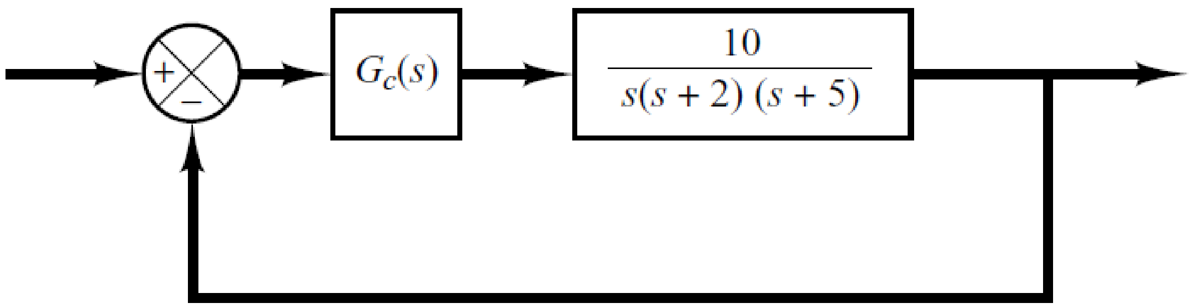
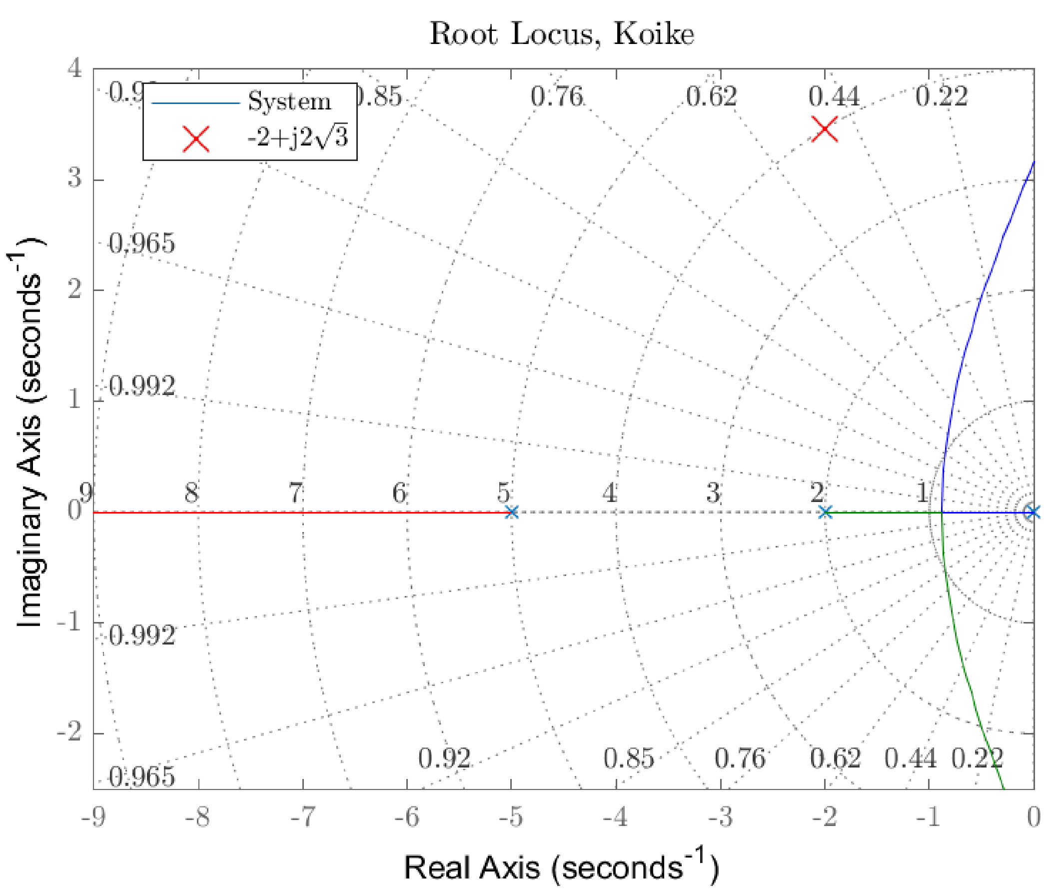


Figure 6–111
Control system.



The angle deficiency, ψ becomes

$$\psi = -180^\circ - \arg[G(s_d)]$$

where $s_d = -2 + j2\sqrt{3}$

$$\psi = -180^\circ - \cancel{\arg(10)}^{\,0} + \arg(-2+j2\sqrt{3})$$
$$+\arg(j2\sqrt{3}) + \arg(3+j2\sqrt{3})$$

$$\psi = 79.1066^\circ$$

When $\psi > 0$, we design a lag-lead compensator

$$G_c(s) = k_c\left(\frac{s+z_1}{s+\beta z_1}\right)\left(\frac{s+z_2}{s+\frac{z_2}{\beta}}\right) \quad (\beta > 1)$$

since $K_v = 50$ sec

$$K_v = \lim_{s\to 0} s\overline{G(s)} = \lim_{s\to 0} s\, G_c(s)G(s)$$

$$K_v = \lim_{s\to 0} s\,(k_c)\left(\frac{s+z_1}{s+\beta z_1}\right)\left(\frac{s+z_2}{s+\frac{z_2}{\beta}}\right)\frac{10}{s(s+2)(s+5)}$$

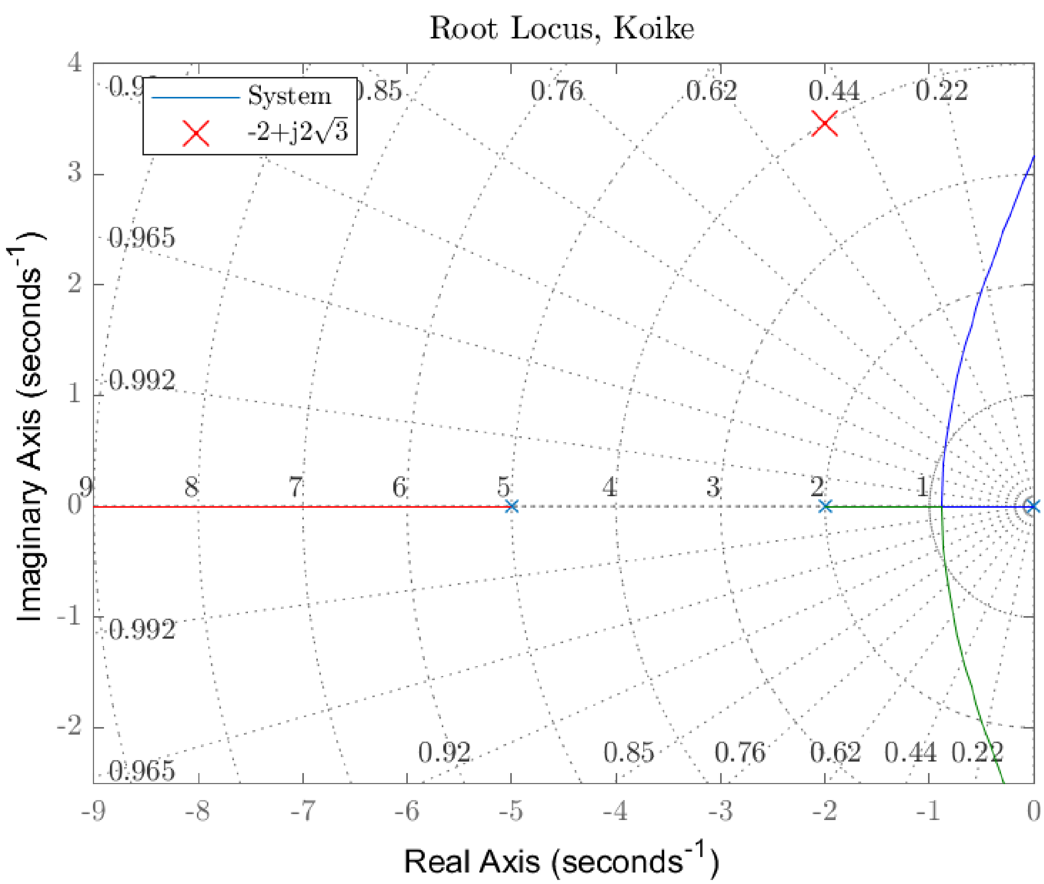
$$50 = k_c\,\frac{10}{10}$$

$$\therefore\ k_c = 50$$

now, say the lead portion must compensate for ψ

$$\Rightarrow \arg(s_d + z_1) - \arg(s_d + \beta z_1) = \psi$$

$$\frac{s_d + z_1}{s_d + \beta z_1} = \tan\psi \quad \cdots \text{①}$$



$$(z_1 - 2)\tan(79.1066^\circ) = 2\sqrt{3}$$

$$z_1 = \frac{2\sqrt{3}}{\tan(79.1066^\circ)} + 2 = 2.6667$$

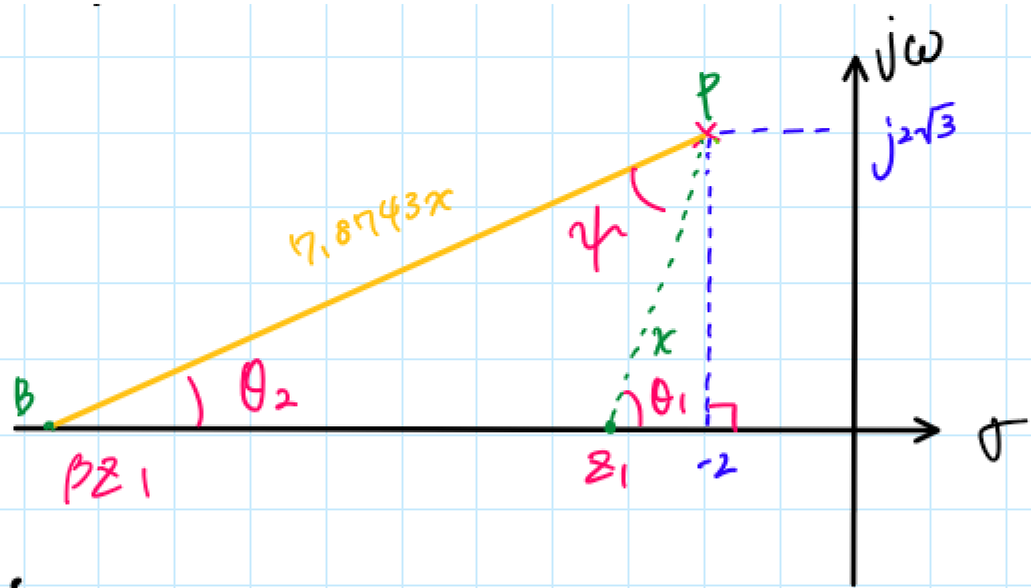
hence, $z_1 < 2.6667$ must be satisfied.

the magnitude condition for lead portion becomes

$$50 \left| \frac{10(s_d + z_1)}{(s_d+\beta z_1) s_d (s_d+2)(s_d+5)} \right| = 1 \quad \cdots ②$$

$$\frac{|s_d + z_1|}{|s_d + \beta z_1|} = \frac{1}{7.8743}$$

we have 2 unknowns z_1 & β
and 2 equations ① & ②
using these we can obtain z_1 & β
using MATLAB (code in Appendix)



$$\begin{cases} \angle APB = \psi = 79.1066^\circ \\ \dfrac{\overline{PA}}{\overline{PB}} = \dfrac{x}{7.8743x} = \dfrac{1}{7.8743} \end{cases}$$

Using trigonometry

$$\frac{(\beta-1)z_1}{\sin\psi} = \frac{x}{\sin\theta_2} = \frac{7.8743x}{\sin(\pi-\theta_1)}$$

$$\Longrightarrow \frac{x}{\sin(\theta_1-\psi)} = \frac{7.8743x}{\sin(\pi-\theta_1)}$$

$$7.8743\sin(\theta_1-\psi) - \sin\theta_1 = 0$$

$$7.8743\left(\sin\theta_1\cos\psi - \cos\theta_1\sin\psi\right) - \sin\theta_1 = 0$$

$$(7.8743\cos\psi)\sin\theta_1 - (7.8743\sin\psi)\cos\theta_1 - \sin\theta_1 = 0$$

$$(7.8743\cos\psi - 1)\sin\theta_1 = (7.8743\sin\psi)\cos\theta_1$$

$$\tan\theta_1 = \frac{7.8743\sin\psi}{7.8743\cos\psi - 1}$$

$$\theta_1 = \arctan\left(\frac{7.8743\sin\psi}{7.8743\cos\psi - 1}\right)$$

since $\psi = 79.1066^\circ$

$$\Longrightarrow \theta_1 = 86.3880^\circ$$

then

$$\frac{2\sqrt{3}}{z_1 - 2} = \tan\theta_1$$

$$z_1 = \frac{2\sqrt{3}}{\tan\theta_1} + 2 = 2.2187$$

and

$$\theta_2 = \theta_1 - \psi = 7.2814^\circ$$

so

$$\frac{2\sqrt{3}}{\beta z_1 - 2} = \tan\theta_2$$

$$\beta = \frac{1}{z_1}\left(\frac{2\sqrt{3}}{\tan\theta_2} + 2\right) = 13.1210$$

$$p_1 = \beta z_1 = 29.1113$$

thus,

$$G_c(s) = 50\left(\frac{s + 2.2187}{s + 29.1113}\right)\left(\frac{s + z_2}{s + \frac{z_2}{13.1210}}\right)$$

next for the lag-component
we arbitrarily choose

$$p_2 = \frac{z_2}{13.1210} = 0.01 < 29.1113 = p_1$$

$$\therefore\ z_2 = 0.13121$$

For the lag

$$\left| \frac{s_d + 0.13121}{s_d + 0.01} \right| = 0.9852$$

$$\arg\left(\frac{s_d + 0.13121}{s_d + 0.01} \right) = -1.5301^\circ$$

from these we can tell that the magnitude and angle contribution of the lag component are trivial enough that the dominant CL poles lie close to the desired pole s_d.

$$\boxed{G_c(s) = 50 \left(\frac{s + 2.2187}{s + 29.1113} \right) \left(\frac{s + 0.13121}{s + 0.01} \right)}$$

B–6–23. Consider the control system shown in Figure 6–113. Design a compensator such that the unit-step response curve will exhibit maximum overshoot of 25% or less and settling time of 5 sec or less.

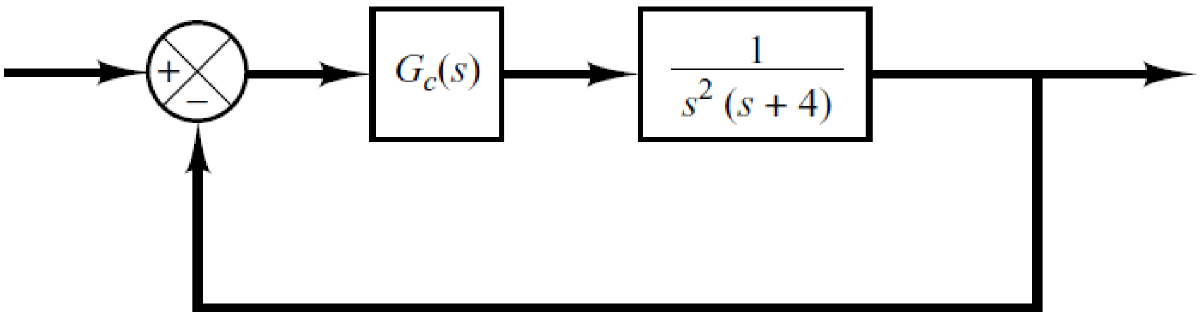


Figure 6–113
Control system.

from the design requirements

$$M_p \leqq 25\% \quad \cdots ①$$

$$t_s \leqq 5 \quad [s] \quad \cdots ②$$

we can find the desired ζ & ω_n

$\because$ ①

$$\zeta = \frac{-\ln(M_p/100)}{\sqrt{\pi^2 + [\ln(M_p/100)]^2}} = 0.4037$$

and $\because$ ②

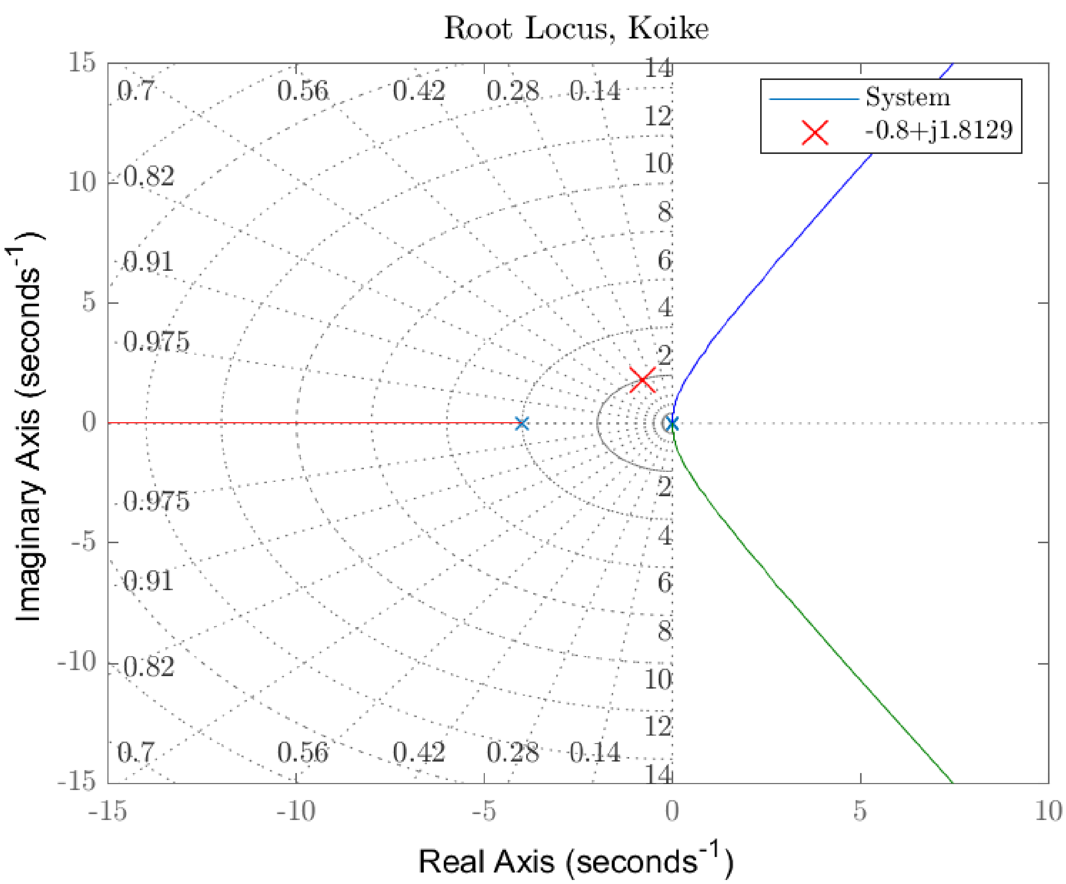
$$t_s = \frac{4}{\zeta \omega_n} = 5 \quad (2\%)$$

$$\omega_n = 1.9816 \text{ rad/s}$$

now the desired pole, s_d becomes

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

$$s_d = -0.8000 \pm j\,1.8129$$



from this RL plot we can tell that the s_d is not on RL.

the angle deficiency ψ becomes

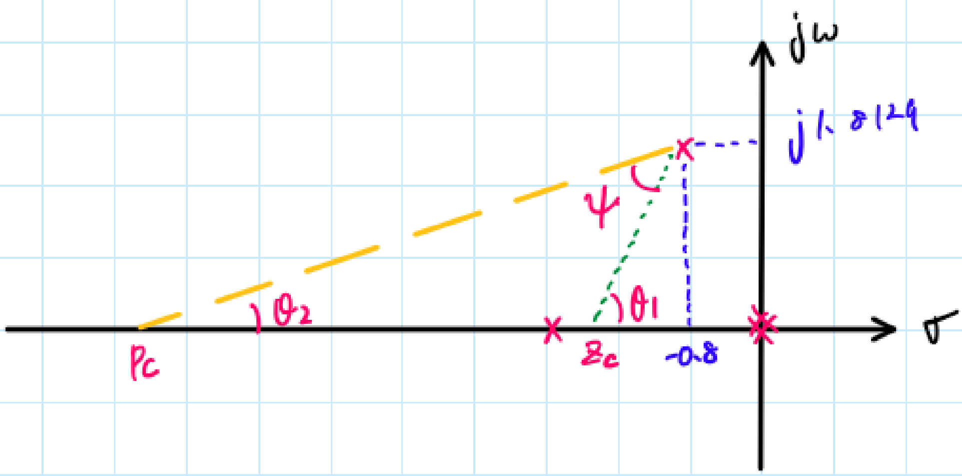
$$\psi = -180^\circ - \arg\left[G(s_d)\right]$$

$$\psi = -180^\circ - \cancel{\arg(1)}^{\,0} + \arg(s_d) + \arg(s_d) + \arg(s_d+4)$$

$$\psi = 77.1545^\circ$$

since, we have to improve only the transient response and set s_d on RL we select a lead compensator design.

$$G_c(s) = K\frac{s+z_c}{s+p_c} \qquad (0 < z_c < p_c)$$



the max (z_c) we can select arbitrarily is

$$z_c = \frac{1.8129}{\tan(77.1545^\circ)} + 0.8 = 1.2134$$

thus, we choose $z_c = 1.2$

then, from trigonometry

$$\tan\theta_1 = \frac{1.8129}{z_c - 0.8}$$

$$\theta_1 = \arctan\left(\frac{1.8129}{1.2-0.8}\right) = 77.5579^\circ$$

then

$$\theta_2 = \theta_1 - \psi = 0.4034^\circ$$

thus,

$$\frac{1.8129}{p_c - 0.8} = \tan\theta_2$$

$$p_c = \frac{1.8129}{\tan(0.4034^\circ)} + 0.8 = 258.3109$$

then, from magnitude condition

$$K \left\| \left(\frac{s+1.2}{s+258.3109}\right)\left(\frac{1}{s^2(s+4)}\right) \right\|_{s=s_d} = 1$$

$$\therefore K = 2003.2$$

Hence, the designed compensator becomes

$$\boxed{G_c(s) = 2003.2 \frac{s+1.2}{s+258.3109}}$$

B–6–24. Consider the system shown in Figure 6–114, which involves velocity feedback. Determine the values of the amplifier gain K and the velocity feedback gain K_h so that the following specifications are satisfied:

1. Damping ratio of the closed-loop poles is 0.5
2. Settling time $\leq$ 2 sec
3. Static velocity error constant $K_v \geq 50 \text{ sec}^{-1}$
4. $0 < K_h < 1$

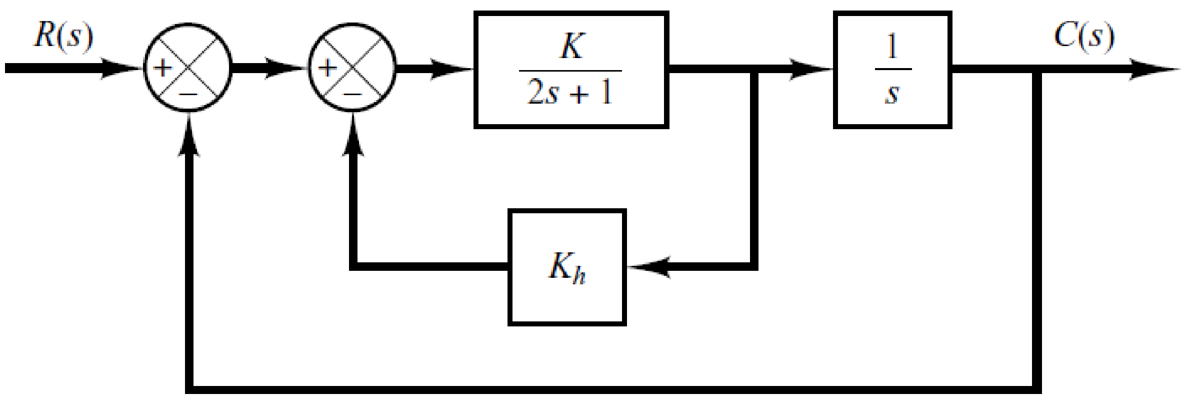
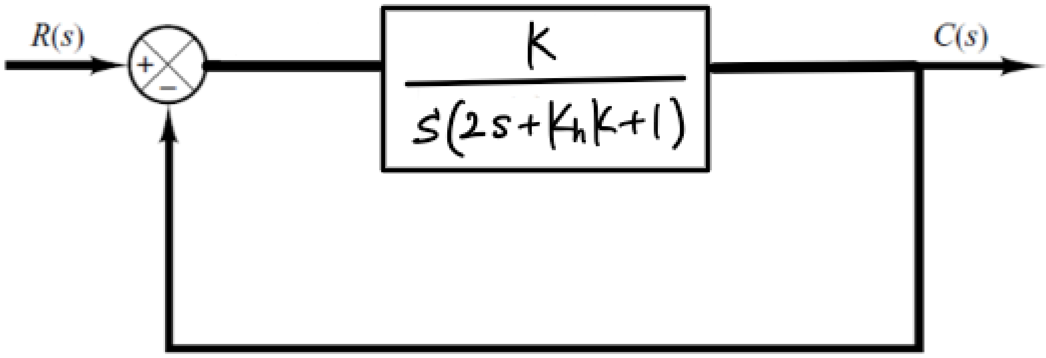


Figure 6–114
Control system.

first, simplify the block diagram

$$G_1(s) = \frac{\frac{K}{2s+1}}{1 + \frac{K_h K}{2s+1}} = \frac{K}{2s + K_h K + 1}$$

$$G(s) = \frac{K}{s(2s + K_h K + 1)}$$



CLTF becomes

$$\frac{\dfrac{K}{s(2s+K_hK+1)}}{1+\dfrac{K}{s(2s+K_hK+1)}} = \frac{K/2}{s^2+\left(\dfrac{1+KK_h}{2}\right)s+\dfrac{K}{2}}$$

thus, $\omega_n = \sqrt{\dfrac{K}{2}} \quad \cdots ①$

and from the given design requirements

$$\zeta = 0.5$$

$$t_s = \frac{4}{\zeta\omega_n} \leq 2$$

$$\omega_n \geq 4 \quad \cdots ②$$

and $\quad 2\zeta\omega_n = \dfrac{1+KK_h}{2}$

$$\omega_n = \frac{1+KK_h}{2} \quad \cdots ③$$

$$\Longrightarrow \quad 1+KK_h \leqq 8 \quad \cdots ④$$

from $K_v \geqq 50 \ \sec^{-1}$

$$K_v = \lim_{s\to 0} s\,G(s) = \lim_{s\to 0} s\,\frac{K}{s(2s+KK_h+1)}$$

$$\geqq 50$$

$$\longrightarrow \quad \frac{K}{KK_h+1} \geqq 50 \quad \cdots ⑤$$

$\because$ ① $\&$ ②

$$\sqrt{\frac{K}{2}} \geqq 4$$

$$K \geqq 32$$

$\because$ ①③⑤

$$\sqrt{2K} = 1+KK_h \quad \cdots ⑥$$

$$\frac{K}{\sqrt{2K}} \geqq 50$$

$$\frac{K}{2} \geqq 2500$$

$$K \geqq 5000$$

$\therefore K = 5000$ を ⑥

$$100 = 1 + 5000 K_h$$

$$K_h = \frac{99}{5000} = 0.0198$$

$$\boxed{K = 5000, \quad K_h = 0.0198}$$

B–6–26. Consider the system shown in Figure 6–116. Plot the root loci as a varies from 0 to ∞. Determine the value of a such that the damping ratio of the dominant closed-loop poles is 0.5.

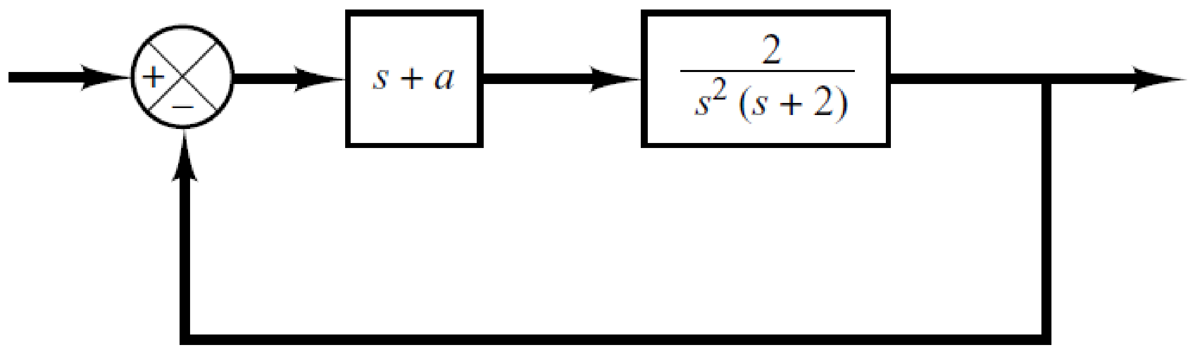


Figure 6–116
Control system.

$$CE: = \quad 1 + \frac{2(s+a)}{s^2(s+2)} = 0$$

$$\Rightarrow \quad s^3 + 2s^2 + 2s + 2a = 0$$

$$\Rightarrow \quad 1 + \frac{2a}{s^3 + 2s^2 + 2s} = 0$$

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

define $\quad a = K$

$$CE: = \quad 1 + \frac{2K}{s(s^2 + 2s + 2)} = 0$$

Find the RL

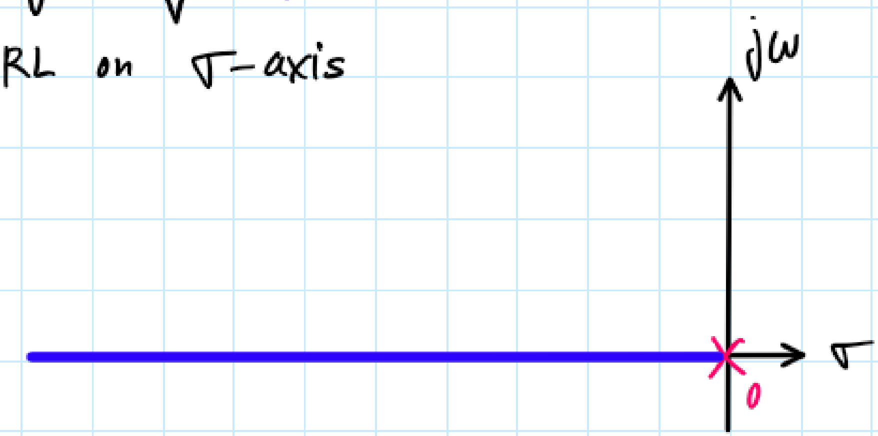
(i) Poles & zeros

p_i: $p_1 = 0$, $p_{2,3} = -1 \pm j$

z_i: none $m = 0$

(ii) symmetry TRUE

(iii) RL on σ-axis



(iv) Asymptotes

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

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

$$\sigma_a = \frac{0 + (-1+j) + (-1-j)}{3} = -\frac{2}{3}$$

(v) Break-in/away points

$$\frac{d}{ds}\left[-\frac{s(s^2+2s+2)}{2}\right]$$

$$= -\frac{6s^2 + 8s + 4}{4} = 0$$

$$\Rightarrow \quad 3s^2 + 4s + 2 = 0$$

$$\hat{s}_{1,2} = -0.6667 \pm j0.4714$$

$\Rightarrow$ NONE

<vi> Departure / Arrival angles

a point s_D near point $(-1+j)$

$$-180^\circ = \arg(2) - \arg(-1+j) - \theta_d - \arg[-1+j-(-1-j)]$$

$$\theta_d = -45$$

(vii) Intersection of RL with $j\omega$

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

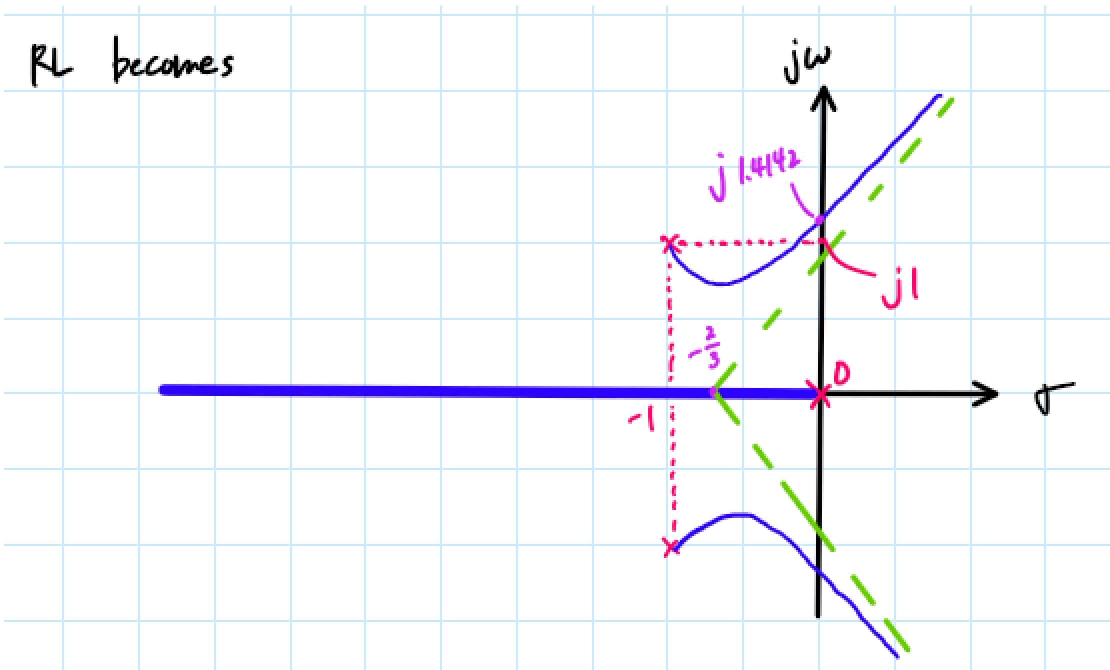
$$\Rightarrow \sigma: \quad 2\hat{k} = 2\hat{\omega}^2$$

$$j\omega: \quad 0 = \hat{\omega}^3 - 2\hat{\omega}$$

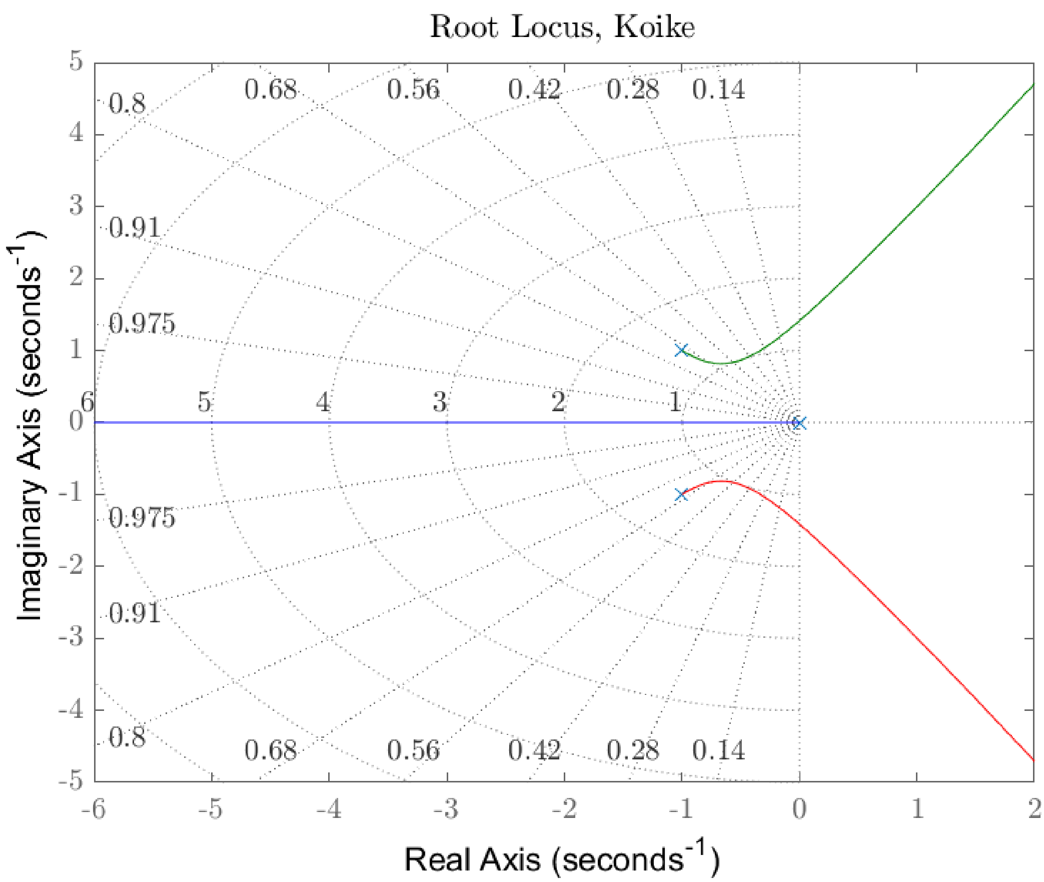
$$\hat{k} = \begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix} \quad \hat{\omega} = \begin{bmatrix} 0 \\ -1.4142 \\ 1.4142 \end{bmatrix}$$

since $k > 0$, ones marked as are elligible

RL becomes



jω
j1.4142
j1
$-\frac{2}{3}$
-1
0
σ



Root Locus, Koike
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})

since the dominant closed loop poles is located @

$$\zeta = 0.5$$
$$\theta = \arccos(0.5) = 60°, -60°$$

the desired pole, $s_d = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2}$

$$s_d = -0.5\omega_n \pm j\omega_n\sqrt{1-0.5^2}$$

then, compute

$$1 + \hat{k}\frac{2}{s(s^2+2s+2)}\bigg|_{s=s_d} = 0$$

$$s_d(s_d^2 + 2s_d + 2) + 2\hat{k} = 0$$

$$\Rightarrow \quad 2\hat{k} = -s_d(s_d^2 + 2s_d + 2)$$

solving this we get

$$2\hat{k} = \omega_n^3 - \omega_n^2 - \omega_n + j\sqrt{3}\omega_n - j\sqrt{3}\omega_n^2$$

$$\Rightarrow \begin{cases} 2\hat{k} = \omega_n^3 - \omega_n^2 - \omega_n \\ 0 = \omega_n - \omega_n^2 \end{cases}$$

$$\omega_n = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \hat{k} = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix}$$

since, $K = \hat{k} = 0.5$ we know that

$$\boxed{a = 0.5}$$

Problem 2: Aircraft Control Example

Figure 1 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.

where $G(s)$ is the transfer function representing the aircraft pitch angle response output to the elevator deflection input. Consider the unity-feedback system in Figure 2 with the plant $G(s)$ representing the aircraft shown in Figure 1.

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

Design a controller $K(s)$ such that the unit step response has the following characteristics:

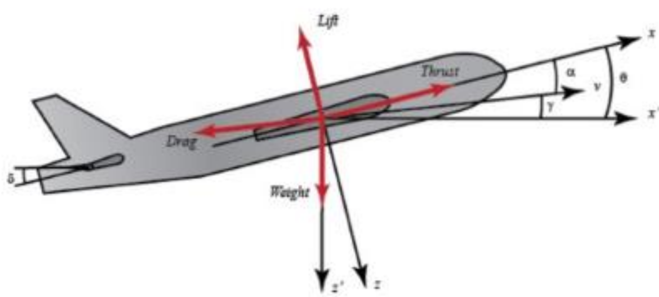


Figure 1: Forces acting on an aircraft in the Longitudinal plane.

1. Settling time ≤ 2 sec (2% criterion)
2. Maximum overshoot $\leq 10\%$
3. Zero steady state error with respect to a unit ramp input

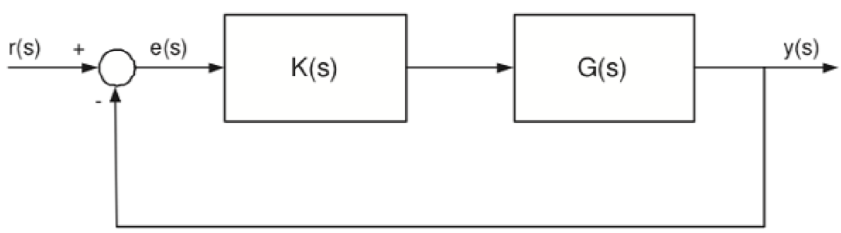


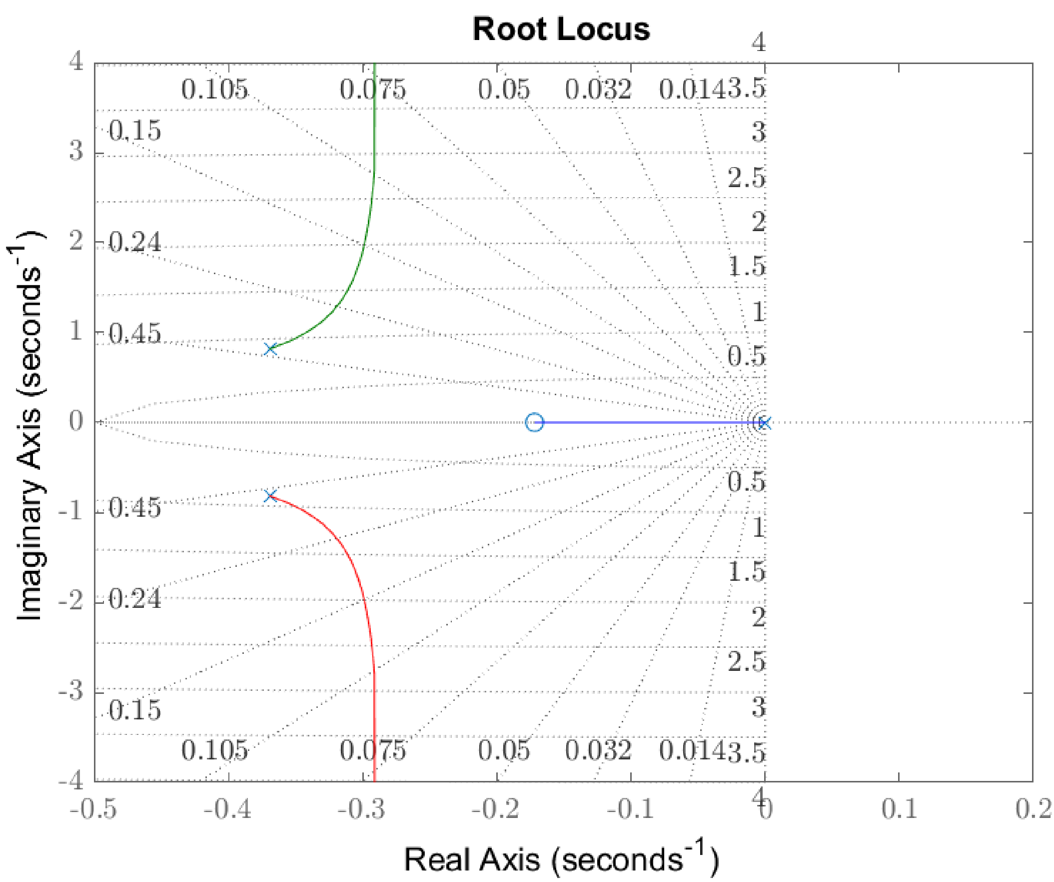
Figure 2: Unity-Feedback System with controller $K(s)$ and plant $G(s)$.

In HW8 we have computed the RL for

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

which is

$$= \frac{(s + 0.1718)}{s(s + 0.3693 + j0.8151)(s + 0.3693 - j0.8151)}$$



from design requirements 1 & 2

settling time : $t_s = \frac{4}{\zeta\omega_n} \leqq 2$ sec (2%)

max overshoot : $M_p \leqq 10\%$

since

$$M_p = \exp\left(\frac{-\zeta\omega_n}{\sqrt{1-\zeta^2}}\right) \times 100$$

$$\Rightarrow \zeta \geqq \frac{-\ln(M_p/100)}{\sqrt{\pi^2 + [\ln(M_p/100)]^2}} = 0.5912$$

then $\zeta\omega_n \geqq 2 \quad \cdots (1)$

and from the third requirement

$$\bar{K}_v = \lim_{s\to 0} s\bar{G}(s) = \lim_{s\to 0} sK(s)G(s) = 0 \quad \cdots (2)$$

to achieve zero steady-state error we design a PID-controller and since we are also required to improve the transient response.

then

$$K(s) = K\frac{(s+z_{lag})(s+z_{lead})}{s}$$

if $\zeta = 0.5912 \Rightarrow \omega_n \geqq \frac{2}{0.5912} = 3.3832$

so say

$$\zeta = 0.5912 \quad \& \quad \omega_n = 3.3832$$

the desired poles become

$$s_{d_0} = -\zeta\omega_n \pm j\,\omega_n\sqrt{1-\zeta^2}$$

$$s_{d_0} = -2.0 \pm j2.7288$$

then, set

$$s_d = -2.0 + j2.7288$$

the angle deficiency w.r.t s_d becomes

$$\psi = -180^\circ - \arg\left[G(s_d)\right]$$

$$\begin{aligned}\psi = -180^\circ &+ \arg(-2.0 + j2.2788) \\ &+ \arg\left[-2.0 + j2.2788 + (0.3693 + j0.8151)\right] \\ &+ \arg\left[-2.0 + j2.2788 + (0.3693 - j0.8151)\right] \\ &- \arg\left[-2.0 + j2.2788 + 0.1718\right]\end{aligned}$$

$$\psi = 67.5654^\circ$$

since $\psi > 0$ the PD-controller will compensate for ψ

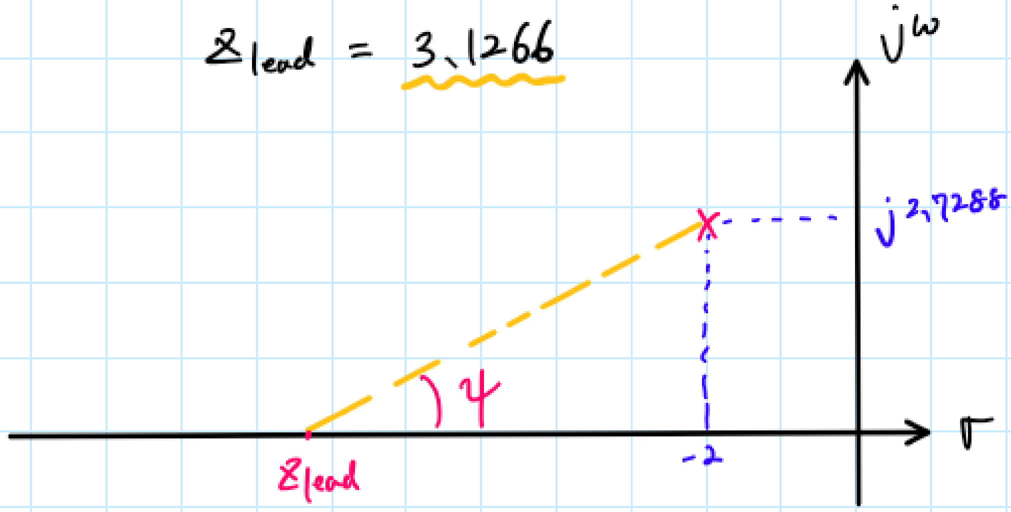
thus,

$$PD: \quad K_1(s + z_{lead})$$

and

$$\arg(s_d + z_{lead}) = \psi$$

$$z_{lead} = 3.1266$$



then from magnitude condition

$$K_1 \left\| \frac{(s_d+3.1266)(1.1057s_d+0.1900)}{(s_d^3+0.7385s_d^2+0.8008s_d)} \right\| = 1$$

$$K_1 = 3.0950$$

thus,

$$K_{PD}(s) = 3.0950(s+3.1266)$$

then, using relation ②

$$\bar{K}_v = \lim_{s\to 0} s K_{PI} K_{PD} G(s) \qquad \text{where} \quad K_{PI} = K_2 \frac{s+z_{lag}}{s}$$

$$= \lim_{s\to 0} \cancel{s} K_{PI} \left(\frac{\cancel{s}^{\,0}+z_{lag}}{\cancel{s}}\right)\left[3.0950(\cancel{s}^{\,0}+3.1266)\right] \times \frac{(\cancel{s}^{\,0}+0.1718)}{\cancel{s}(\cancel{s}^{\,0}+0.3693+j0.8151)(\cancel{s}^{\,0}+0.3693-j0.8151)}$$

$= \infty$ if $K_v \to \infty$, $e_{ss} = \frac{1}{K_v} = 0$

steady state error is $\emptyset$

next we select z_{lag} that is smaller than z_{lead} and keeps $\psi = 0$

that is

$$\arg(s_d + z_{lag}) - \arg(s_d) \simeq 0$$

$z_{lag} \ll 1 \implies$ so we choose

$$z_{lag} = 0.01$$

then

$$\arg(s_d + 0.01) - \arg(s_d) = -0.1368^\circ$$

this angle is small enough.

and then, from magnitude condition

$$K_1 K_2 \left\| \frac{(s_d+0.01)(s_d+3.1266)(1.1057 s_d + 0.1900)}{s_d (s_d^3 + 0.7385 s_d^2 + 0.8008 s_d)} \right\| = 1$$

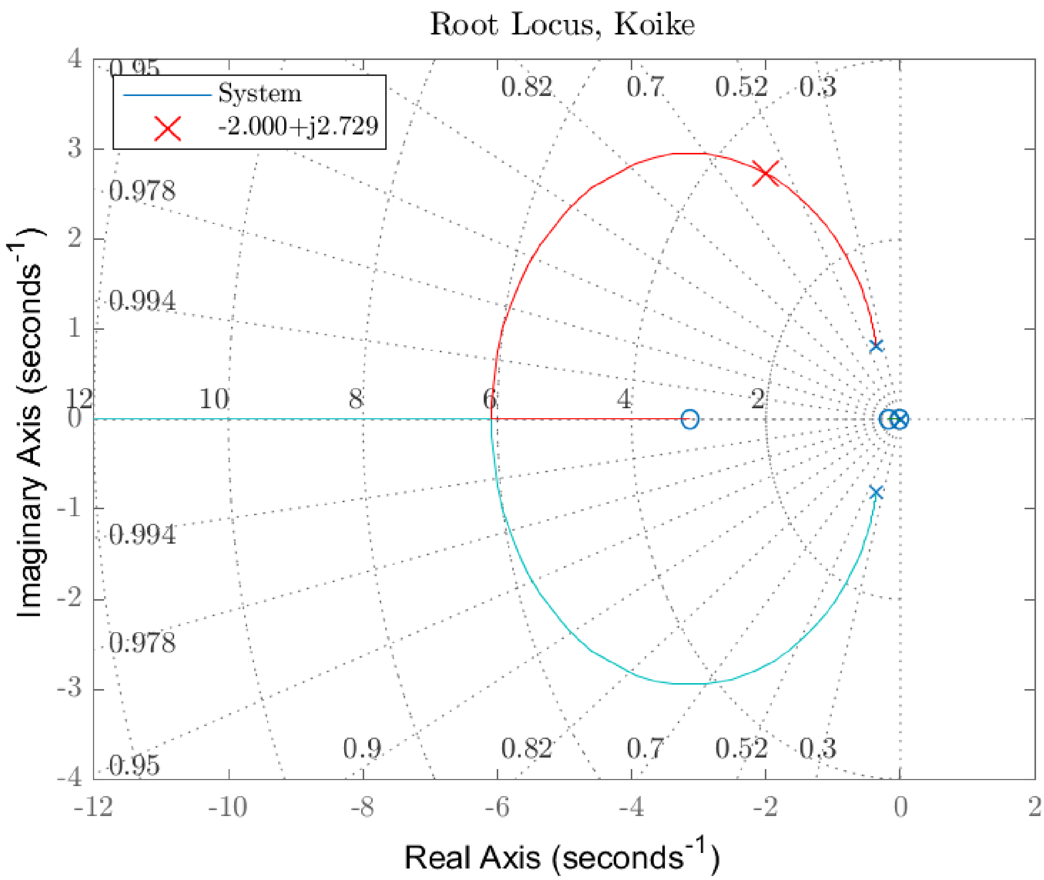
$$\therefore K_2 = 1.0017$$

then, $K = K_1 K_2 = 3.1004$

Thus, the controller $K(s)$ becomes

$$\boxed{K(s) = 3.1004\,\frac{(s+0.01)(s+3.1266)}{s}}$$

then the RL becomes



Appendix

AAE364 HW9 MATLAB CODE

```
clear all; close all; clc;
fdir = 'C:\Users\Tomo\Desktop\studies\2020-
Spring\AAE364\matlab\matlab_output\hw9';
set(groot, 'defaulttextinterpreter',"latex");
set(groot, 'defaultAxesTickLabelInterpreter',"latex");
set(groot, 'defaultLegendInterpreter',"latex");
```

B-6-19

```
% Define OLTF G(s)
num = 16;                   % numerator
den = conv([1 0],[1 4]);  % denominator

% Desired pole
sd = -2 + 2i*sqrt(3);

% Compute deficiency angle [deg]
psi = calc_ang_deficiency(num,den,sd);

% Plotting the RL (negative feedback)
fig1 = figure("Renderer","painters");
    rlocusplot(tf(num,den)); sgrid;
    title('Root Locus, Koike',"Interpreter","latex")
    hold on
    plot(real(sd),imag(sd),'xr','MarkerSize',12)
    hold off
    ylim([-2.5 3.6])
    legend('System','-2+j2$\sqrt{3}$',"location","best")
saveas(fig1,fullfile(fdir,'B-6-19_RL.png'));
```

B-6-21

```
% Define OLTF G(s)
num = [0 10];               % numerator
den = conv([1 0],[1 2]);  % denominator
den = conv(den,[1 5]);

% Desired pole
Kv = 50;  % [s-1]
sd = -2 + 2i*sqrt(3);

% Compute deficiency angle [deg]
psi = calc_ang_deficiency(num,den,sd)

% Plotting the RL (negative feedback)
fig2 = figure("Renderer","painters");
    rlocusplot(tf(num,den)); sgrid;
```

```
    title('Root Locus, Koike',"Interpreter","latex")
    hold on
    plot(real(sd),imag(sd),'xr','MarkerSize',12)
    hold off
    ylim([-2.5 4.0])
    legend('System','-2+j2$\sqrt{3}$',"location","best")
saveas(fig2,fullfile(fdir,'B-6-21_RL.png'));
% Finding z1, p1 (beta*z1), and Kc
Kc = Kv;
z1_lim = imag(sd)/tand(psi) - real(sd);  % condition for z1
theta1 = atan(7.8743*sind(psi)/(7.8743*cosd(psi) - 1));
theta1_deg = rad2deg(theta1);
z1 = imag(sd)/tan(theta1) + 2;
theta2_deg = theta1_deg - psi;
beta = (imag(sd)/tand(theta2_deg) + 2)/z1;

% lag component
p2 = 0.01;
mag_lag = abs((sd + p2*beta)/(sd + p2));
ang_lag = rad2deg(angle(sd + p2*beta) - angle(sd + p2));
```

B-6-23

```
% Design requirements
Mp = 25;  % [%]
ts = 5;  % settling time
% Design parameters
zeta = calc_zetaFromMOS_or_MOSFromzeta(Mp,"zeta");
wn = 4/5/zeta;
% Desired pole (positive imag)
sd = -zeta*wn + wn*1i*sqrt(1 - zeta^2);

% OLTF
num = [0 1];
den = conv([1 0],[1 0]);
den = conv(den,[1 4]);
% Plotting the RL (negative feedback)
fig3 = figure("Renderer","painters");
    rlocusplot(tf(num,den)); sgrid;
    title('Root Locus, Koike',"Interpreter","latex")
    hold on
    plot(real(sd),imag(sd),'xr','MarkerSize',12)
    hold off
    sd_txt = sprintf("%0.3f+j%0.3f",real(sd),imag(sd));
    legend('System',sd_txt,"location","best")
saveas(fig3,fullfile(fdir,'B-6-23_RL.png'));
% Angle deficiency
psi = calc_ang_deficiency(num,den,sd)

% Selecting zc, pc, and K
zc_lim = imag(sd)/tand(psi) - real(sd)  % condition for zc
zc = 1.2;
```

```
theta1_deg = atand(imag(sd)/(zc + real(sd)));
theta2_deg = theta1_deg - psi;
pc = imag(sd)/tand(theta2_deg) - real(sd);
% Calculate K
syms K
Gc = (sd + zc)/(sd + pc);
G = 1/sd^2/(sd + 4);
eqn = K*abs(Gc*G) == 1
K = double(solve(eqn,K));
```

B-6-24

```
% Design parameters
zeta = 0.5;
ts = 2;
wn = 4/ts/zeta
% Desired pole (positive imag)
sd = -zeta*wn + wn*1i*sqrt(1 - zeta^2)
Kh = 99/5000;
```

B-6-26

```
% RL
num = [0 2];
den = [1 2 2 0];
[poles,zrs,angs,sigma,bi_pt,T_P,T_Z,k,w,fig1] =
rootLocus_stepBystep_negFeedback(num,den)
saveas(fig1,fullfile(fdir,"B-6-26_RL.png"))
% Find gain, K when zeta = 0.5
zeta = 0.5;

syms K wn s
assume(K,'real');
assume(wn,'real');
p = -zeta*wn + wn*1j*sqrt(1 - zeta^2);
RHS = (s*(s^2 + 2*s + 2))
RHS = subs(RHS,s,p)
RHS = expand(RHS)
eqn1 = 2*K == -real(RHS)
eqn2 = 0 == -imag(RHS)
res = solve([eqn1 eqn2],[wn K]);
wn = double(res.wn)
Kh = double(res.K)
Kh = nonzeros(Kh)
```

P2

```
% Define system
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);
% Design parameters
zeta = calc_zetaFromMOS_or_MOSFromzeta(10,"zeta");
wn = 2/zeta;
```

```
% Desired pole (positive imag)
sd = -zeta*wn + wn*1i*sqrt(1 - zeta^2);
% Angle deficiency
psi = calc_ang_deficiency(num,den,sd)
% Find z_lead
syms z_lead
assume(z_lead,{'real','positive'})
eqn = angle(sd + z_lead) == deg2rad(psi);
z_lead = double(solve(eqn,z_lead))

% Find K1
syms K1
num2 = conv(num,[1 z_lead]);
G = create_TF_expression(num2,den);  % Expression with "syms" of "s"
G_sd = subs(G,s,sd);
eqn = K1*abs(G_sd) == 1;
K1 = double(solve(eqn,K1))
% Find z_lag
z_lag = 0.01;
d_ang = rad2deg(angle(sd + z_lag) - angle(sd))
% Find K2
syms K2
num3 = conv(num2,[1 z_lag]);
den2 = conv(den,[1 0]);
G = create_TF_expression(num3,den2);
G_sd = subs(G,s,sd);
eqn = K1*K2*abs(G_sd) == 1;
K2 = double(solve(eqn,K2))

% Find K
K = K1*K2
% Plot RL
fig6 = figure("Renderer","painters")
    rlocus(tf(K*num3,den2))
    sgrid
    title("Root Locus, Koike",'Interpreter',"latex")
    hold on
    plot(real(sd),imag(sd),'xr','MarkerSize',12)
    hold off
    sd_txt = sprintf("%0.3f+j%0.3f",real(sd),imag(sd));
    legend('System',sd_txt,"location","best")
saveas(fig6,fullfile(fdir,'P2_RL_new.png'));
```

```
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)
      title('Root Locus, Koike','interpreter','Latex')
      sgrid
end
```

```matlab
function psi = calc_ang_deficiency(num,den,s_d)
    %{
        Function:    calc_ang_deficiency
        Author:      Tomoki Koike
        Description: Computes the deficiency angle for a certain system from
                     its given open-loop transfer function and desired
                     pole.
        >>Inputs
            num: the numerator of the open-loop transfer function
            den: the denominator of the open-loop transfer function
            s_d: the desired pole
        Outputs<<
            psi: the angle deficiency
    %}

    % Get the length of each numerator and denominator
    num_len = length(num);
    den_len = length(den);

    % Preset an array with the order of magnitudes (i.e. s^3, s^2, s^1, s^0)
    % corresponding to the numerator and denominator
    O_num = (num_len-1):-1:0;
    O_den = (den_len-1):-1:0;

    % Define a system equation of s to compute deficiency angle
    syms s
    N = factor(dot(num,s.^O_num),'FactorMode','complex');
    D = factor(dot(den,s.^O_den),'FactorMode','complex');
    N_angs = angle(subs(N,s,s_d));
    D_angs = angle(subs(D,s,s_d));
    psi = -pi - sum(N_angs) + sum(D_angs);
    psi = double(rad2deg(psi));
end
```

```matlab
function output = calc_zetaFromMOS_or_MOSFromzeta(MOS_or_zeta, type)
    %{
      inputs:  1) MOS_or_zeta: maximum overshoot or zeta (damping ratio) input
                  the one of the two will be chosen depending on the second
                  input "type"
               2) type: string "MOS" or "zeta" indicates what output the
                  user requires
      outputs: 1) output: returns either the MOS or zeta
    %}
    if type == "MOS"
        zeta = MOS_or_zeta;
        output = exp(-zeta*pi/sqrt(1-zeta^2))*100;
    elseif type == "zeta"
```

```
        MOS = MOS_or_zeta;
        output = -log(MOS/100)/sqrt(pi^2 + (log(MOS/100))^2);
    end
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 [table_P, table_Z] = departure_arrival_angle_calc(zrs, poles)
    %{
       NAME:    DEPARTURE_ARRIVAL_ANGLE_CALC
       AUTHOR:  TOMOKI KOIKE
       INPUTS:  (1) zrs:   THE ZEROS OF THE TRANSFER FUNCTION
                (2) poles: THE POLES OF THE TRANSFER FUNCTION
       OUTPUTS: (1) TABLE_P : TABLE OF ALL THE POLES' DEPARTURE ANGLES
                (2) TABLE_Z : TABLE OF ALL THE ZOLES' DEPARTURE ANGLES
       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
```

```matlab
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
```