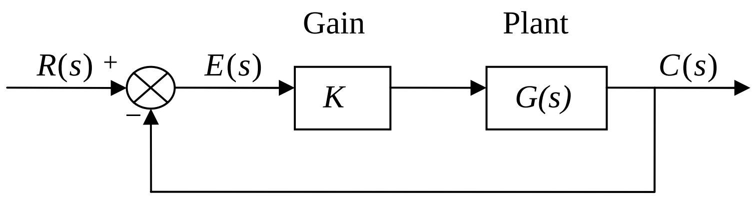
<u>Example 2</u>: Lead Compensator Design

A unit feedback system



with $KG(s) = \dfrac{K}{s(s+4)(s+6)}$ is operating with 30% overshoot ($\zeta = 0.358$).

(a) Design a compensator that will reduce the settling time by a factor of 2.

(b) Use control toolkit to simulate the uncompensated and compensated systems.

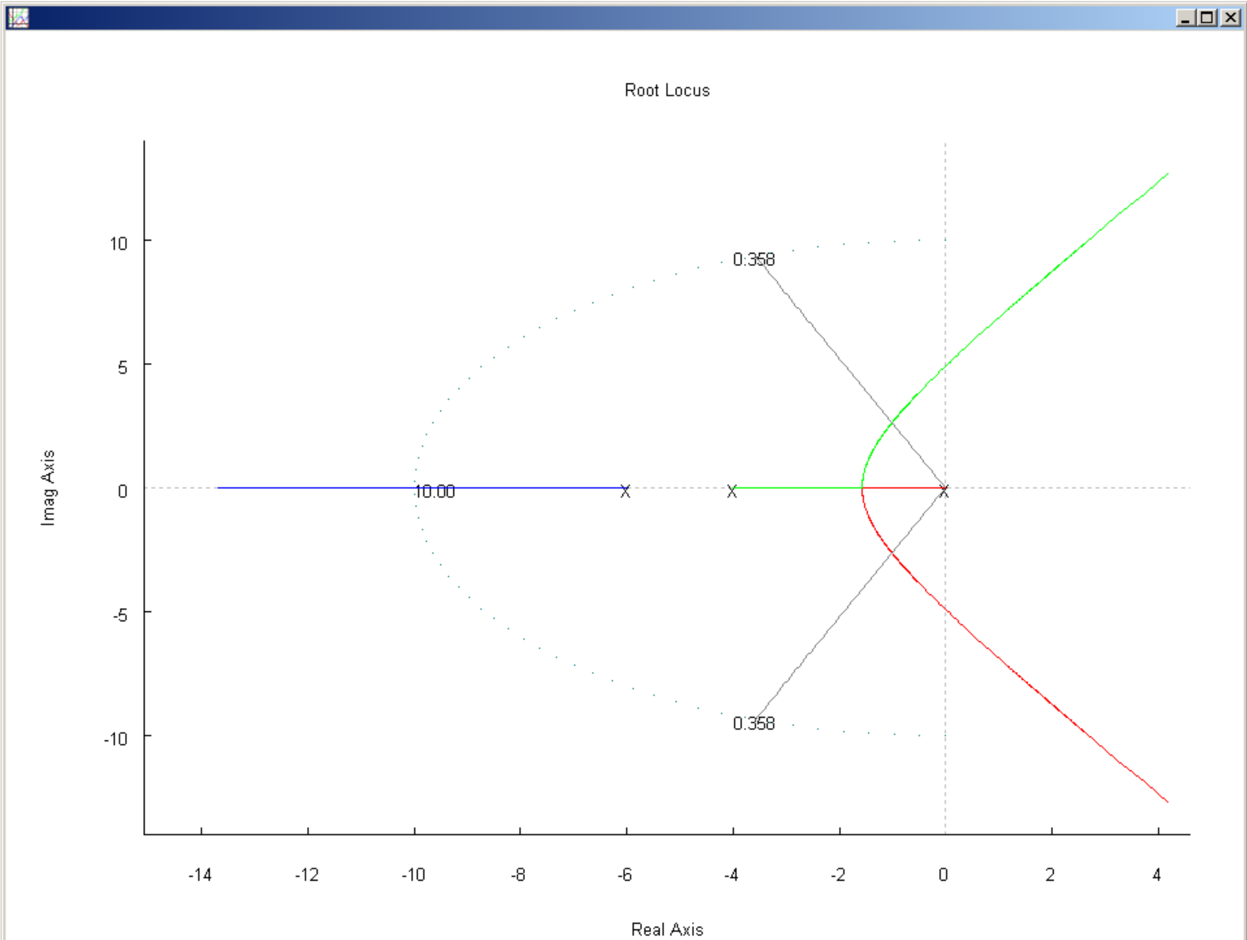
Solution:

(a)

1. Use <u>program 1</u> to plot the root locus of the ***<u>uncompensated system</u>***.

```
Program 1:

/**************************
 * File name: rl_uncomp.ch
 **************************/

#include <control.h>

int main() {
    double k = 1;
    array double complex p[3] = {complex(0, 0),
                                 complex(-4, 0),
                                 complex(-6, 0)};
    array double zeta[1] = {0.358};
    array double omega[1] = {10};
    CControl sys;
    CPlot plot;

    sys.model("zpk", NULL, p, k);
    sys.sgrid(1, zeta, omega);
    sys.rlocus(&plot, NULL, NULL);

    return 0;
}
```

```
Output:
```



2. From the root locus, measure the real and imaginary parts of the dominant poles.

The dominant poles are found to be $-1.007 \pm i2.63$.

3. Use program 2 with one of the dominant poles to find the gain K and the third pole.

The gain K and the third pole are found to be 63 and -7.987, respectively. Since the third pole is more than 5 times of the real part of the dominant poles, *the second order approximation is valid*.

Program 2:

```
/***************************
 * File name: rlf_uncomp.ch
 ***************************/

#include <control.h>

int main() {
    // default system gain
    double dk = 1;

    // uncompensated system poles
    array double complex up[3] = {complex(0, 0),
                                  complex(-4, 0),
                                  complex(-6, 0)};

    // dominant pole selected from the root locus
    array double complex dp[1] = {complex(-1.007, 2.63)};

    // closed-loop poles when one of the dominant poles is selected
    array double complex p[3];

    // system gain when one of the dominant poles is selected
    array double k[1];

    CControl sys;

    sys.model("zpk", NULL, up, dk);
    sys.rlocfind(k, p, dp);
    printf("k: %f\n", k);
    printf("poles: %f\n", p);

    return 0;
}
```

Output:

```
k: 63.321714

poles: complex(-7.987851,0.000000) complex(-1.006075,2.629651)
complex(-1.006075,-2.629651)
```

4. Find the settling time of the uncompensated system.

$$T_s = \frac{4}{\zeta\omega} = \frac{4}{1.007} = 3.972 \text{ sec.}$$

5. Find the settling time, dominant poles of the compensated system.

$$T_s' = \frac{T_s}{2} = \frac{3.972}{2} = 1.986 \text{ sec.}$$

The real part of the dominant poles is $-\zeta\omega = -\frac{4}{T_s'} = -2.014$.

The imaginary part of the dominant poles is $\omega_d = \frac{\zeta\omega}{\tan\left(\sin^{-1}\zeta\right)} = \frac{2.014}{\tan\left(\sin^{-1}0.358\right)} = 5.252$.

6. Select the compensator zero to be -5, which means $z_c = 5$.

The compensated plant is $L(s) = \frac{(s+z_c)}{(s+p_c)s(s+4)(s+6)}$.

To the dominant poles, $\angle L(s) = \theta_{zc} - \theta_{pc} - \theta_1 - \theta_2 - \theta_3 = (2k+1)180^\circ \quad (k = 0, \pm 1, \pm 2, ...)$.

Choose $k = -1$, $\theta_{pc} = 180^\circ + \theta_{zc} - \theta_1 - \theta_2 - \theta_3$.

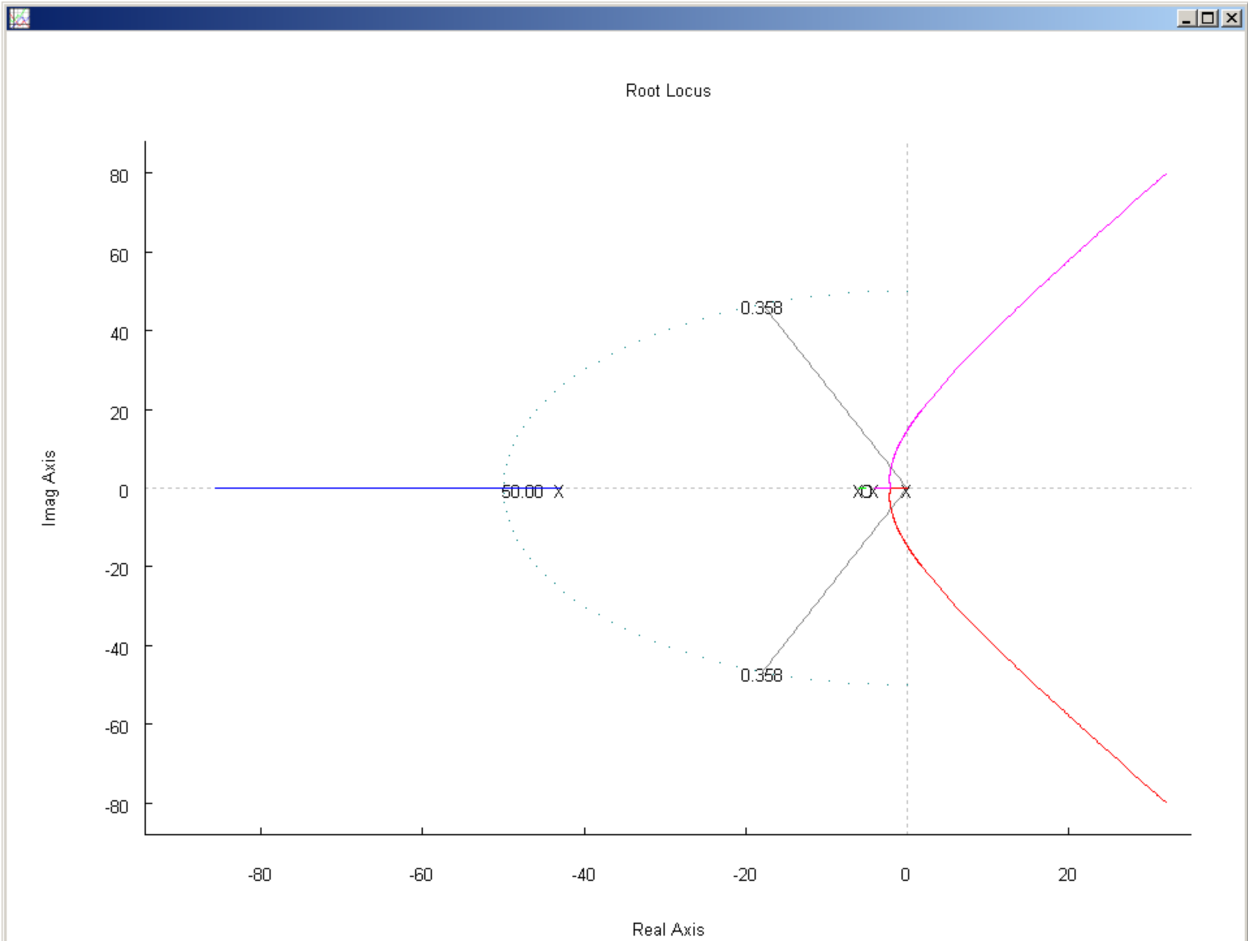
Choose the dominant pole at $-2.014 + i5.252$, θ_{pc} can be found to be 7.31°.

$\frac{5.252}{p_c - 2.014} = \tan(\theta_{pc}) = \tan(7.31^\circ)$, $p_c = 42.96$, whch means the compensator pole is -42.96.

7. Use <u>program 3</u> to plot the root locus of the <u>***compensated system***</u>.

```
Program 3:

/**************************
 * File name: rl_comp.ch
 **************************/

#include <control.h>

int main() {
    double k = 1;
    array double complex z[1] = {complex(-5, 0)};
    array double complex p[4] = {complex(-42.96, 0),
                                 complex(0, 0),
                                 complex(-4, 0),
                                 complex(-6, 0)};
    array double zeta[1] = {0.358};
    array double omega[1] = {50};
    CControl sys;
    CPlot plot;

    sys.model("zpk", z, p, k);
    sys.sgrid(1, zeta, omega);
    sys.rlocus(&plot, NULL, NULL);

    return 0;
}
```

Output:



Root Locus
Imag Axis
Real Axis
80
60
40
20
0
-20
-40
-60
-80
-80
-60
-40
-20
0
20
0.358
0.358
50.00

8. Use program 4 with one of the dominant poles to find the gain K and other closed-loop poles. The gain K is found to be 1423. The third and fourth poles are found to be -43.8 and -5.134, respectively. Since the third pole at -43.8 is more than 20 times of the real part of the dominant poles, the effect of the third pole is negligible. Since the fourth pole at -5.134 is close to the zero at -5, the pole-zero cancellation stands. As a result, *the second order approximation is valid*.

```
Program 4:

/***************************
 * File name: rlf_comp.ch
 ***************************/

#include <control.h>

int main() {
    // default system gain
    double dk = 1;

    // compensated system zero
    array double complex cz[1] = {complex(-5, 0)};

    // compensated system poles
    array double complex cp[4] = {complex(-42.96, 0),
                                  complex(0, 0),
                                  complex(-4, 0),
                                  complex(-6, 0)};

    // dominant pole selected from the root locus
    array double complex dp[1] = {complex(-2.014, 5.252)};

    // closed-loop poles when one of the dominant poles is selected
    array double complex p[4];

    // system gain when one of the dominant poles is selected
    array double k[1];

    CControl sys;

    sys.model("zpk", cz, cp, dk);
    sys.rlocfind(k, p, dp);
    printf("k: %f\n", k);
    printf("poles: %f\n", p);

    return 0;
}

Output:

k: 1422.905212

poles: complex(-43.797920,0.000000) complex(-2.014025,5.252003)
complex(-5.134029,0.000000) complex(-2.014025,-5.252003)
```

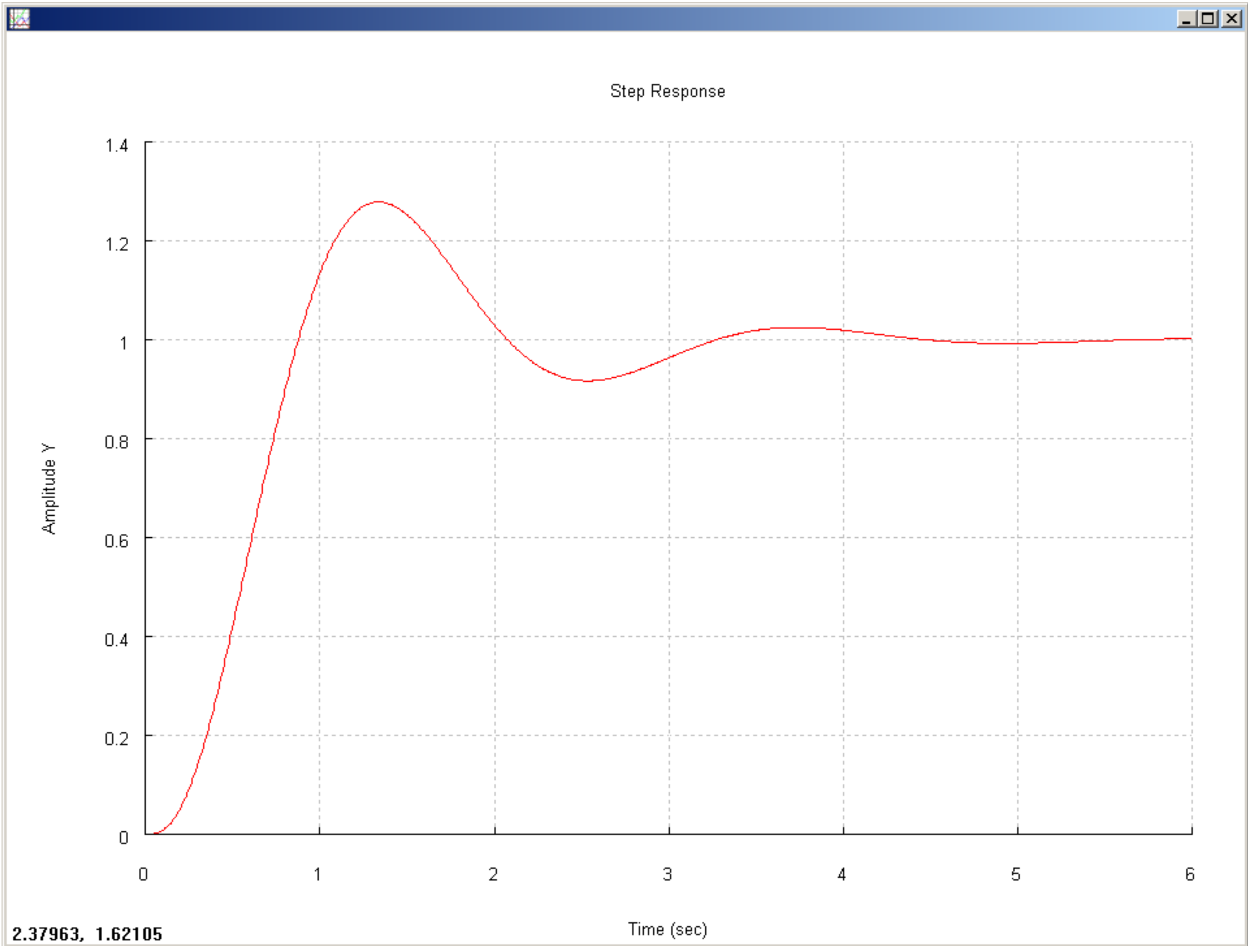
(b)

1. Use program 5 to obtain the step response of the ***uncompensated system***.

Program 5:

```
/****************************
 * File name: step_uncomp.ch
 ****************************/

#include <control.h>

int main() {
    // system gain
    double k = 63;

    // uncompensated system poles
    array double complex up[3] = {complex(0, 0),
                                  complex(-4, 0),
                                  complex(-6, 0)};
    array double num[1] = {1};
    array double den[1] = {1};
    double tf = 6;

    CControl sys1, sys2, *sys3;
    CPlot plot;

    sys1.model("zpk", NULL, up, k);
    sys2.model("tf", num, den);
    sys3 = sys1.feedback(&sys2);
    sys3->grid(1);
    sys3->step(&plot, NULL, NULL, NULL, tf);

    return 0;
}
```

Output:

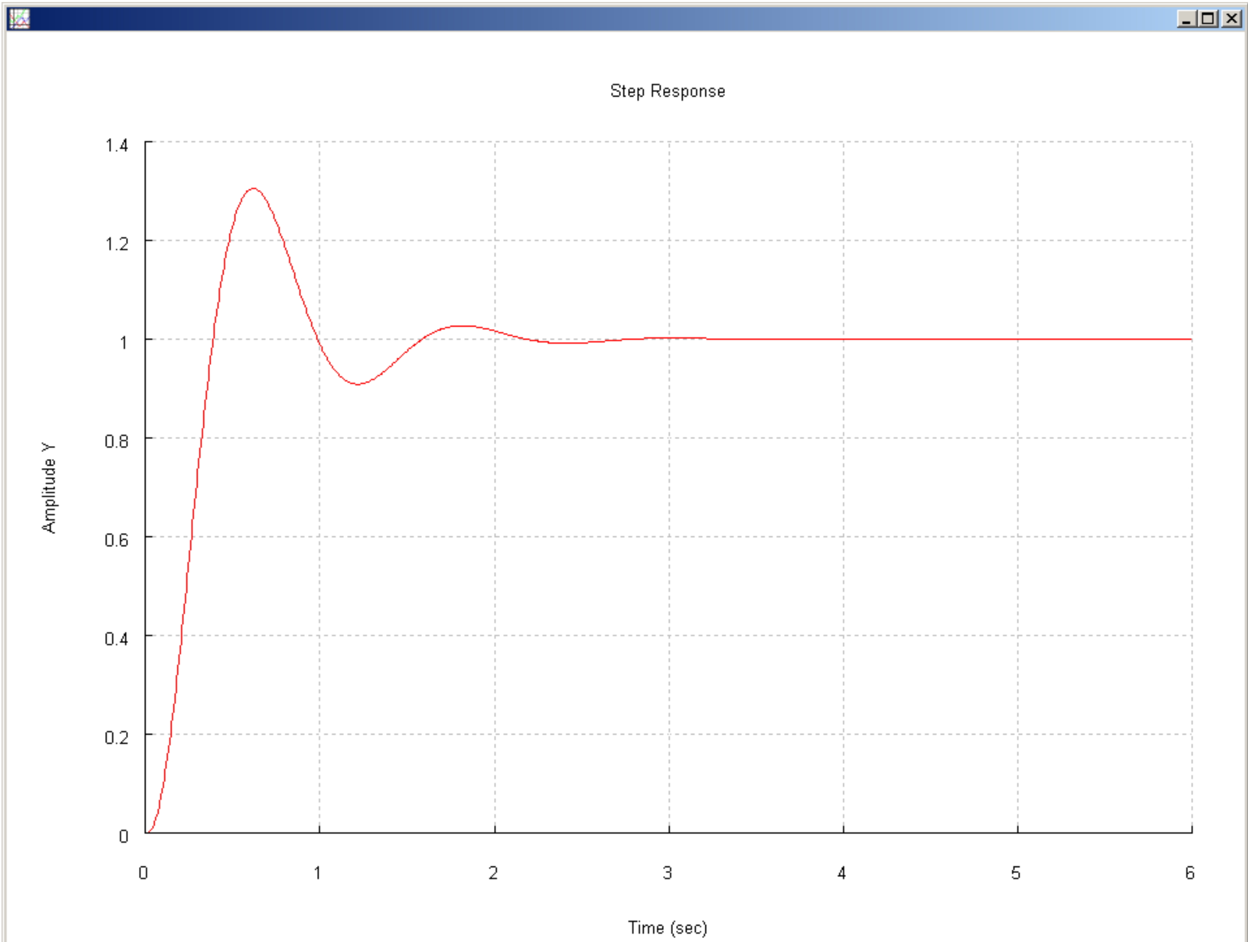

Step Response
Amplitude Y
Time (sec)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
6
2.37963, 1.62105

2. Use program 6 to obtain the step response of the ***compensated system***.

Program 6:

```
/****************************
 * File name: step_comp.ch
 ****************************/

#include <control.h>

int main() {
    // system gain
    double k = 1423;

    // compensated system zero
    array double complex cz[1] = {complex(-5, 0)};

    // compensated system poles
    array double complex cp[4] = {complex(-42.96, 0),
                                  complex(0, 0),
                                  complex(-4, 0),
                                  complex(-6, 0)};
    array double num[1] = {1};
    array double den[1] = {1};
    double tf = 6;

    CControl sys1, sys2, *sys3;
    CPlot plot;

    sys1.model("zpk", cz, cp, k);
    sys2.model("tf", num, den);
    sys3 = sys1.feedback(&sys2);
    sys3->grid(1);
    sys3->step(&plot, NULL, NULL, NULL, tf);

    return 0;
}
```

Output:


Step Response
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Amplitude Y
0
1
2
3
4
5
6
Time (sec)