

**PES UNIVERSITY**

**(Established under Karnataka Act No. 16 of 2013)**

**100 Feet Ring Road, BSK III Stage, Bengaluru-560 085**

**Department of Electronics and Communications Engineering**

Title:

Temperature Control using

PID Controller

Team member details:

PES1UG21EC071 - C P Sindhu

PES1UG21EC072 - Carol Miriam Joby

PES1UG21EC098 - G Tanushree

Course details:

Course name: Control Systems

Course code: UE21EC241B

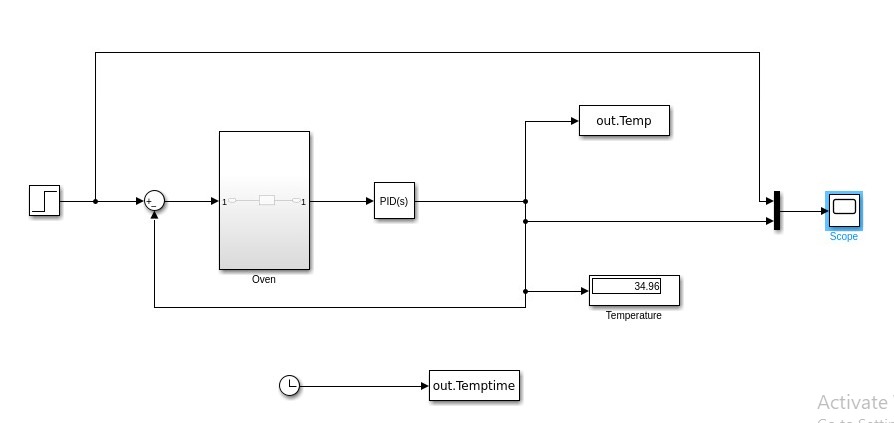
**ABSTRACT:**

A PID controller is a popular feedback control mechanism used for temperature control in various industrial, commercial, and domestic applications. It is a closed-loop control system that continuously receives feedback from temperature sensors and adjusts the input signal to minimize the error between the desired temperature and the actual temperature of the system.

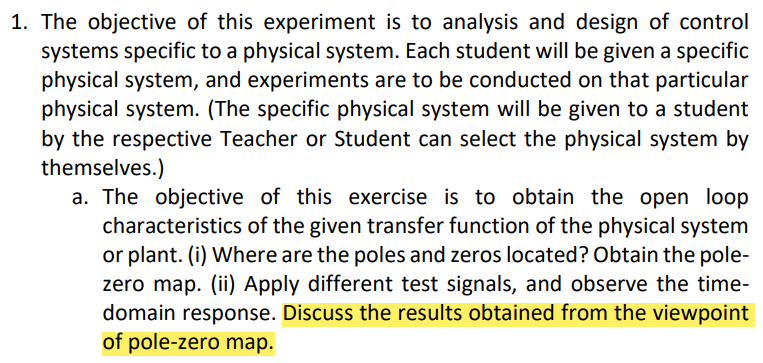
The PID controller comprises three main terms: proportional, integral, and derivative. The proportional term is proportional to the error between the desired temperature and the actual temperature of the system. The integral term accumulates the error over time to eliminate any steady-state error. The derivative term is proportional to the rate of change of the error and helps to reduce overshoot and oscillations in the system.

PID controllers are widely used for temperature control in various applications, including HVAC (Heating, Ventilation, and Air Conditioning) systems, industrial processes, and laboratory equipment. By providing accurate and responsive control, PID controllers play a crucial role in ensuring that these systems operate efficiently and reliably, while maintaining the desired temperature setpoint.

**MECHANICAL MODELING:**



**QUESTION 1:**

****

**CODE:**

clc;

clear all;

close all;

s=tf("s");

sys=tf([0 0 1],[1 1.2 4]);

disp("Poles:")

disp(pole(sys))

disp("Zeros:")

disp(zero(sys))

figure;

pzmap(sys);

grid on;

figure;

subplot(221)

impulse(sys);

subplot(222)

step(sys);

title('Step response');

subplot(223)

step(sys/s);

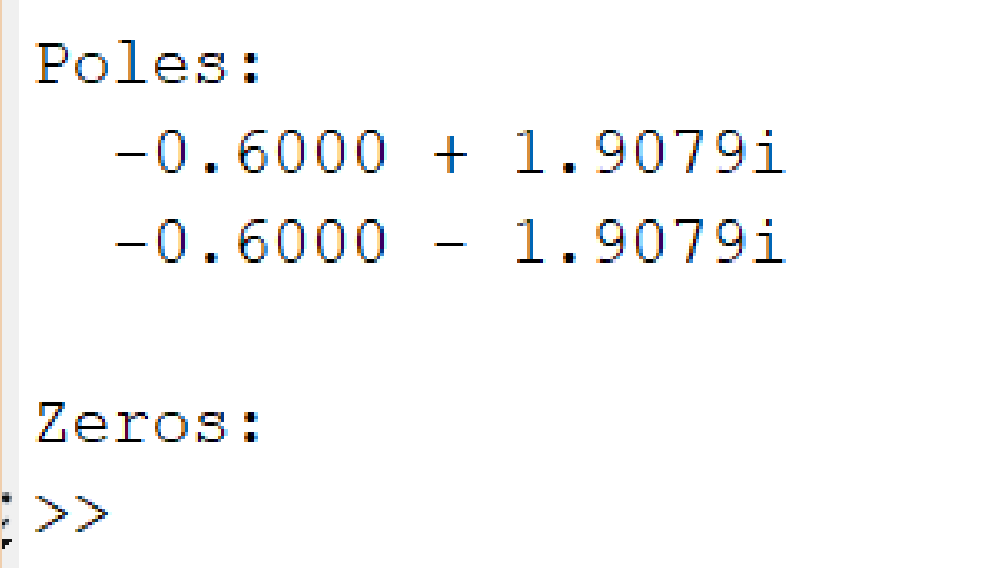
title('Ramp response');

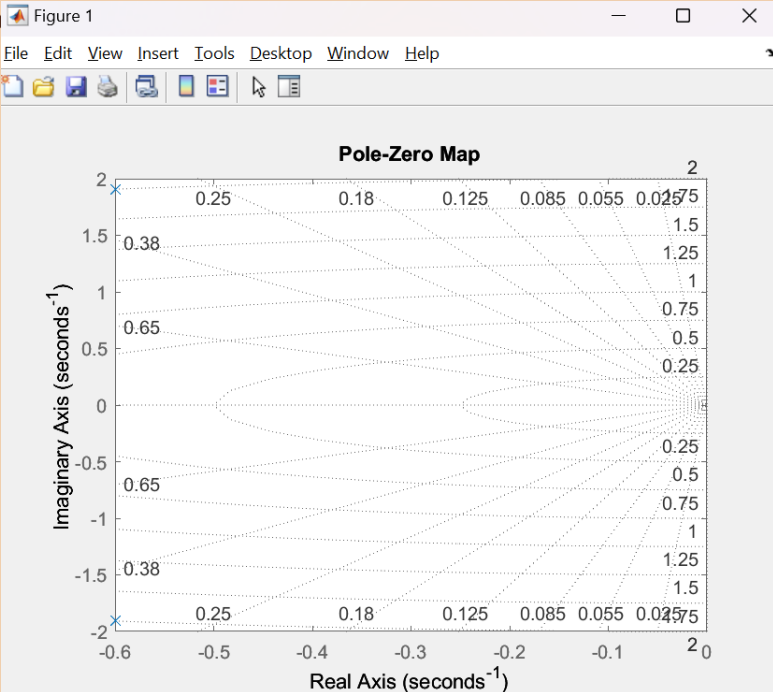
subplot(224)

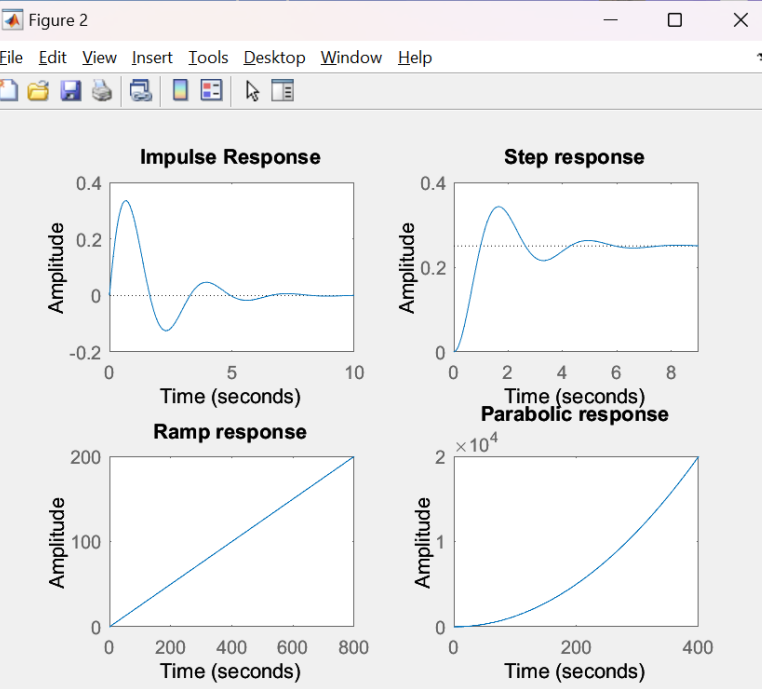
step(sys/s^2);%parabolic

title('Parabolic response');

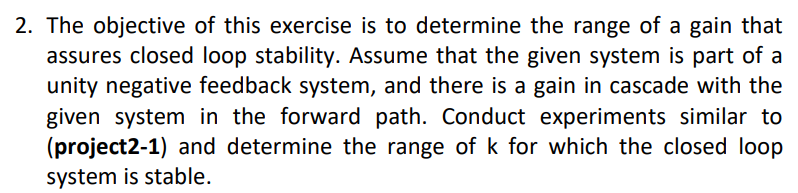
**OUTPUT:**







**QUESTION 2:**



**CODE:**

clc;

clear all;

close all;

%proj32

for k = -20:10:50

sys = tf([0 0 k],[1 1.2 4])

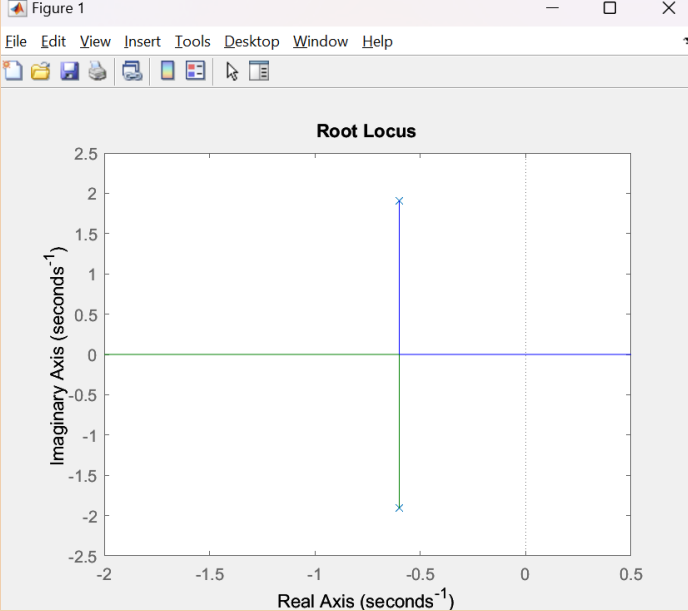
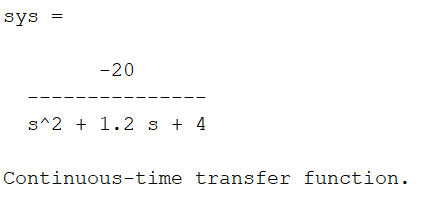
figure;

rlocus(sys)

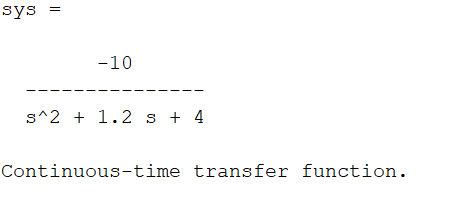
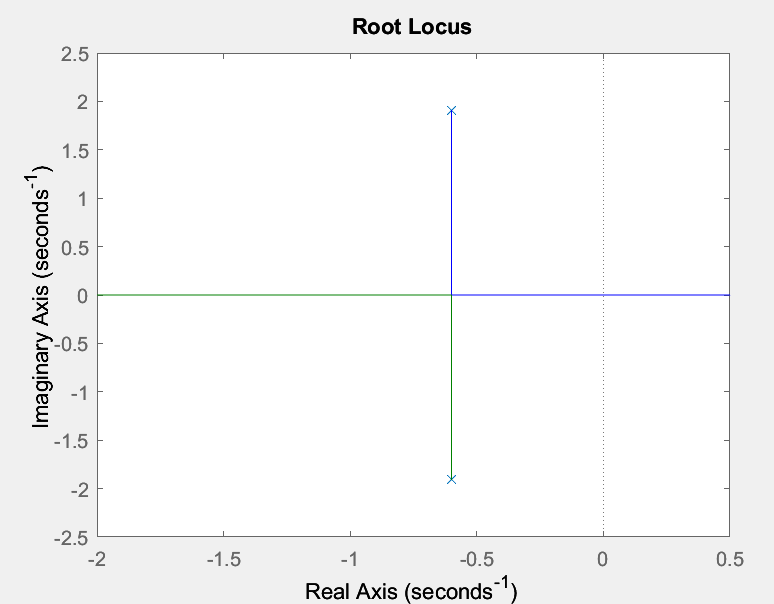
end

**OUTPUT:**

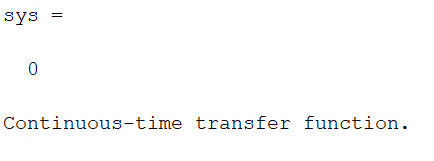
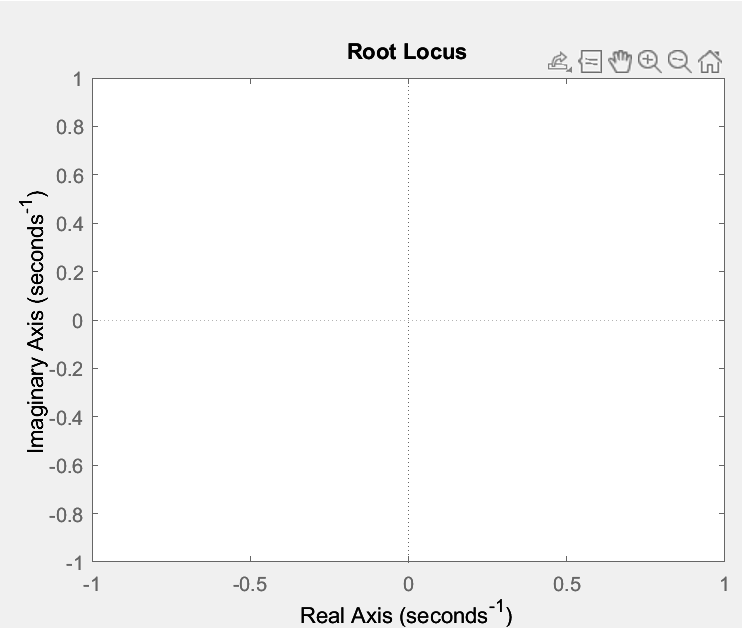
**Gain=-20, Unstable System**

****

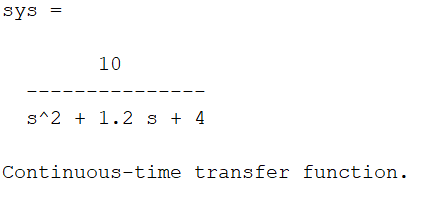
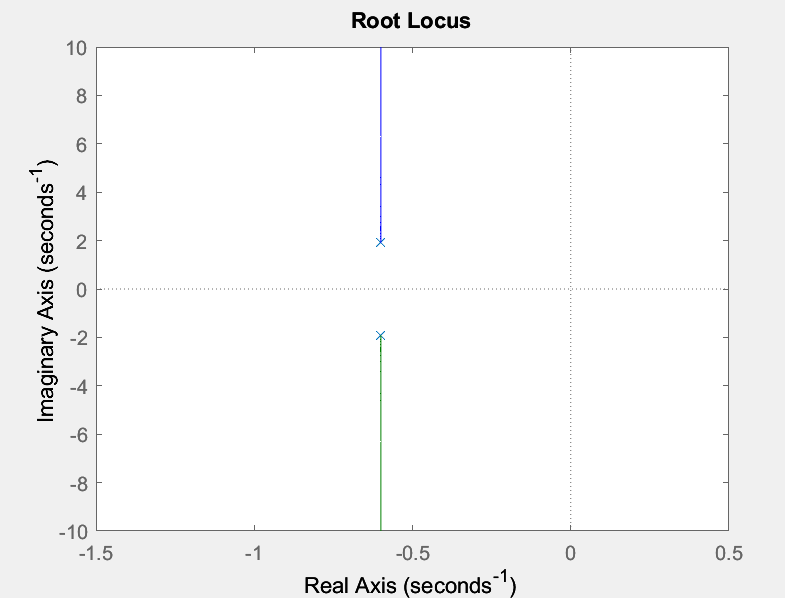
**Gain=-10, Unstable System**

**** ****

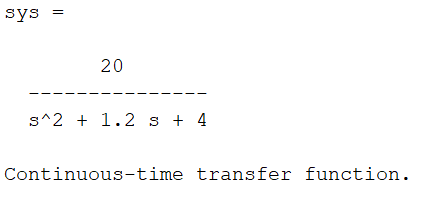
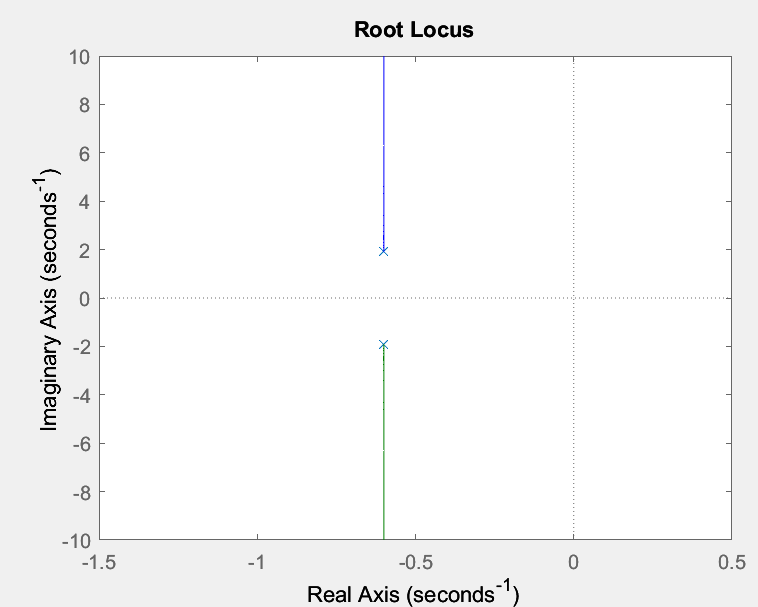
**Gain=0**

****

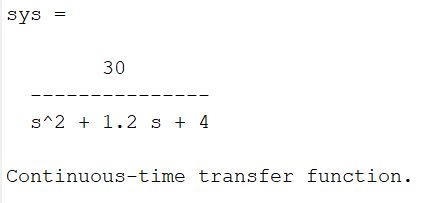
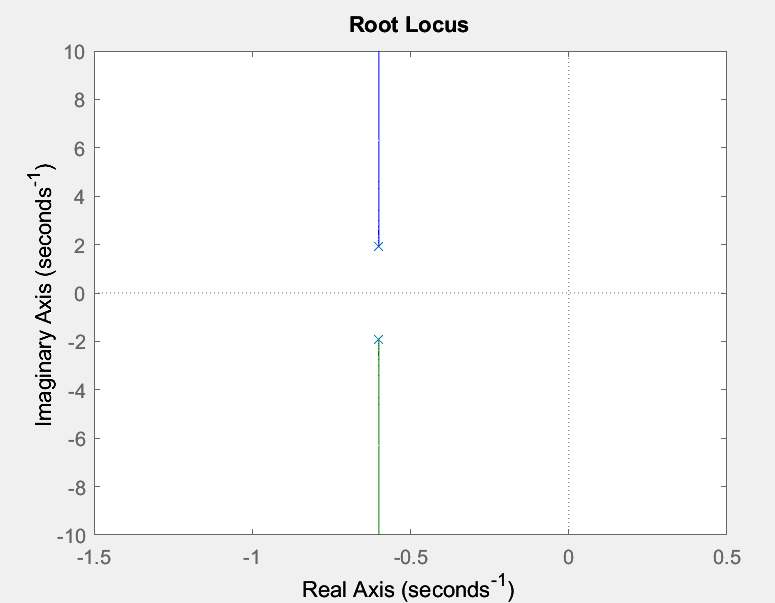
**Gain=10, Stable System**

**** ****

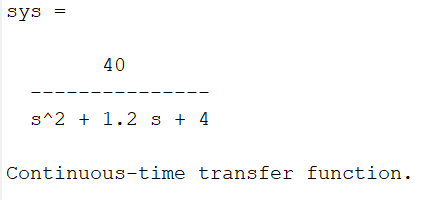
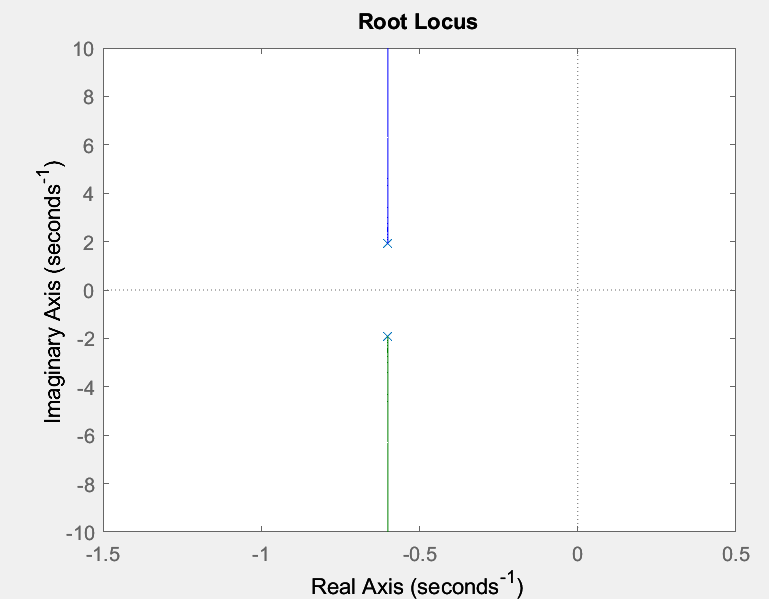
**Gain=20, Stable System**

**** ****

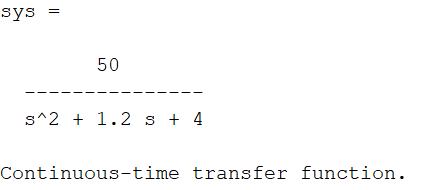
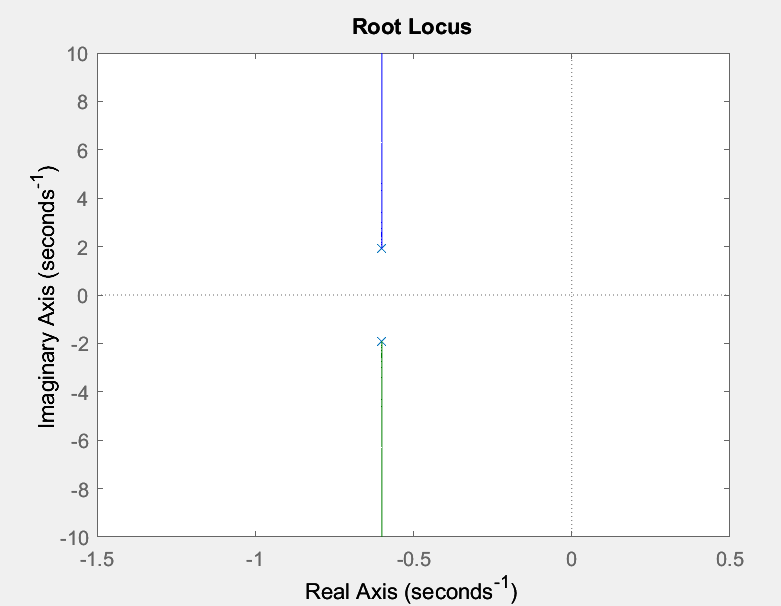
**Gain=30, Stable System**

**** ****

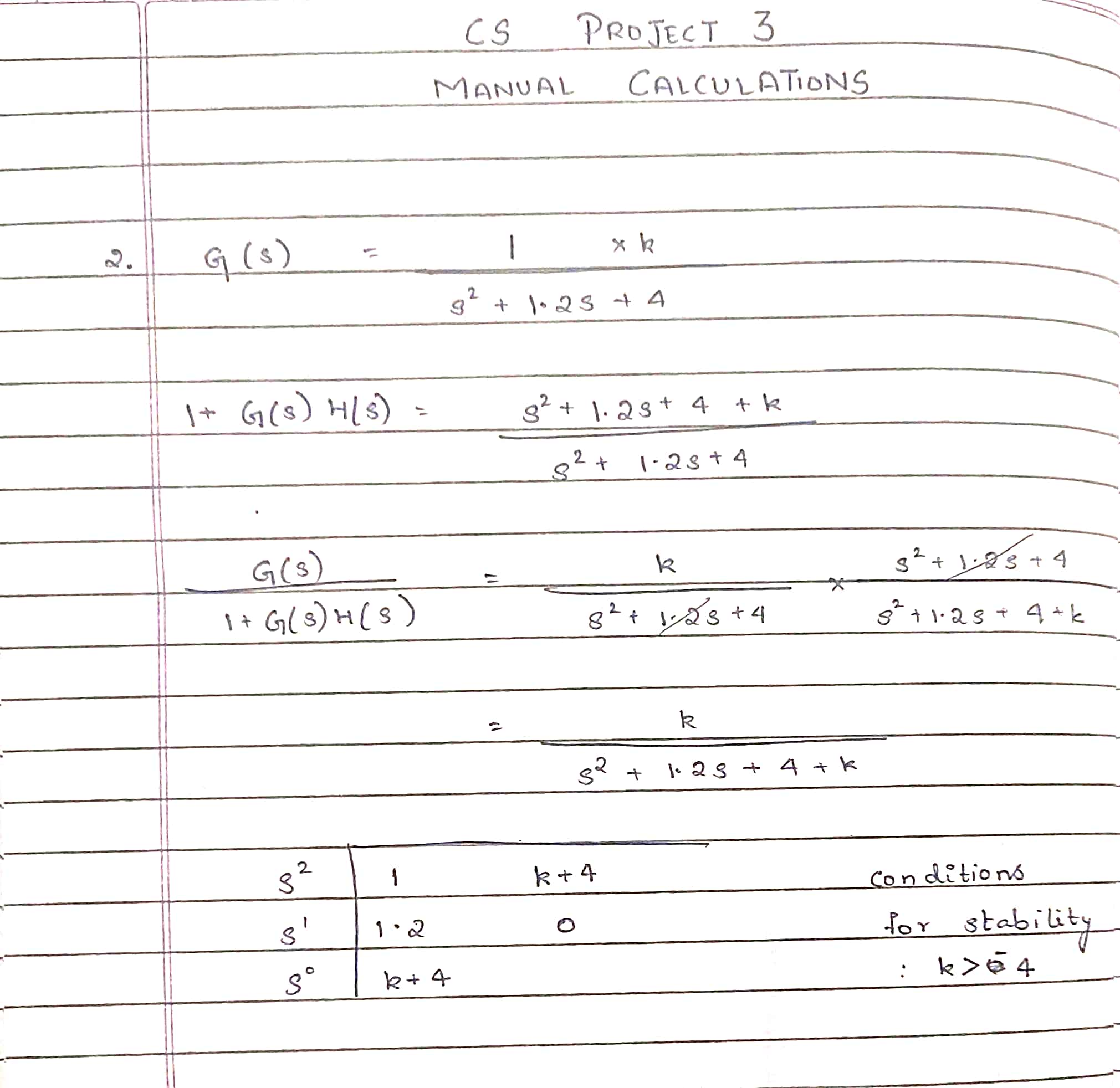
**Gain=40, Stable System**

**** ****

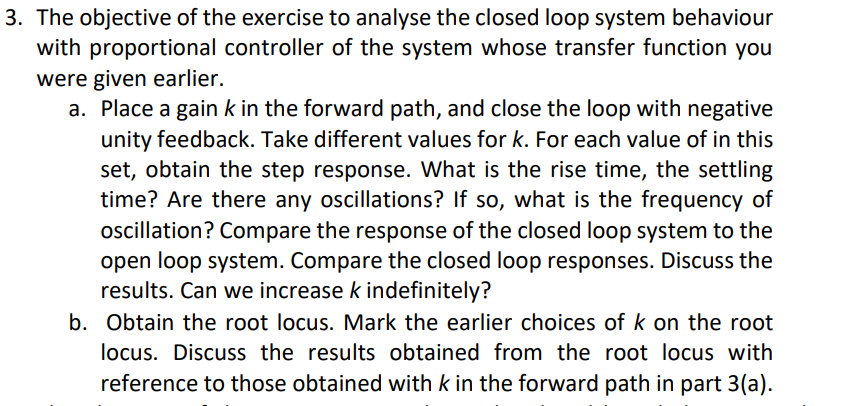
**Gain=50, Stable System**

**** ****

**MANUAL CALCULATIONS:**



**QUESTION 3:**



**CODE:**

clc;

clear all;

close all;

%proj33

for k = -20:10:50

sys = tf([0 0 k],[1 1.2 4])

figure;

rlocus(sys)

sys2=sys/(1+sys)

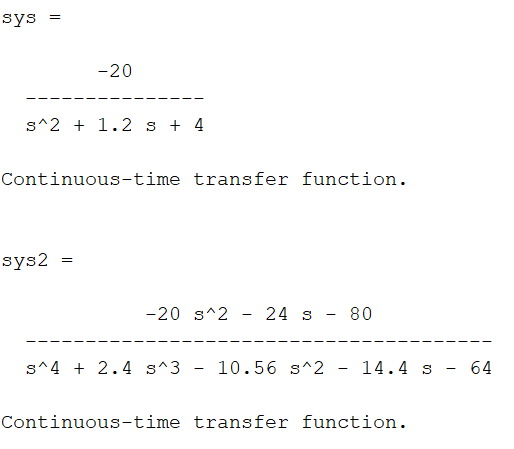
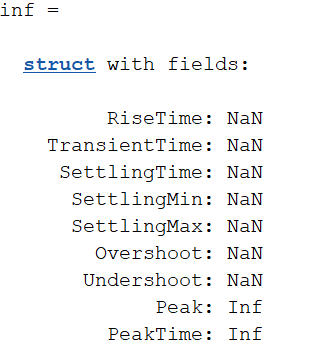
inf=stepinfo(sys2)

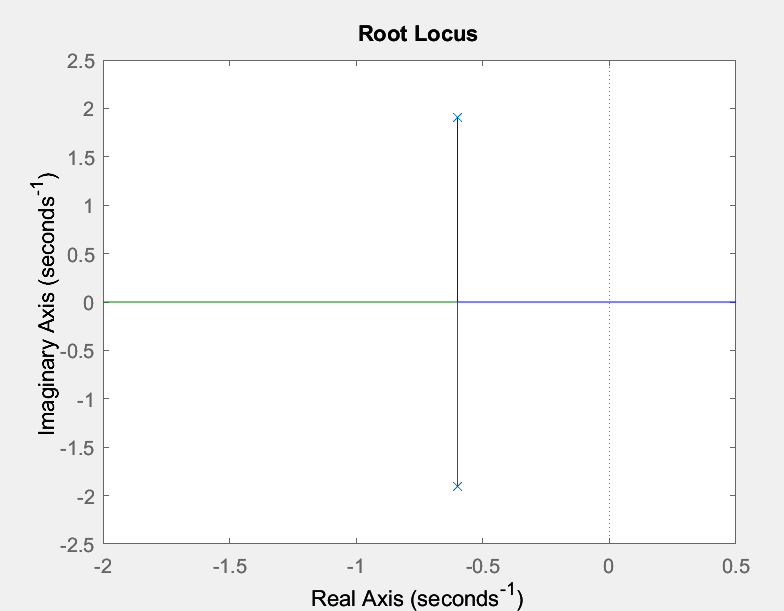
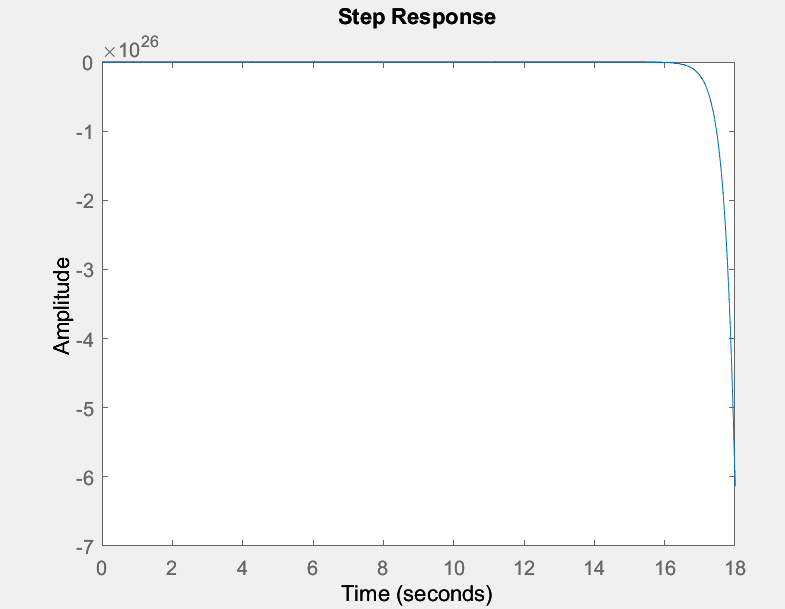
figure;

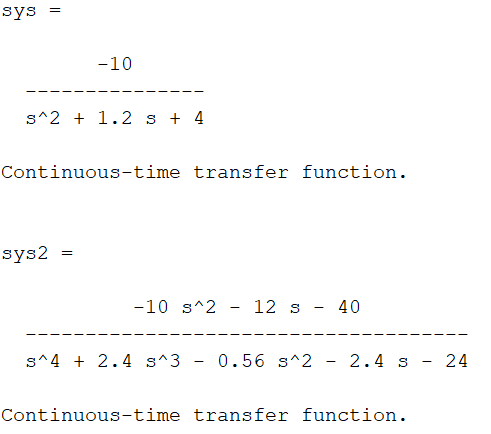
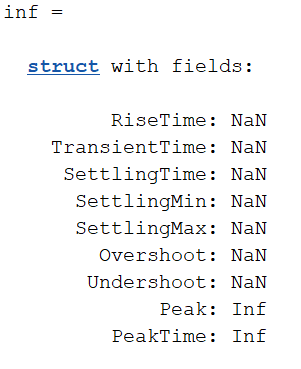
step(sys2)

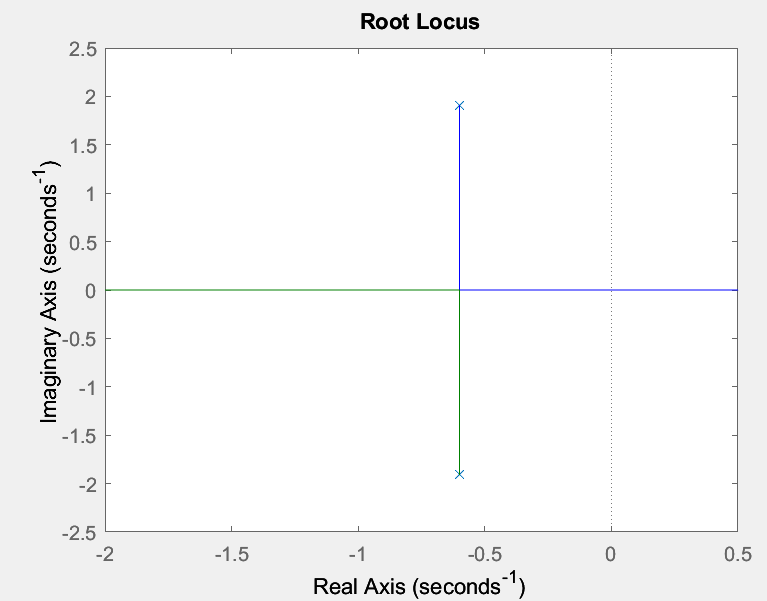
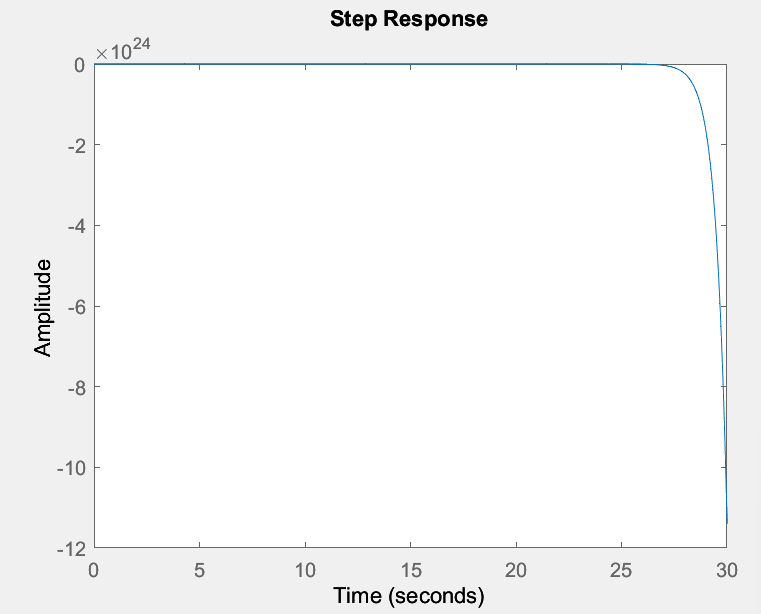
end

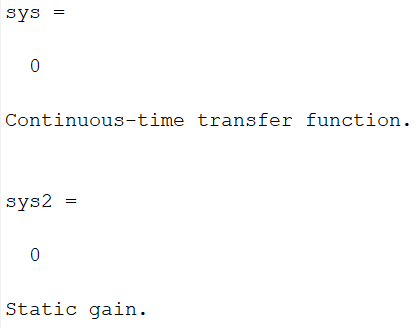
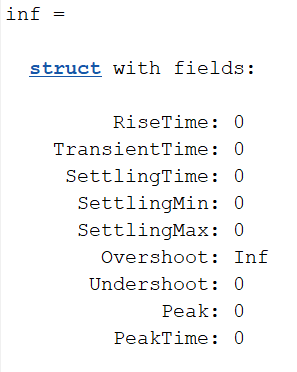
**OUTPUT:**

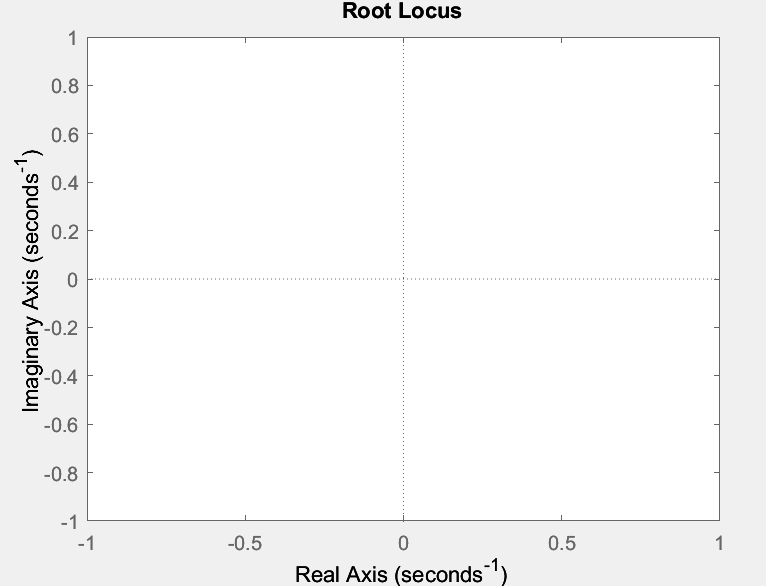
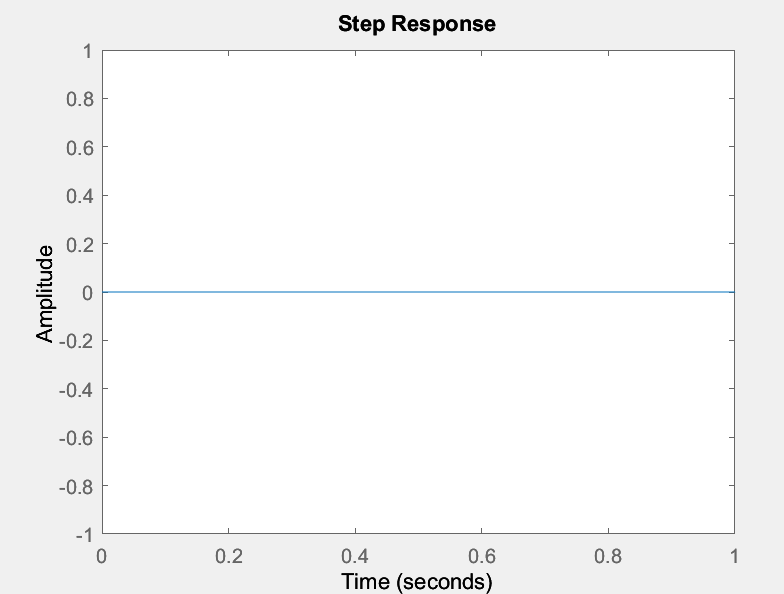
**** ****

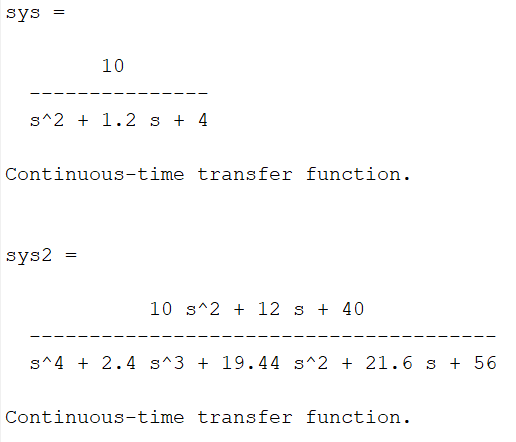
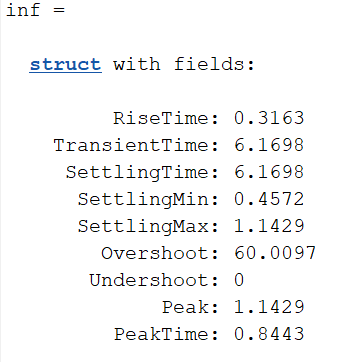
 

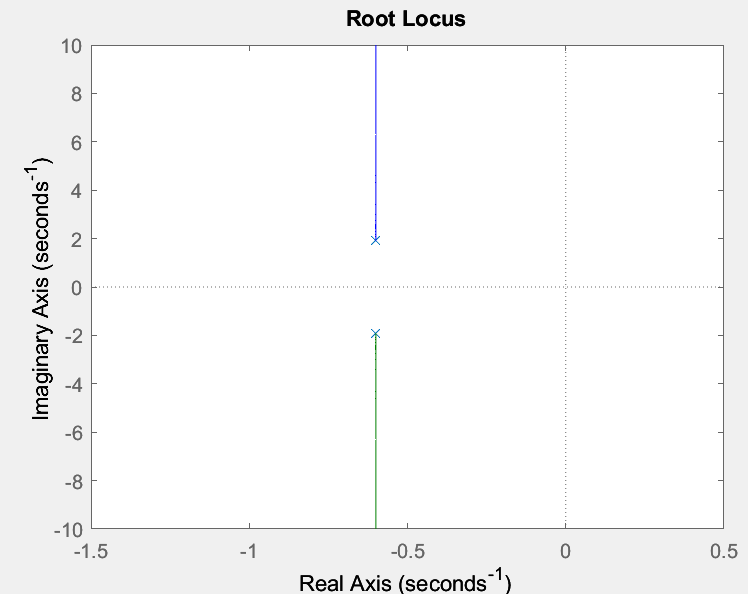
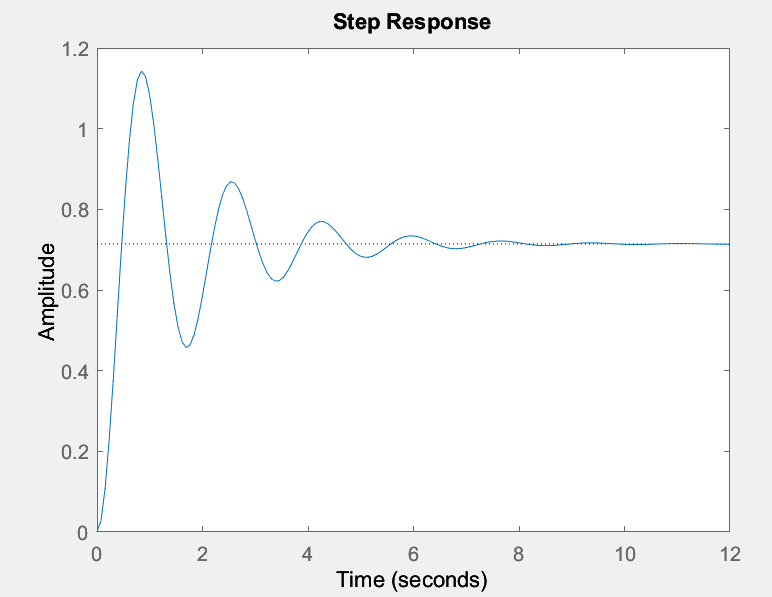
**** ****

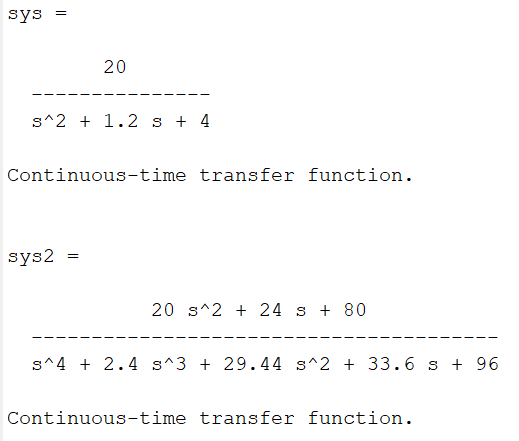
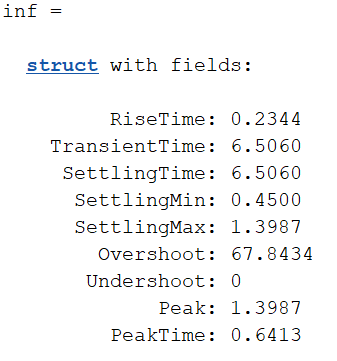
 

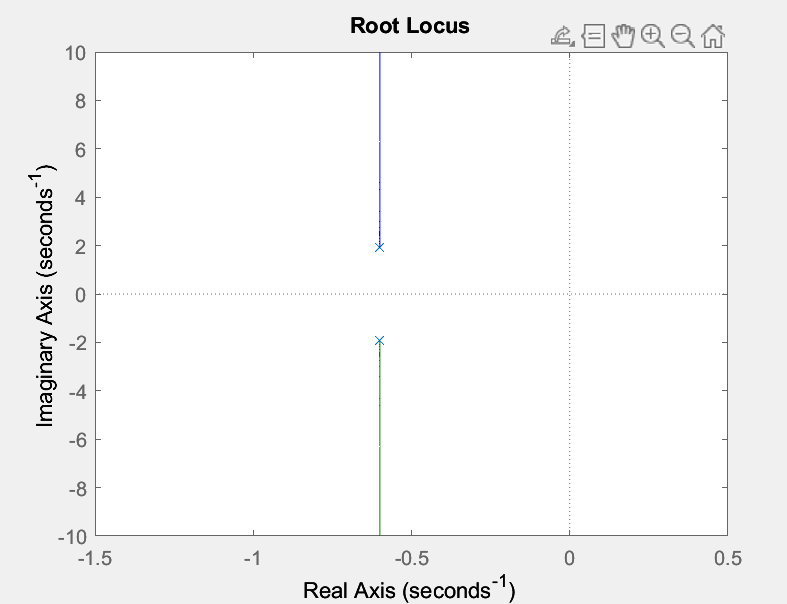
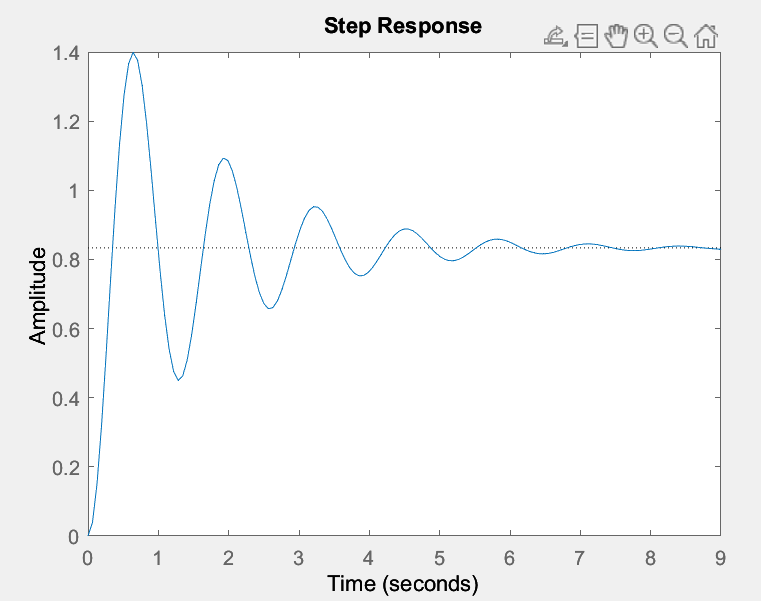
**** ****

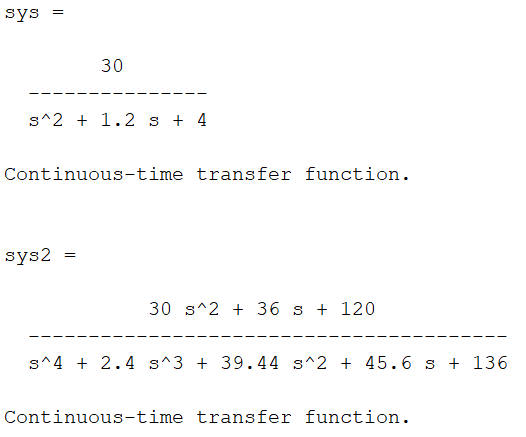
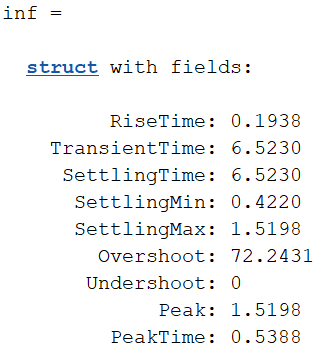
 

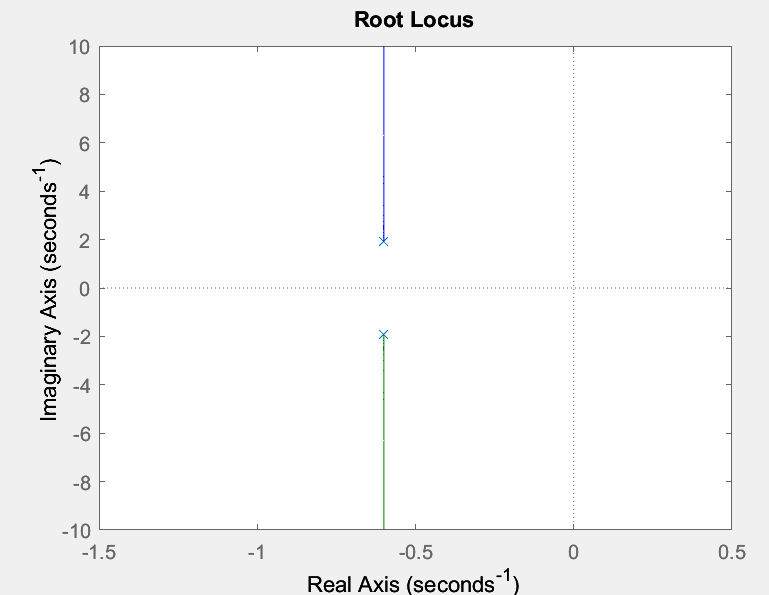
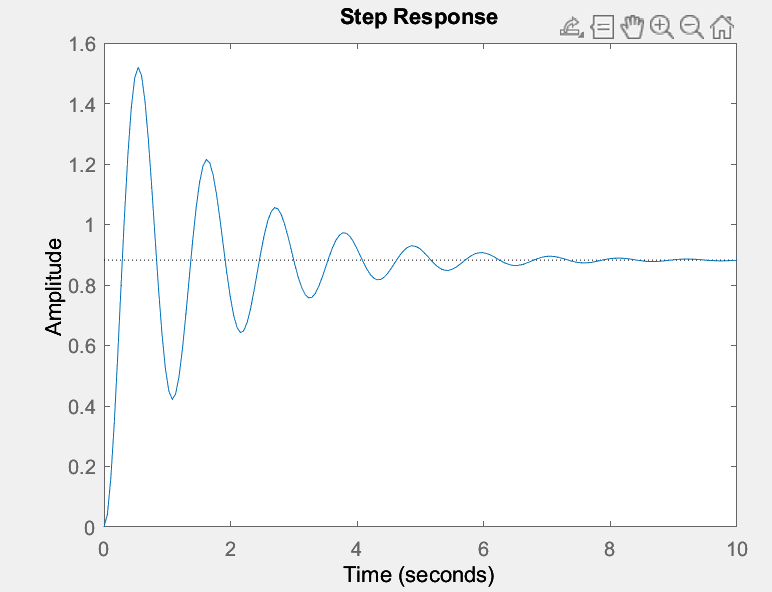
**** ****

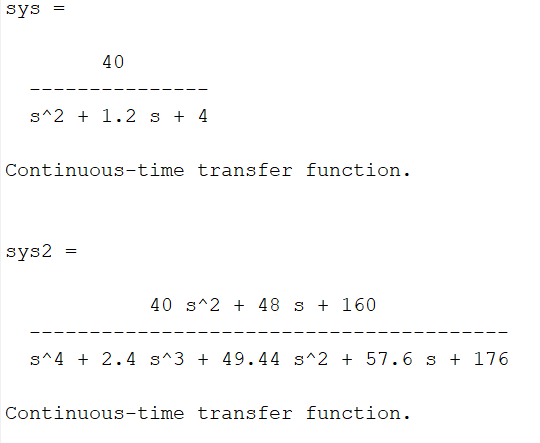
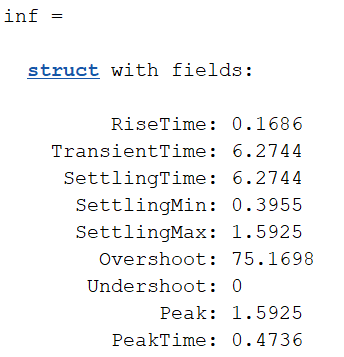
 

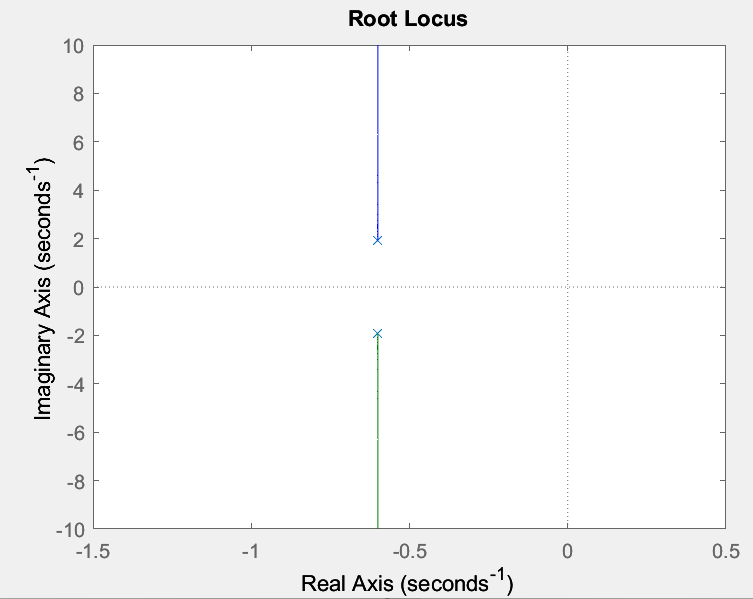
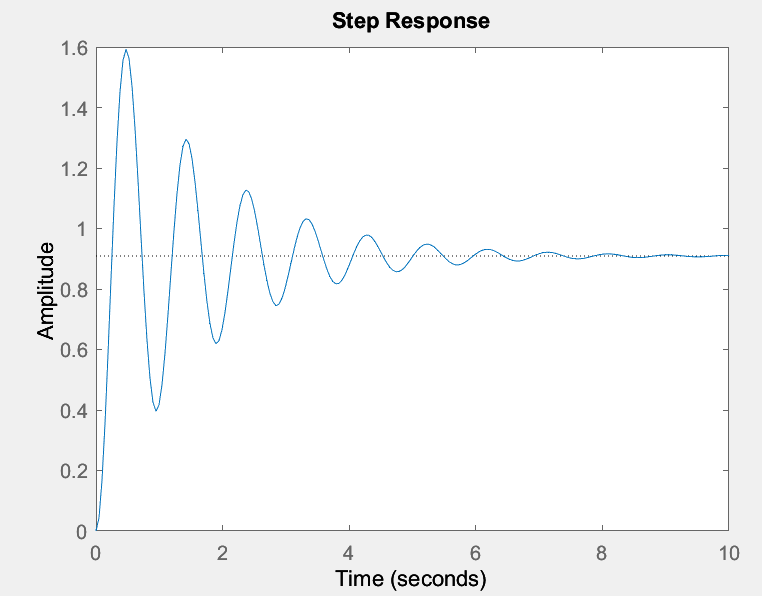
**** ****

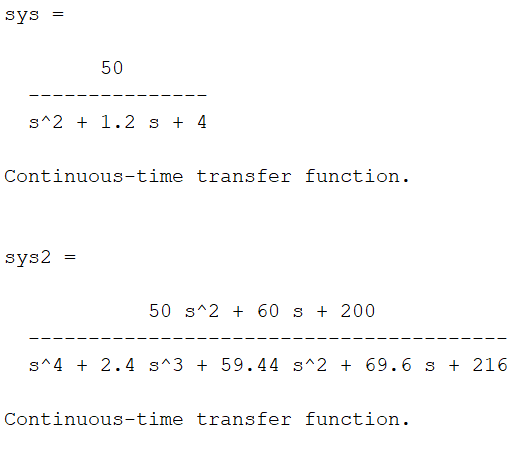
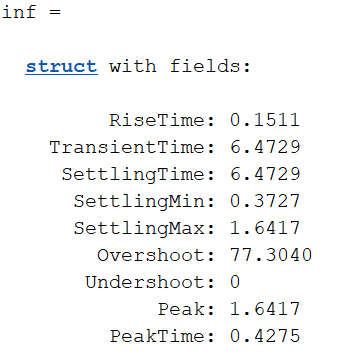
 

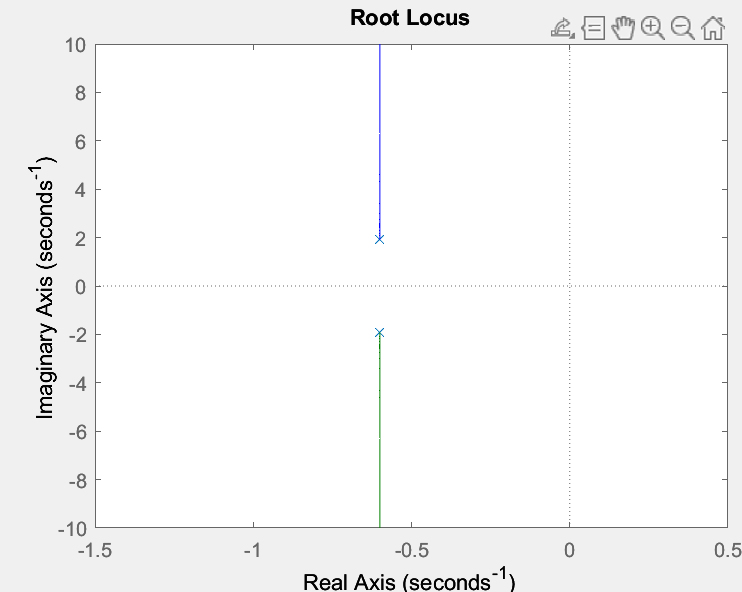
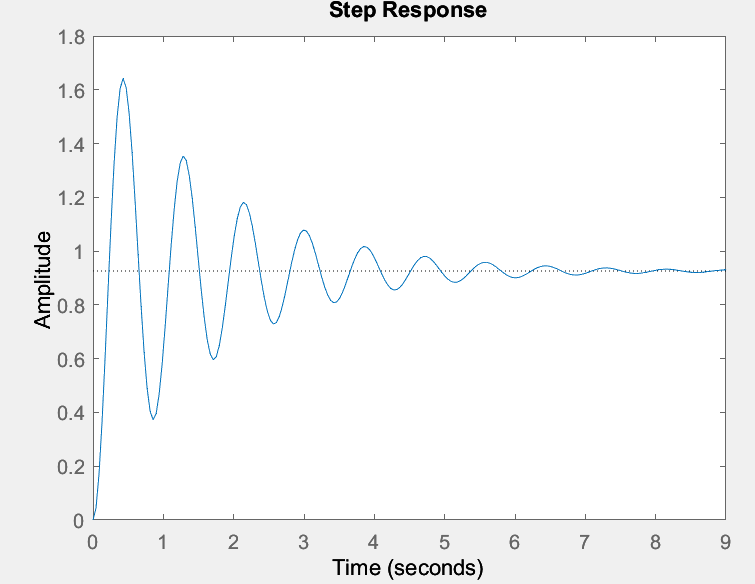
**** ****

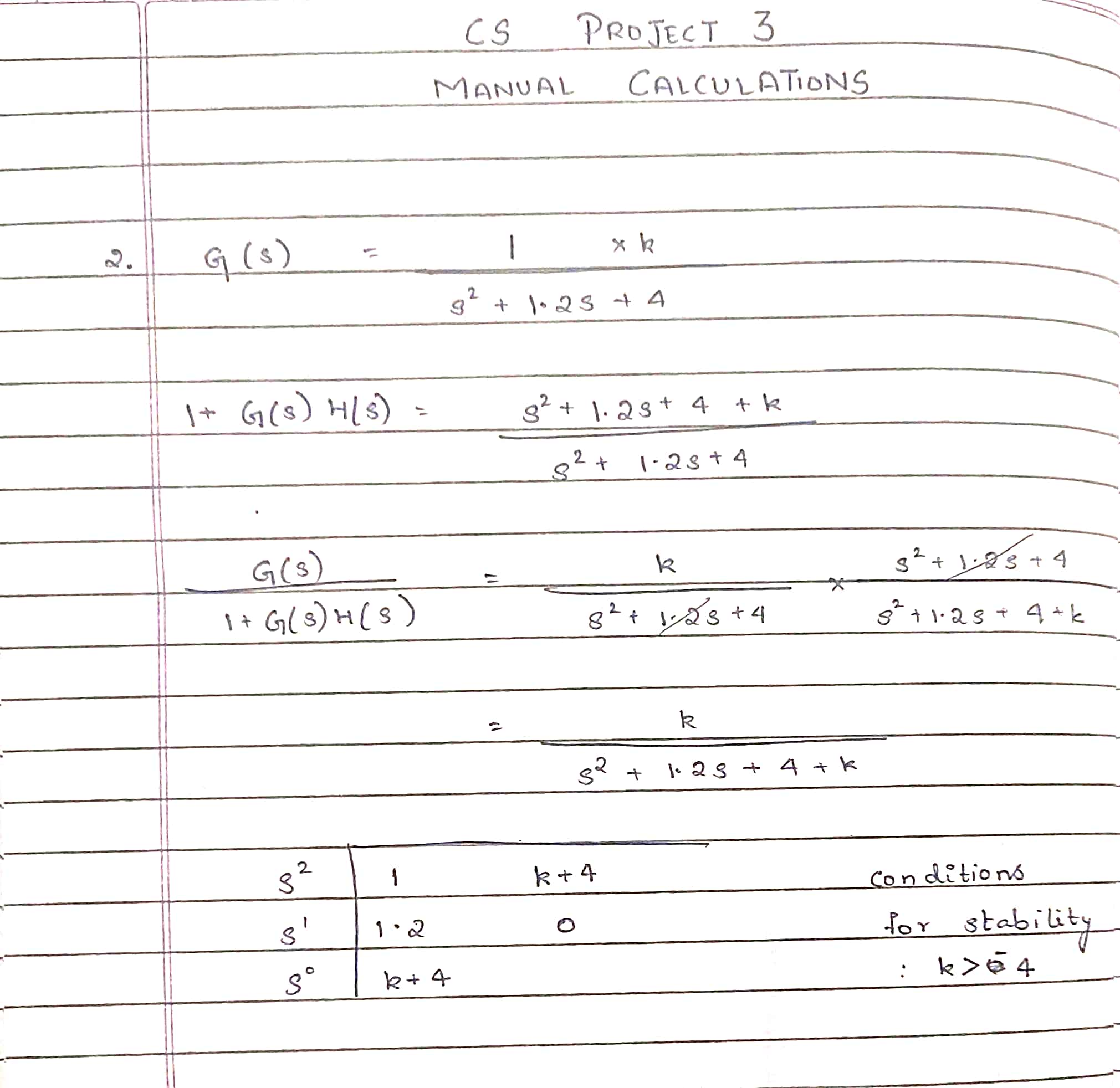
**** ****

**** ****

**** ****

**MANUAL CALCULATIONS:**



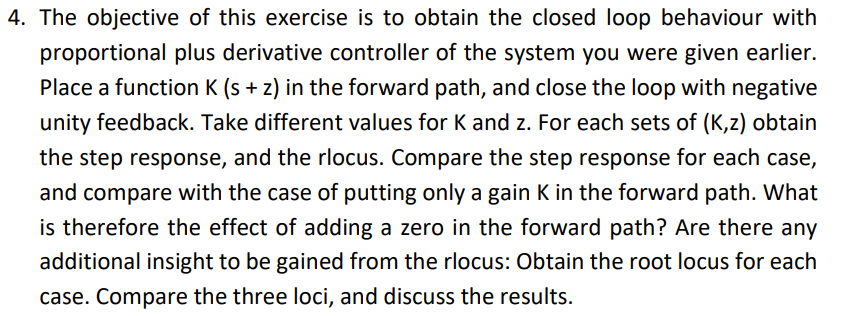
**INFERENCE:**

* Regarding response, the closed-loop system has a faster response time and a smaller steady-state error than the open-loop system.
* No, the gain parameter, k, cannot be increased indefinitely in a closed-loop system response because it might induce system instability.
* The root locus plot demonstrates how the positions of the closed-loop poles in the complex plane change as the gain parameter, k, is adjusted.
* The closed-loop poles on the root locus plot move as k rises until they either approach one another or move into the right half of the complex plane, indicating instability.

**TABLE:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P (gain) | Rise time | Settling time | Are there any oscillations? | Stability |
| -20 | NaN | NaN | No | Unstable |
| -10 | NaN | NaN | No | Unstable |
| 10 | 0.3163 | 6.1698 | Yes | Stable |
| 20 | 0.2344 | 6.5060 | Yes | Stable |
| 30 | 0.1938 | 6.5230 | Yes | Stable |
| 40 | 0.1686 | 6.2744 | Yes | Stable |
| 50 | 0.1511 | 6.4729 | Yes | Stable |

**QUESTION 4:**



**CODE:**

clc;

clear all;

close all;

%csp34

for k = -0.04:0.02:0.04

for z=-1:1:1

%a = tf([k,k\*z],[1]);

sys = tf([0 k k\*z],[1 1.2 4]);

%b=a\*sys;

figure;

subplot(121);

rlocus(sys)

title(k);

subtitle(k\*z);

sys2=sys/(1+sys);

inf=stepinfo(sys2)

subplot(122)

step(sys2)

title(k);

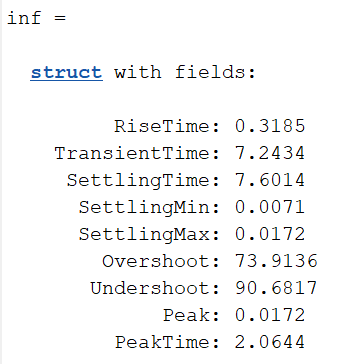
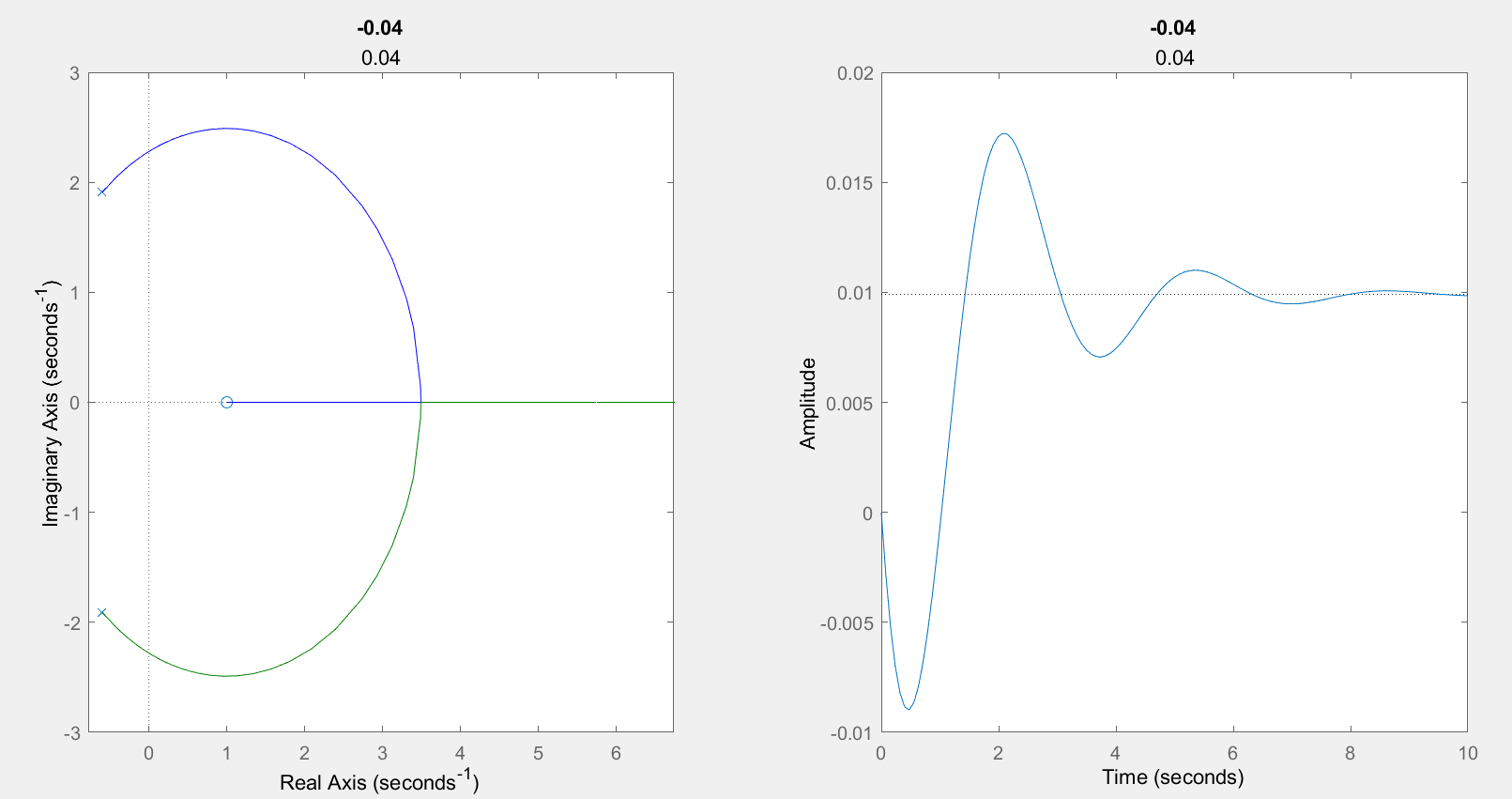
subtitle(k\*z);

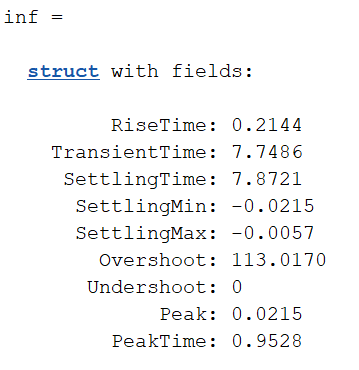
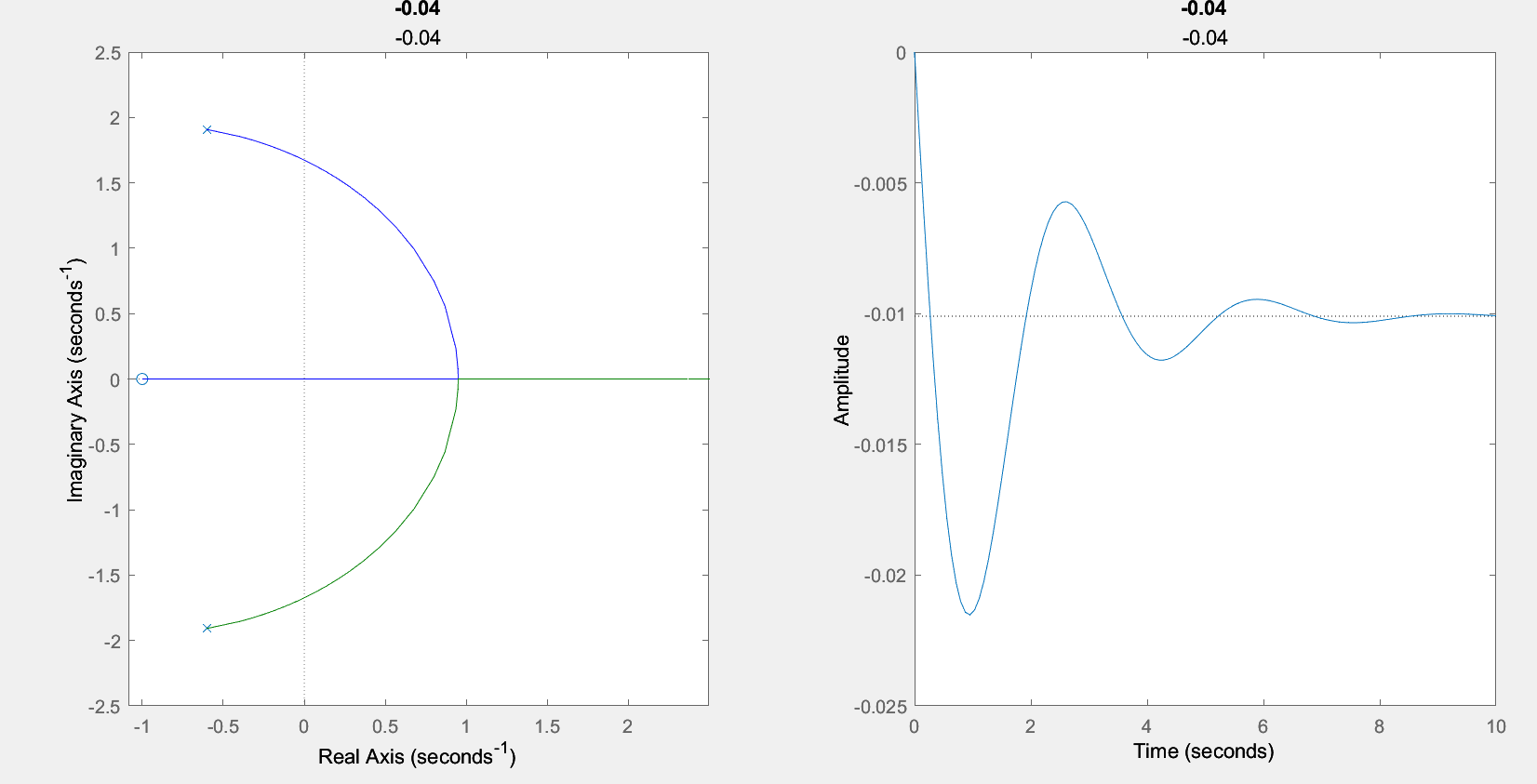
end

end

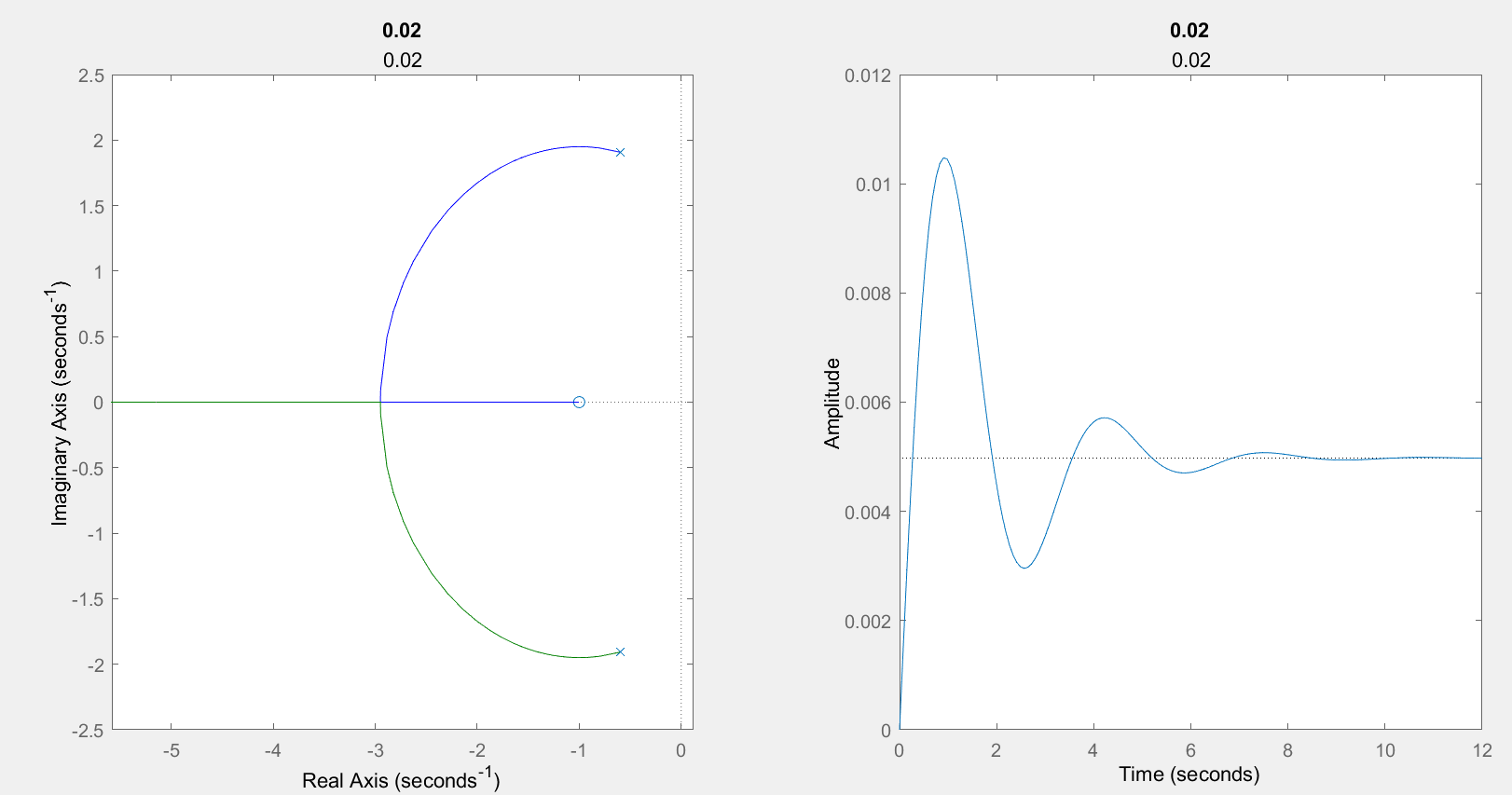
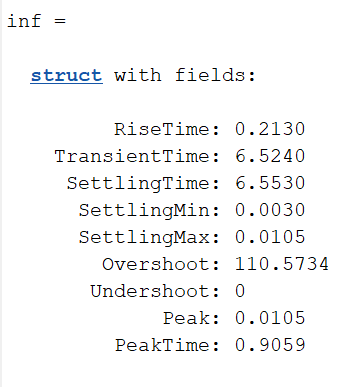
**OUTPUT:**

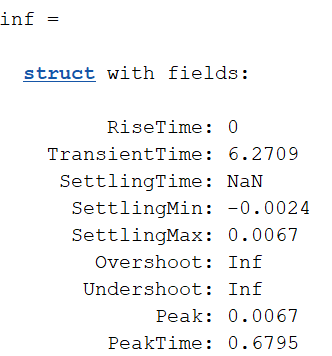
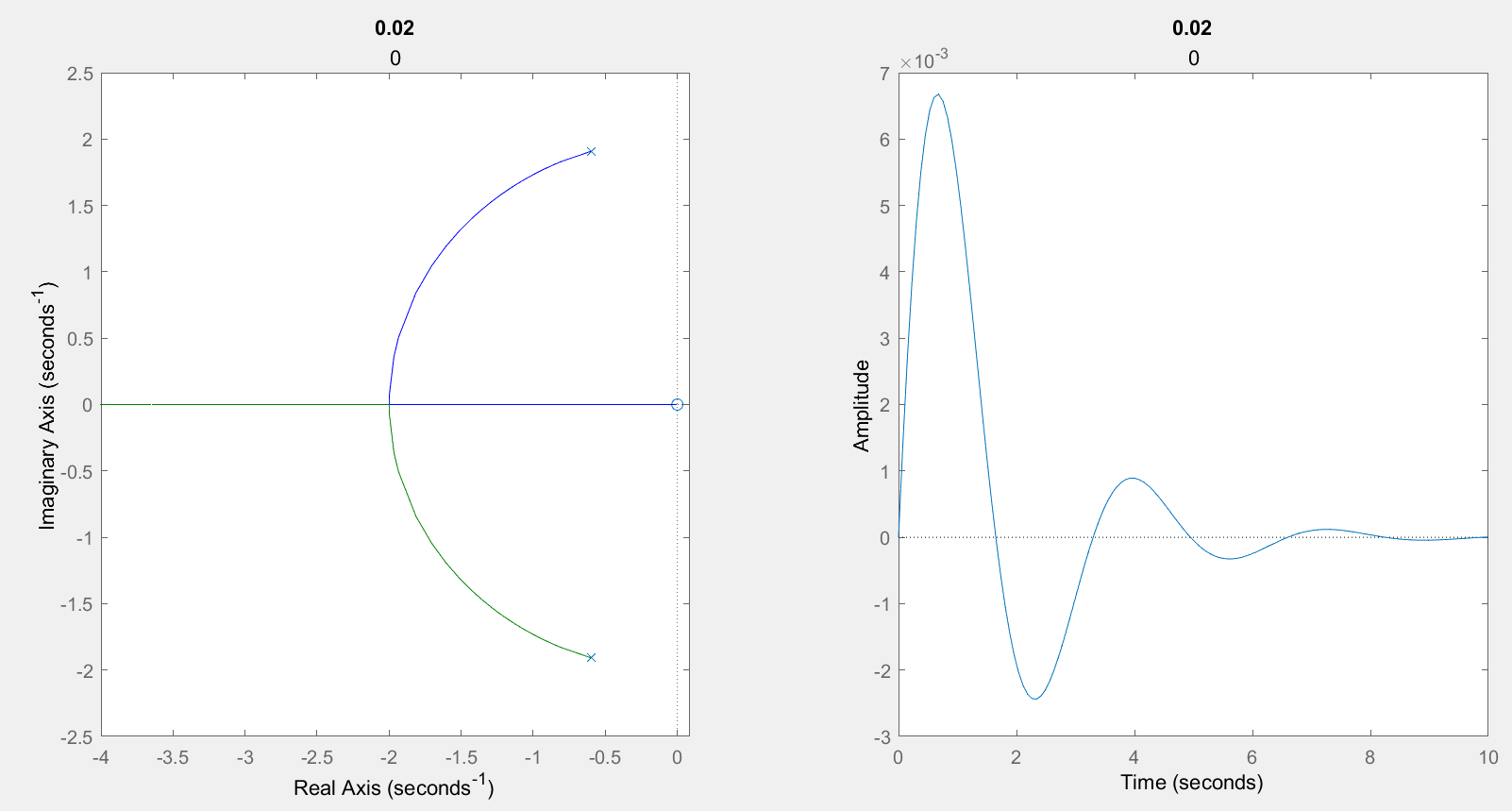
**Stable**

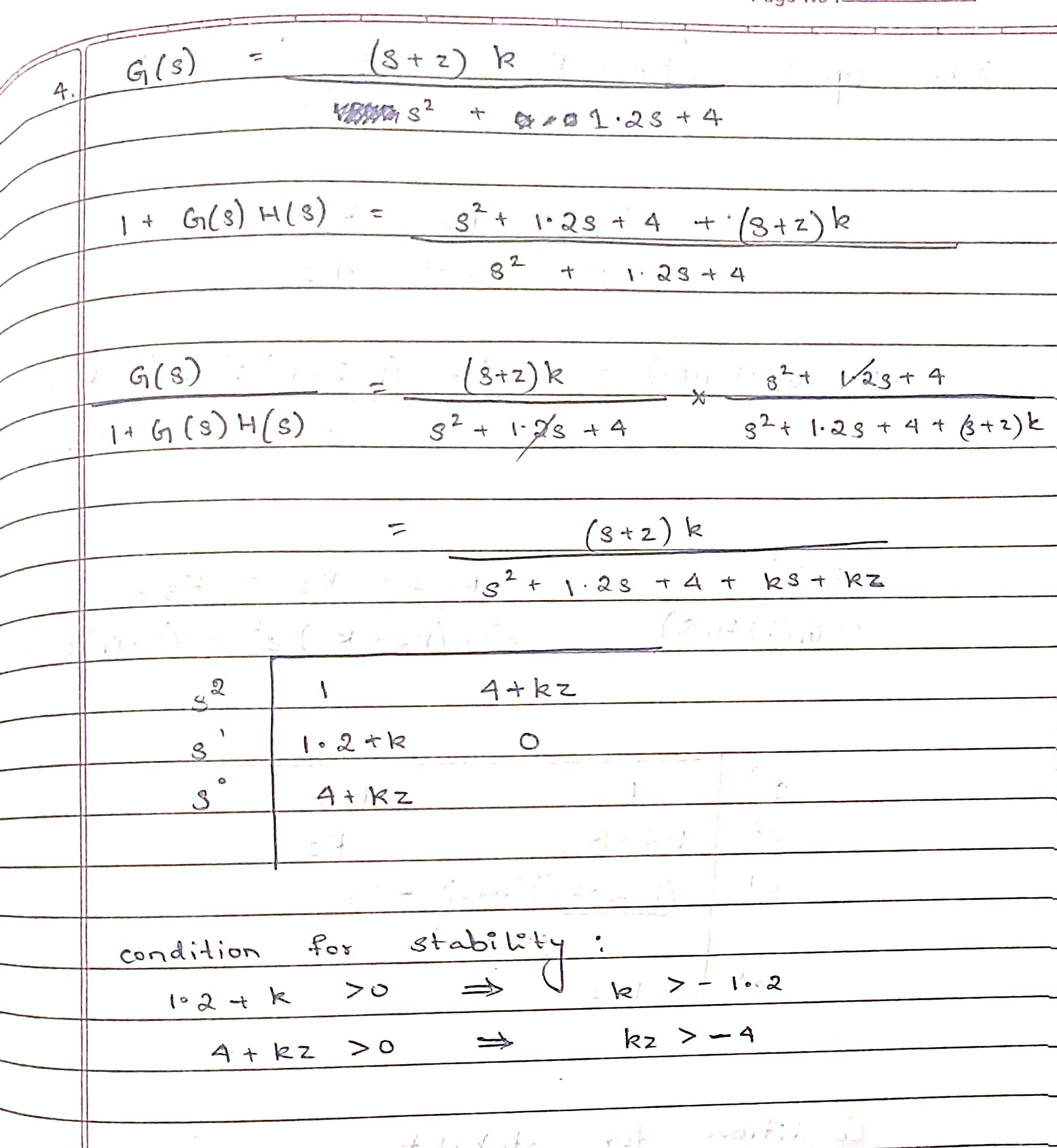
 

**Stable**



**MANUAL CALCULATIONS:**



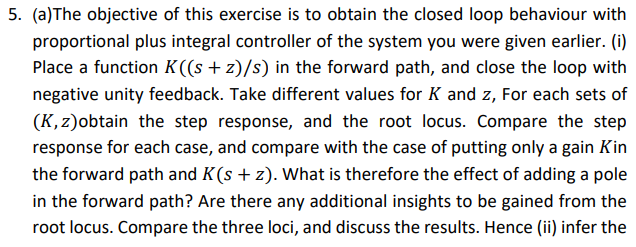
**INFERENCE:**

* The system is stable for k value >-1.2 **and** kz value > -4
* Derivative(Zero) softens or smooths oscillations , ie., it damps the system’s response making it more stable.

**TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| K | Z | Rise time(s) | Settling time(s) |
| 0.02 | 1 | 0.2130 | 6.5530 |
| 0.02 | 0 | 0 | NaN |
| -0.04 | -1 | 0.3185 | 7.6014 |
| -0.04 | 1 | 0.2144 | 7.8721 |

**QUESTION 5:**





**CODE (a):**

clc;

clear all;

close all;

%csp35

t=1;

for k = -1:1:1

for z=-1:1:1

sys = tf([0 0 k k\*z],[1 1.2 4 0]);

figure;

title(k);

subtitle(k\*z);

subplot(121)

rlocus(sys)

sys2=sys/(1+sys);

inf=stepinfo(sys2);

subplot(122)

step(sys2)

end

end

%k>0;z<0.0000133;

sys = tf([0 0 1 0.00001],[1 1.2 4 0]);

figure;

subplot(121)

rlocus(sys)

title(k);

subtitle(k\*z);

sys2=sys/(1+sys);

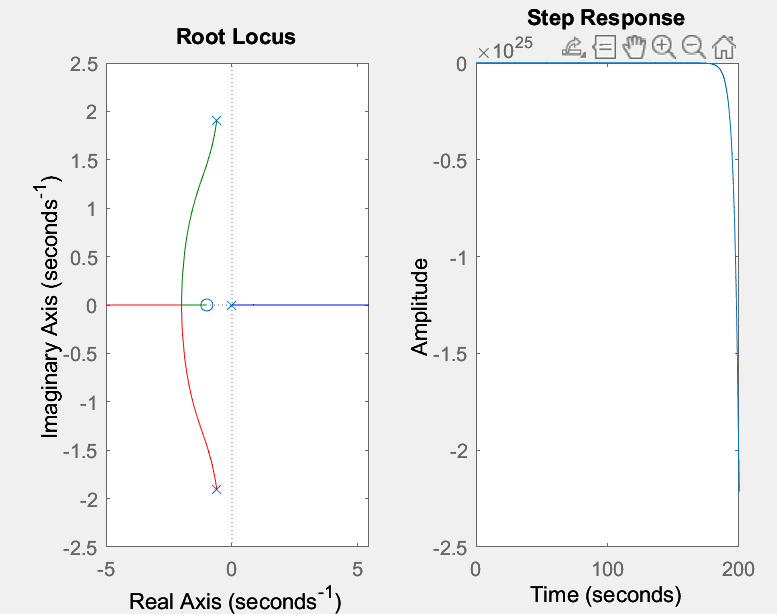
inf=stepinfo(sys2);

subplot(122)

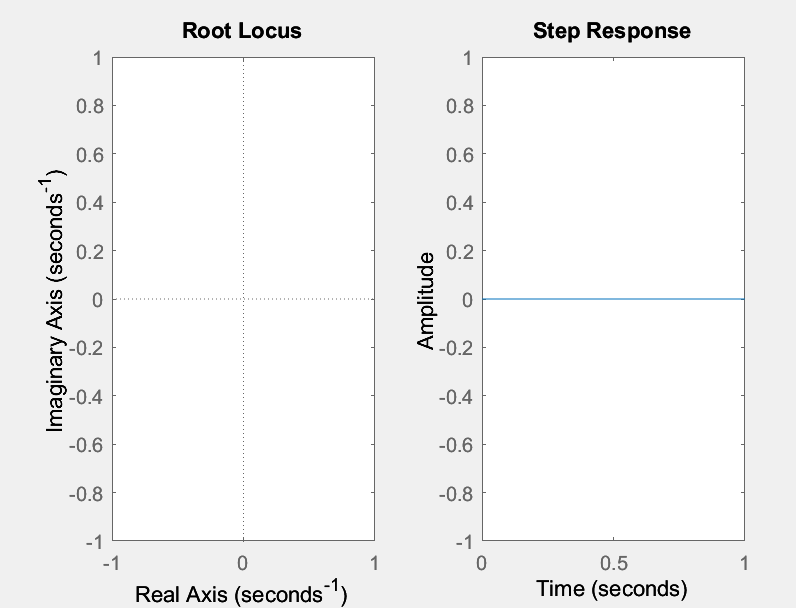
step(sys2)

**OUTPUT:**

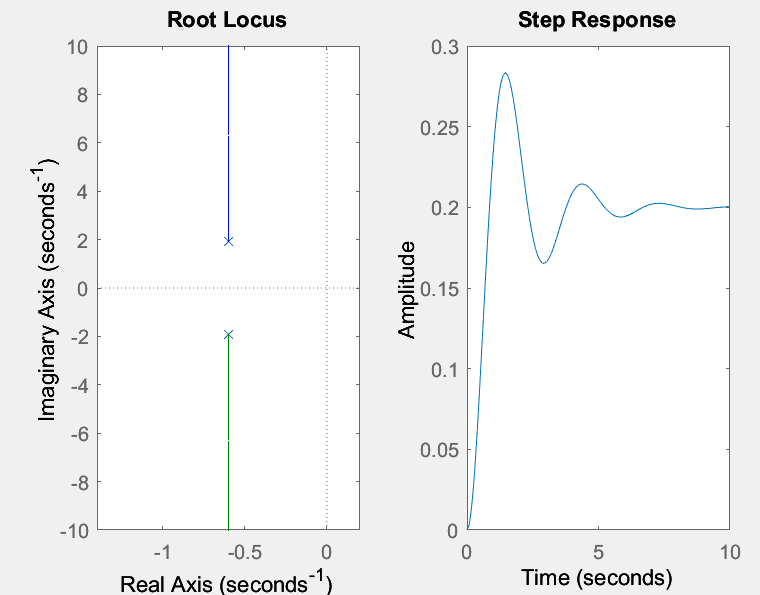
Unstable (k=-1 ,kz=-1)



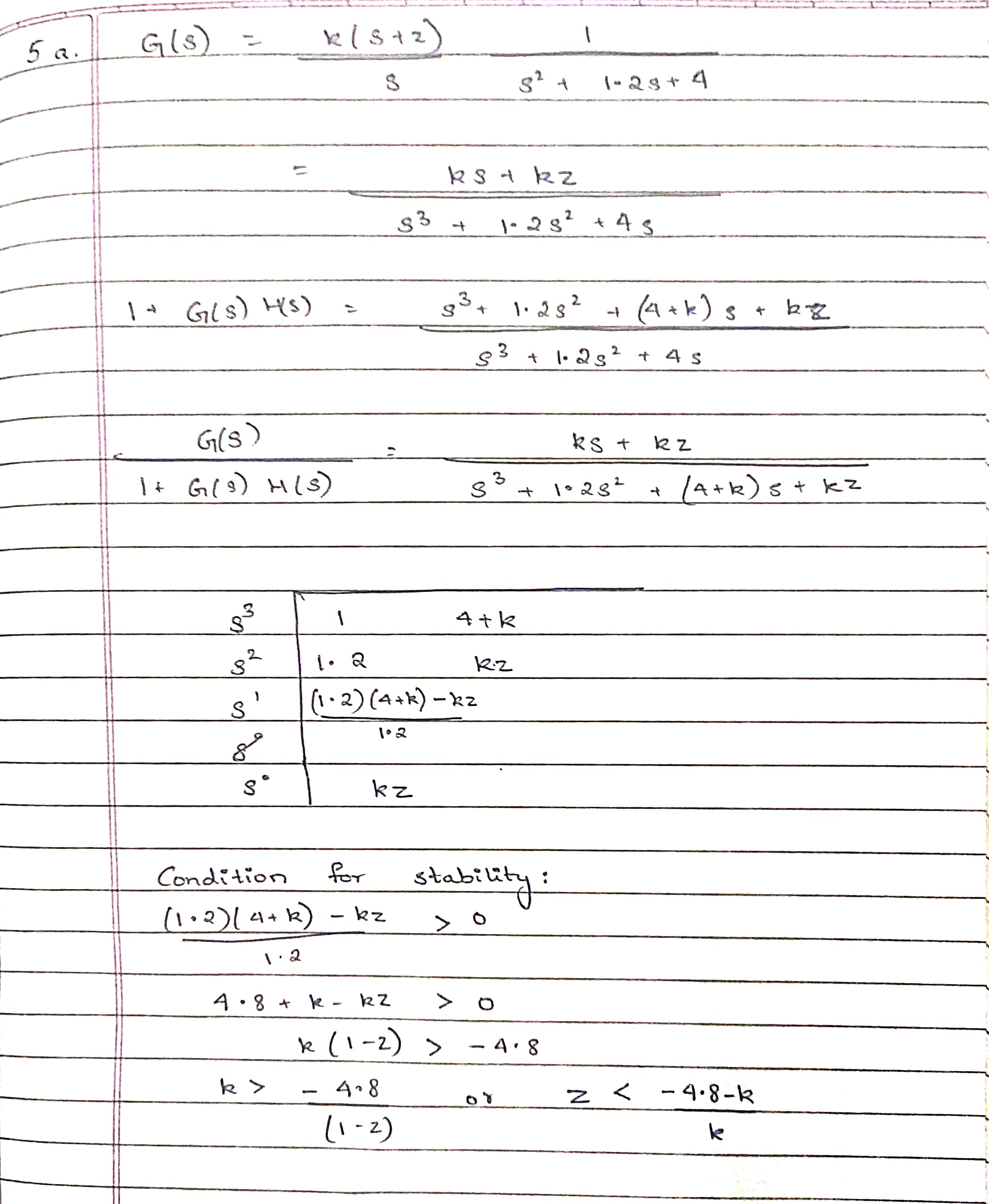
(k=0 ,kz=0)



Stable (k=1 ,kz=0)

****

**MANUAL CALCULATIONS:**



**CODE 5(b):**

clc;

clear all;

close all;

%csp35

t=1;

for k1 = -1:1:1

for k2=-1:1:1

for k3= -1:1:1

sys = tf([0 k3 k1 k2],[1 1.2 4 0]);

figure;

rlocus(sys)

disp("K1:");

disp(k1)

disp("K2:");

disp(k2)

disp("K3:");

disp(k3)

sys2=sys/(1+sys);

inf=stepinfo(sys2);

t=t+1;

end

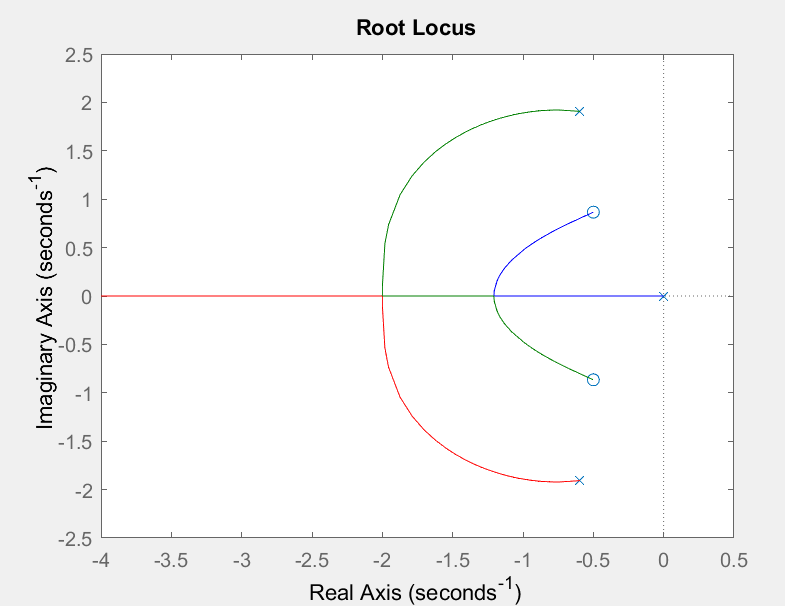
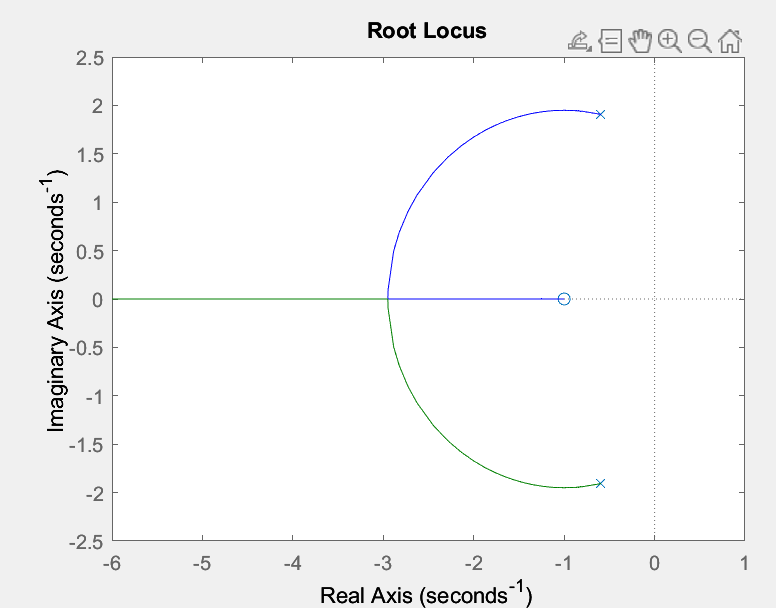
end

end

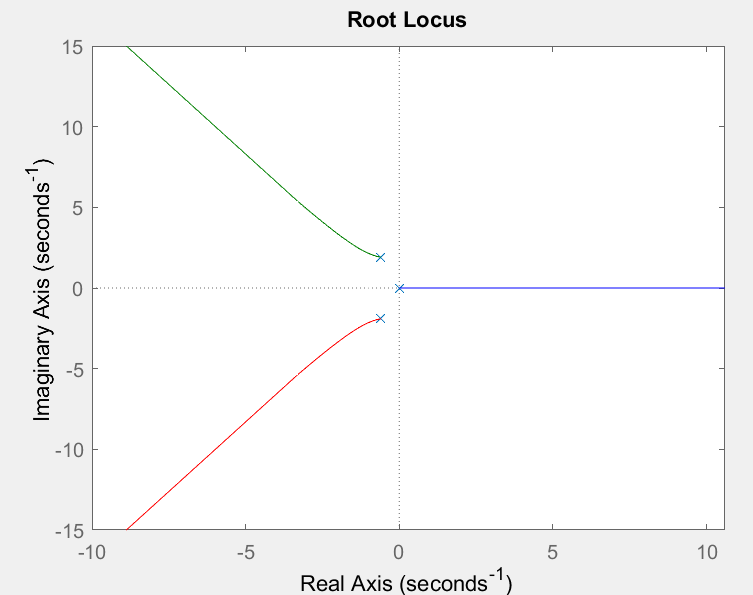
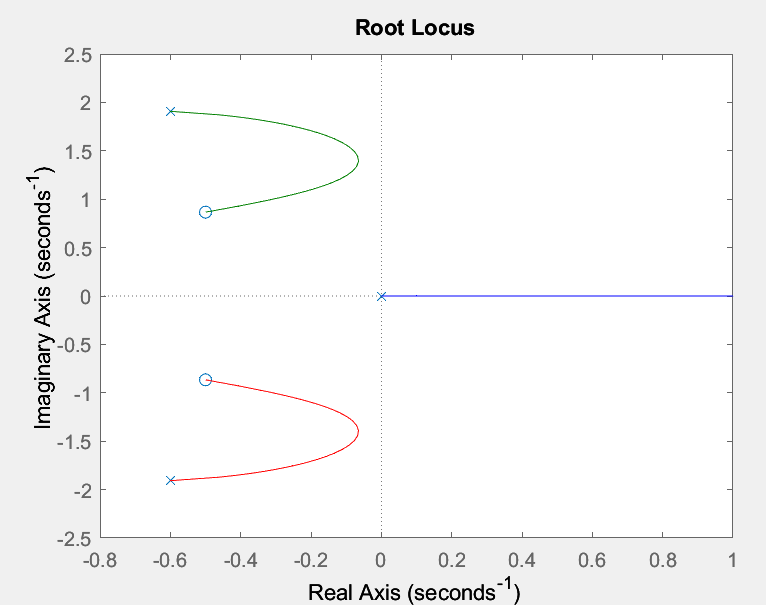
**OUTPUT:**

Stable (k2>0, k3>-1.2, 4.8+4k3+1.2k1+k1k3>0)

**K1 = 1, K2 = 1, K3 = 1 K1 = 0, K2 = 1, K3 = -1**

** **

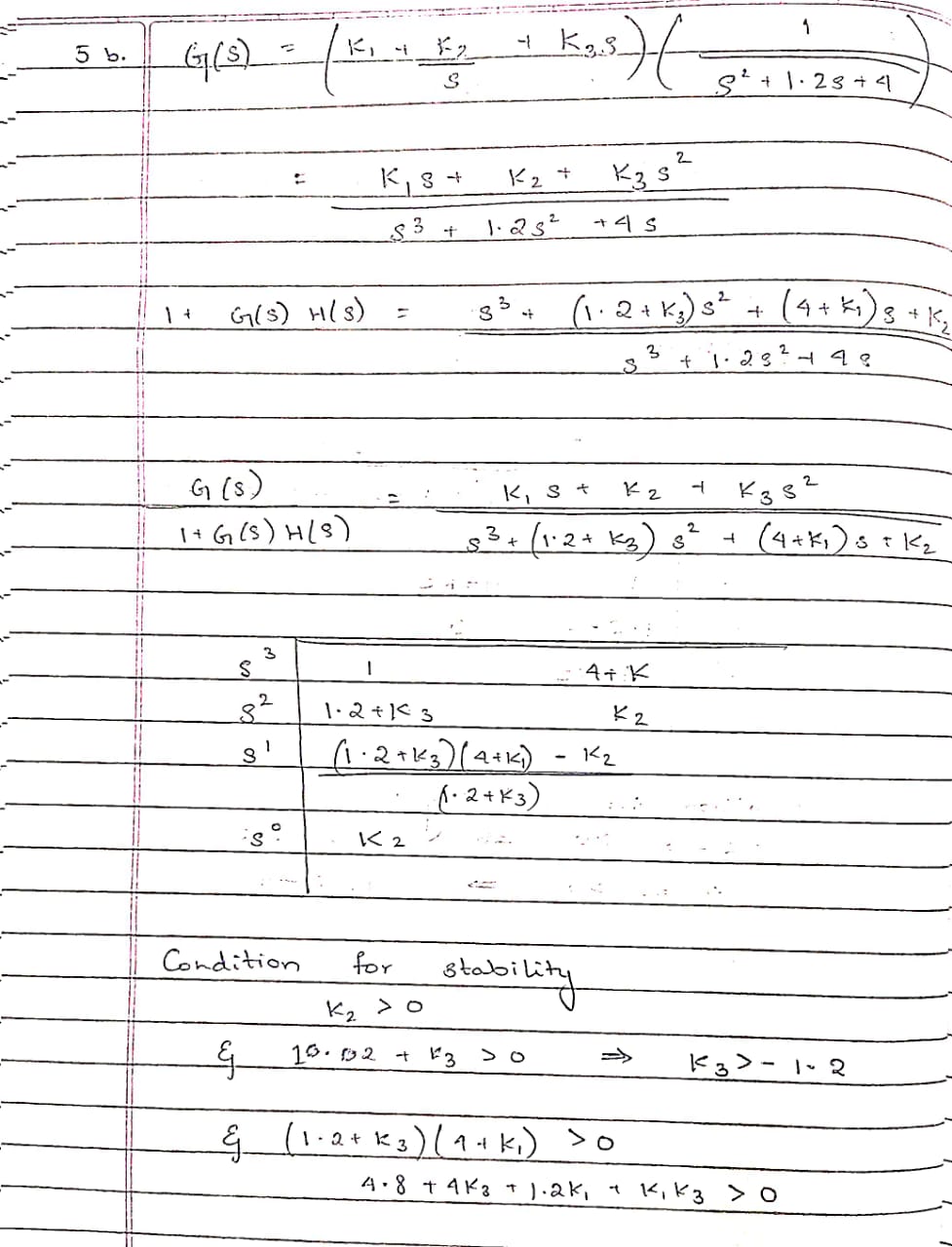
Unstable

**K1 = 0, K2 = -1, K3 = 0 K1 = -1, K2 = -1, K3 = -1  **

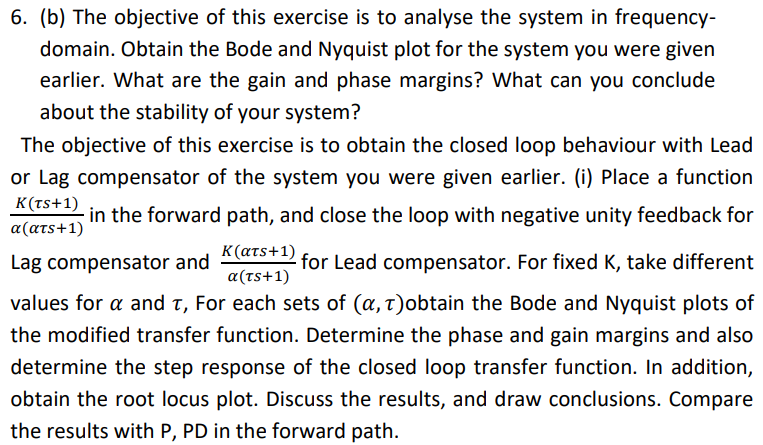
**INFERENCE:**

* The effect of adding a pole in the forward path is to introduce a time delay in the system's response. This can result in slower response and longer settling time, as the system takes longer to reach its steady-state value.
* A P controller's root locus consists of a single branch, whereas a PD controller has two branches and a PID controller has three branches. The complexity of the controller increases the number of branches of the root locus. The derivative term in the controller responds faster to changes in the error signal, reducing overshoot and improving transient response. However, it can also make the system more sensitive to noise and disturbances, causing the closed-loop poles to shift to the right side of the complex plane and destabilise the system.

**MANUAL CALCULATIONS:**



**QUESTION 6:**



**CODE:**

n=[0 0 1];

d=[1 1.2 4];

sys = tf(n,d);

figure;

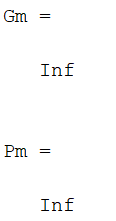
bodeplot(sys);

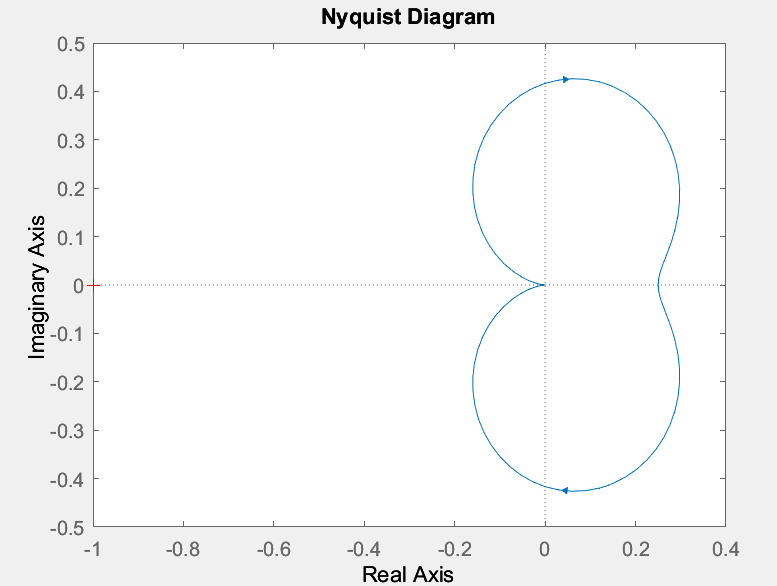
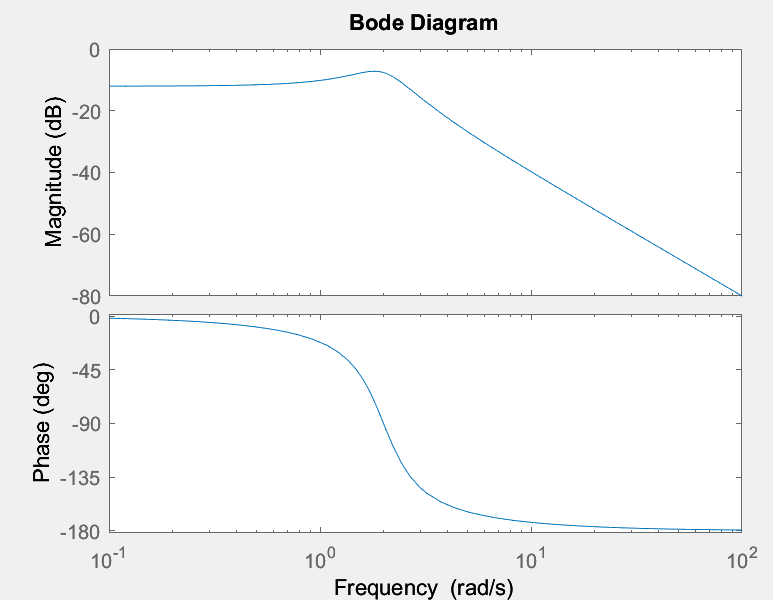
figure;

nyquistplot(sys);

[Gm,Pm]=margin(n,d)

**OUTPUT:**





**INFERENCE:**

Pm and Gm are both positive, thus stable system.

**INFERENCE:**

If a system has an infinite gain margin, it indicates that its gain may be raised indefinitely without the system becoming unstable. In other words, the system is extremely resistant to changes in its gain. Similarly, if a system has an infinite phase margin, it signifies that the phase lag may be extended indefinitely without the system becoming unstable. This indicates that the system is highly robust to changes in its phase.

* In practical terms, an infinite gain margin and an infinite phase margin are highly desirable properties for a control system. They indicate that the system is highly stable and can handle large disturbances or uncertainties without becoming unstable
* Lead and Lag compensators are used to increase and decrease the phase margin respectively.

**APPLICATIONS:**

PID controllers are commonly used in temperature control applications to regulate the temperature of a system by adjusting the input power to the heater or cooling system. Here are a few examples of PID controller applications for temperature control:

1. Industrial ovens: PID controllers can be used to maintain the temperature inside industrial ovens that are used for baking, drying, or curing products.

2. HVAC systems: PID controllers are used to maintain a comfortable temperature inside buildings by controlling the heating, ventilation, and air conditioning (HVAC) systems.

3. Refrigeration systems: PID controllers are used to maintain the temperature inside refrigeration systems, such as freezers and refrigerators, to ensure that the temperature remains within a desired range.

4. Chemical processing: In chemical processing, PID controllers are used to maintain the temperature of chemical reactors to ensure that the reactions occur at the desired rate.

5. Water heating: PID controllers can be used to regulate the temperature of water heaters to ensure that the water temperature remains within a desired range.

In all of these applications, the PID controller continuously measures the temperature of the system and adjusts the input power to the heating or cooling system to maintain the temperature within a desired range. The PID controller can be tuned to provide the desired response time and stability for the specific application.

Question 6b:

Showing the lead and lag compensation effect

Note: Since our transfer function is perfectly stable we have used another transfer function to depict the compensation effect.

Code:

clc;

clear all;

close all;

%Lead compensator

for a = -18:20:100

for t=-10:10:10

n=[0 0 a\*t 1];

d=[1500\*a\*t 0.02\*a\*t+1500\*a 0.02\*a 0];

sys = tf(n,d);

figure;

subplot(221)

rlocus(sys)

sys2=sys/(1+sys);

subplot(222)

bodeplot(sys);

subplot(223)

nyquistplot(sys);

subplot(224)

step(sys2);

[Gm,Pm]=margin(n,d)

end

end

%Lag compensator

for a = -18:20:100

for t=-10:10:10

n=[0 0 t 1];

d=[1500\*(a^2)\*t 0.02\*(a^2)\*t+1500\*a 0.02\*a 0];

sys = tf(n,d);

figure;

subplot(221)

rlocus(sys)

sys2=sys/(1+sys);

subplot(222)

bodeplot(sys);

subplot(223)

nyquistplot(sys);

subplot(224)

step(sys2);

[Gm,Pm]=margin(n,d)

end

end

Output:

