BIRZEIT UNIVERSITY

Faculty of Engineering and Technology

Department of Electrical and Computer Engineering

Control systems

(ENEE4302)

"MATLAB assignment"

Prepared by:

Name: Mohammad Shehadeh Number:1201458

Section:1 Date:5/1/24

Contents

List of Figures

Introduction:

The block diagram of a pitch control system for Unmanned Free-Swimming Submersible Vehicles (UFSS) typically consists of several subsystems working together to control the pitch angle of the vehicle. The aim of the pitch control system is to submerge or rise the vehicle by varying a vertical force.

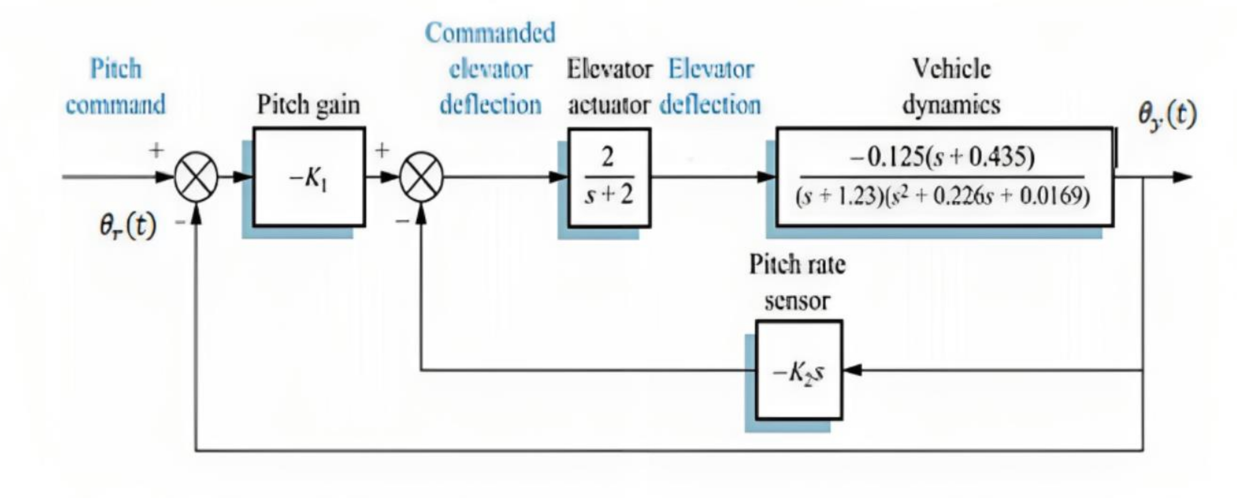

Figure 1 system block diagram

The block diagram consists of a Pitch Gain: This subsystem represents the gain applied to the pitch control system. It determines the sensitivity of the system to changes in the pitch angle. Elevator Actuator: The elevator actuator is responsible for controlling the elevator deflection, which in turn affects the pitch angle of the vehicle. It receives input from the pitch rate feedback and adjusts the elevator deflection accordingly. Vehicle Dynamics: This subsystem represents the dynamics of the UFSS vehicle. It considers factors such as mass, inertia, and hydrodynamic forces that affect the pitch angle of the vehicle. The vehicle dynamics subsystem is typically modeled using transfer functions. Pitch Rate Sensor: The pitch rate sensor provides feedback to the closed-loop system. It measures the rate of change of the pitch angle and provides this information to the elevator actuator for control purposes.

Procedure, Results and Discussion:

(all assignment objectives are here but maybe not in the same order as the assignment)

Part 1:Transfer Function

1- Transfer function in terms using symbolic k1, k2 using math operation

In the below code I used math operations to manually reduce the block diagram into one Transfer Function while using k1, k2 and s as symbolic variables.

When k_1, k_2 are symbolic:

```
clear all;
syms s k1 k2;
%k1=5;
%k2=8;
g1=-k1;
g2=2/(s+2);
l1=(-0.125)/(s+1.23);
l2=(s+0.435)/(s^2+0.226*s+0.0169);
g3=l1*l2;
g4=(-k2*s);
cas1=g2*g3;%series g2 g3
feedback1=cas1/(1+(cas1*g4));%cas1 g4 feedback
cas2=g1*feedback1;%series g1 feedback1
system_tf=cas2/(1+cas2);%feedback cas2 1
system=simplify(system_tf);
system=(system);
[num,den]=numden(system);
num = simplifyFraction(num,"Expand",true);%expand k1*()
num=num/10^6;
den=den/10^6;
system=num/den;
system=vpa(system);
disp(system);
```

$$\frac{0.10875\,k_1 + 0.25\,k_1\,s}{0.10875\,k_1 + 0.610547\,s + 0.25\,k_1\,s + 0.10875\,k_2\,s + 0.25\,k_2\,s^2 + 3.20688\,s^2 + 3.456\,s^3 + s^4 + 0.041574}$$

Figure 2 Transfer function in terms of k1, k2.

When $k_1 = 5$ and k_2 is symbolic.

If we change the above code and remove k1 from the 2nd line and removing '%" from the 3rd line therefore transferring it from being a comment into a constant we get the following transfer function.

$$\frac{1.25\,s + 0.54375}{1.860547\,s + 0.10875\,k_2\,s + 0.25\,k_2\,s^2 + 3.20688\,s^2 + 3.456\,s^3 + s^4 + 0.585324}$$

Figure 3 transfer function using k1=5.

When $k_1 = 5$ and $k_2 = 8$.

Doing the same procedure we can get the transfer function when $k_1 = 5,\ k_2 = 8$.

$$\frac{1.25\,s + 0.54375}{s^4 + 3.456\,s^3 + 5.20688\,s^2 + 2.730547\,s + 0.585324}$$

Figure 4 system transfer function k1=5, k2=8.

When k_1 is symbolic and $k_2 = 8$.

$$\frac{0.10875\,k_1 + 0.25\,k_1\,s}{0.10875\,k_1 + 1.480547\,s + 0.25\,k_1\,s + 5.20688\,s^2 + 3.456\,s^3 + s^4 + 0.041574}$$

Figure 5 system transfer function k1 symbol k2=8

2-Transfer function using Simulink:

1-using the non accurate model

The system in the figure below was built using Simulink but this system isn't an accurate representation of the desired system since the linearization model of the derivative block is $\frac{s}{cs+1}$

Not s.

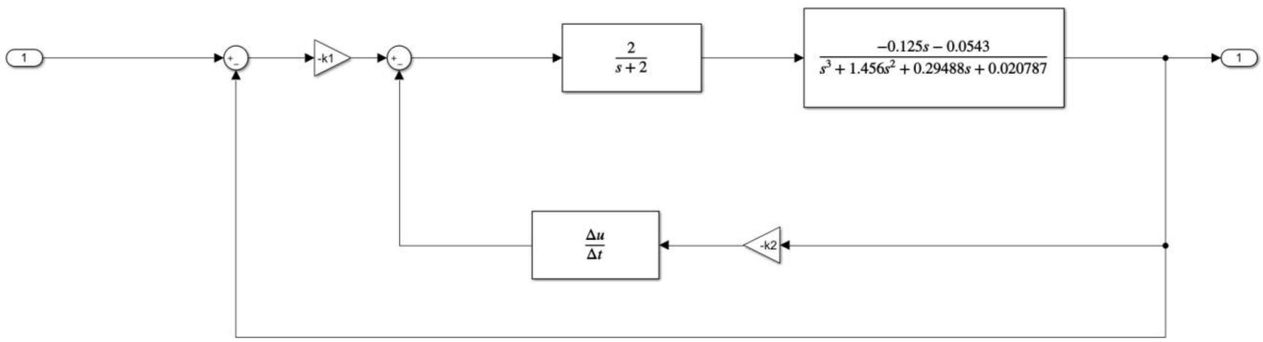

Figure 6 Simulink block diagram

Below is the written code and the transfer function of the system at k1=5,k2=8.

```
[a,b,c,d]=linmod('block_diagram') %linearized model

 a = 4×4
     -1.4560   -0.2949   -0.0208    2.0000
      1.0000         0         0         0
           0    1.0000         0         0
           0   -0.6250   -0.2715   -2.0000

 b = 4×1
       0
       0
       0
      -5

 c = 1×4
           0   -0.1250   -0.0543         0

d = 0
```

```
[num,den]=ss2tf(a,b,c,d)%state space to transfer function

 num = 1×5
           0         0         0    1.2500    0.5430

 den = 1×5
      1.0000    3.4560    3.2069    1.8605    0.5846
```

```
system=tf(num,den)%G(s)=

system =

                 1.25 s + 0.543
  ---------------------------------------------
  s^4 + 3.456 s^3 + 3.207 s^2 + 1.861 s + 0.5846

Continuous-time transfer function.
```

Figure 7 non accurate transfer function (code)

2- Accurate Simulink block diagram

Below is manually reduced block diagram so I didn't have to use the derivative block which introduced the error to the previous system.

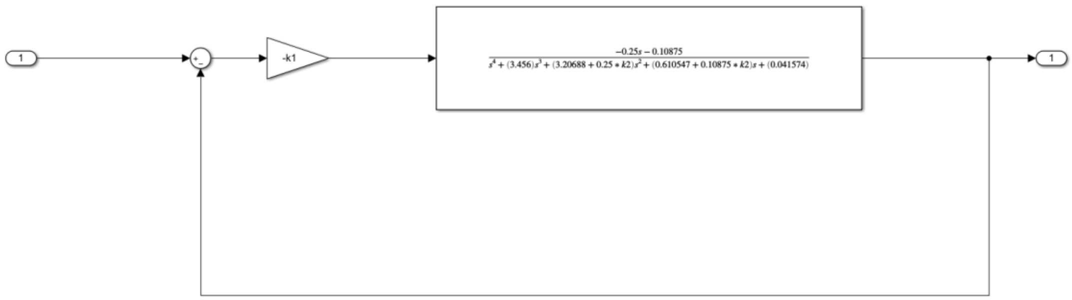

Figure 8 accurate Simulink model

The code used and result Transfer Function:

```
[a,b,c,d]=linmod('block_t'); %linearized model
[num,den]=ss2tf(a,b,c,d);%state space to transfer function
system=tf(num,den);%G(s)=
system=simplify(system)
```

```
system =

                 1.25 s + 0.5437
  ----------------------------------------------
  s^4 + 3.456 s^3 + 5.207 s^2 + 2.731 s + 0.5853

Continuous-time transfer function.
```

3-Transfer function using control system library in MATLAB:

Here we use the function in control system library such as series, feedback, tf and stepinfo which make the system easier to analyze here is the code I used to get the transfer function.

```
k1=5;
k2=8;
g1=tf([-k1],[1]);
g2=tf([2],[1 2]);
l1=tf([-0.125],[1 1.23]);
l2=tf([1 0.435], [1 0.226 0.0169]);
g3=l1*l2;
g4=tf([-k2 0],[1]);
cas1=series(g2,g3);
feedback1=feedback(cas1,g4);
cas2=series(g1,feedback1);
system_tf=feedback(cas2,1)
```

```
system_tf =

                1.25 s + 0.5437
  ---------------------------------------------
  s^4 + 3.456 s^3 + 5.207 s^2 + 2.731 s + 0.5853

Continuous-time transfer function.
```

2-Root locus Plot

Using the function rlocus(system_tf) or rlocusplot(system_tf)

We can plot the root locus.

1-Variable k1 and k2

While using different values of k1 and k2, 3 stable values each

```
k1=[5,10,15] ;%values of k1
k2=[8,4,1];%values of k2
```

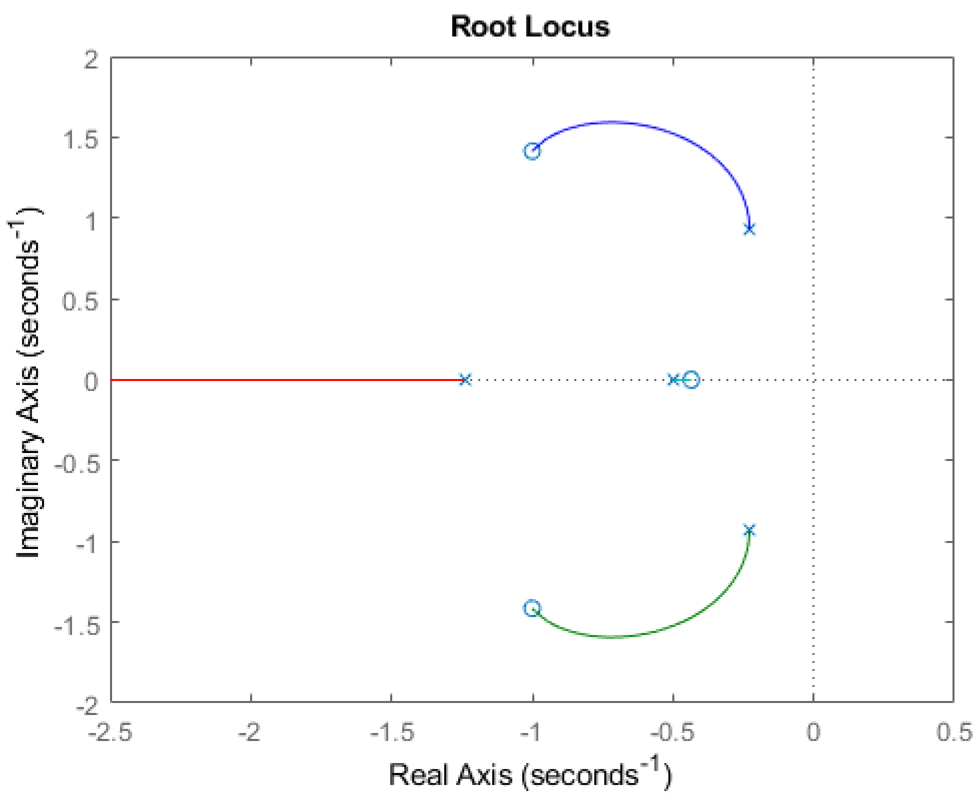

Figure 9 root locus for variable k1 and k2

2- Variable k2 fixed k1

Now root locus for k1=5 and different values of k2

```
k1=5 ;
k2=[8,4,1];
```

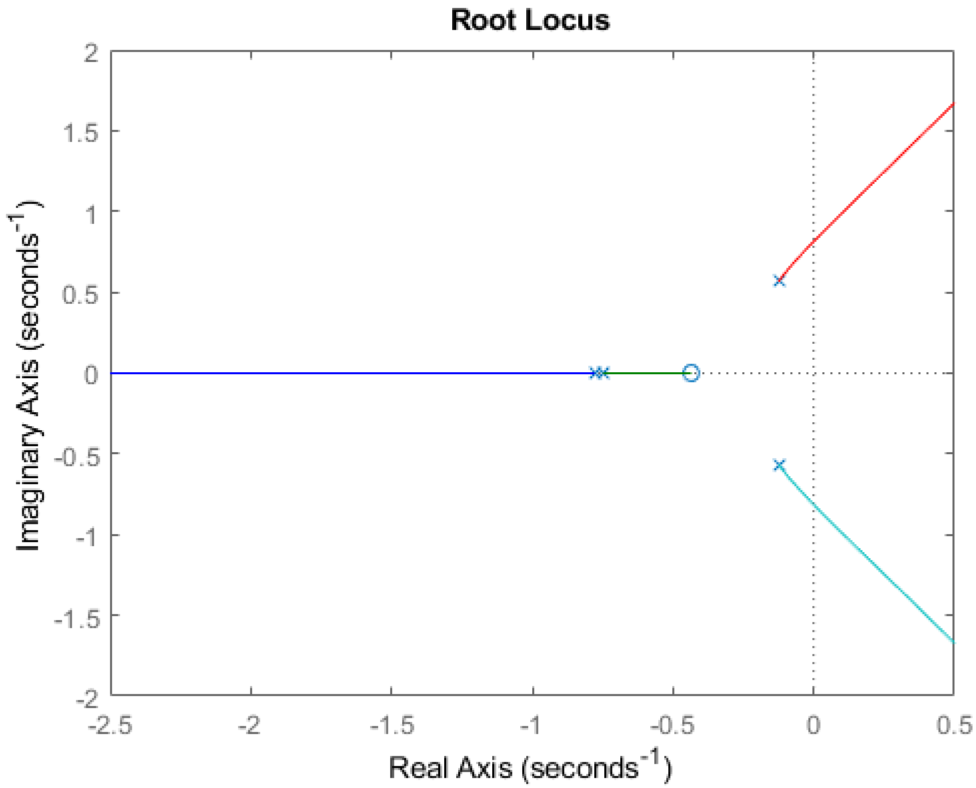

Figure 10 root locus for k1=5 and 3 different values of k2

3- Variable k1 fixed k2

```
k1=[5,10,15] ;
k2=8;
```

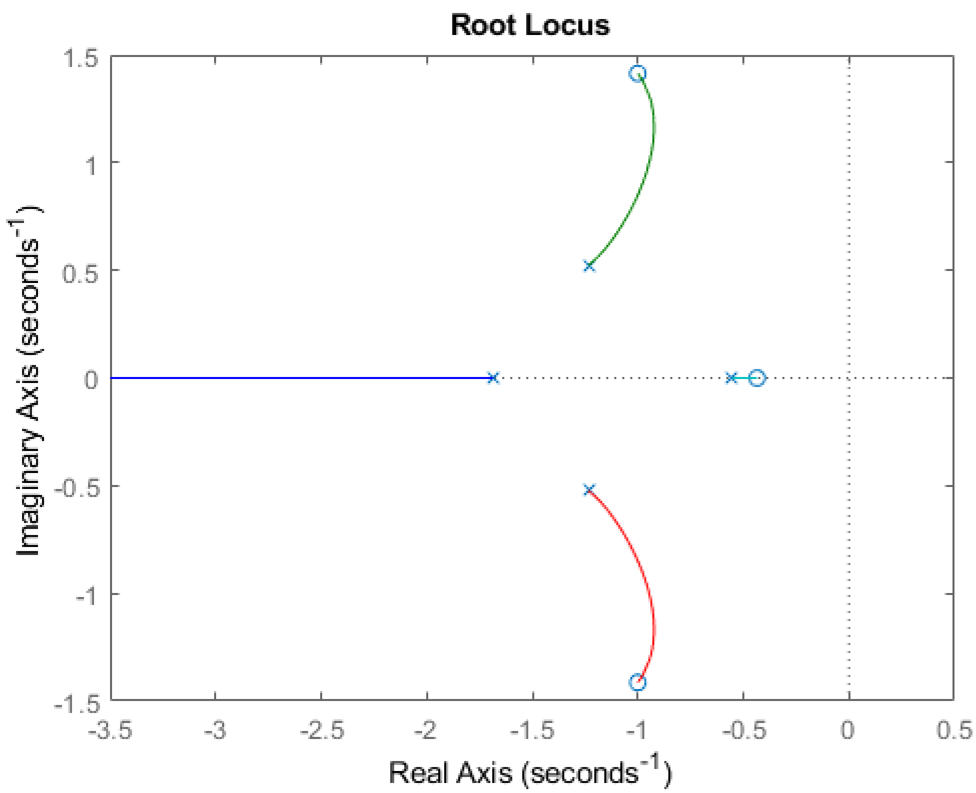

Figure 11 k1 variable k2 =8

4- Constant k1,k2

k1=5 k2=8

```
k1=5 ;%values of k1
k2=8;%values of k2
```

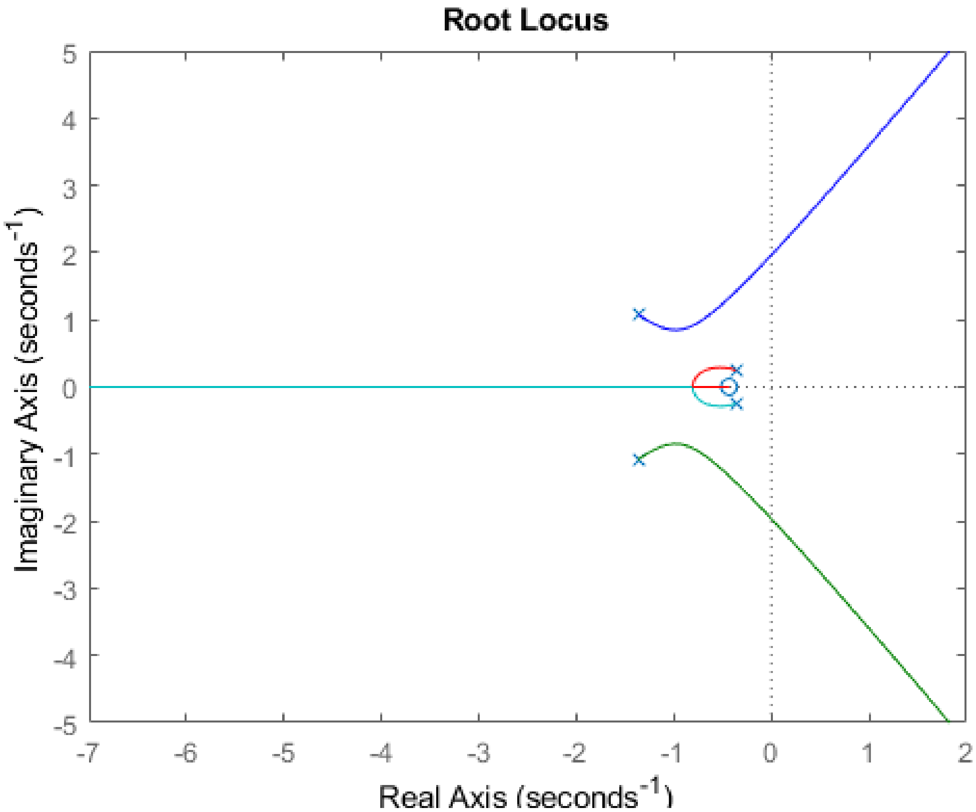

Figure 12 root locus k1=5 k2=8

3- Testing system stability for k1

Using Routh Hurwitz, we can test the system stability based on the change of the signs.

$$\frac{0.10875\,k_1 + 0.25\,k_1\,s}{0.10875\,k_1 + 1.480547\,s + 0.25\,k_1\,s + 5.20688\,s^2 + 3.456\,s^3 + s^4 + 0.041574}$$

Figure 13 Transfer function k2=8

s^4	1	5.20688	0.041574+0.25k1	+
s^3	3.456	1.48+0.25k1		+
s^2	4.778-0.0723k1	0.041574+0.25k1		
s^1	$\frac{-0.018075k_1^2 + 0.2235k_1 + 6.93}{4.778 - 0.0723k_1}$			
s^0	0.041574+0.25k1			

Table 1 Routh Hurwitz table k_2=8.

K2=8

K1

$s^3 s^4$ +++

s^2 ++(66.057)---------------------------------------

s^1--------(-6.244)+++++++++++++++++++(49.97)-----------------------(82.47)-----------------------

s^0-----------------------(-0.3822)+++

K2=4

$s^3 s^4$ +++

s^2 +++++++++++++++++++++++++++++++++++++53.973--

s^1--------(-4.5304)++++++++++++++++++(37.809)------------------------(70.49)-----------------------

s^0-----------------------(-0.3822)+++

Stable between [-0.3822,37.809]

K2=1

$s^3 s^4$ +++

s^2 +++++++++++++++++++++++++++++++++(44.91072)--

s^1--------(-3.23)+++++++++++++++++(28.645)-----------------------(61.529)-----------------------

s^0-----------------------(-0.3822)++

4-Step response for different K_1 values along with its transient parameters and root locus plot.

1-Stable oscillatory $k_1 = 25, k_2 = 8$

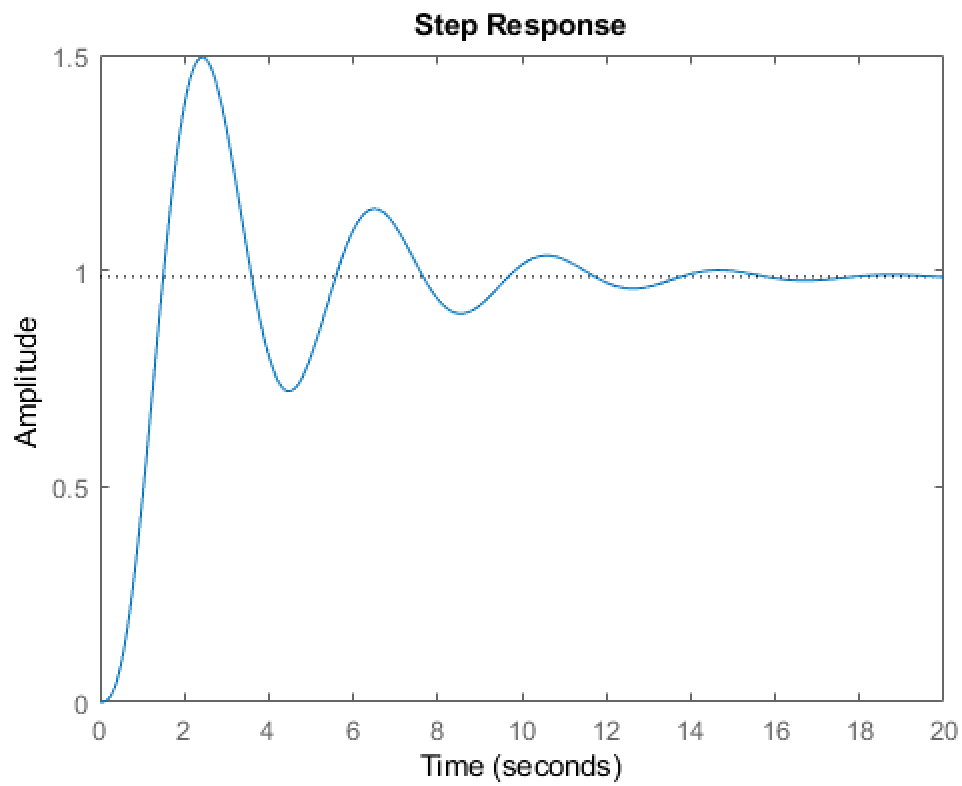

Figure 14 Stable oscillatory

Poles and zeros:

```
zeros=zero(system_tf)

zeros = -0.4350
poles=pole(system_tf)
poles = 4×1 complex
  -2.4301 + 0.0000i
  -0.2814 + 1.5408i
  -0.2814 - 1.5408i
  -0.4630 + 0.0000i
```

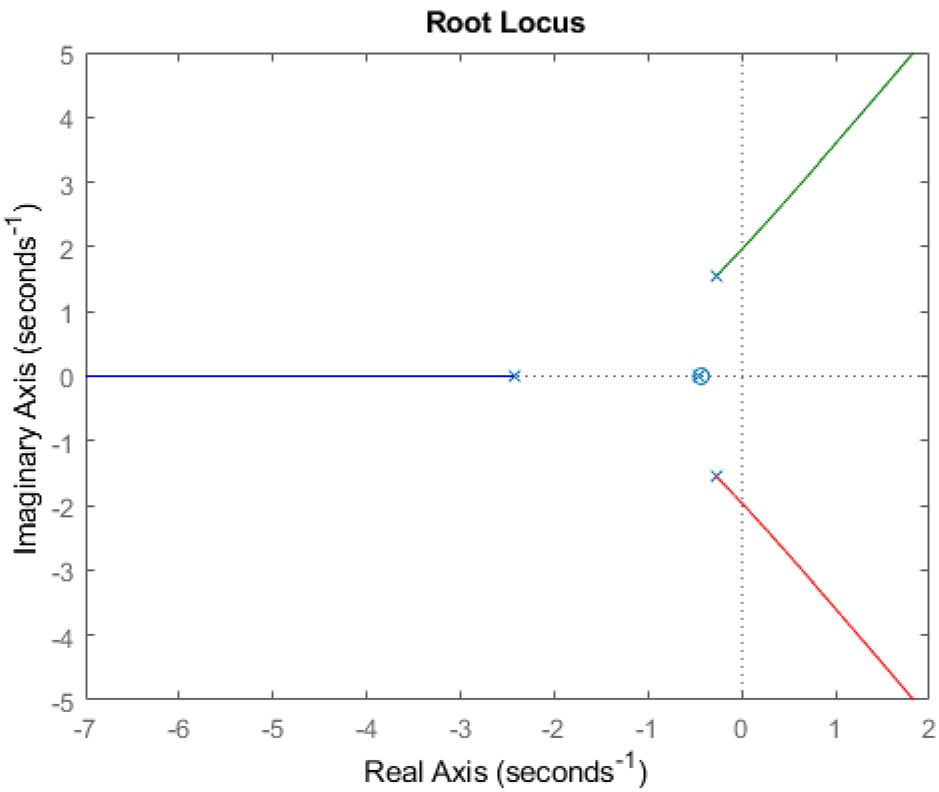

Figure 15 root locus k1=25,k2=8

Transient parameters:

Notice that since it is oscillatory the settling time is relatively high but the rise time is relatively low which made the overshoot high due to the poles being far from the real axis, also there is no undershoot which is expected of a stable oscillatory system.

```
stepinfo(system_tf)
```

```
ans = struct with fields:
        RiseTime:    0.8812
    SettlingTime:   13.1379
     SettlingMin:    0.7217
     SettlingMax:    1.4951
       Overshoot:   51.7937
      Undershoot:         0
            Peak:    1.4951
        PeakTime: 2.4257
```

Steady state error

Using this code we can find the steady state error.

```
[y,tf]=step(system_tf);
sp=1;
ss_error=abs(sp-y(end))
```

```
ss_error = 0.0115
```

2-stable non-oscillatory $k_1 = 1, k_2 = 8$

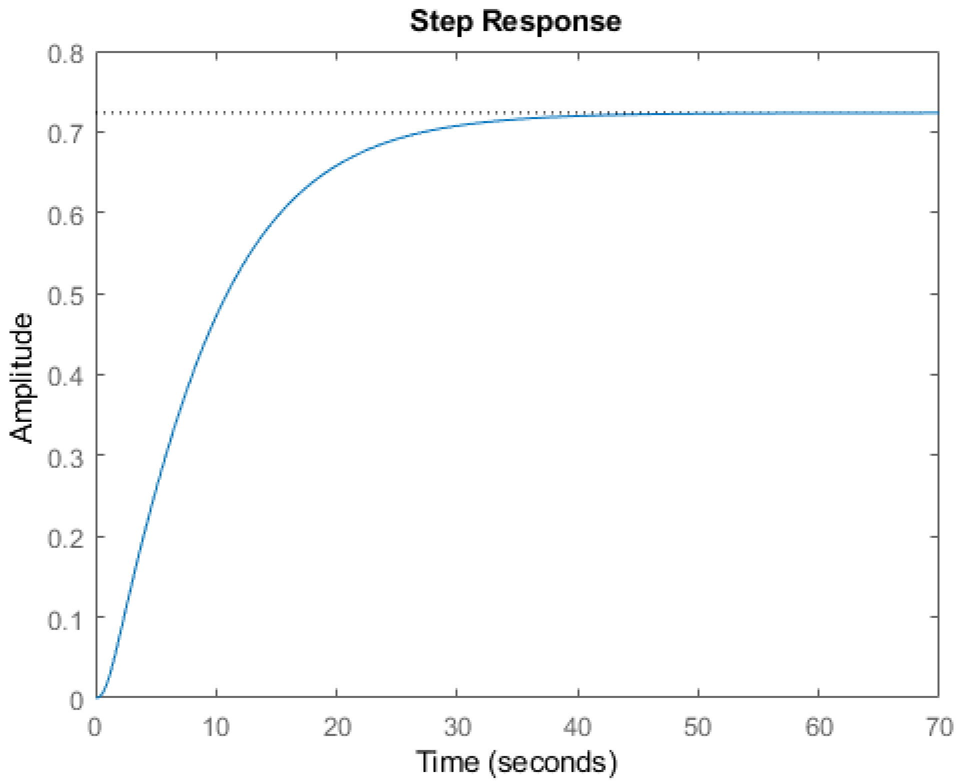

Figure 16 Stable non-oscillatory

Poles and zeros

```
zeros=zero(system_tf)
zeros = -0.4350
poles=pole(system_tf)
poles = 4×1 complex
  -1.5218 + 1.2639i
  -1.5218 - 1.2639i
  -0.2701 + 0.0000i
  -0.1422 + 0.0000i
```

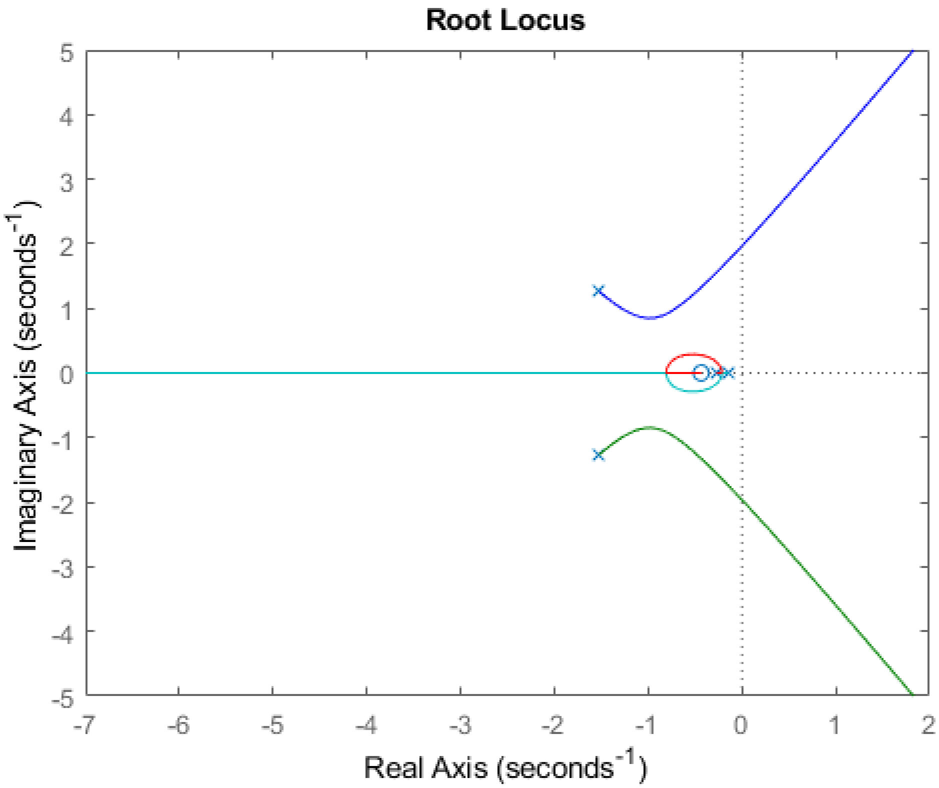

Figure 17 root locus for stable non-oscillatory

Transient parameters

Notice that the rise time is high because the poles are far from the real axis and the settling time is also high because the poles are close to the imaginary axis and because the system is non oscillatory there is no overshoot or under shoot.

```
stepinfo(system_tf)
ans = struct with fields:
        RiseTime:    17.3169
    SettlingTime:    30.7223
     SettlingMin:    0.6529
     SettlingMax:    0.7234
       Overshoot:    0
      Undershoot:    0
            Peak:    0.7234
        PeakTime: 77.4015
```

Steady state error

Using this code we can find the steady state error.

```
[y,tf]=step(system_tf);
sp=1;
ss_error=abs(sp-y(end))
```

```
ss_error = 0.2766
```

3-unstable $k_1 = 50, k_2 = 8$

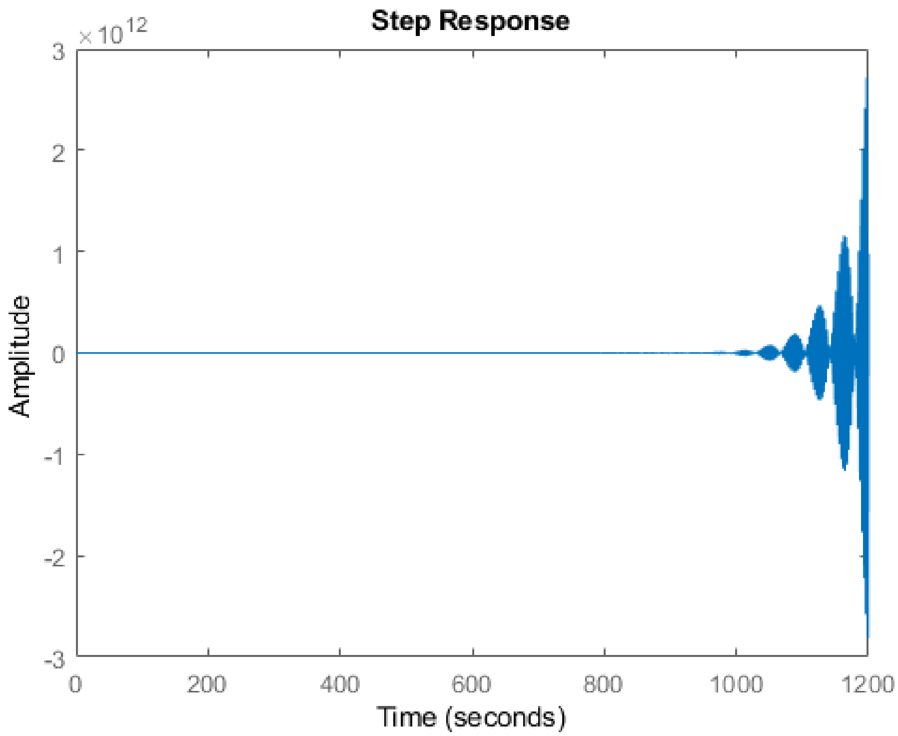

Figure 18 unstable

Poles and zeros

```
zeros=zero(system_tf)
```

```
zeros = -0.4350
```

```
poles=pole(system_tf)
```

```
poles = 4×1 complex
  -3.0570 + 0.0000i
   0.0240 + 2.0020i
   0.0240 - 2.0020i
  -0.4471 + 0.0000i
```

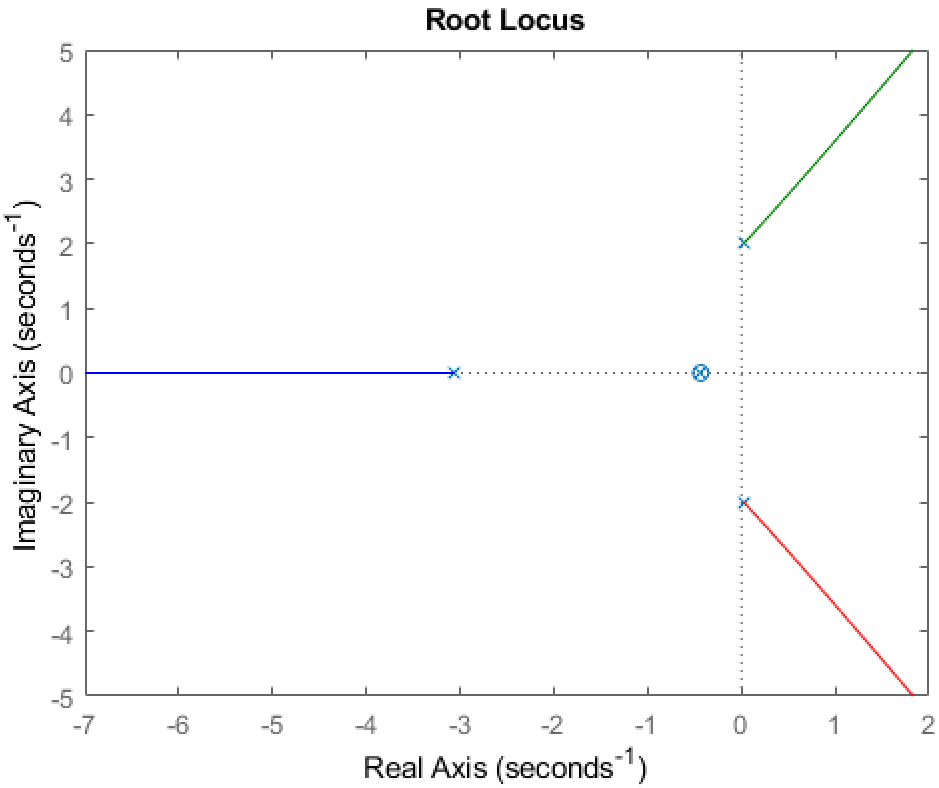

Figure 19 root locus for unstable

Transient parameters

There is no transient parameters for a non-stable system

```
stepinfo(system_tf)
ans = struct with fields:
        RiseTime:     NaN
    SettlingTime:     NaN
     SettlingMin:     NaN
     SettlingMax:     NaN
       Overshoot:     NaN
      Undershoot:     NaN
            Peak:     Inf
        PeakTime: Inf
```

Steady state error

Since this isn't stable there is no steady state error but when using the same code we get this error

Which is huge and indicates that the system isn't stable (if you didn't notice from the poles or the step response).

```
[y,tf]=step(system_tf);
sp=1;%setpoint
ss_error=abs(sp-y(end))
ss_error = 2.8637e+12
```

4-At the limit of stability ($k_1 = 35.196, k_2 = 4$)

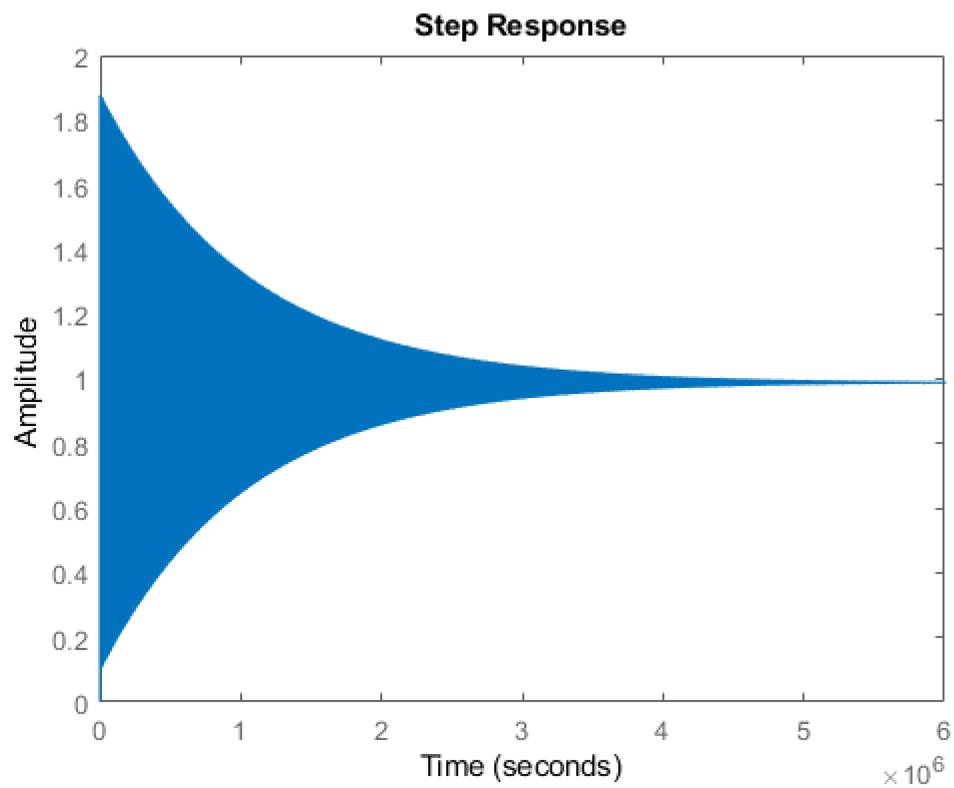

Figure 20 At the limit of stability

Poles and zeros

```
poles=pole(system_tf)
poles = 4×1 complex
  -3.0038                                    +                             0.0000i
  -0.0000                                    +                             1.6878i
  -0.0000                                    -                             1.6878i
  -0.4522 + 0.0000i
```

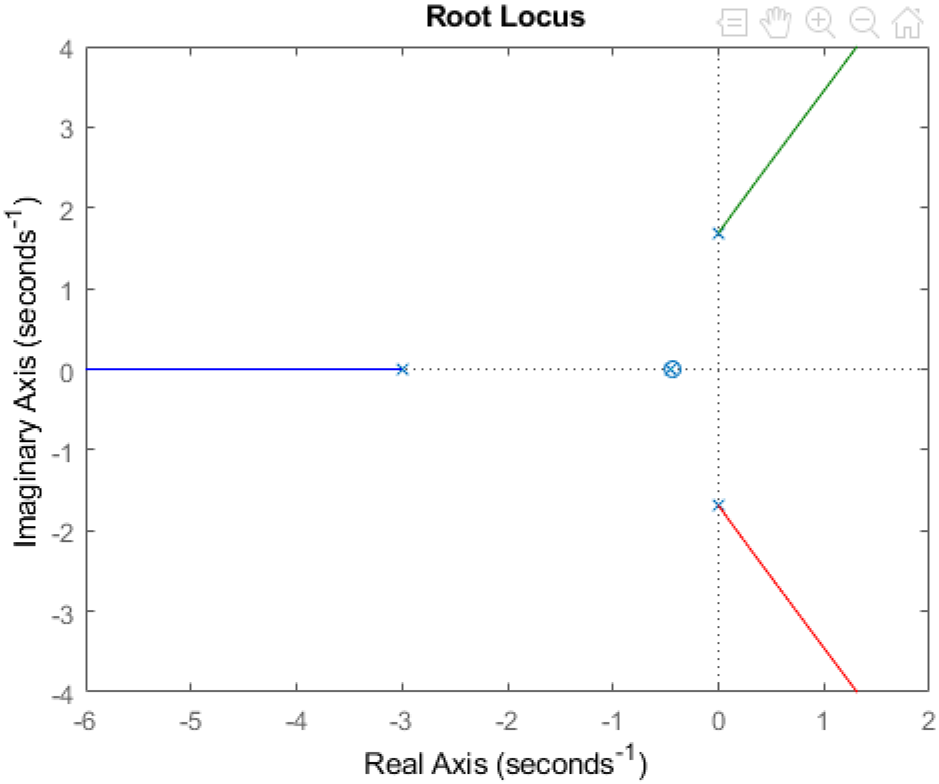

Figure 21 root locus for the limit of stability

Transient parameters

Rise time is high, settling time is so high due to it almost being marginally stable (undamped) but it should have been infinity.

```
stepinfo(system_tf)
ans = struct with fields:
        RiseTime:    179.7022
    SettlingTime:    3.9899e+06
     SettlingMin:    0.0983
     SettlingMax:    1.8772
       Overshoot:    89.7584
      Undershoot:    0
            Peak:    1.8772
        PeakTime: 579.1197
```

Steady state error

Using the same code we get a relatively low steady state error.

```
[y,tf]=step(system_tf);
sp=1;
ss_error=abs(sp-y(end))
ss_error = 0.0137
```

5-The desired system ($k_1 = \mathbf{5}, \boldsymbol{k_2} = \mathbf{8}$)

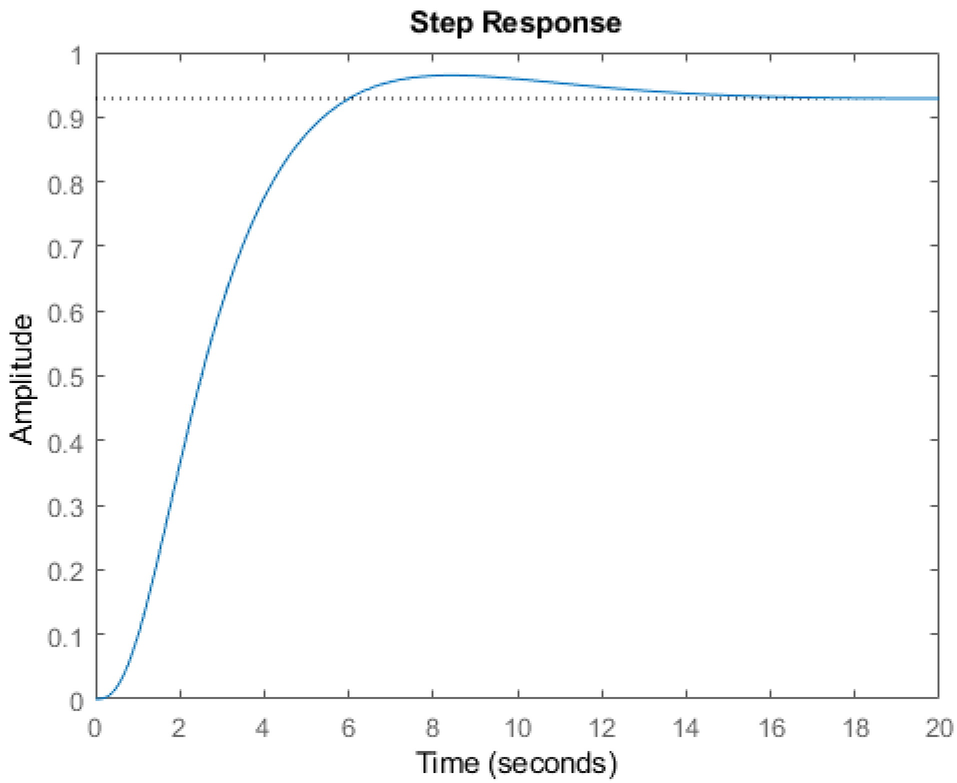

Figure 22 step response k1=5 k2=8

Poles and zeros

```
zeros=zero(system_tf)
zeros = -0.4350
poles=pole(system_tf)
poles = 4×1 complex
  -1.3643 + 1.0805i
  -1.3643 - 1.0805i
  -0.3637 + 0.2469i
  -0.3637 - 0.2469i
```

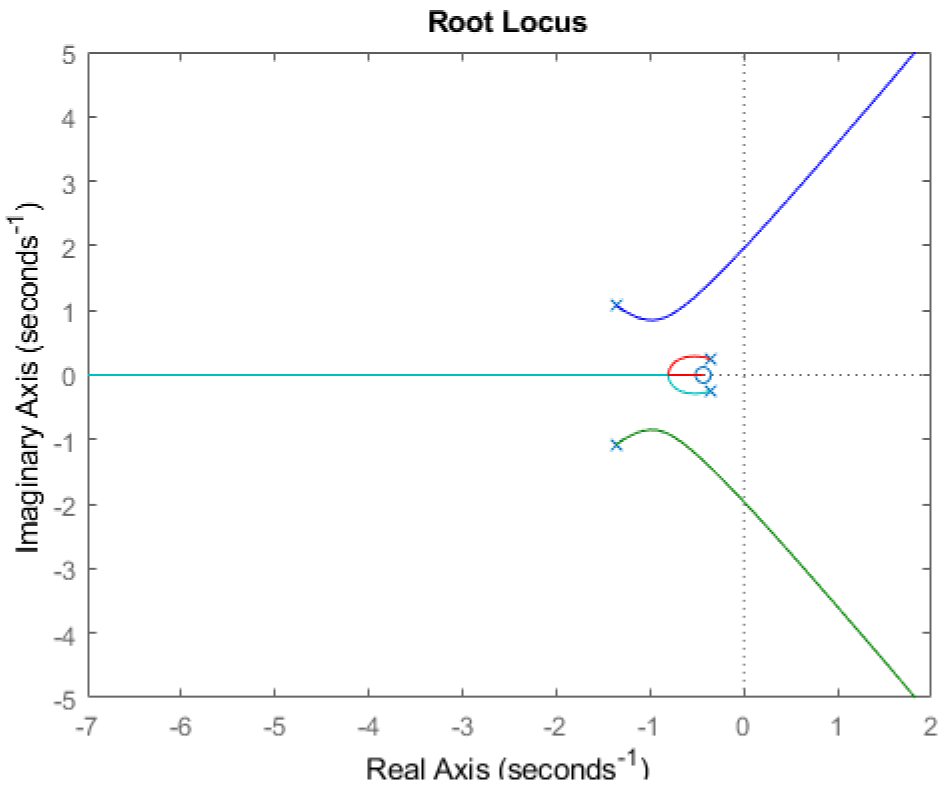

Figure 23 root locus k1=5 k2=8

Transient parameters

```
stepinfo(system_tf)
ans = struct with fields:
         RiseTime: 3.5648
     SettlingTime: 11.7887
      SettlingMin: 0.8402
      SettlingMax: 0.9647
        Overshoot: 3.8475
       Undershoot: 0
             Peak: 0.9647
         PeakTime: 8.4388
```

Steady state error

```
[y,tf]=step(system_tf);
sp=1;
ss_error=abs(sp-y(end))
ss_error = 0.0710
```

The effect of turning the controller off ($k_2 = 0$)

1-stable oscillatory $k_1 = 5$

The transfer function would become like this.

```
system_tf =

                 1.25 s + 0.5437
  ---------------------------------------------
  s^4 + 3.456 s^3 + 3.207 s^2 + 1.861 s + 0.5853
```

the poles and zeros would be

```
zeros=zero(system_tf)
zeros = -0.4350
poles=pole(system_tf)
poles = 4×1 complex
  -2.4007 + 0.0000i
  -0.2239 + 0.5927i
  -0.2239 - 0.5927i
  -0.6074 + 0.0000
```

The root locus plot:

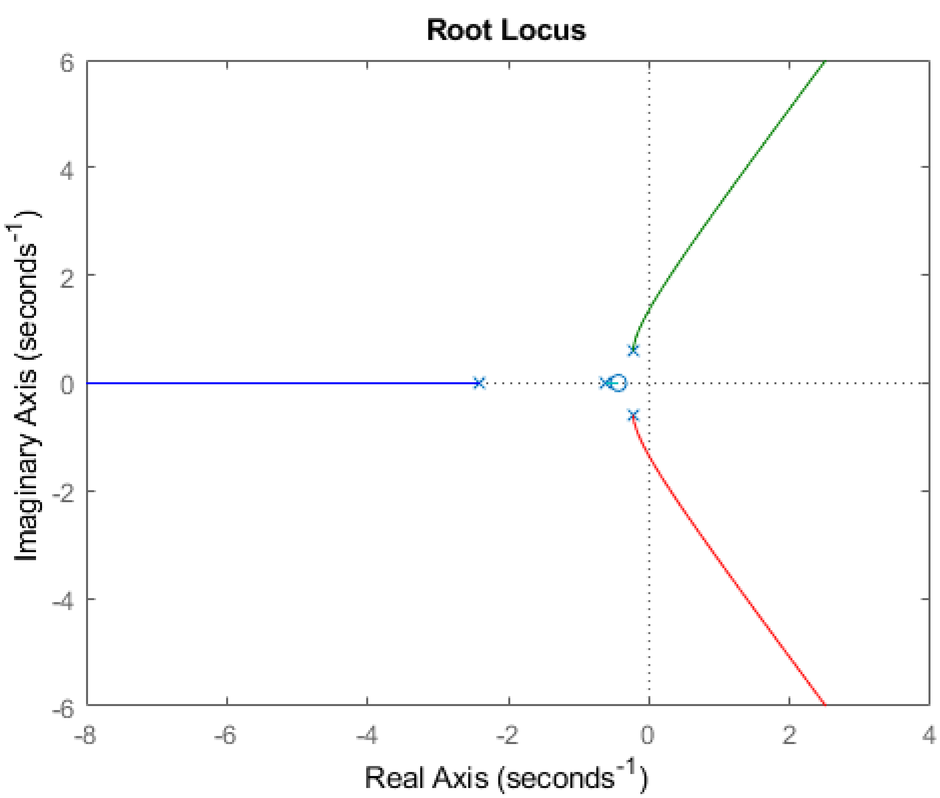

Figure 24 root locus for $k_2 = 0$ $k_1 = 5$

Step response:

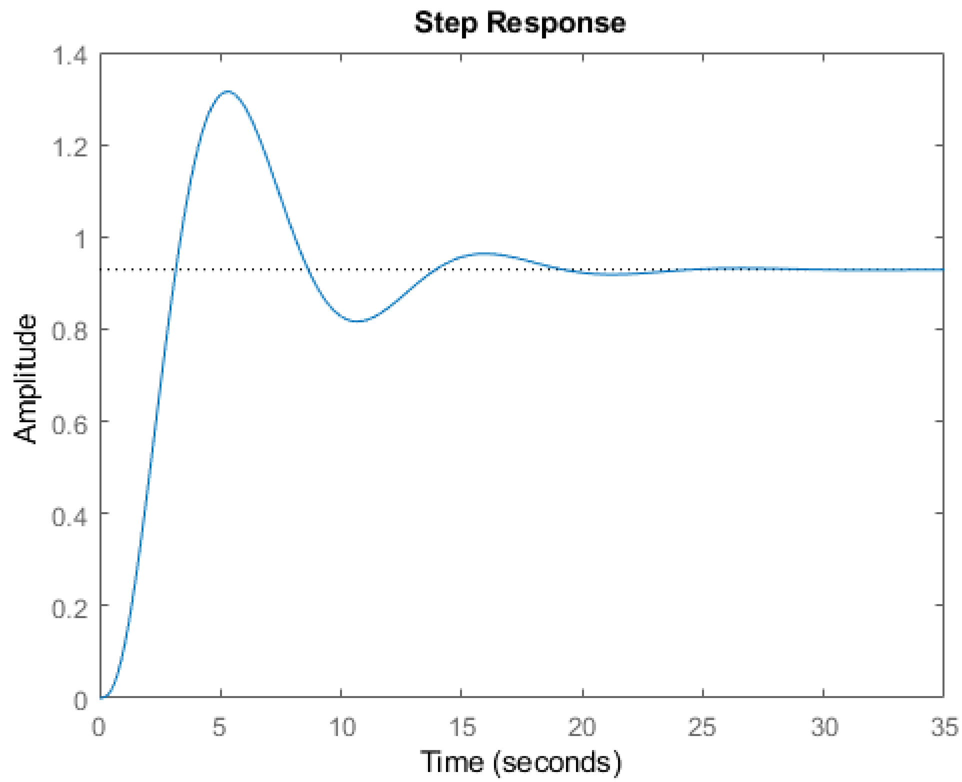

Figure 25 Step response for k2=0 k1=5

Transient parameters:

```
stepinfo(system_tf)
ans = struct with fields:
        RiseTime: 1.9688
    SettlingTime: 17.7732
     SettlingMin: 0.8166
     SettlingMax: 1.3152
       Overshoot: 41.5771
      Undershoot: 0
            Peak: 1.3152
        PeakTime: 5.3071
```

Steady state error

```
[y,tf]=step(system_tf);
sp=1;
ss_error=abs(sp-y(end))
ss_error = 0.0719
```

2-Unstable $k_1 = 50$,

The transfer function would become like this:

```
system_tf =

                 12.5 s + 5.438
  ---------------------------------------------
  s^4 + 3.456 s^3 + 3.207 s^2 + 13.11 s + 5.479
```

the poles and zeros would be:

```
zeros=zero(system_tf)
zeros = -0.4350
poles=pole(system_tf)
poles = 4×1 complex
  -3.4857 + 0.0000i
   0.2379 + 1.8617i
   0.2379 - 1.8617i
  -0.4462 + 0.0000i
```

The root locus plot:

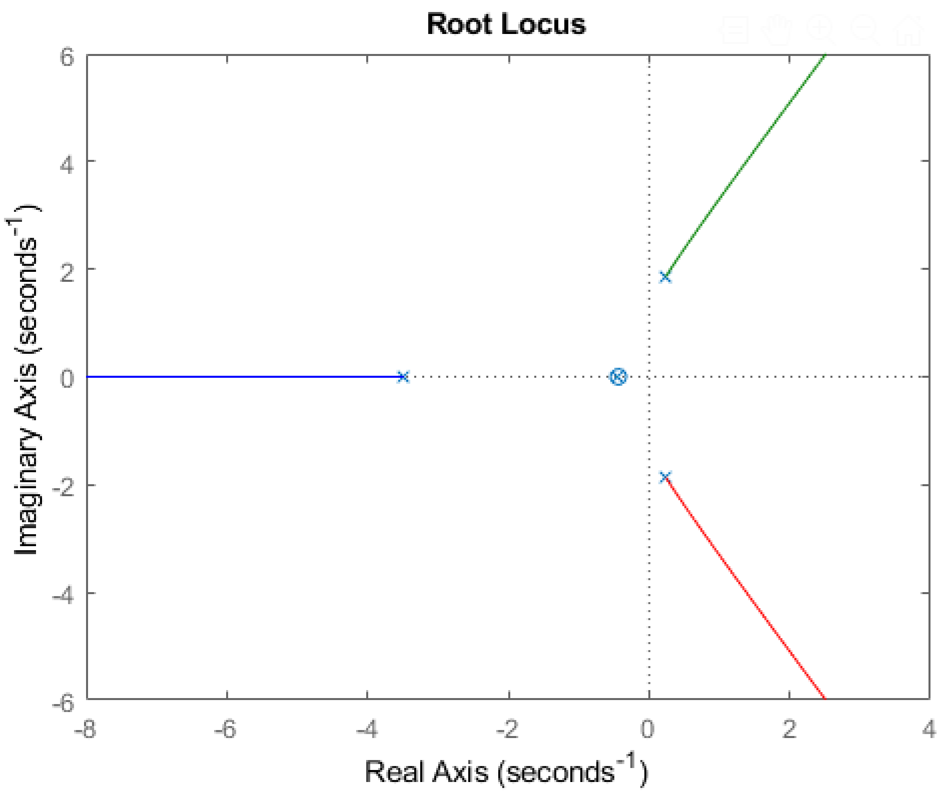

Figure 26 root locus for $k_2 = 0$ $k_1 = 50$(unstable)

Step response:

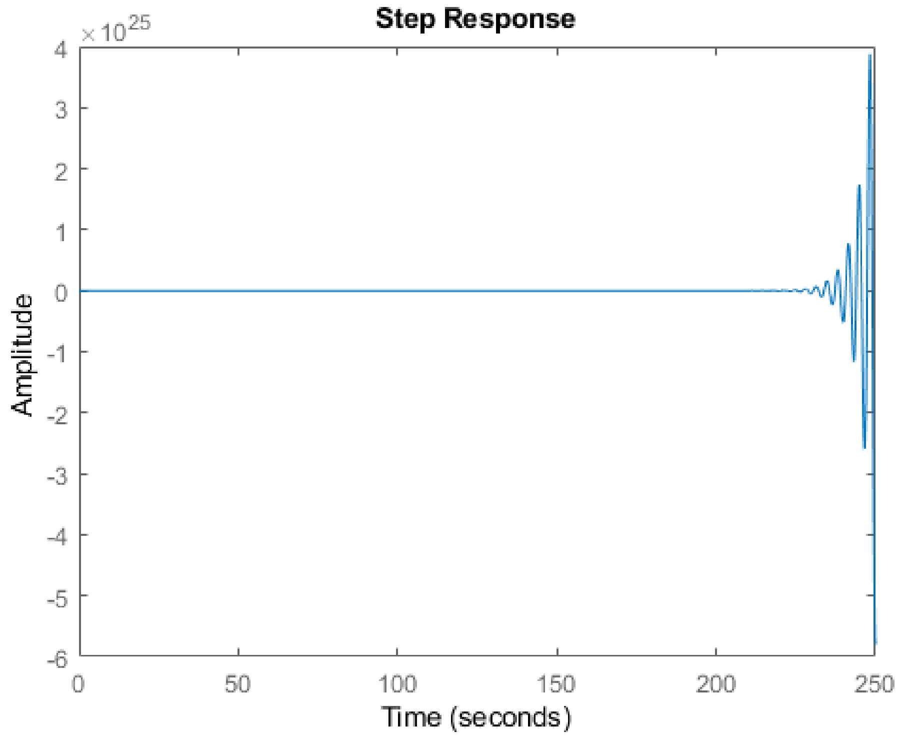

Figure 27 Step response for k2=0 k1=50(unstable)

Transient parameters:

No info about the transient parameters because its unstable

```
stepinfo(system_tf)
ans = struct with fields:
        RiseTime: NaN
    SettlingTime: NaN
     SettlingMin: NaN
     SettlingMax: NaN
       Overshoot: NaN
      Undershoot: NaN
            Peak: Inf
        PeakTime: Inf
```

Steady state error

(no steady state) not stable

```
[y,tf]=step(system_tf);
sp=1;
ss_error=abs(sp-y(end))
ss_error = 2.2096e+24
```

3-stable non oscillatory $k_1 = 0.001$

The transfer function would become like this:

```
system_tf =

system_tf =

               0.00025 s + 0.0001088
  ------------------------------------------------
  s^4 + 3.456 s^3 + 3.207 s^2 + 0.6108 s + 0.04168
```

the poles and zeros would be:

```
zeros=zero(system_tf)
zeros = -0.4350
poles=pole(system_tf)
poles = 4×1 complex
  -2.0001 + 0.0000i
  -1.2298 + 0.0000i
  -0.1130 + 0.0646i
  -0.1130 - 0.0646i
```

The root locus plot (pole-zero map):

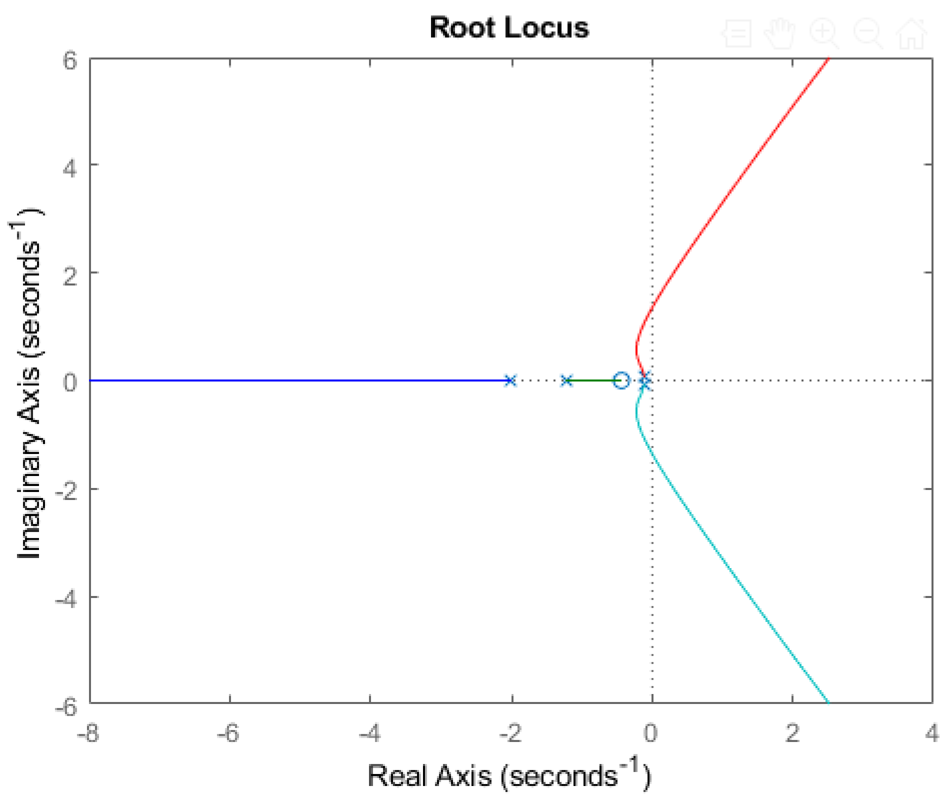

Figure 28 root locus for $k_2 = 0$ $k_1 = 0.001$(stable non oscillatory)

Step response:

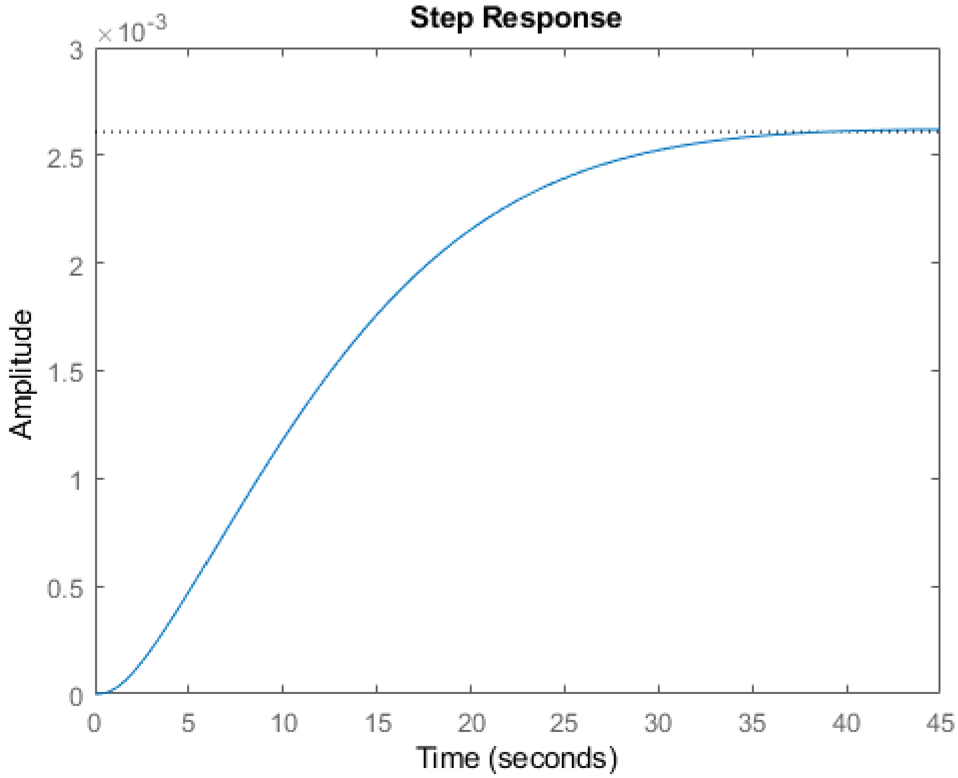

Figure 29 Step response for k2=0 k1=0.001(stable non oscillatory)

Transient parameters:

```
stepinfo(system_tf)
ans = struct with fields:
        RiseTime: 20.3699
    SettlingTime: 32.1435
     SettlingMin: 0.0024
     SettlingMax: 0.0026
       Overshoot: 0.4327
      Undershoot: 0
            Peak: 0.0026
        PeakTime: 46.8536
```

Steady state error

```
[y,tf]=step(system_tf);
sp=1;
ss_error=abs(sp-y(end))
ss_error = 0.9974
```

5-canonical controller state-space representation of the feedback system

```
[A, B, C, D] = tf2ss(system_tf.num{1}, system_tf.den{1})%controller canonical form
```

```
A = 4×4
   -3.4560   -5.2069   -2.7305   -0.5853
    1.0000         0         0         0
         0    1.0000         0         0
         0         0    1.0000         0
B = 4×1
     1
     0
     0
     0
C = 1×4
         0         0    1.2500    0.5437
D = 0
```