Task 1:

```
clear all
clc
```

a)

```
% First enter the transfer function G(s)
numG = 1;
denG = conv ( conv ( [1 0], [1 1] ), [0.2 1] );
% Convert to state-space model
[ Ag, Bg, Cg, Dg ] = tf2ss ( numG, denG );
% Check the controllability and observability of the system
    system_order = length(Ag) % equals "3"
```

```
system_order = 3
```

```
M = ctrb(Ag, Bg);
rank_of_M = rank(M) % equals "3"
```

```
rank_of_M = 3
```

```
N = obsv(Ag, Cg);
rank_of_N = rank(N) % equals "3"
```

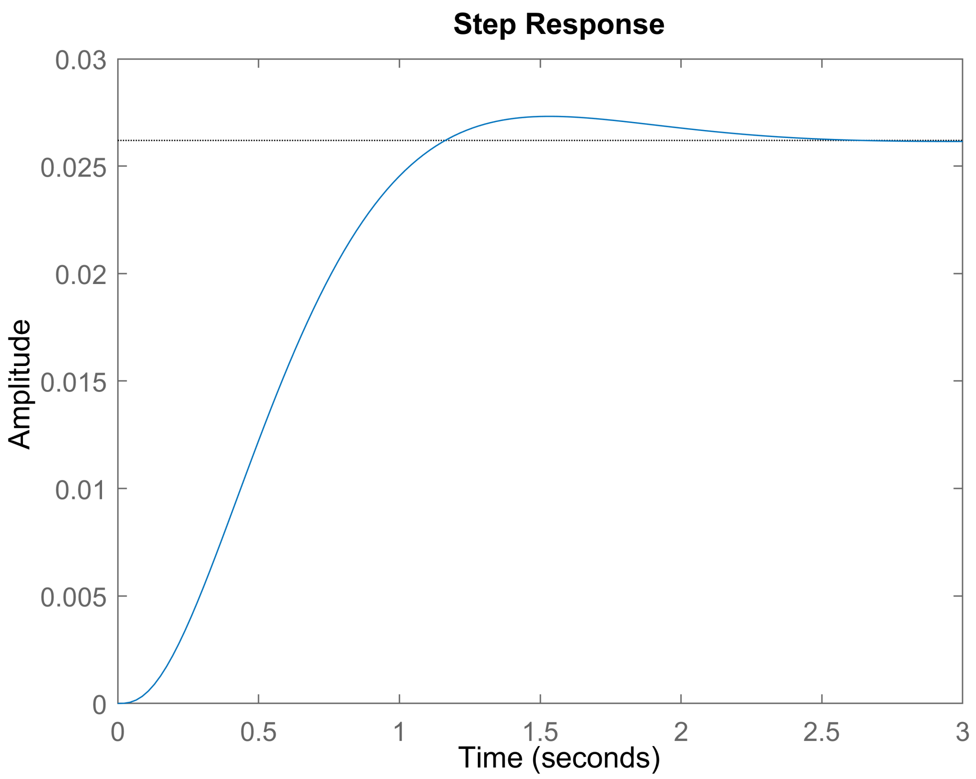
```
rank_of_N = 3
```

```
% Compute the poles of a second-order system
damping = 0.707;
wn = 3;
[ num2, den2 ] = ord2 (wn, damping);
% Select desired poles to include poles of the second-order system
dominant = roots(den2);
desiredpoles = [dominant' 10 * real( dominant(1) ) ];
% Compute the controller gain K
K = acker (Ag, Bg, desiredpoles);
% Compute the closed-loop state variable feedback system
Asf = Ag - Bg * K; Bsf = Bg; Csf = Cg; Dsf = 0;
[numsf, densf] = ss2tf (Asf, Bsf, Csf, Dsf);
% Select observer poles to be 10 times faster than controller poles
observerpoles = 10 * desiredpoles;
% Compute the observer gain L
L = acker (Ag', Cg', observerpoles);
% Compute the closed-loop system with both controller and observer
Areg = [ (Ag - Bg * K) Bg * K; zeros( size(Ag) ) (Ag - L' * Cg) ];
Breg = [ Bg; zeros( size(Bg) ) ];
Creg = [ Cg zeros( size(Cg) ) ];
Dreg = 0;
[numreg, denreg] = ss2tf ( Areg, Breg, Creg, Dreg );
damp(denreg);
```

```
         Pole              Damping       Frequency      Time Constant
                                       (rad/TimeUnit)    (TimeUnit)
```

```
 -2.12e+02                 1.00e+00       2.12e+02         4.71e-03
 -2.12e+01 + 2.12e+01i     7.07e-01       3.00e+01         4.71e-02
 -2.12e+01 - 2.12e+01i     7.07e-01       3.00e+01         4.71e-02
 -2.12e+01                 1.00e+00       2.12e+01         4.71e-02
 -2.12e+00 + 2.12e+00i     7.07e-01       3.00e+00         4.71e-01
 -2.12e+00 - 2.12e+00i     7.07e-01       3.00e+00         4.71e-01
```

```
G=tf(numsf,densf);
step(numsf,densf);
```



```
stepinfo(G,'RiseTimeLimits',[0.1 0.9])
```

```
ans = struct with fields:
         RiseTime: 0.7242
    TransientTime: 2.0357
     SettlingTime: 2.0357
      SettlingMin: 0.0239
      SettlingMax: 0.0273
        Overshoot: 4.2749
       Undershoot: 0
             Peak: 0.0273
         PeakTime: 1.5416
```

```
K
```

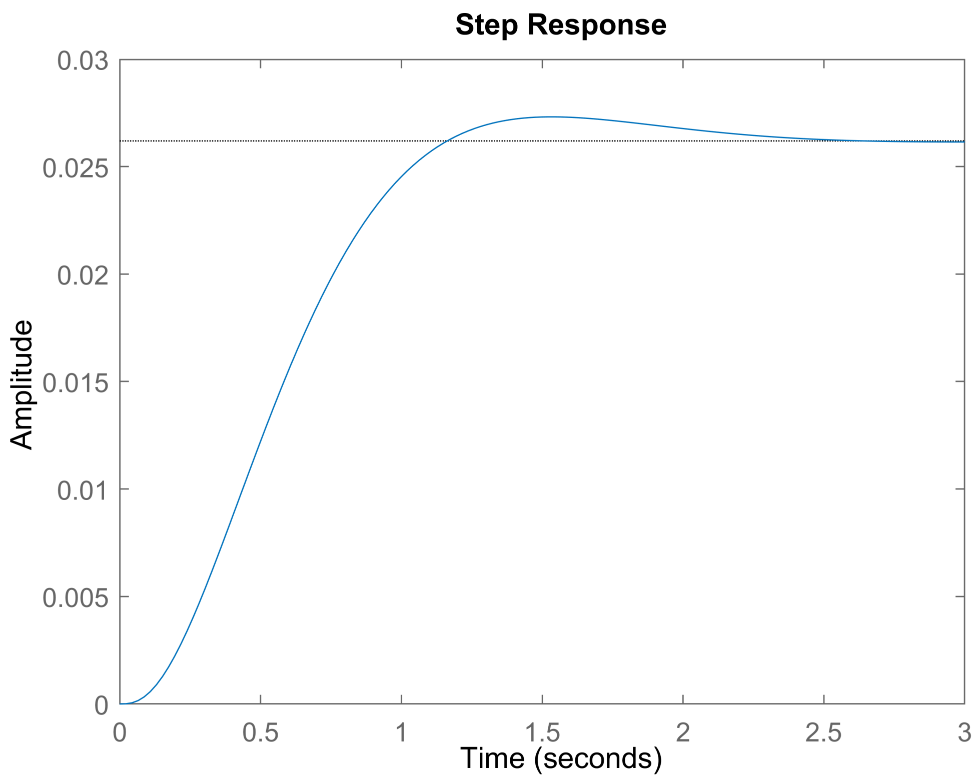
```
K = 1×3
   19.4520   93.9728  190.8900
```

```
desiredpoles2 = [dominant' 20 * real( dominant(1) ) ];
% Compute the controller gain K
K2 = acker (Ag, Bg, desiredpoles2);
```

```
% Compute the closed-loop state variable feedback system
Asf = Ag - Bg * K; Bsf = Bg; Csf = Cg; Dsf = 0;
[numsf2, densf2] = ss2tf (Asf, Bsf, Csf, Dsf);
% Select observer poles to be 10 times faster than controller poles
observerpoles2 = 10 * desiredpoles2;
% Compute the observer gain L
L2 = acker (Ag', Cg', observerpoles2);
% Compute the closed-loop system with both controller and observer
Areg2 = [ (Ag - Bg * K) Bg * K; zeros( size(Ag) ) (Ag - L2' * Cg) ];
Breg2 = [ Bg; zeros( size(Bg) ) ];
Creg2 = [ Cg zeros( size(Cg) ) ];
Dreg2 = 0;
[numreg2, denreg2] = ss2tf ( Areg2, Breg2, Creg2, Dreg2 );
damp(denreg2);
```

```
         Pole                 Damping       Frequency      Time Constant
                                          (rad/TimeUnit)    (TimeUnit)

 -4.24e+02                    1.00e+00       4.24e+02         2.36e-03
 -2.12e+01 + 2.12e+01i        7.07e-01       3.00e+01         4.71e-02
 -2.12e+01 - 2.12e+01i        7.07e-01       3.00e+01         4.71e-02
 -2.12e+01                    1.00e+00       2.12e+01         4.71e-02
 -2.12e+00 + 2.12e+00i        7.07e-01       3.00e+00         4.71e-01
 -2.12e+00 - 2.12e+00i        7.07e-01       3.00e+00         4.71e-01
```

```
G2=tf(numsf2,densf2);
step(numsf2,densf2);
```



```
stepinfo(G2,'RiseTimeLimits',[0.1 0.9])
```

```
ans = struct with fields:
         RiseTime: 0.7242
    TransientTime: 2.0357
     SettlingTime: 2.0357
      SettlingMin: 0.0239
      SettlingMax: 0.0273
        Overshoot: 4.2749
       Undershoot: 0
             Peak: 0.0273
         PeakTime: 1.5416
```

```
K2
```

```
K2 = 1×3
   40.6620  183.9456  381.7800
```

In this example we can observe that the gain calculated with the regulator.m file is approximately 2 times smaller, than the gain of the initial system. Correspondingly the numbers obtained were: 40.66, 183,94 & 381.78 without regulator.m and 19.45, 93.97 & 190.89 with it. The change in gain has an influence on the position of the poles of the system. We know that as the gain continues to increase the location of closed-loop poles moves from the location of open-loop poles to the location of open-loop zeros. The overall dynamics of the system with and without regulator.m have not been significantly affected with the exception of frequency, which is directly related to the gain of the poles.

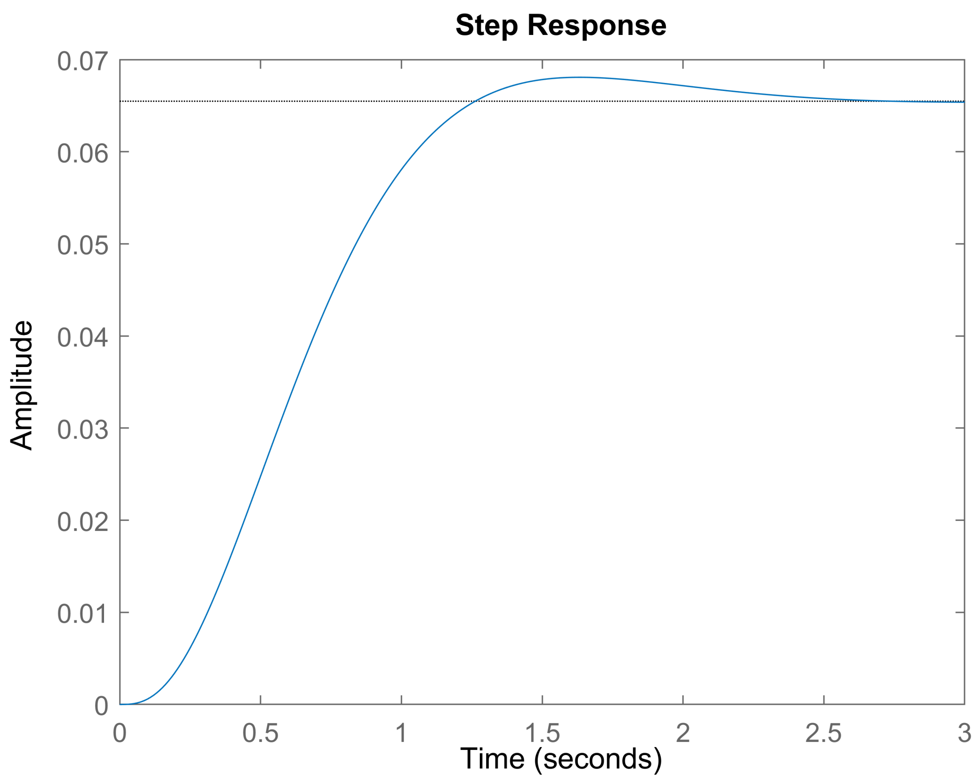
b)

```
desiredpoles = [dominant' 4 * real( dominant(1) ) ];
% Compute the controller gain K.
K = acker (Ag, Bg, desiredpoles);
% Compute the closed-loop state variable feedback system.
Asf = Ag - Bg * K; Bsf = Bg; Csf = Cg; Dsf = 0;
[numsf, densf] = ss2tf (Asf, Bsf, Csf, Dsf);
% Select observer poles to be 10 times faster than controller.
observerpoles = 10 * desiredpoles;
% Compute observer gain L.
L = acker (Ag', Cg', observerpoles);
% Compute the closed-loop system with controller and observer.
Areg = [ (Ag - Bg * K) Bg * K; zeros( size(Ag) ) (Ag - L' *Cg) ];
Breg = [ Bg; zeros( size(Bg) ) ];
Creg = [ Cg zeros ( size(Cg) ) ];
Dreg = 0;
[numreg, denreg] = ss2tf ( Areg, Breg, Creg, Dreg );
damp (denreg);
```

```
         Pole                Damping        Frequency       Time Constant
                                          (rad/TimeUnit)     (TimeUnit)

 -8.48e+01                   1.00e+00        8.48e+01          1.18e-02
 -2.12e+01 + 2.12e+01i       7.07e-01        3.00e+01          4.71e-02
 -2.12e+01 - 2.12e+01i       7.07e-01        3.00e+01          4.71e-02
 -8.48e+00                   1.00e+00        8.48e+00          1.18e-01
```

```
  -2.12e+00 + 2.12e+00i     7.07e-01       3.00e+00          4.71e-01
  -2.12e+00 - 2.12e+00i     7.07e-01       3.00e+00          4.71e-01
```

```
G2=tf(numsf,densf);
step(numsf,densf);
```



```
stepinfo(G2,'RiseTimeLimits',[0.1 0.9])
```

```
ans = struct with fields:
         RiseTime: 0.7647
    TransientTime: 2.1069
     SettlingTime: 2.1069
      SettlingMin: 0.0593
      SettlingMax: 0.0681
        Overshoot: 3.9657
       Undershoot: 0
             Peak: 0.0681
         PeakTime: 1.6284
```

```
K
```

```
K = 1×3
    6.7260   39.9891   76.3560
```

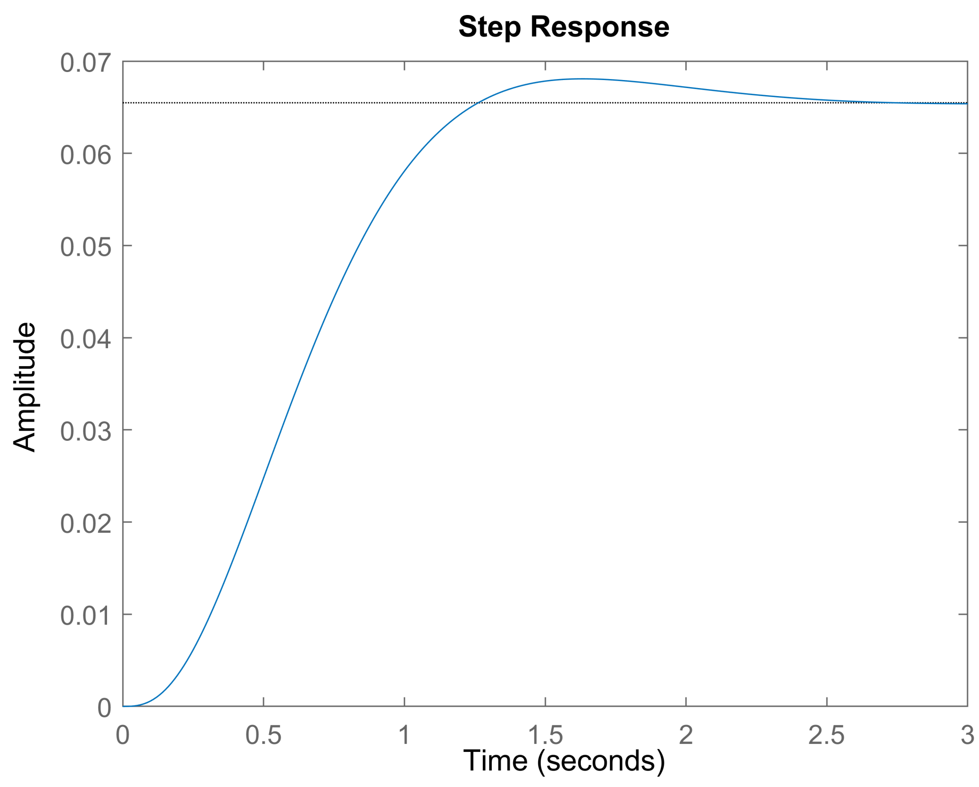
Compared to the other 2 set ups, in this case the obtained gain K is smaller, therefore the poles are closer to the imaginary axis. In general the closer non-dominant poles are to the dominant pole, the more significant effect they have on the overall system performance. In particular in our case they resulted in higher rise time and peak time compared to the previous instances.

c)

```
desiredpoles4 = [dominant' 10 * real( dominant(1) ) ];
% Compute the controller gain K.
K4 = acker (Ag, Bg, desiredpoles4);
% Compute the closed-loop state variable feedback system.
Asf = Ag - Bg * K; Bsf = Bg; Csf = Cg; Dsf = 0;
[numsf4, densf4] = ss2tf (Asf, Bsf, Csf, Dsf);
% Select observer poles to be 10 times faster than controller.
observerpoles4 = 20 * desiredpoles4;
% Compute observer gain L.
L4 = acker (Ag', Cg', observerpoles4);
% Compute the closed-loop system with controller and observer.
Areg4 = [ (Ag - Bg * K) Bg * K; zeros( size(Ag) ) (Ag - L4' *Cg) ];
Breg4 = [ Bg; zeros( size(Bg) ) ];
Creg4 = [ Cg zeros ( size(Cg) ) ];
Dreg4 = 0;
[numreg4, denreg4] = ss2tf ( Areg4, Breg4, Creg4, Dreg4 );
damp (denreg4);
```

```
         Pole                Damping       Frequency       Time Constant
                                         (rad/TimeUnit)      (TimeUnit)

 -4.24e+02                   1.00e+00       4.24e+02          2.36e-03
 -4.24e+01 + 4.24e+01i       7.07e-01       6.00e+01          2.36e-02
 -4.24e+01 - 4.24e+01i       7.07e-01       6.00e+01          2.36e-02
 -8.48e+00                   1.00e+00       8.48e+00          1.18e-01
 -2.12e+00 + 2.12e+00i       7.07e-01       3.00e+00          4.71e-01
 -2.12e+00 - 2.12e+00i       7.07e-01       3.00e+00          4.71e-01
```

```
G4=tf(numsf4,densf4);
step(numsf4,densf4);
```



```
stepinfo(G4,'RiseTimeLimits',[0.1 0.9])
```

```
ans = struct with fields:
         RiseTime: 0.7647
    TransientTime: 2.1069
     SettlingTime: 2.1069
      SettlingMin: 0.0593
      SettlingMax: 0.0681
        Overshoot: 3.9657
       Undershoot: 0
             Peak: 0.0681
         PeakTime: 1.6284
```

```
K4
```

```
K4 = 1×3
   19.4520   93.9728  190.8900
```

In this case, by increasing the distance of the observer poles (or their "speed") the values in the original L matrix are significantly smaller (2.7848e+04, 1.6802e+03, 49.7040 vs 2.6104e+05, 7.3132e+03, 100.6080). When the observer poles are faster, the state estimation is more responsive to changes in the system's dynamics. This can be beneficial in some cases because it allows the controller to respond quickly to changes in the system and improve the system's performance. However, faster observer poles can also introduce noise amplification and sensitivity to modeling errors, which can degrade the system's stability and performance.

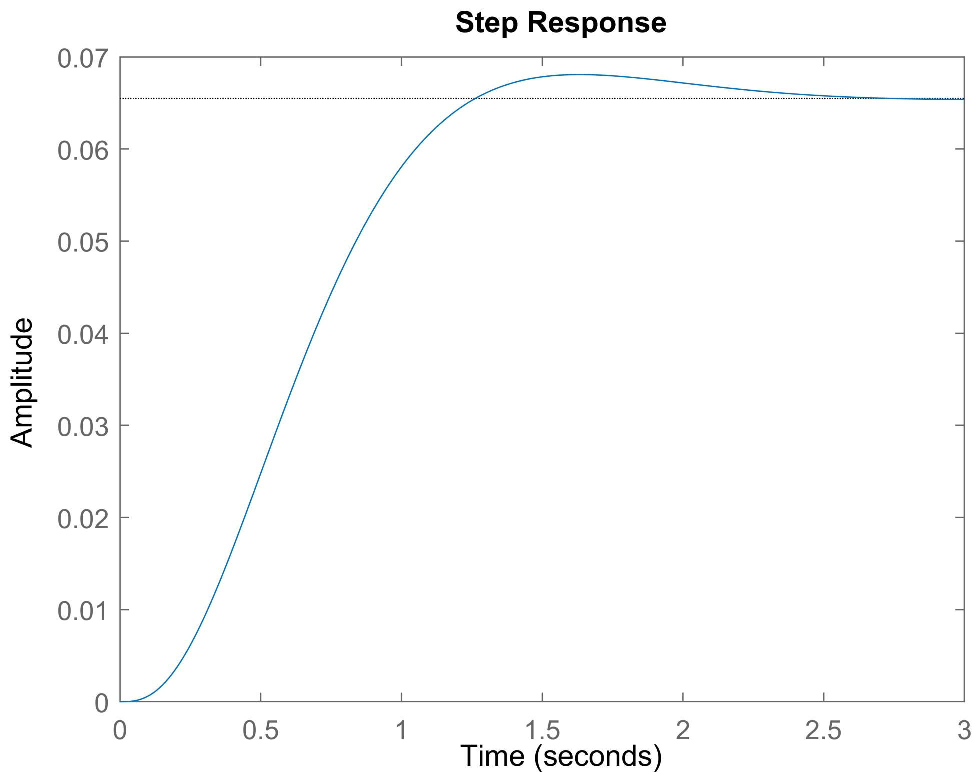
d)

```
desiredpoles5 = [dominant' 10 * real( dominant(1) ) ];
% Compute the controller gain K.
```

```
K5 = acker (Ag, Bg, desiredpoles5);
% Compute the closed-loop state variable feedback system.
Asf = Ag - Bg * K; Bsf = Bg; Csf = Cg; Dsf = 0;
[numsf5, densf5] = ss2tf (Asf, Bsf, Csf, Dsf);
% Select observer poles to be 10 times faster than controller.
observerpoles5 = 4 * desiredpoles5;
% Compute observer gain L.
L5 = acker (Ag', Cg', observerpoles5);
% Compute the closed-loop system with controller and observer.
Areg5 = [ (Ag - Bg * K) Bg * K; zeros( size(Ag) ) (Ag - L5' *Cg) ];
Breg5 = [ Bg; zeros( size(Bg) ) ];
Creg5 = [ Cg zeros ( size(Cg) ) ];
Dreg5 = 0;
[numreg5, denreg5] = ss2tf ( Areg5, Breg5, Creg5, Dreg5 );
damp (denreg5);
```

```
         Pole                  Damping       Frequency      Time Constant
                                           (rad/TimeUnit)     (TimeUnit)

 -8.48e+01                     1.00e+00       8.48e+01         1.18e-02
 -8.48e+00 + 8.49e+00i         7.07e-01       1.20e+01         1.18e-01
 -8.48e+00 - 8.49e+00i         7.07e-01       1.20e+01         1.18e-01
 -8.48e+00                     1.00e+00       8.48e+00         1.18e-01
 -2.12e+00 + 2.12e+00i         7.07e-01       3.00e+00         4.71e-01
 -2.12e+00 - 2.12e+00i         7.07e-01       3.00e+00         4.71e-01
```

```
G5=tf(numsf5,densf5);
step(numsf5,densf5);
```



```
stepinfo(G5,'RiseTimeLimits',[0.1 0.9])
```

```
ans = struct with fields:
         RiseTime: 0.7647
    TransientTime: 2.1069
     SettlingTime: 2.1069
      SettlingMin: 0.0593
      SettlingMax: 0.0681
        Overshoot: 3.9657
       Undershoot: 0
             Peak: 0.0681
         PeakTime: 1.6284
```

```
K5
```

```
K5 = 1×3
   19.4520   93.9728  190.8900
```

In this case observer gain L was equal to: 1.1431e+03 200.7434 19.1616. It's lower than in the previous case, because the "speed" of observer poles was smaller. There were no changes in RiseTime, Overshoot, PeakTime and so on. On the other hand we can see changes in poles, damping, frequency and time constant.

e)

```
for wn=3:-0.001:0
     [ num2, den2 ] = ord2 (wn, damping);
     dominant=roots(den2);
     desiredpoles = [dominant' 10 * real( dominant(1) ) ];
     K = acker (Ag, Bg, desiredpoles);

     if ((K(3)<10)&&(K(2)<10)&&(K(1)<10))
         wn
         K
         break
     end

end
```

```
wn = 1.1220
K = 1×3
    3.5190    8.8439    9.9861
```

In this exercise we have shown that the maximum natural frequency omega n obtained while all the elements of matrix K are smaller than 10 is 1.122.

Task 2:

```
clear all
clc
```

```
%system nb 1
A1=[1 4 3; 0 2 16; 0 -25 -20]
```

```
A1 = 3×3
     1     4     3
     0     2    16
     0   -25   -20
```

```
B1=[-1;0;0]
```

```
B1 = 3×1
    -1
     0
     0
```

```
C1=[-1 3 0]
```

```
C1 = 1×3
    -1     3     0
```

```
D1=[0]
```

```
D1 = 0
```

```
%system nb 2
A2=[1 0 0; 0 0 0; -2 -4 -3]
```

```
A2 = 3×3
     1     0     0
     0     0     0
    -2    -4    -3
```

```
B2=[-1;0;-1]
```

```
B2 = 3×1
    -1
     0
    -1
```

```
C2=[1 0 0]
```

```
C2 = 1×3
     1     0     0
```

```
D2=[0]
```

```
D2 = 0
```

```
%checking controllability of the 1st system
poles1 = eig(A1) %one unstable pole at p=1
```

```
poles1 = 3×1 complex
   1.0000 + 0.0000i
  -9.0000 +16.7033i
  -9.0000 -16.7033i
```

```
%finding controllability staircase form
[Af1, Bf1, Cf1, T1, k1] = ctrbf(A1, B1, C1)
```

```
Af1 = 3×3
   -20   -25     0
    16     2     0
    -3    -4     1
Bf1 = 3×1
     0
     0
     1
Cf1 = 1×3
     0     3     1
T1 = 3×3
     0     0     1
     0     1     0
    -1     0     0
k1 = 1×3
     1     0     0
```

```

% the poles of the transformed system are the same as the
% ones of the original system.
poles1_new = eig(Af1)
```

```
poles1_new = 3×1 complex
   1.0000 + 0.0000i
  -9.0000 +16.7033i
  -9.0000 -16.7033i
```

```
controllable_poles = eig(Af1(3,3))
```

```
controllable_poles = 1
```

```
uncontrollable_poles = eig(Af1(1:2, 1:2))
```

```
uncontrollable_poles = 2×1 complex
  -9.0000 +16.7033i
  -9.0000 -16.7033i
```

```
system_order = length(A1)
```

```
system_order = 3
```

```
M = ctrb(A1, B1);
rank_of_M = rank(M)
```

```
rank_of_M = 1
```

```
%checking observability of the 1st system
N = obsv(A1, C1);
rank_of_N = rank(N)
```

```
rank_of_N = 3
```

The 1st system is observable,and beacouse the one unstable pole is controllable we can design a variable feedback controller to stabilize it. This system has one unstable pole at p=1, the system is therefore unstable.

```
%checking controllability of the 2nd system
poles2 = eig(A2) %one unstable pole at p=1 and one pole at the origin p=0
```

```
poles2 = 3×1
    -3
     1
     0
```

```
%finding controllability staircase form
[Af2, Bf2, Cf2, T2, k2] = ctrbf(A2, B2, C2)
```

```
Af2 = 3×3
         0         0         0
   -2.8284   -0.0000    3.0000
    2.8284    1.0000   -2.0000
Bf2 = 3×1
         0
   -0.0000
    1.4142
Cf2 = 1×3
         0   -0.7071   -0.7071
T2 = 3×3
         0    1.0000         0
   -0.7071         0    0.7071
   -0.7071         0   -0.7071
k2 = 1×3
     1     1     0
```

```


% the poles of the transformed system are the same as the
% ones of the original system.
poles2_new = eig(Af2)
```

```
poles2_new = 3×1
   -3.0000
    1.0000
         0
```

```
controllable_poles = eig(Af2(2:3,2:3))
```

```
controllable_poles = 2×1
    1.0000
   -3.0000
```

```
uncontrollable_poles = eig(Af2(1, 1))
```

```
uncontrollable_poles = 0
```

```
system_order = length(A2)
```

```
system_order = 3
```

```
M = ctrb(A2, B2);
rank_of_M = rank(M)
```

```
rank_of_M = 2
```

```
%checking observability of the 2nd system
N = obsv(A2, C2);
```

```
rank_of_N = rank(N)
```

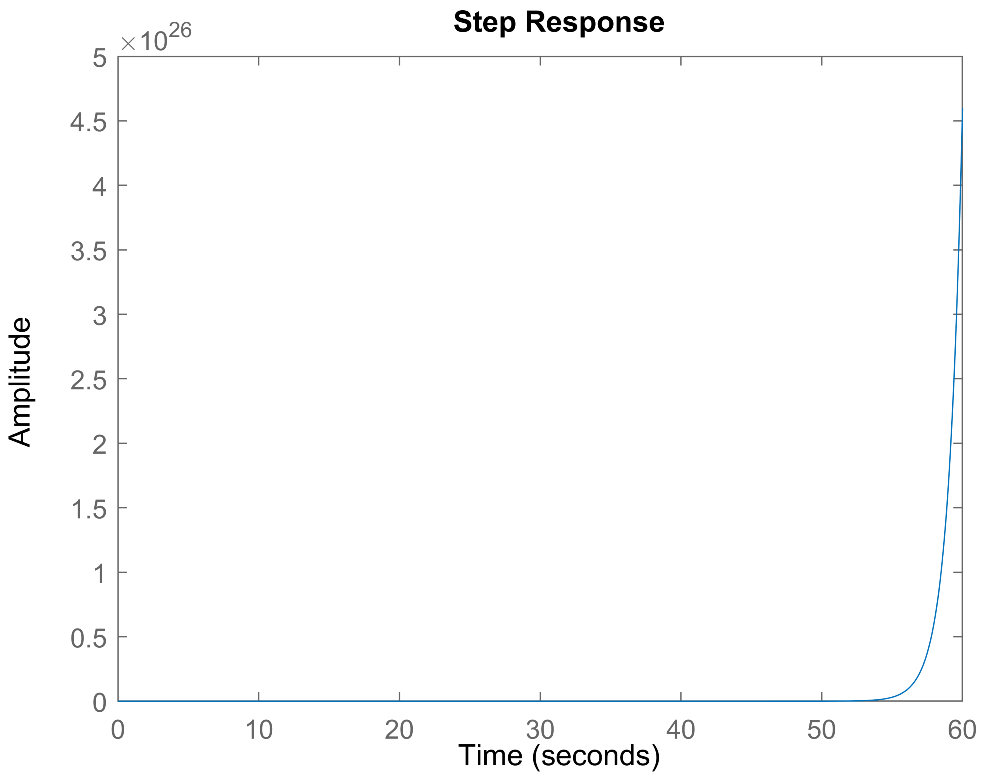
```
rank_of_N = 1
```

In the case of 2nd system the pole at p=0 is uncontrollable, therefore we cannot move it and as a result we are unable to create variable feedback controller for this system. Similarly to the system nb 1 it has an unstable pole at p=1, therfefore it is also unstable.

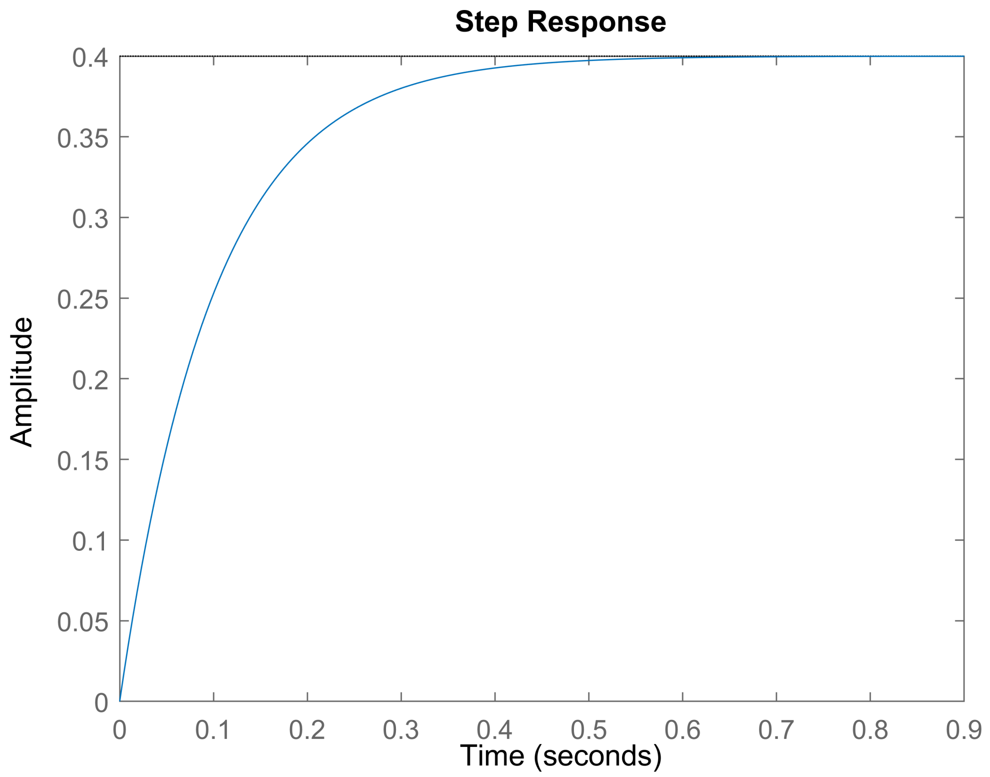
Task 3:

```
clear all
clc
```

```
% given state-space model:
A = [ -5, -2; -3 0];
B = [ 1; -3];
C = [1, -1];
D = 0;
%checking observability
N = obsv(A,C);
%checking controllability
[Af, Bf, Cf] = ctrbf(A,B,C); %unstable pole p=1 is controllable
controllable_poles = eig(Af(2,2));
%applying Ackerman's formula to design variable feedback controller
char_eqn = [ 1, 16, 60];
desired_poles = roots(char_eqn);
L = acker( A', C', desired_poles);
hfl = L'*C;
Areg = minus(A, hfl);
Breg = B;
Creg = C;
Dreg = D;
%converting stabilized system to transfer function form
[numreg, denreg] = ss2tf(Areg, Breg, Creg, Dreg);
G2 = tf(numreg, denreg);
%information about initial system
rank_of_N = rank(N);
order_of_sys = length(A);
[num,den] = ss2tf(A,B,C,D);
G1 = tf(num,den);
figure()
step(G1);
```



```
%information about stabilized system
figure()
step(G2);
```



In this example we have applied Ackerman's formula to create a variable feedback controller used to move the controllable unstable pole at p=1 to a desired stable pole p=-10. As shown on the 2nd step response polot, the system is now stable. We did not have to move the 2nd stable pole located at p=-6 to obtain stability.

Task 4:

```
% checking the controllability and observability of the system
numG = 1;
denG = conv(conv([1 0], [1 1]),[0.2 1]);
% conversion to state-space
[Ag,Bg,Cg,Dg] = tf2ss(numG,denG);
system_order = length(Ag)
```

```
system_order = 3
```

```
M = ctrb(Ag,Bg); % computing controllability matrix
rank_of_M = rank(M)
```

```
rank_of_M = 3
```

```
N = obsv(Ag,Cg); % computing observability matrix
rank_of_N = rank(N)
```

```
rank_of_N = 3
```

```
% a)
damping = 0.707;
wn = 3;
[num2,den2] = ord2(wn, damping);
dominant = roots(den2); % dominant complex pole pair
```

```
% b)
desired_poles = [dominant' 10*real(dominant(1))];
```

```
% c)
K = acker(Ag,Bg, desired_poles);
```

```
% d)
Asf = Ag - Bg * K;
Bsf = Bg;
Csf = Cg;
Dsf = 0;
[numsf,densf] = ss2tf(Asf, Bsf, Csf, Dsf);
```

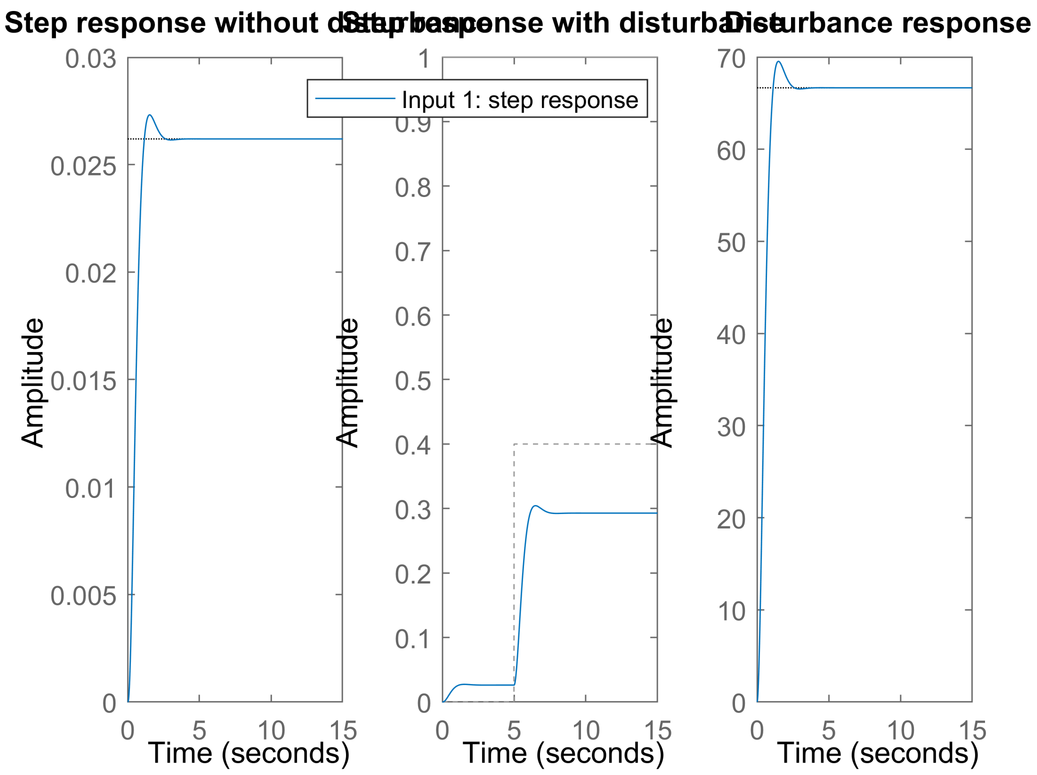
```
%% creating inputs for simulation
t = 0:0.01:15; % total time
t1 = 0:0.01:4.99; % the interval in which the disturbance equals to 0
t2 = 5:0.01:15; % the interval in which the disturbance equals to 1
input1 = 1.0 * ones(size(t)); % step signal
input2 = [0.4 * zeros(size(t1)) 0.4 * ones(size(t2))];
```

```
%% simulating the system
figure(1)
```

```
subplot(1,3,1);
step(Asf,Bsf,Csf,Dsf,1,t);
title('Step response without disturbance');
subplot(1,3,2);
lsim(Asf,[1 0; 0 1; 0 0], Csf, Dsf, [input1' input2'],t); % Bsf = [1 0; 0 1; 0 0]
title('Step response with disturbance');
legend('Input 1: step response', 'Input 2: disturbance signal');
```

```
Warning: Ignoring extra legend entries.
```

```
subplot(1,3,3);
step(Asf, [0; 100; 0], Csf, Dsf,1,t); % Bsf = [0; 100; 0]
title('Disturbance response');
```



Conclusion A:

1) When the system is subjected to input disturbances, the state feedback controller without integral control may not be able to reject the disturbances completely, leading to steady-state errors.

2) The state feedback controller doesn't always track the reference signal accurately due to its inherent steady-state error.

3) The response time of the variable state feedback controller is very good.

```
%% integral control
% a)
Ae = [Ag zeros(size(Ag(:,1))); Cg zeros(size(Cg(:,1)))];% Ae = [Ag 0; Cg 0]
Be = [Bg; zeros(size(Bg(1,:)))];% Be = [Bg; 0]
```

```
Ce = [Cg zeros(size(Cg(:,1)))];% Ce = [Cg 0]
De = [0];
```

```
% b)
system_order = length(Ae)
```

```
system_order = 4
```

```
M = ctrb(Ae, Be);
rank_of_M = rank(M)
```

```
rank_of_M = 4
```

```
N = obsv(Ae, Ce);
rank_of_N = rank(N)
```

```
rank_of_N = 3
```

```
% c)
desiredpoles_ext = [dominant' 10*real(dominant(1)) 20*real(dominant(1))];
```

```
% d)
K_e = acker(Ae, Be, desiredpoles_ext)
```

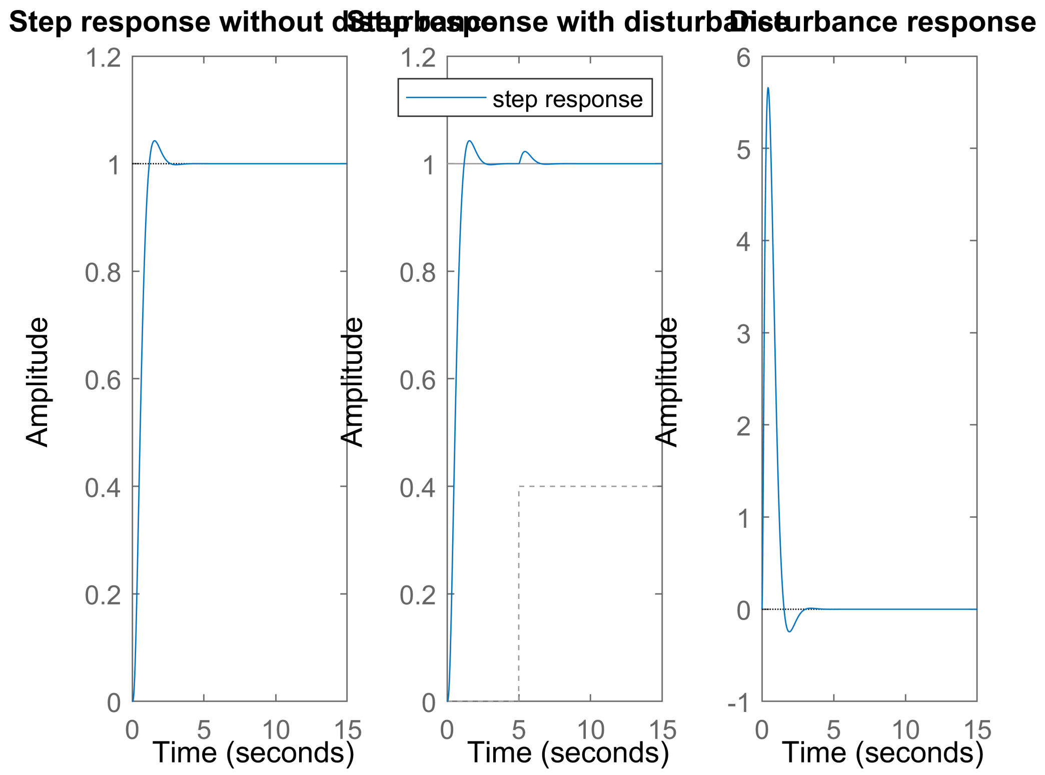
```
K_e = 1×4
10^3 ×
    0.0619    1.1736    4.3893    1.6195
```

```
% e)
Asf_ext = Ae - Be * K_e;
B_r = [0; 0; 0; -1];
[numsf_ext, densf_ext] = ss2tf(Asf_ext, B_r, Ce, De);
SYS_ext = tf(numsf_ext, densf_ext);
```

```
%simulating system
figure(2);
subplot(1,3,1);
step(Asf_ext, B_r, Ce, De, 1, t);
title('Step response without disturbance');
axis([0 15 0 1.2]);
subplot(1,3,2);
lsim(Asf_ext, [0 0; 0 1; 0 0; -1 0], Ce, De, [input1' input2'], t);
title('Step response with disturbance');
legend('step response', 'disturbance signal')
```

```
Warning: Ignoring extra legend entries.
```

```
axis([0 15 0 1.2]);
subplot(1,3,3);
step(Asf_ext, [0; 100; 0; 0], Ce, De, 1, t);
title('Disturbance response');
```



conclusion B:

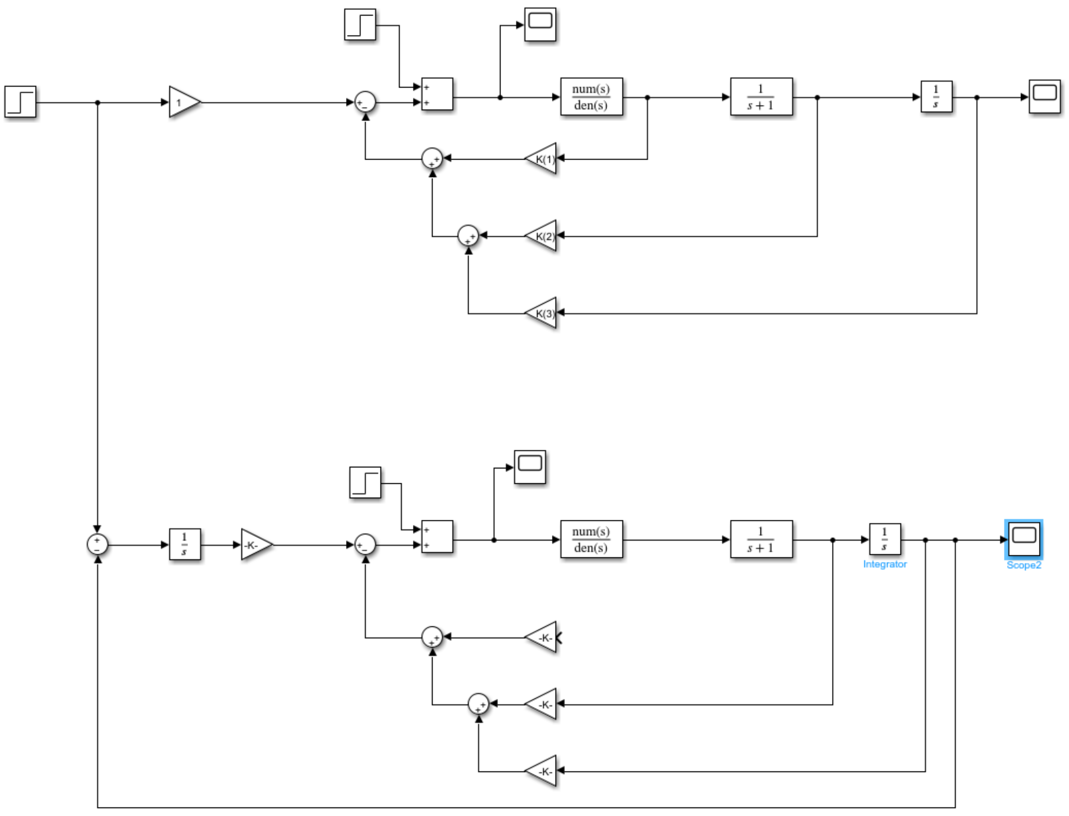
1) In comparaison to a system without integral control the steady state error can be completely eliminated thanks to continuous integration of the error signal.

2) Beacouse of the integal control 2nd system is capable of eliminating steady state error caused by the const. reference signals therefore offering better reference tracking.

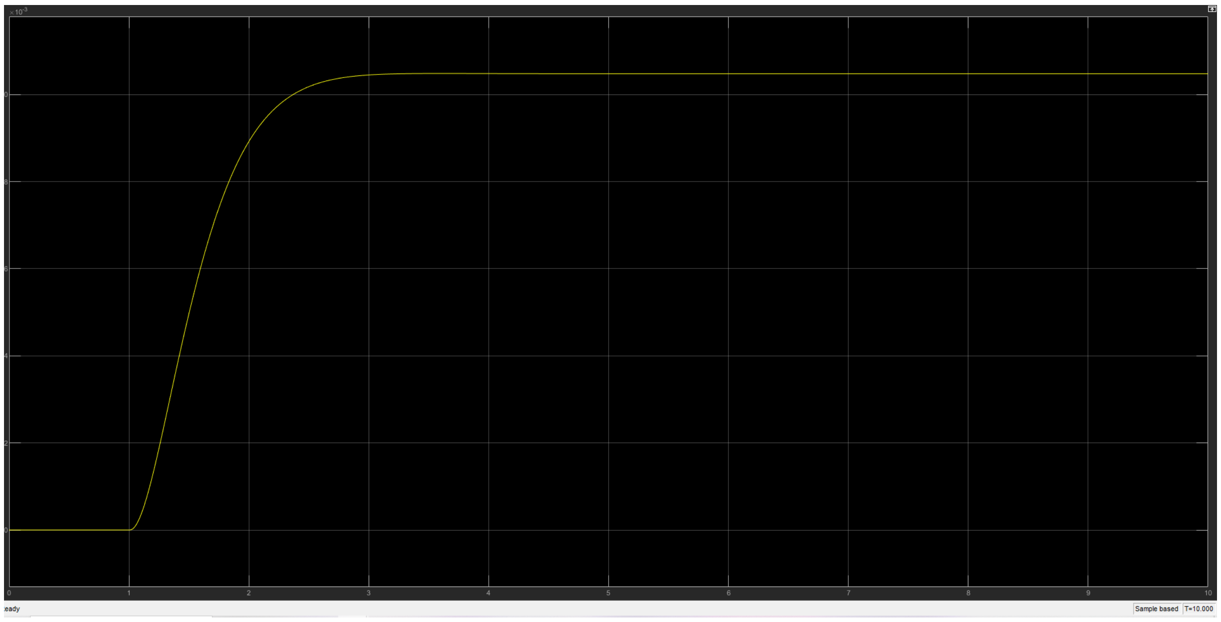
3) the response time of an integral control system is generally longer then the pure variable feedback controller.

Results of a simulink simulation:

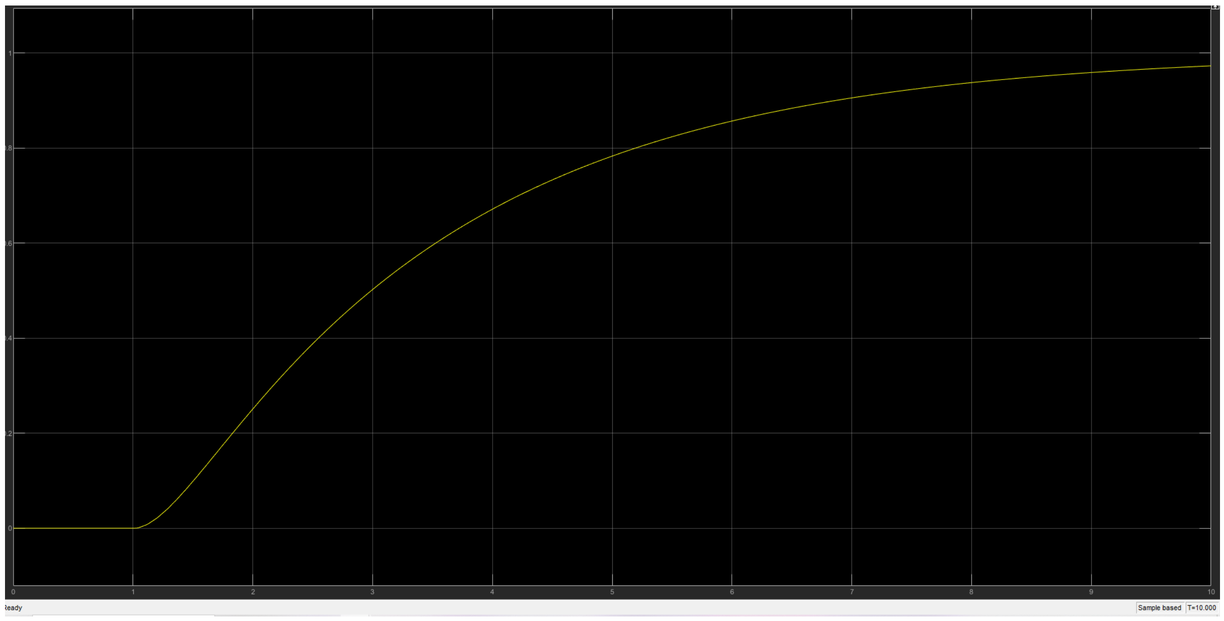
1) Simulink model:



2) scope of a system without integral control:



3) scope of a system with integral control:


1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
6
7
8
9
10
Ready
Sample based
T=10.000

```
clear
clc
```

A)

Checking controllability and observability of the system

```
Ag=[0 1 0 0 0;-0.1 -0.5 0 0 0;0.5 0 0 0 0;0 0 10 0 0;0.5 1 0 0 0]
```

```
Ag = 5×5
         0    1.0000         0         0         0
   -0.1000   -0.5000         0         0         0
    0.5000         0         0         0         0
         0         0   10.0000         0         0
    0.5000    1.0000         0         0         0
```

```
Bg=[0 1 0 0 0]'
```

```
Bg = 5×1
     0
     1
     0
     0
     0
```

```
Cg=[0 0 0 1 0]
```

```
Cg = 1×5
     0     0     0     1     0
```

```
Dg=0
```

```
Dg = 0
```

```

system_order = length(Ag)
```

```
system_order = 5
```

```
M = ctrb(Ag, Bg);
rank_of_M = rank(M)
```

```
rank_of_M = 4
```

```
N = obsv(Ag, Cg);
rank_of_N = rank(N)
```

```
rank_of_N = 4
```

The uncontrollable state indicates that M and N does not have full rank of system_order therefore the system is not controllable and observable

B)

Developing a controllable state variable model

```
[NUM,DEN] = ss2tf(Ag,Bg,Cg,Dg)
```

```
NUM = 1×6
         0         0         0         0    5.0000         0
DEN = 1×6
    1.0000    0.5000    0.1000         0         0         0
```

After canceling common factor z

```
NUM=[0 0 0 0 5];
DEN=[1 0.5 0.1 0 0];
[Ag, Bg, Cg, Dg] = tf2ss(NUM, DEN)
```

```
Ag = 4×4
   -0.5000   -0.1000         0         0
    1.0000         0         0         0
         0    1.0000         0         0
         0         0    1.0000         0
Bg = 4×1
     1
     0
     0
     0
Cg = 1×4
     0     0     0     5
Dg = 0
```

C)

Checking controllability and observability

```
system_order = length(Ag)
```

```
system_order = 4
```

```
M = ctrb(Ag, Bg);
rank_of_M = rank(M)
```

```
rank_of_M = 4
```

```
N = obsv(Ag, Cg);
rank_of_N = rank(N)
```

```
rank_of_N = 4
```

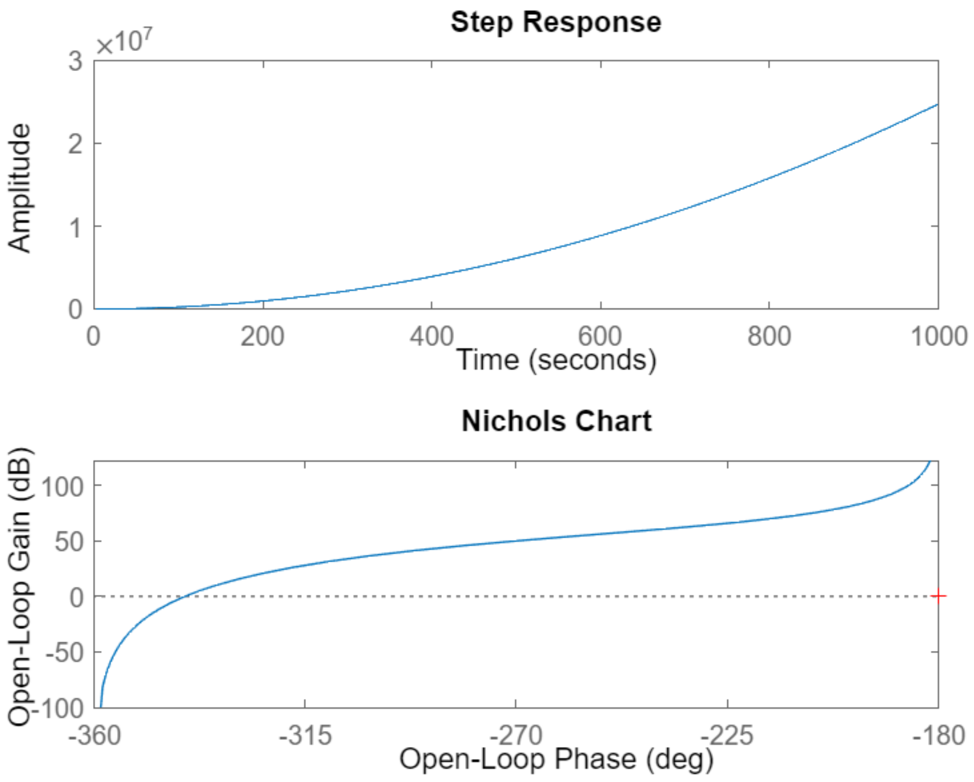
The system is controllable and observable because the M and N matrices have the same rank as the system.

D)

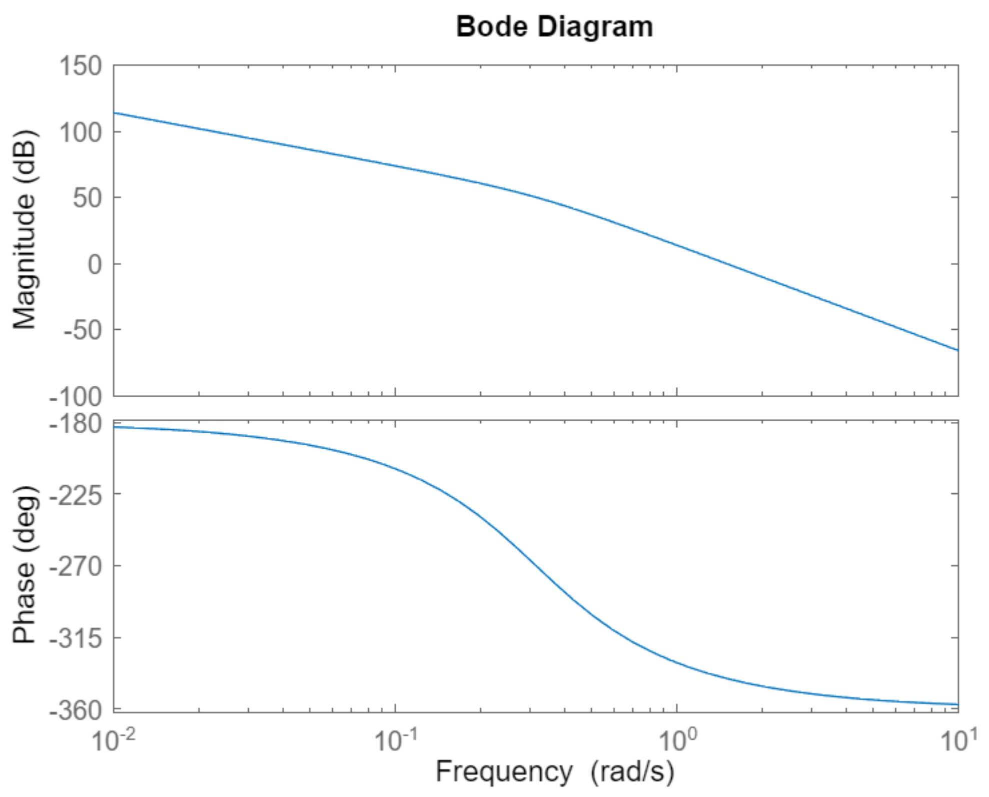
```
sys=tf(NUM,DEN)
```

```
sys =
             5
  ----------------------
  s^4 + 0.5 s^3 + 0.1 s^2

Continuous-time transfer function.
```

```
subplot(211)
step(sys)
subplot(212)
nichols(sys)
```



```
figure
bode(sys)
```



System is unstable

Designing a state variable feedback controller to stabilize the system

```
[A,B,C,D] = tf2ss(NUM,DEN)
```

```
A = 4×4
   -0.5000   -0.1000         0         0
    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     0     5
D = 0
```

New desired poles for a stable system

```
new_desired_poles = [-0.25+0.25i -0.25-0.25i -0.5+0.5i -0.5-0.5i];
K = acker(A, B, new_desired_poles)
```

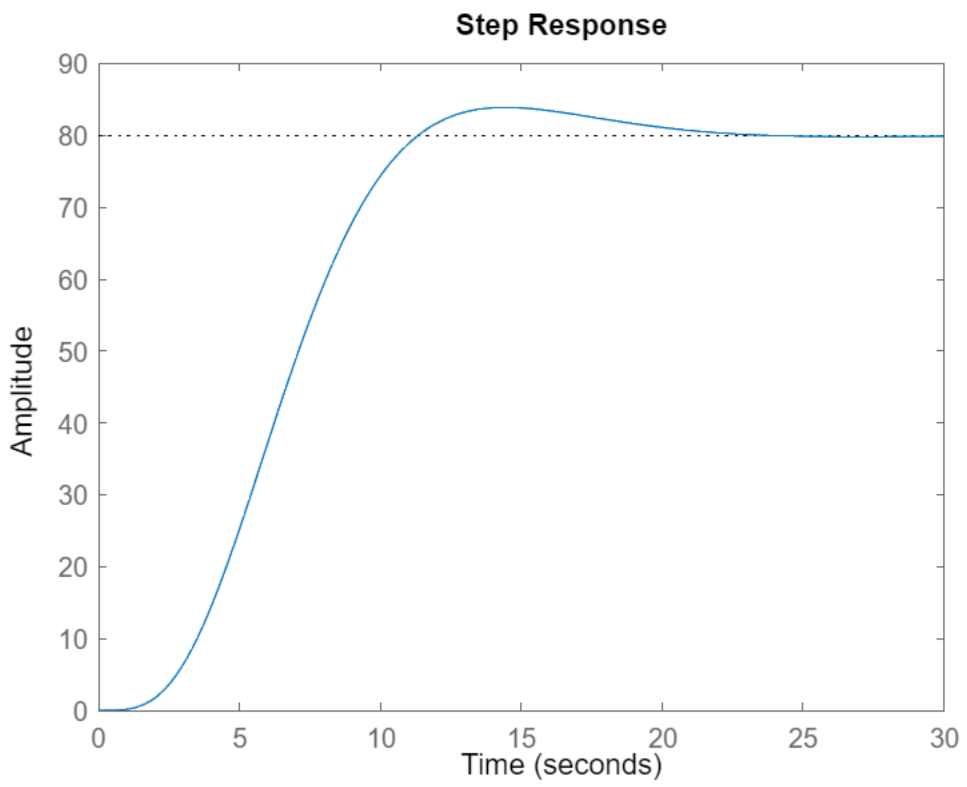
```
K = 1×4
    1.0000    1.0250    0.3750    0.0625
```

```
[num2,den2] = ss2tf(A - B * K,B,C,D);
```

```
H = tf(num2,den2)
```

```
H =

                      5
  -------------------------------------------
  s^4 + 1.5 s^3 + 1.125 s^2 + 0.375 s + 0.0625

Continuous-time transfer function.
```

```
step(H);
```



System has been stabilized using Ackermann's formula

The original system of 5th order was uncontrollable, unobservable and unstable. After pole-zero cancelation the system became observable and controllable yet unstable. The introduction of a controller rendered the system stable.

The number of state variables i.e. the rank of A matrix must be the same as the controllability matrix for the system to be controllable.

Task 6

A)

Checking the stability of system below.

```
Ag=[0 0 0 1 0 0;0 0 0 0 1 0;0 0 0 0 0 1;...
    7.3809 0 0 0 2 0;0 -2.1904 0 -2 0 0;0 0 -3.1904 0 0 0]
```

```
Ag = 6×6
         0         0         0    1.0000         0         0
         0         0         0         0    1.0000         0
         0         0         0         0         0    1.0000
    7.3809         0         0         0    2.0000         0
         0   -2.1904         0   -2.0000         0         0
         0         0   -3.1904         0         0         0
```

```
Bg=[[0 0 0 1 0 0]' [0 0 0 0 1 0]' [0 0 0 0 0 1]']
```

```
Bg = 6×3
     0     0     0
     0     0     0
     0     0     0
     1     0     0
     0     1     0
     0     0     1
```

```

e=eig(Ag)
```

```
e = 6×1 complex
  -2.1587 + 0.0000i
   2.1587 + 0.0000i
   0.0000 + 1.8626i
   0.0000 - 1.8626i
   0.0000 + 1.7862i
   0.0000 - 1.7862i
```

The equilibrium point in not a stable position because one pole is on the right side of the complex plane.

B-C)

Controllability:

```
system_order = length(Ag)
```

```
system_order = 6
```

```
M = ctrb(Ag, Bg(:,1));
rank_of_M = rank(M)
```

```
rank_of_M = 4
```

```
M = ctrb(Ag, Bg(:,2));
rank_of_M = rank(M)
```

```
rank_of_M = 4
```

```
M = ctrb(Ag, Bg(:,3));
rank_of_M = rank(M)
```

```
rank_of_M = 2
```

From the fact that the controllability matrix for u1 u2 and u3 is of lower rank than the order of the system we conclude the system is not controllable from any input u.

D)

Determining the transfer function for y=[0 1 0 0 0 0]x

```
Cg=[0 1 0 0 0 0];
Dg=0;

[NUM,DEN]=ss2tf(Ag,Bg(:,2),Cg,Dg)
```

```
NUM = 1×7
         0         0    1.0000    0.0000   -4.1905    0.0000  -23.5480
DEN = 1×7
    1.0000    0.0000    1.9999    0.0000  -19.9653    0.0000  -51.5796
```

```
sys=tf(NUM,DEN)
```

```
sys =
 
              s^4 + 4.441e-16 s^3 - 4.191 s^2 + 1.776e-15 s - 23.55
  -----------------------------------------------------------------------------
  s^6 + 1.943e-16 s^5 + 2 s^4 + 3.325e-15 s^3 - 19.97 s^2 + 8.629e-15 s - 51.58
 
Continuous-time transfer function.
```

E)

Controlabillity for y=[0 1 0 0 0 0]x

```
zeros=zero(sys)
```

```
zeros = 4×1 complex
  -2.7168 + 0.0000i
   2.7168 + 0.0000i
   0.0000 + 1.7862i
   0.0000 - 1.7862i
```

```
poles=pole(sys)
```

```
poles = 6×1 complex
  -2.1587 + 0.0000i
   2.1587 + 0.0000i
   0.0000 + 1.8626i
   0.0000 - 1.8626i
  -0.0000 + 1.7862i
  -0.0000 - 1.7862i
```

```
K=1;
zeros=[zeros(1:2)]
```

```
zeros = 2×1
   -2.7168
    2.7168
```

```
poles=[poles(1:4)]
```

```
poles = 4×1 complex
  -2.1587 + 0.0000i
   2.1587 + 0.0000i
   0.0000 + 1.8626i
   0.0000 - 1.8626i
```

```
sys=zpk(zeros,poles,K)
```

```
sys =

         (s+2.717) (s-2.717)
  ---------------------------------
  (s+2.159) (s-2.159) (s^2 + 3.469)

Continuous-time zero/pole/gain model.
```

```
NUM=conv([1 2.717],[1 -2.717])
```

```
NUM = 1×3
    1.0000         0   -7.3821
```

```
DEN=conv([1 2.159],[1 -2.159]);
DEN=conv(DEN,[1 0 3.469])
```

```
DEN = 1×5
    1.0000         0   -1.1923         0  -16.1700
```

```
[A,B,C,D] = tf2ss(NUM,DEN)
```

```
A = 4×4
         0    1.1923         0   16.1700
    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    1.0000         0   -7.3821
D = 0
```

```


system_order = length(A)
```

```
system_order = 4
```

```
M = ctrb(A,B);
rank_of_M = rank(M)
```

```
rank_of_M = 4
```

```
N = obsv(Ag, Cg);
```

```
rank_of_N = rank(N)
```

```
rank_of_N = 4
```

The system is now controllable given the uniformity of the rank of controllability matx with system order.

F)

Designing a controller for system stabilisation.

```
desiredpoles = [-1+1i;-1-1i;-10;-10]
```

```
desiredpoles = 4×1 complex
  -1.0000 + 1.0000i
  -1.0000 - 1.0000i
 -10.0000 + 0.0000i
 -10.0000 + 0.0000i
```

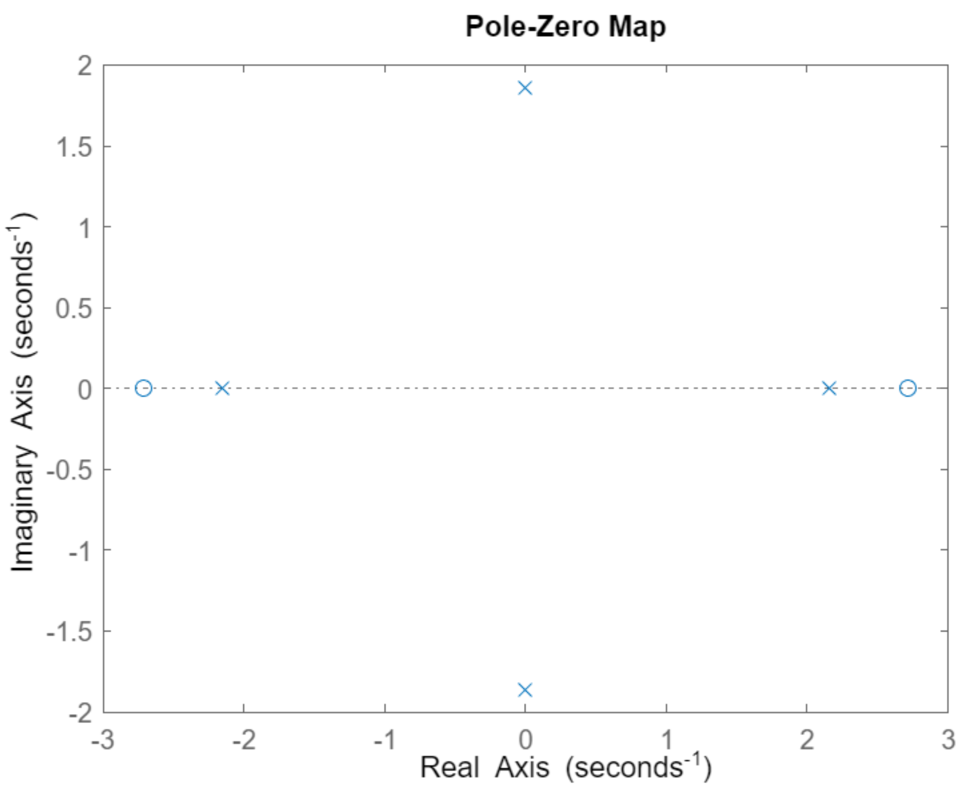
```
K = acker(A,B,desiredpoles)
```

```
K = 1×4
   22.0000  143.1923  240.0000  216.1700
```

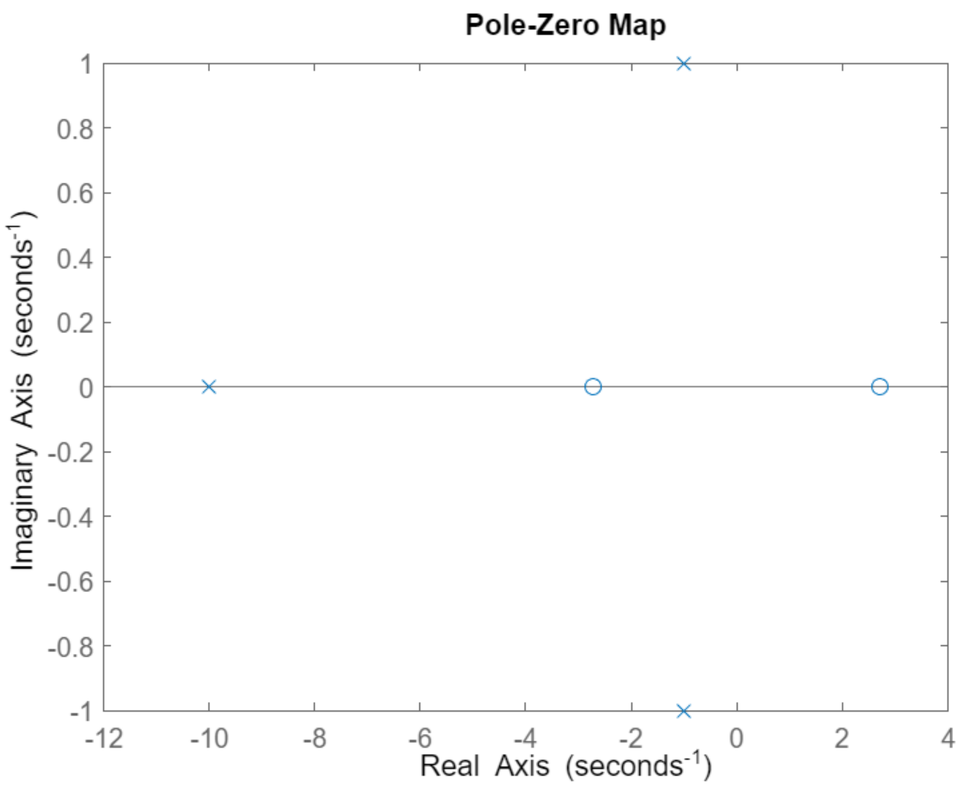
```


[num2,den2] = ss2tf(A - B * K,B,C,D);
G = tf(num2,den2)
```

```
G =

              s^2 - 7.382
  ------------------------------------
  s^4 + 22 s^3 + 142 s^2 + 240 s + 200

Continuous-time transfer function.
```

```
pzmap(sys)
```


Pole-Zero Map
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-3
-2
-1
0
1
2
3

```
pzmap(G)
```


Pole-Zero Map
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-12
-10
-8
-6
-4
-2
0
2
4

```
roots(den2)
```

```
ans = 4×1 complex
 -10.0000 + 0.0000i
 -10.0000 - 0.0000i
  -1.0000 + 1.0000i
  -1.0000 - 1.0000i
```

No positve real parts of poles imply stabilized system.

Task 7)

A)

```
Z=[];
P=[-0.01 -0.2 -1];
K=8;
sys=zpk(Z,P,K)
```

```
sys =
 
            8
  ----------------------
  (s+0.01) (s+0.2) (s+1)
 
Continuous-time zero/pole/gain model.
```

```
[A,B,C,D] = zp2ss(Z,P,K);

system_order = length(A)
```

```
system_order = 3
```

```

M = ctrb(A, B);
rank_of_M = rank(M)
```

```
rank_of_M = 3
```

```
N = obsv(A, C);
rank_of_N = rank(N)
```

```
rank_of_N = 3
```

```

desiredpoles = [-0.2+0.2i;-0.2-0.2i;-10]
```

```
desiredpoles = 3×1 complex
  -0.2000 + 0.2000i
  -0.2000 - 0.2000i
 -10.0000 + 0.0000i
```

```

K = acker(A,B,desiredpoles)
```

```
K = 1×3
    9.1900   -7.1600   -2.3255
```

```

[num2,den2] = ss2tf(A - B * K,B,C,D);
G2 = tf(num2,den2)
```

```
G2 =
 
                8
  -----------------------------
  s^3 + 10.4 s^2 + 4.08 s + 0.8

Continuous-time transfer function.
```

```
K2 = place(A,B,desiredpoles)
```

```
K2 = 1×3
    9.1900   -7.1600   -2.3255
```

```
[num3,den3] = ss2tf(A - B * K,B,C,D);
G3 = tf(num3,den3)
```

```
G3 =

               8
  -----------------------------
  s^3 + 10.4 s^2 + 4.08 s + 0.8

Continuous-time transfer function.
```

```
stepinfo(G3)
```

```
ans = struct with fields:
         RiseTime: 7.5999
    TransientTime: 21.1814
     SettlingTime: 21.1814
      SettlingMin: 9.1071
      SettlingMax: 10.4319
        Overshoot: 4.3186
       Undershoot: 0
             Peak: 10.4319
         PeakTime: 15.8878
```

Matlab functions like Acker or Place ensure good stability, performance and ease of use due to being optimized for Matlab. While Acker is better suited for lower order systems (up to 4th) and Place function performs better for higher order systems, both yielded the same result for a 3rd order system.

The Root-Locus method is quick to solve and provides a good visualization of pole mobility making this method very intuitive. Unfortunately it introduces a lot of human error, lacking precision and problematic for higher order systems.

Ziegler-Nichols method results in poor initial results making additional optimizations necessary but all calculations are first hand mathematical operations performed by the user which gives a better control/customizability for the user.

	Peak Amplitude	Overshoot	Rise time [s]	Settling time
Root-Locus	1,05	8,25%	5,53	19,5
Ziegler-Nichols	1,15	16,49%	1,63	7,98
Matlab functions	4,31	4,31%	7,6	21,18