National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Electrical Engineering and Computer Science

Faculty Member: Ma'am Neelma Naz **Dated:** 21/09/2022

Semester: 6^{th} **Section:** BEE 12C

EE379: Control Systems

Lab 7: **Performance of systems**

Lab Instructor: Sir. Yasir Rizwan

Group Members

Student Name	Reg. No.	Lab Report Marks / 10	Viva Marks / 5	Total /15
Muhammad Ahmed Mohsin	333060			
Imran Haider	332569			
Zafar Azhar	340908			

1 TABLE OF CONTENTS

State space, response of systems to various inputs, and interconnections of systems

2 OBJECTIVES

Learn how to compute the transient and steady state characteristics of a system in MATLAB.

3 PERFORMANCE

As you should have studied in the lectures and we also mentioned in earlier lab handouts, the step input is very common in control systems, e.g., if you want the elevator to go from the ground floor to the fourth floor, then the desired behavior is a step of magnitude 4. Therefore, the performance of a control system is often based on the response of the system to a step input. The step response has two categories of performance measures: the performance of transient response and the performance of steady state response. The characteristics of the transient response include the rise time, settling time, peak time and the maximum overshoot. For the steady state we are interested in the steady state error. In this handout we will learn how to calculate these performance measures in MATLAB.

4 STEP RESPONSE CHARACTERISTICS IN MATLAB

You already know that the step response of a system can be found in MATLAB using the function step(). The information about the transient response of a system when excited by a step input, can be found by the function stepinfo(). There is no function to find the steady state error in response to a step input. However, you can use the code

```
abs(1-dcgain(sys))
```

to find the steady state error for a step input. An example code is given below:

4.1 Code

```
% Example
my_tf = tf(1,[1 2]);
stepinfo(my_tf);
steady_state_error=abs(1-dcgain(my_tf));
```

4.2 Output

steady_state_error =

0.5000

RiseTime: 1.0985
TransientTime: 1.9560
SettlingTime: 1.9560
SettlingMin: 0.4523
SettlingMax: 0.5000
Overshoot: 0
Undershoot: 0
Peak: 0.5000
PeakTime: 5.2729

5 Exercise 1

Find the rise time, peak time, peak value, overshoot, settling time and the steady state error for step input of the following systems:

5.1 Code

```
num1=[2 2];
dem1=[1 9 20];
tf1=tf(num1,dem1)
stepinfo(tf1)
steadyerror1=abs(1-dcgain(tf1))
num2=[1 1];
dem2=[1 12 47 60];
tf2=tf(num2,dem2);
stepinfo(tf2);
steaderror2=abs(1-dcgain(tf2));
num3=[1];
dem3=[1 10];
```

```
tf3=tf(num3,dem3);
stepinfo(tf3);
steadyerror3=abs(1-dcgain(tf3));
```

5.2 OUTPUT

```
tf1 =

      2 s + 2
  --------------
  s^2 + 9 s + 20

Continuous-time transfer function.
```

```
ans =

  struct with fields:

        RiseTime: 0.0510
   TransientTime: 1.5940
    SettlingTime: 1.5940
     SettlingMin: 0.0917
     SettlingMax: 0.1949
       Overshoot: 94.9219
      Undershoot: 0
            Peak: 0.1949
        PeakTime: 0.2878


steadyerror1 =

    0.9000
```

```
tf2 =

            s + 1
  ------------------------
  s^3 + 12 s^2 + 47 s + 60

Continuous-time transfer function.
```

```
ans =

  struct with fields:

        RiseTime: 0.2247
   TransientTime: 2.2150
    SettlingTime: 2.2150
     SettlingMin: 0.0151
     SettlingMax: 0.0240
       Overshoot: 43.7489
      Undershoot: 0
            Peak: 0.0240
        PeakTime: 0.6908


steaderror2 =

    0.9833
```

```
tf3 =

    1
  ------
  s + 10

Continuous-time transfer function.
```

```
ans =

  struct with fields:

        RiseTime: 0.2197
    TransientTime: 0.3912
     SettlingTime: 0.3912
      SettlingMin: 0.0905
      SettlingMax: 0.1000
        Overshoot: 0
       Undershoot: 0
             Peak: 0.1000
         PeakTime: 1.0546


steadyerror3 =

    0.9000
```

6 EXERCISE 2

$$\frac{p}{s-p}$$

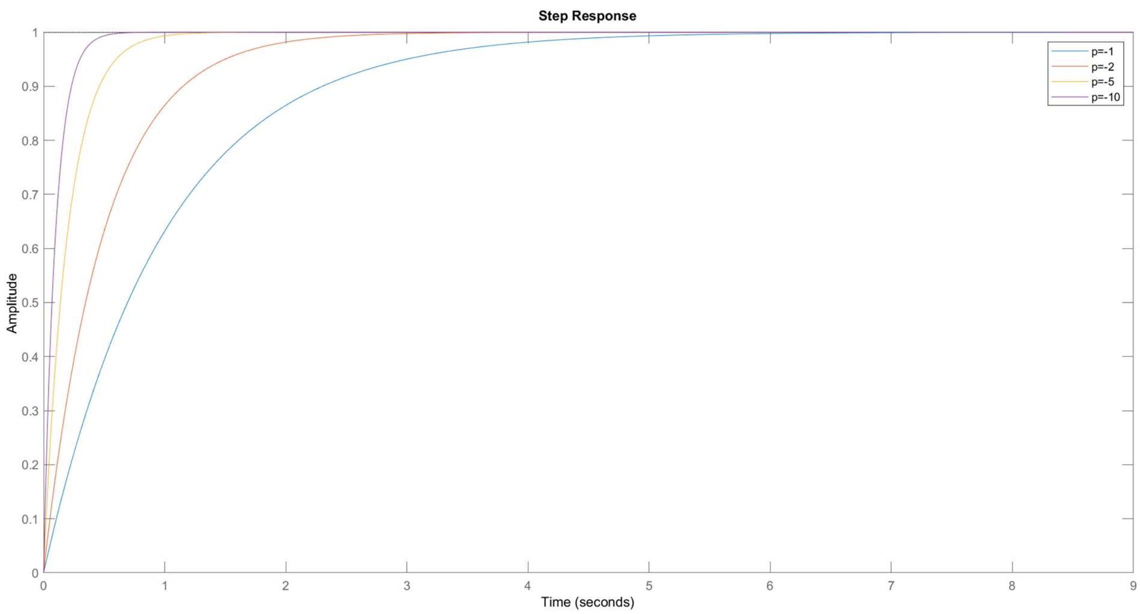
This system has a pole at , it has no zeros and the gain is equal to the negative of the pole i.e. - . Using Matlab, plot the step response of systems of this form for p= -1, -2, -5, -10. Plot all the step responses on a single figure. A sample code is given below.

6.1 CODE:

```
clc
clear
close
hold on;
sys1 = zpk([],-1,1);
tf1=tf(sys1);
step(ss(sys1));
sys2 = zpk([],-2,2);
```

```
tf2=tf(sys2);
step(ss(sys2));
sys3 = zpk([],-5,5);
tf3=tf(sys3);
step(ss(sys3));
sys4 = zpk([],-10,10);
tf4=tf(sys4);
step(ss(sys4));
legend('p=-1','p=-2','p=-5','p=-10');

stepinfo(sys1);
steadyerror1=abs(1-dcgain(tf1));
stepinfo(sys2);
steadyerror2=abs(1-dcgain(tf2));
stepinfo(sys3);
steadyerror3=abs(1-dcgain(tf3));
stepinfo(sys4);
steadyerror4=abs(1-dcgain(tf4));
```

6.2 Output



```
ans =

  struct with fields:

        RiseTime: 2.1970
   TransientTime: 3.9121
    SettlingTime: 3.9121
     SettlingMin: 0.9045
     SettlingMax: 1.0000
       Overshoot: 0
      Undershoot: 0
            Peak: 1.0000
        PeakTime: 10.5458


steadyerror1 =

     0
```

```
ans =

  struct with fields:

        RiseTime: 1.0985
   TransientTime: 1.9560
    SettlingTime: 1.9560
     SettlingMin: 0.9045
     SettlingMax: 1.0000
       Overshoot: 0
      Undershoot: 0
            Peak: 1.0000
        PeakTime: 5.2729


steadyerror2 =

     0
```

```
ans =

  struct with fields:

        RiseTime: 0.4394
   TransientTime: 0.7824
    SettlingTime: 0.7824
     SettlingMin: 0.9045
     SettlingMax: 1.0000
       Overshoot: 0
      Undershoot: 0
            Peak: 1.0000
        PeakTime: 2.1092


steadyerror3 =

     0
```

```
ans =

  struct with fields:

        RiseTime: 0.2197
   TransientTime: 0.3912
    SettlingTime: 0.3912
     SettlingMin: 0.9045
     SettlingMax: 1.0000
       Overshoot: 0
      Undershoot: 0
            Peak: 1.0000
        PeakTime: 1.0546


steadyerror4 =

     0
```

6.3 COMMENTS

The introduction of an additional pole into a system leads to a reduction in the speed at which the system responds, the extent of which is contingent on the proximity of the pole to the imaginary jw axis.

7 Exercise 3:

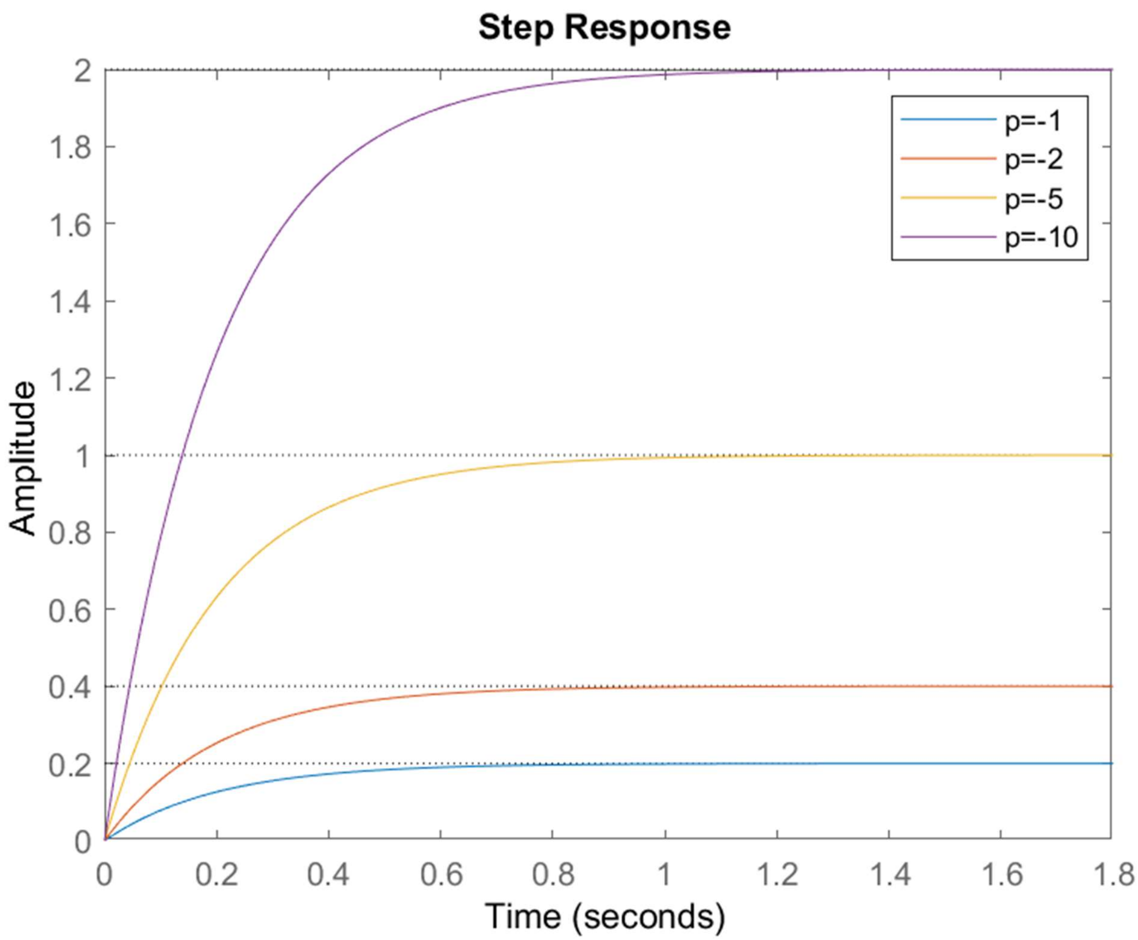
$$\frac{k}{s-p}$$

Using Matlab, plot the step response of systems of this form for = -5 and k= 1,2,5,10. Plot all the step responses on a single figure. For each system also find the values of the various performance characteristics. Comment on the effects of changing the gain.

7.1 Code:

```
clc
clear
close
hold on;
sys1 = zpk([],-5,1);
tf1=tf(sys1);
step(ss(sys1));
sys2 = zpk([],-5,2);
tf2=tf(sys2);
step(ss(sys2));
sys3 = zpk([],-5,5);
tf3=tf(sys3);
step(ss(sys3));
sys4 = zpk([],-5,10);
tf4=tf(sys4);
step(ss(sys4));
legend('p=-1','p=-2','p=-5','p=-10');

stepinfo(sys1)
steadyerror1=abs(1-dcgain(tf1))
stepinfo(sys2)
steadyerror2=abs(1-dcgain(tf2))
stepinfo(sys3)
steadyerror3=abs(1-dcgain(tf3))
stepinfo(sys4)
steadyerror4=abs(1-dcgain(tf4))
```

7.2 Output:


Step Response
Amplitude
Time (seconds)
p=-1
p=-2
p=-5
p=-10
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

```
ans =

  struct with fields:

        RiseTime: 0.4394
    TransientTime: 0.7824
     SettlingTime: 0.7824
      SettlingMin: 0.9045
      SettlingMax: 1.0000
        Overshoot: 0
       Undershoot: 0
             Peak: 1.0000
         PeakTime: 2.1092


steadyerror3 =

     0


ans =

  struct with fields:

        RiseTime: 0.4394
    TransientTime: 0.7824
     SettlingTime: 0.7824
      SettlingMin: 1.8090
      SettlingMax: 1.9999
        Overshoot: 0
       Undershoot: 0
             Peak: 1.9999
         PeakTime: 2.1092


steadyerror4 =

     1
```

```
ans =

  struct with fields:

        RiseTime: 0.4394
    TransientTime: 0.7824
     SettlingTime: 0.7824
      SettlingMin: 0.1809
      SettlingMax: 0.2000
        Overshoot: 0
       Undershoot: 0
             Peak: 0.2000
         PeakTime: 2.1092


steadyerror1 =

    0.8000


ans =

  struct with fields:

        RiseTime: 0.4394
    TransientTime: 0.7824
     SettlingTime: 0.7824
      SettlingMin: 0.3618
      SettlingMax: 0.4000
        Overshoot: 0
       Undershoot: 0
             Peak: 0.4000
         PeakTime: 2.1092


steadyerror2 =

    0.6000
```

7.3 COMMENTS:

Altering the gain of a system does not affect the overall speed of the system's response; instead, it modifies the ultimate value at which the system stabilizes.

8 Exercise 4:

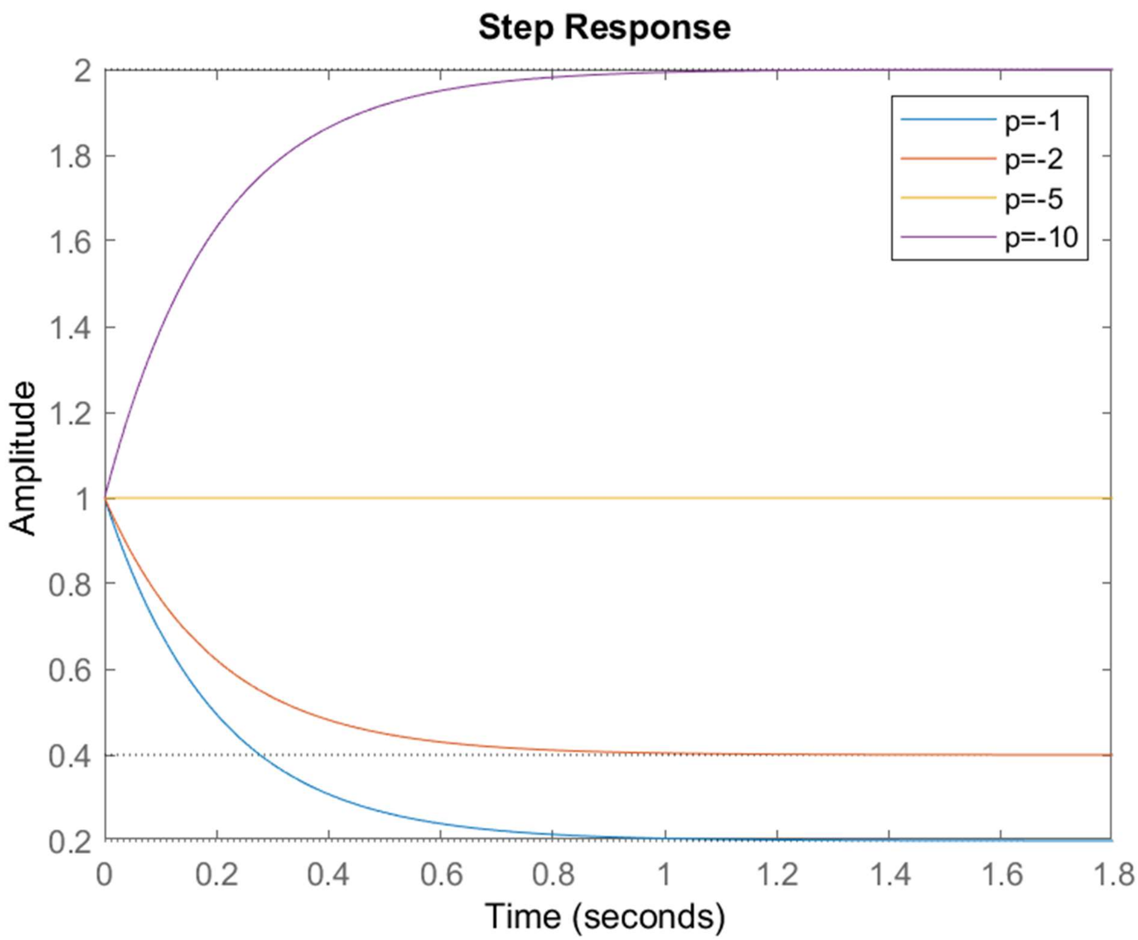
$$\frac{k(s-z)}{s-p}$$

Using Matlab, plot the step response of systems of this form for = -5, k= 1 and = -1, -2, -5, -10. Also have a plot for no zero. Plot all the step responses on a single figure. For each system also find the values of the various performance characteristic. Comment on the effects of changing the zeros.

8.1 Code:

```
clc
clear
close
hold on;
sys1 = zpk(-1,-5,1);
tf1=tf(sys1);
step(ss(sys1));
sys2 = zpk(-2,-5,1);
tf2=tf(sys2);
step(ss(sys2));
sys3 = zpk(-5,-5,1);
tf3=tf(sys3);
step(ss(sys3));
sys4 = zpk(-10,-5,1);
tf4=tf(sys4);
step(ss(sys4));
legend('p=-1','p=-2','p=-5','p=-10');

stepinfo(sys1)
steadyerror1=abs(1-dcgain(tf1))
stepinfo(sys2)
steadyerror2=abs(1-dcgain(tf2))
stepinfo(sys3)
steadyerror3=abs(1-dcgain(tf3))
stepinfo(sys4)
steadyerror4=abs(1-dcgain(tf4))
```

8.2 Output:


Step Response
Amplitude
Time (seconds)
p=-1
p=-2
p=-5
p=-10
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8

```
ans =

  struct with fields:

         RiseTime: 0
    TransientTime: 0.7824
     SettlingTime: 1.0597
      SettlingMin: 0.2000
      SettlingMax: 1
        Overshoot: 400.0000
       Undershoot: 0
             Peak: 1
         PeakTime: 0


steadyerror1 =

    0.8000


ans =

  struct with fields:

         RiseTime: 0
    TransientTime: 0.7824
     SettlingTime: 0.8635
      SettlingMin: 0.4000
      SettlingMax: 1
        Overshoot: 150
       Undershoot: 0
             Peak: 1
         PeakTime: 0


steadyerror2 =

    0.6000
```

8.3 COMMENTS:

Incorporating an additional zero in a system amplifies the velocity of the system's response, with the degree of escalation hinging on the proximity of the pole to the imaginary jw axis.

9 Exercise 5

Use the formulas given above to find the values of the pole of a first order system that would give

- rise times of 0.1, 0.5 and 1
- settling times of 1, 1.5 and 2

9.1 Code:

```
risetime=[0.1 0.5 1];
pole=2./risetime;
settlingtime=[1 1.5 2];
pole1=4./settlingtime;
```

9.2 Output:

```
>> pole

pole =

    20     4     2

>> pole1

pole1 =

    4.0000    2.6667    2.0000
```

10 Exercise 6:

Find the damping ratio and the natural frequency of the following systems:

$$\frac{5}{s^2-4s+5}, \quad \frac{2}{s^2-2s+2}, \quad \frac{5}{s^2-2s+5}$$

10.1 Code:

```
tf1=tf(5,[1 -4 5]);
tf2=tf(2,[1 -2 2]);
tf3=tf(5,[1 -2 5]);
damp(tf1)
damp(tf2)
damp(tf3)
```

10.2 Output:

```
>> lab6lcs

        Pole              Damping       Frequency      Time Constant
                                      (rad/seconds)     (seconds)

  2.00e+00 + 1.00e+00i   -8.94e-01      2.24e+00        -5.00e-01
  2.00e+00 - 1.00e+00i   -8.94e-01      2.24e+00        -5.00e-01

        Pole              Damping       Frequency      Time Constant
                                      (rad/seconds)     (seconds)

  1.00e+00 + 1.00e+00i   -7.07e-01      1.41e+00        -1.00e+00
  1.00e+00 - 1.00e+00i   -7.07e-01      1.41e+00        -1.00e+00

        Pole              Damping       Frequency      Time Constant
                                      (rad/seconds)     (seconds)

  1.00e+00 + 2.00e+00i   -4.47e-01      2.24e+00        -1.00e+00
  1.00e+00 - 2.00e+00i   -4.47e-01      2.24e+00        -1.00e+00
>>
```

11 Exercise 7:

Write a MATLAB function that takes the damping ratio and natural frequency as arguments and returns a transfer function of the form given in equation (2).

$$\frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

11.1 Output

```
>> lab6lcs
What is the b 20
What is the a 10

Tf =

20/(s^2 + 20*s + 20)
```

11.2 Code

```
syms s;
prompt1="What is the b ";
prompt2="What is the a ";
b=input(prompt1);
a=input(prompt2);

Tf=(b)/(s^2+2*s*a+b)
```

12 Exercise 8

Write a MATLAB function that takes the damping ratio and natural frequency as arguments and returns a transfer function of the form given in equation (2). Call this function my_second_order_tf.

12.1 Code:

```
syms s;
prompt1="What is the natural Frequency ";
prompt2="What is the Damping Ratio ";
naturalfrequency=input(prompt1);
dampingratio=input(prompt2);

Tf=(naturalfrequency^2)/(s^2+2*dampingratio*s*naturalfrequency+naturalfrequency^2)
```

12.2 OUTPUT:

```
>> lab6lcs
What is the natural Frequency 10
What is the Damping Ratio 20

Tf =

100/(s^2 + 400*s + 100)
```

13 EXERCISE: 9

Using the function that you have just created, my_second_order_tf, make transfer functions for the following sets of damping ratios and natural frequencies Set 1: = 0, - = 1,2,5 (See the note given below) Set 2: = 1, - = 1,2,5 Set 3: = 0,0.5, 1,2, - = 1 For each set of values plot the step responses on a single figure. For each system also find the values of the various performance characteristic. Comment on the effects of changing the natural frequency and the damping ratio. Classify each of the above systems as undamped, underdamped, critically damped or overdamped.

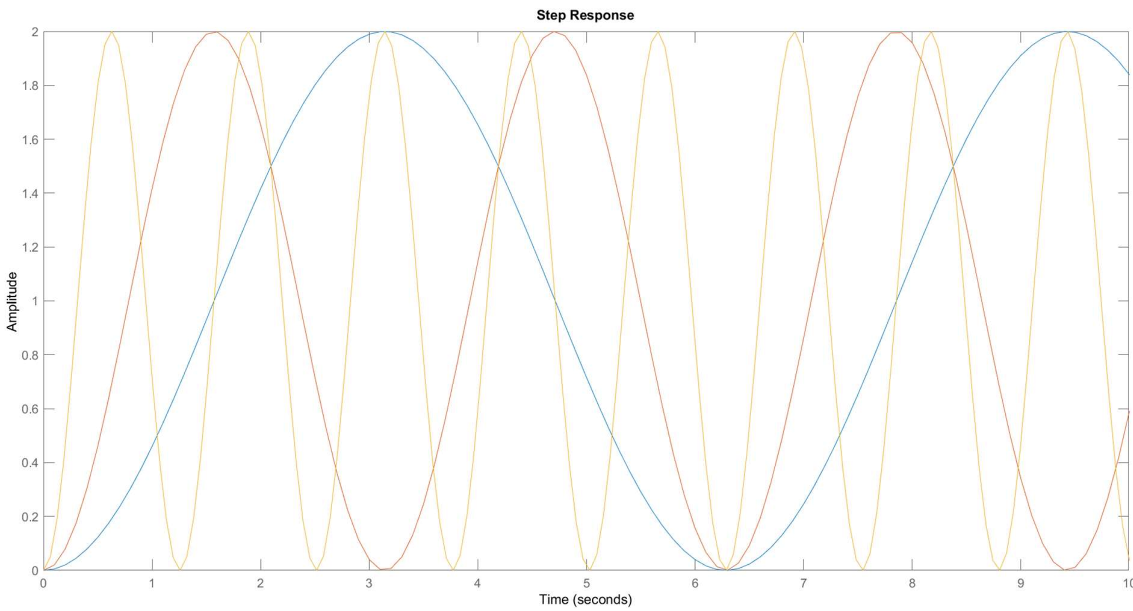
13.1 CODE:

```
prompt1="What is the natural Frequency ";
prompt2="What is the Damping Ratio ";
naturalfrequency1=input(prompt1);
dampingratio1=input(prompt2);
naturalfrequency2=input(prompt1);
dampingratio2=input(prompt2);
naturalfrequency3=input(prompt1);
dampingratio3=input(prompt2);

Tf1=tf([naturalfrequency1^2],[1 2*dampingratio1*naturalfrequency1
naturalfrequency1^2]);
Tf2=tf([naturalfrequency2^2],[1 2*dampingratio2*naturalfrequency2
naturalfrequency2^2]);
Tf3=tf([naturalfrequency3^2],[1 2*dampingratio3*naturalfrequency3
naturalfrequency3^2]);
hold on;
step(Tf1,10)
step(Tf2,10)
step(Tf3,10)
```

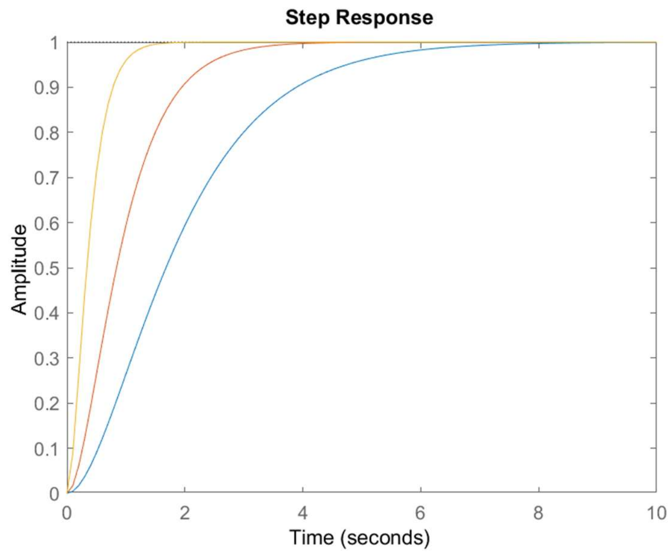
```
stepinfo(Tf1)
stepinfo(Tf2)
stepinfo(Tf3)
```

13.2 OUTPUT:

13.2.1 Part 1

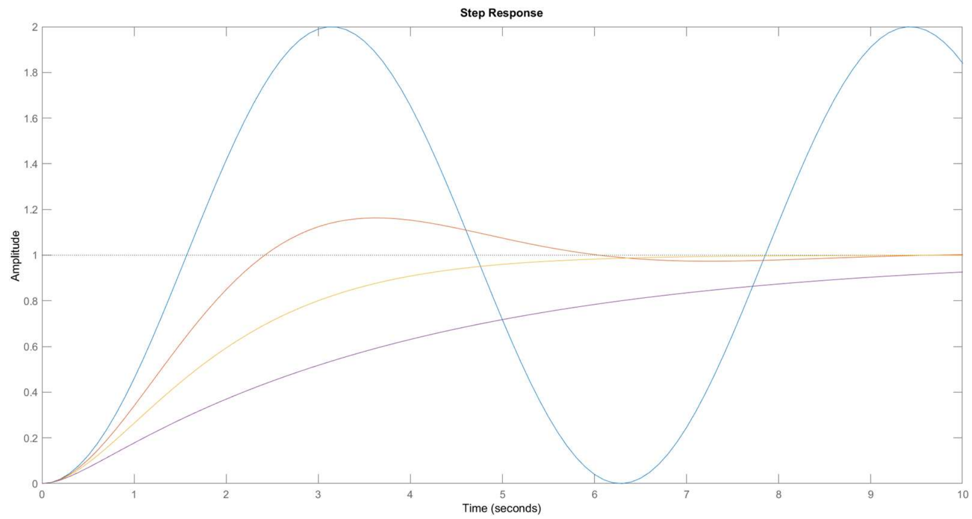


13.2.2 Part 2




```
ans =

  struct with fields:

        RiseTime: 3.3579
   TransientTime: 5.8339
    SettlingTime: 5.8339
     SettlingMin: 0.9000
     SettlingMax: 0.9994
       Overshoot: 0
      Undershoot: 0
            Peak: 0.9994
        PeakTime: 9.7900


ans =

  struct with fields:

        RiseTime: 1.6790
   TransientTime: 2.9170
    SettlingTime: 2.9170
     SettlingMin: 0.9008
     SettlingMax: 0.9991
       Overshoot: 0
      Undershoot: 0
            Peak: 0.9991
        PeakTime: 4.6900


ans =

  struct with fields:

        RiseTime: 0.6717
   TransientTime: 1.1668
    SettlingTime: 1.1668
     SettlingMin: 0.9008
     SettlingMax: 0.9999
       Overshoot: 0
      Undershoot: 0
            Peak: 0.9999
        PeakTime: 2.3900
```

13.2.3 Part 3




```
ans =

  struct with fields:

        RiseTime: NaN
   TransientTime: NaN
    SettlingTime: NaN
     SettlingMin: NaN
     SettlingMax: NaN
       Overshoot: NaN
      Undershoot: NaN
            Peak: Inf
        PeakTime: Inf


ans =

  struct with fields:

        RiseTime: 1.6390
   TransientTime: 8.0759
    SettlingTime: 8.0759
     SettlingMin: 0.9315
     SettlingMax: 1.1629
       Overshoot: 16.2929
      Undershoot: 0
            Peak: 1.1629
        PeakTime: 3.5920


ans =

  struct with fields:

        RiseTime: 3.3579
   TransientTime: 5.8339
    SettlingTime: 5.8339
     SettlingMin: 0.9000
     SettlingMax: 0.9994
       Overshoot: 0
      Undershoot: 0
            Peak: 0.9994
        PeakTime: 9.7900
```

14 EXERCISE 10

Using the formulas given above, find the values of damping ratio and natural frequency that result in %OS=10 and = 1.

14.1 CODE:

```
OS=10;
Ts=.01;
Zeta = -(log(OS/100))/sqrt((log(OS/100))^2 + pi^2)
Wn = 4/(Zeta*Ts)
```

14.2 OUTPUT:

```
w_n_ans =

    6.7664

>> zeta_ans

zeta_ans =

    0.5912
```

15 EXERCISE: 11

For the systems given below, find the natural frequency, the damping ratio and transient characteristics. Also plot their step responses on a single graph.

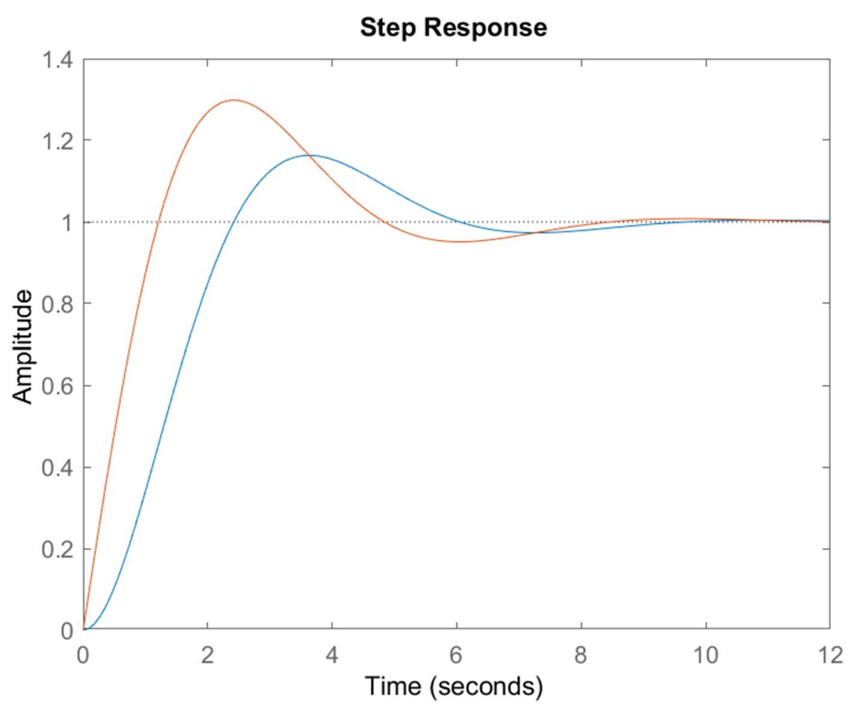
$$\frac{1}{s^2+s+1}, \qquad \frac{s+1}{s^2+s+1}$$

15.1 CODE:

```
num=[1];
dem=[1 1 1];
```

```
tf1=tf(num,dem);
hold on;
damp(tf1);
step(tf1);
stepinfo(tf1);
num1=[1 1];
dem1=[1 1 1];
tf2=tf(num1,dem1);
damp(tf2);
step(tf2);
stepinfo(tf2);
```

15.2 Output



15.3 Comments

Introducing an extra zero to a second-order system engenders a swifter pace at which the step response advances when juxtaposed with the initial system.

16 Exercise 12:

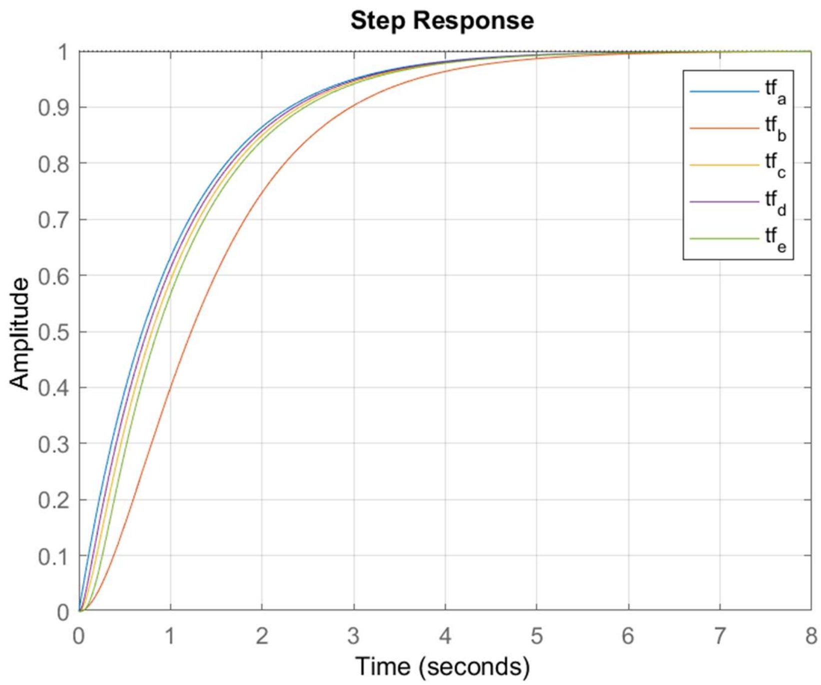
For the systems given below, find the natural frequency, the damping ratio and transient characteristics. Also plot their step responses on a single graph.

$$\frac{1}{(s+1)}, \quad \frac{2}{(s+1)(s+2)}, \quad \frac{10}{(s+1)(s+10)}, \quad \frac{20}{(s+1)(s+20)},$$

16.1 Code:

```
tf_a = zpk([], -1, 1);
tf_b = zpk([], [-1 -2], 2);
tf_c = zpk([], [-1 -10], 10);
tf_d = zpk([], [-1 -20], 20);
tf_e = zpk([], [-1 -10-5*1i -10+5*1i], 125);
figure
t = 0:0.01:8;
hold on
step(tf_a, t)
step(tf_b, t)
step(tf_c, t)
step(tf_d, t)
step(tf_e, t)
grid
legend('tf_a', 'tf_b', 'tf_c', 'tf_d', 'tf_e')
```

16.2 OUTPUT:



Comments:

The introduction of an additional pole to a first-order system results in a deceleration of the speed at which the step response progresses when compared to the original system.

damp_a:
Pole Damping Frequency Time Constant
(rad/seconds) (seconds)
-1.00e+00 1.00e+00 1.00e+00 1.00e+00
damp_b:
Pole Damping Frequency Time Constant
(rad/seconds) (seconds)
-1.00e+00 1.00e+00 1.00e+00 1.00e+00
-2.00e+00 1.00e+00 2.00e+00 5.00e-01
damp_c:
Pole Damping Frequency Time Constant
(rad/seconds) (seconds)
-1.00e+00 1.00e+00 1.00e+00 1.00e+00
-1.00e+01 1.00e+00 1.00e+01 1.00e-01

damp_d:
Pole Damping Frequency Time Constant
(rad/seconds) (seconds)
-1.00e+00 1.00e+00 1.00e+00 1.00e+00
-2.00e+01 1.00e+00 2.00e+01 5.00e-02
damp_e:
Pole Damping Frequency Time Constant
(rad/seconds) (seconds)
-1.00e+00 1.00e+00 1.00e+00 1.00e+00
-1.00e+01 - 5.00e+00i 8.94e-01 1.12e+01 1.00e-01
-1.00e+01 + 5.00e+00i 8.94e-01 1.12e+01 1.00e-01
step_a:
RiseTime: 2.1970
TransientTime: 3.9121
SettlingTime: 3.9121
SettlingMin: 0.9045
SettlingMax: 1.0000
Overshoot: 0
Undershoot: 0
Peak: 1.0000
PeakTime: 10.5458
step_b:
RiseTime: 2.5901
TransientTime: 4.6002
SettlingTime: 4.6002
SettlingMin: 0.9023
SettlingMax: 0.9992
Overshoot: 0
Undershoot: 0
Peak: 0.9992
PeakTime: 7.7827
step_c:
RiseTime: 2.2150
TransientTime: 4.0174
SettlingTime: 4.0174
SettlingMin: 0.9005
SettlingMax: 0.9993

Overshoot: 0
Undershoot: 0
Peak: 0.9993
PeakTime: 7.3591
EE-371: Linear Control Systems Page 17
step_d:
RiseTime: 2.2000
TransientTime: 3.9634
SettlingTime: 3.9634
SettlingMin: 0.9040
SettlingMax: 0.9993
Overshoot: 0
Undershoot: 0
Peak: 0.9993
PeakTime: 7.3222
step_e:
RiseTime: 2.2117
TransientTime: 4.0769
SettlingTime: 4.0769
SettlingMin: 0.9001
SettlingMax: 0.9994
Overshoot: 0
Undershoot: 0
Peak: 0.9994
PeakTime: 7.5433

17 EXERCISE 13

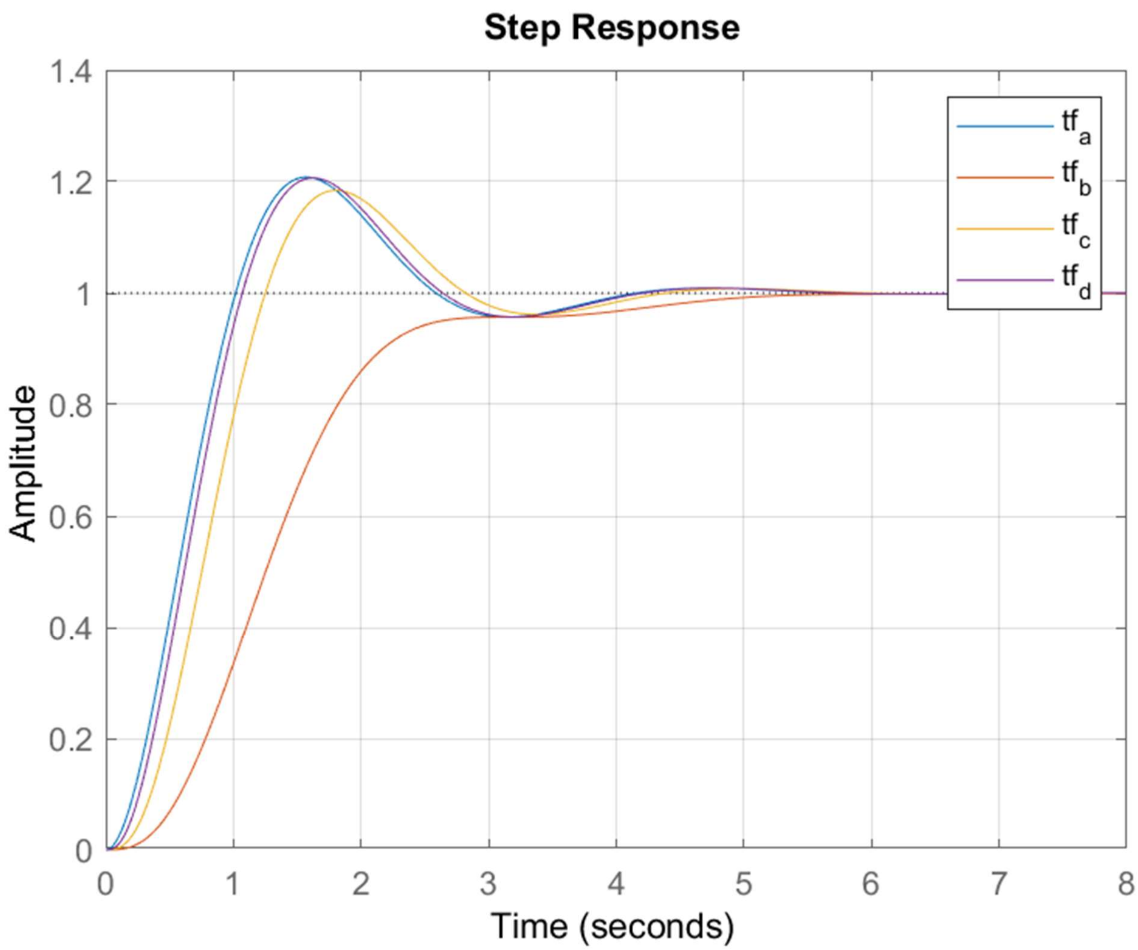
For the systems given below, find the natural frequency, the damping ratio and transient characteristics. Also plot their step responses on a single graph.

$$\frac{5}{(s+1+2i)(s+1-2i)}$$
$$\frac{5}{(s+1)(s+1+2i)(s+1-2i)}$$
$$\frac{25}{(s+5)(s+1+2i)(s+1-2i)}$$
$$\frac{100}{(s+20)(s+1+2i)(s+1-2i)}$$

17.1 CODE:

```
tf_a = zpk([], [-1-2*1i -1+2*1i], 5);
tf_b = zpk([], [-1 -1-2*1i -1+2*1i], 5);
tf_c = zpk([], [-5 -1-2*1i -1+2*1i], 25);
tf_d = zpk([], [-20 -1-2*1i -1+2*1i], 100);
figure
t = 0:0.01:8;
hold on
step(tf_a, t)
step(tf_b, t)
step(tf_c, t)
step(tf_d, t)
grid
legend('tf_a', 'tf_b', 'tf_c', 'tf_d')
```

17.2 Output:



17.3 Comments

Incorporating an extra pole into a second-order system causes a deceleration in the speed at which the step response proceeds in comparison to the initial system.

18 CONCLUSION:

In conclusion, the lab was successful in achieving its objectives of plotting step response of first and second order equations in MATLAB and determining the effect of pole zero locations on step response. The lab also enabled us to find the rise time, transient response, and overshoot for second order systems. Through the experiments conducted, we were able to observe how the placement of poles and zeros in the transfer function of a system can have a significant impact on its step response characteristics, including rise time, settling time, and overshoot. These insights will be invaluable in designing and analyzing control systems in various engineering applications. Overall, this lab provided a comprehensive introduction to the fundamental concepts of system analysis and MATLAB simulation, which will serve as a foundation for further exploration in the field of control systems engineering. In conclusion, the lab was successful in achieving its objectives of plotting step response of first and second order equations in MATLAB and determining the effect of pole zero locations on step response. The lab also enabled us to find the rise time, transient response, and overshoot for second order systems. Through the experiments conducted, we were able to observe how the placement of poles and zeros in the transfer function of a system can have a significant impact on its step response characteristics, including rise time, settling time, and overshoot. These insights will be invaluable in designing and analyzing control systems in various engineering applications. Overall, this lab provided a comprehensive introduction to the fundamental concepts of system analysis and MATLAB simulation, which will serve as a foundation for further exploration in the field of control systems engineering.