EE324: Control Systems Lab
Problem sheet 3

Vinit Awale, 18D070067

January 31, 2021

Contents

1 On pole zero cancellation

1.1 Problem a

We consider the transfer function

$$G_a(s) = \frac{s+5+a}{s^2+11s+30}$$

Now we vary a from -1 to 1 in the steps of 0.1 and observe the step response of the system. For a = 0, the system transfer function reduces to

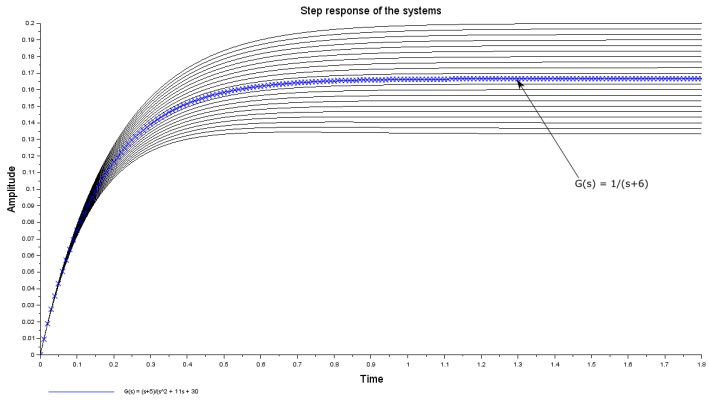
$$G_0(s) = \frac{s+5}{s^2+11s+30} = \frac{1}{s+6}$$

We also observe the step response for this first order transfer function.
The Scilab code for it is given below.

```
clear
close
clc
clf()

s = poly(0,'s');
a = -1:0.1:1;
t = 0:0.01:1.8;

for i = -1:0.1:1
    g = (s+5+i)/(s^2+11*s+30)
    G = syslin('c', g);
    gs = csim('step' , t , G);
    if (i==0)
        plot2d(t,gs, style =2 , leg = 'G(s) = (s+5)/(s^2 + 11
   s + 30')
    else
        plot2d(t,gs)
    end
end

g1 = 1/(s+6);
G1 = syslin('c', g1);
gs1 = csim('step' , t , G1);

plot(t,gs1 ,'x')

xlabel('Time','fontsize',4)
```

```
28 ylabel('Amplitude','fontsize',4)
29
30 title('Step response of the systems', 'fontsize',4)
```

The following plot is obtained.



From the above plot we can easily see that the step response remains for the transfer functions $G_0(s) = \frac{s+5}{s^2+11s+30}$ and $G(s) = \frac{1}{s+6}$

1.2 Problem b

Now consider the transfer function

$$G(s) = \frac{1}{s^2 - s - 6}$$

We plot the step response of the system and obtain its poles using the following Scilab code.

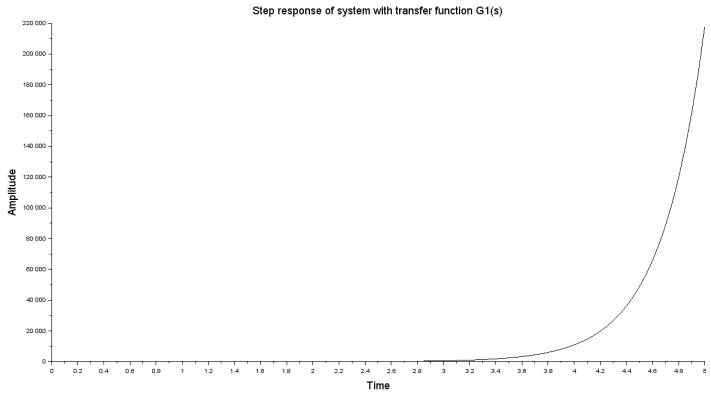
```
1  clear
2  close
3  clc
4
5  s = poly(0,'s');
6  g1 = 1/(s^2 -s-6);
7
8  G1 = syslin('c', g1);
9  t = 0:0.05:5;
10
11 gs1 = csim('step' , t , G1);
```

```
12
13 plot2d(t,gs1)
14 xlabel('Time','fontsize',4)
15 ylabel('Amplitude','fontsize',4)
16
17 title('Step response of system with transfer function G1(s)',
       'fontsize',4)
18
19 [n,d] = simp(1, s^2 - s - 6);
20 poles = roots(d);
21 disp(poles,"The poles of the system are as follows")
```

The poles obtained are as follows

```
The poles of the system are as follows

  3.
 -2.
```

Hence the system has a right half plane pole.
The following plot is obtained.



From the plot obtained above we can see that step response of the system is not bounded. Hence we can say that the system is unstable as we get unbounded output for a bounded input (step signal). This is as expected because our system has a right half plane pole.

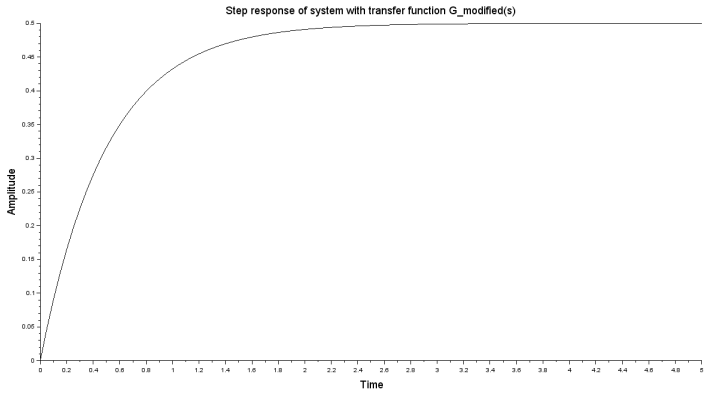
Now, we make a system by cancelling the pole on the right, by introducing a zero at the same location. The transfer function we use is as follows

$$G_{modified}(s) = \frac{s-3}{s^2-s-6}$$

The Scilab code for plotting the step response of this system is given below

```
clear
close
clc

s = poly(0,'s');
g = (s-3)/(s^2 -s-6);

G_modified = syslin('c', g);
t = 0:0.05:5;

gs = csim('step' , t , G_modified);

plot2d(t,gs)
xlabel('Time','fontsize',4)
ylabel('Amplitude','fontsize',4)

title('Step response of system with transfer function
   G_modified(s)', 'fontsize',4)
```

The following plot is obtained.



We can see from the plot that response of the system is bounded. Hence, the system is stable.

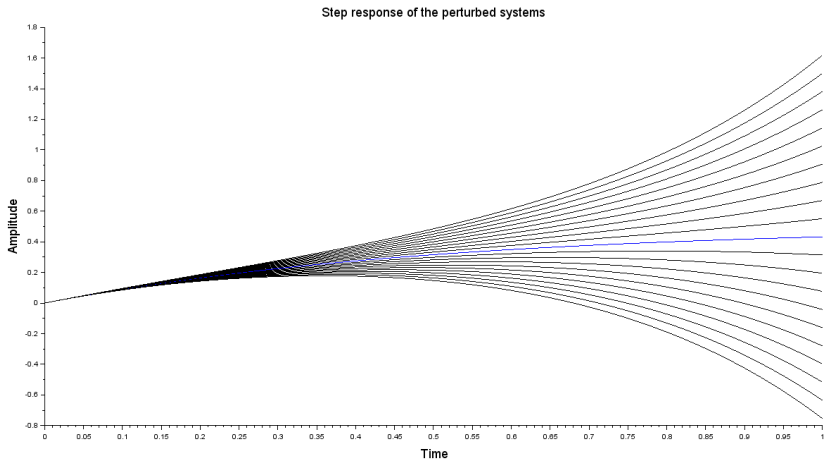
Now we shift the zero of the system slightly. We introduce a variable a and vary it from -1 to 1 in steps of 0.1 and observe the step response of the system. The transfer function used is as follows

$$G_{perturbed} = \frac{s-3+a}{s^2-s-6}$$

The Scilab code to plot the step response for the varying values of a is as follows.

```
clear
close
clc
clf()

s = poly(0,'s');
a = -1:0.1:1;
t = 0:0.01:1;

for i = -1:0.1:1
    g = (s-3+i)/(s^2-s-6)
    G = syslin('c', g);
    gs = csim('step' , t , G);
    if (i==0)
        plot2d(t,gs, style =2)
    else
        plot2d(t,gs)
    end
end

xlabel('Time','fontsize',4)
ylabel('Amplitude','fontsize',4)

title('Step response of the perturbed systems', 'fontsize',4)
```

The obtained plot



From the above plot we can easily see that the step response is bounded only when a = 0. For negative a, the step response goes to $-\infty$ and for positive a, the step response goes to ∞. Hence any slight variation in the value of a makes the system unstable. Hence, we can say that an unstable plant cannot be rendered stable by cancelling unstable poles by adding zeros attempting to cancel the unstable pole.

2 On second order approximation

2.1 Problem a

The given third order transfer function is

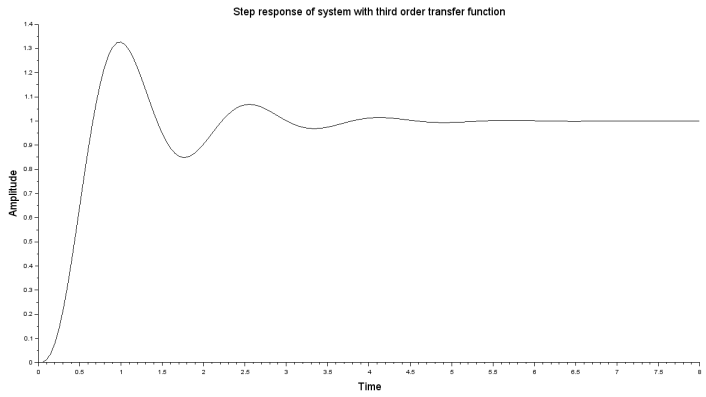
$$G(s) = \frac{85}{s^3 + 7s^2 + 27s + 85}$$

The Scilab code for plotting its step response is given below

```
clear
close
clc

```

```
s = poly(0,'s');
g = 85/(s^3 + 7*s^2 + 27*s + 85);

G = syslin('c', g);
t = 0:0.05:8;

gs = csim('step',t,G);

plot2d(t,gs)

xlabel('Time','fontsize',4)
ylabel('Amplitude','fontsize',4)

title('Step response of system with third order transfer
   function', 'fontsize',4)
```

The obtained step response is as follows



The given transfer function can be written as:

$$G(s) = \frac{85}{(s+5)(s^2+2s+17)}$$

The poles of the system are at -5, $-1 \pm 4i$. Since, -5 is 5 times the real part of the dominant poles, we can find an equivalent second order system with transfer function

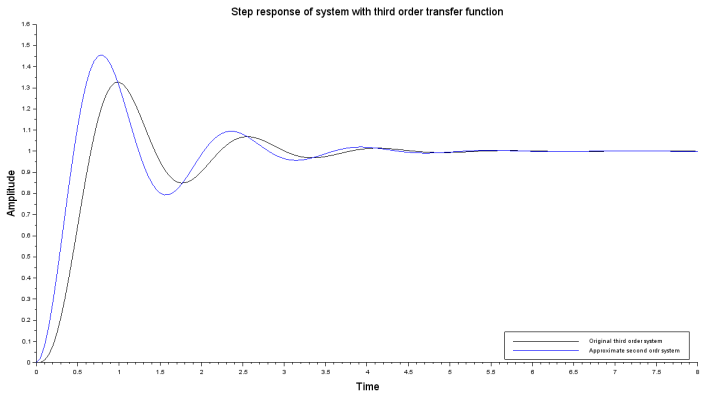
$$G_{approx}(s) = \frac{17}{s^2+2s+17}$$

Here, we have scaled the numerator appropriately.
Now, let us compare the step response of the two systems

Scilab code

```
clear
close
clc

s = poly(0,'s');
g = 85/(s^3 + 7*s^2 + 27*s + 85);
g_app = 17/(s^2 + 2*s + 17)
G = syslin('c', g);
G_app = syslin('c', g_app);
t = 0:0.05:8;

gs = csim('step',t,G);
gs_app = csim('step',t,G_app);
plot2d(t,gs)
plot2d(t , gs_app,style = 2)

h = legend(['Original third order system' , 'Approximate
    second ordr system'], 'fontsize',4)

xlabel('Time','fontsize',4)
ylabel('Amplitude','fontsize',4)

title('Step response of system with third order transfer
    function', 'fontsize',4)
```

Obtained plots



Hence we can easily see that the response of the two systems is similar.

2.2 Problem b

The given transfer function is

$$G(s) = \frac{s+0.01}{s^3 + (101/50)s^2 + (126/25)s + 0.1}$$

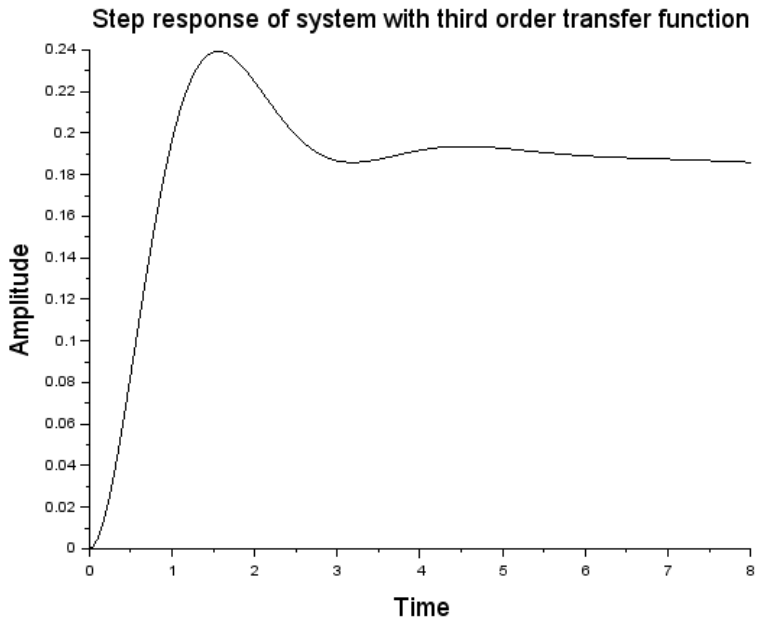
We plot its step response.

Scilab code

```
clear
close
clc

s = poly(0,'s');
g = (s+0.01)/(s^3 + (101/50)*s^2 + (126/25)*s + 0.1);

G = syslin('c', g);
t = 0:0.05:8;

gs = csim('step',t,G);

```

```
13 plot2d(t,gs)
14
15 xlabel('Time','fontsize',4)
16 ylabel('Amplitude','fontsize',4)
17
18 title('Step response of system with third order transfer
      function', 'fontsize',4)
```

Obtained plots



Now, we approximate the given system by a second order system as follows

$$G(s) = \frac{s+0.01}{s^3+(101/50)s^2+(126/25)s+0.1} \implies G(s) = \frac{s+0.01}{(s^2+2s+5)(s+0.02)}$$

Now, to check if we can approximate the system by a second order system by pole zero cancellation, we need to check the residue at the pole -0.02.

```
--> elts = pfss(G)
 elts  =


      elts(1)


  1.0039916 + 0.002016s
  ---------------------
                2
       5 + 2s + s


      elts(2)


  -0.002016
  ----------

  0.02 + s
```

As it is clear from the partial fraction expansion, we can approximate the system using a second order transfer function as follows

$$G_{approx}(s) = \frac{1}{s^2 + 2s + 5}$$

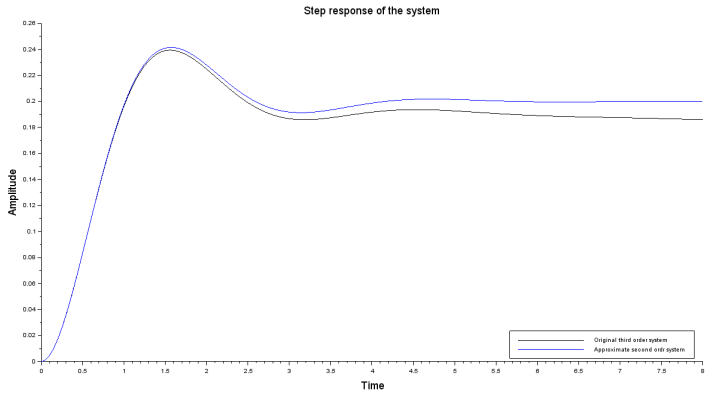
Now, we plot the step response of both the original system and the approximate system

Scilab code

```
clear
close
clc

s = poly(0,'s');
g = (s+0.01)/(s^3 + (101/50)*s^2 + (126/25)*s + 0.1);
g_app = 1/(s^2 + 2*s + 5)
G = syslin('c', g);
G_app = syslin('c', g_app);
t = 0:0.05:8;

gs = csim('step',t,G);
gs_app = csim('step',t,G_app);
plot2d(t,gs)
plot2d(t , gs_app,style = 2)

```

```
17 h = legend(['Original third order system' , 'Approximate
       second ordr system'], 'fontsize',4)
18
19 xlabel('Time','fontsize',4)
20 ylabel('Amplitude','fontsize',4)
21
22 title('Step response of the system', 'fontsize',4)
```

Obtained plots



Hence, from the two plots we can see that the two systems have similar step responses.

3 Effect of additional poles and zeros

3.1 Problem a

The transfer function under consideration is as follows

$$G(s) = \frac{9}{s^2 + 2 * s + 9}$$

The Scilab code and obtained poles of the system using Scilab are as follows:

```
1 clear;
2 close;
3 clc;
4
```

```
5
6  s = poly(0,'s');
7
8  g = 9/(s^2 + 2*s + 9);
9  poles = roots(g.den)
10
11 disp(poles , 'Poles of the system are ')
```

```
Poles of the system are

 -1. + 2.8284271i
 -1. - 2.8284271i
```

Now we add a zero to the system and define a new transfer function as follows

$$G_1(s) = \frac{9(s+1)}{s^2+2s+9}$$

The Scilab code for finding the rise time and percentage overshoot of systems $G(s)$ and $G_1(s)$ is as follows

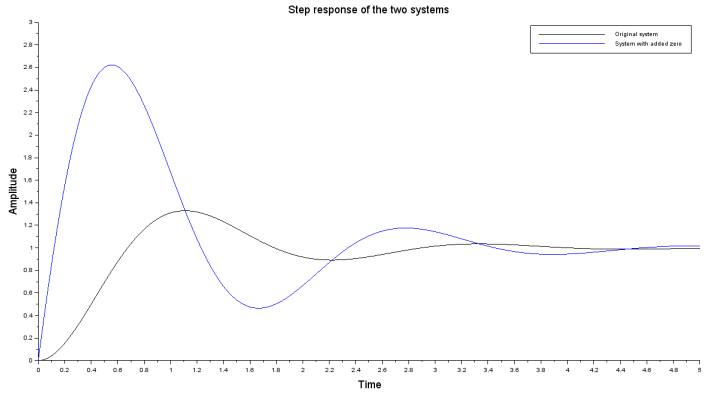
```
1  clear;
2  close;
3  clc;
4
5
6  s = poly(0,'s');
7
8  g = 9/(s^2 + 2*s + 9);
9
10 // Adding zeros to the system
11
12 g1 = g * (s+1)
13 G = syslin('c', g);
14 G1 = syslin('c', g1);
15 t = 0:0.0001:5;
16 gs = csim('step' , t , G);
17 gs1 = csim('step' , t , G1);
18
19 // Finding rise time
20 g_steady = gs($);
21 g1_steady = gs1($);
22
23 ten_percent_indx_g = find(abs(gs - 0.1*g_steady)<0.0001)
```

```
ninety_percent_indx_g = find(abs(gs - 0.9*g_steady)<5e-6)

rise_time_g = t(ninety_percent_indx_g) - t(ten_percent_indx_g
    );
disp(rise_time_g,' Rise time of the original sytem is ')

ten_percent_indx_g1 = find(abs(gs1 - 0.1*g1_steady)<0.0001)
ninety_percent_indx_g1 = find(abs(gs1 - 0.9*g1_steady)<5e-5)

rise_time_g1 = t(ninety_percent_indx_g1) - t(
    ten_percent_indx_g1);
disp(rise_time_g1,' Rise time of the sytem with added zero is
     ')

// Finding percentage overshoot
max_g = max(gs);
max_g1 = max(gs1);

os_g = (max_g - g_steady)/g_steady *100;
os_g1 = (max_g1 - g1_steady)/g1_steady *100;

disp(os_g , 'Percentage overshoot of the original sytem is ')
    ;
disp(os_g1, 'Percentage overshoot of the sytem with added
    zero is ');


plot2d(t , gs, style =1)
plot2d(t, gs1,style =2)

h = legend(['Original system' , 'System with added zero'])
xlabel('Time','fontsize',4)
ylabel('Amplitude','fontsize',4)

title('Step response of the two systems', 'fontsize',4)
```

The output obtained is as follows

```
 Rise time of the original sytem is

  2.1824

 Rise time of the sytem with added zero is

  2.2367

Percentage overshoot of the original sytem
is
  33.245107

Percentage overshoot of the sytem with adde
d zero is
  157.39963
```

The plots of the step responses is as follows:



3.2 Problem b

3.2.1 Adding pole closer to the origin

The transfer function used for adding pole close to the origin

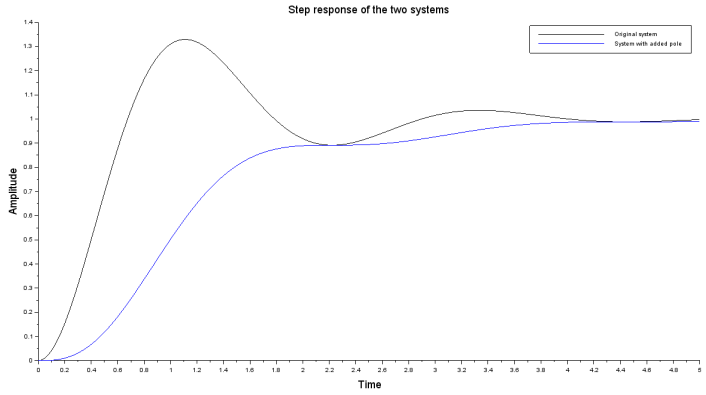
$$G(s) = \frac{9}{(s+1)(s^2+2s+9)}$$

Scilab code

```
clear;
close;
clc;


s = poly(0,'s');

g = 9/(s^2 + 2*s + 9);

// Adding poles to the system

g1 = g/(s+1)
G = syslin('c', g);
G1 = syslin('c', g1);
t = 0:0.001:5;
gs = csim('step' , t , G);
gs1 = csim('step' , t , G1);

// Finding rise time
g_steady = gs($);
g1_steady = gs1($);



ten_percent_indx_g = find((abs(gs - 0.1*g_steady)<0.001))($)
ninety_percent_indx_g = find(abs(gs - 0.9*g_steady)<0.001)(1)

rise_time_g = t(ninety_percent_indx_g) - t(ten_percent_indx_g
    );
disp(rise_time_g,' Rise time of the original sytem is ')

ten_percent_indx_g1 = find(abs(gs1 - 0.1*g1_steady)<0.001)($)
ninety_percent_indx_g1 = find(abs(gs1 - 0.9*g1_steady)<0.001)
    (1)

rise_time_g1 = t(ninety_percent_indx_g1) - t(
    ten_percent_indx_g1);
disp(rise_time_g1,' Rise time of the sytem with added pole is
     ')
// Finding percentage overshoot
max_g = max(gs);
max_g1 = max(gs1);

os_g = (max_g - g_steady)/g_steady *100;
os_g1 = (max_g1 - g1_steady)/g1_steady *100;
```

```
42
43 disp(os_g , 'Percentage overshoot of the original sytem is ')
       ;
44 disp(os_g1, 'Percentage overshoot of the sytem with added
       pole is ');
45
46
47 plot2d(t , gs, style =1)
48 plot2d(t, gs1,style =2)
49
50 h = legend(['Original system' , 'System with added pole'])
51 xlabel('Time','fontsize',4)
52 ylabel('Amplitude','fontsize',4)
53
54 title('Step response of the two systems', 'fontsize',4)
```

Obtained result



```
 Rise time of the original sytem is

  0.453

 Rise time of the sytem with added pole is

  1.553

Percentage overshoot of the original sytem
is
  33.245096

Percentage overshoot of the sytem with adde
d pole is
  0.
```

3.2.2 Adding pole far from the origin

The transfer function used for adding pole close to the origin

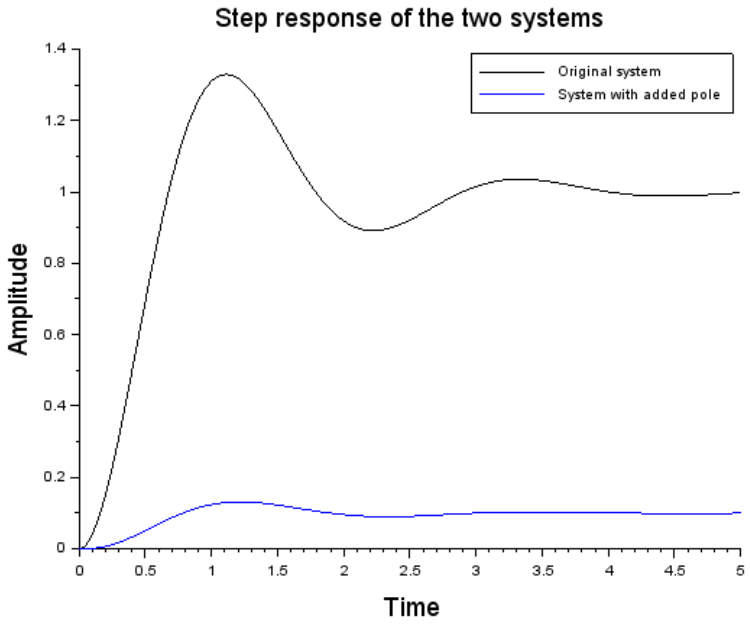
$$G(s) = \frac{9}{(s+100)(s^2+2s+9)}$$

Scilab code

```
clear;
close;
clc;


s = poly(0,'s');

g = 9/(s^2 + 2*s + 9);

// Adding poles to the system

g1 = g/(s+10)
G = syslin('c', g);
G1 = syslin('c', g1);
t = 0:0.001:5;
gs = csim('step' , t , G);
gs1 = csim('step' , t , G1);

// Finding rise time
g_steady = gs($);
g1_steady = gs1($);
```

```
22
23
24
25 ten_percent_indx_g = find((abs(gs - 0.1*g_steady)<0.001))($)
26 ninety_percent_indx_g = find(abs(gs - 0.9*g_steady)<0.001)(1)
27
28 rise_time_g = t(ninety_percent_indx_g) - t(ten_percent_indx_g
       );
29 disp(rise_time_g,' Rise time of the original sytem is ')
30
31 ten_percent_indx_g1 = find(abs(gs1 - 0.1*g1_steady)<0.001)($)
32 ninety_percent_indx_g1 = find(abs(gs1 - 0.9*g1_steady)<0.001)
       (1)
33
34 rise_time_g1 = t(ninety_percent_indx_g1) - t(
       ten_percent_indx_g1);
35 disp(rise_time_g1,' Rise time of the sytem with added pole is
        ')
36 // Finding percentage overshoot
37 max_g = max(gs);
38 max_g1 = max(gs1);
39
40 os_g = (max_g - g_steady)/g_steady *100;
41 os_g1 = (max_g1 - g1_steady)/g1_steady *100;
42
43 disp(os_g , 'Percentage overshoot of the original sytem is ')
       ;
44 disp(os_g1, 'Percentage overshoot of the sytem with added
       pole is ');
45
46
47 plot2d(t , gs, style =1)
48 plot2d(t, gs1,style =2)
49
50 h = legend(['Original system' , 'System with added pole'])
51 xlabel('Time','fontsize',4)
52 ylabel('Amplitude','fontsize',4)
53
54 title('Step response of the two systems', 'fontsize',4)
```

Obtained result




```
 Rise time of the original sytem is

  0.453

 Rise time of the sytem with added pole is

  0.468

Percentage overshoot of the original sytem is

  33.245096

Percentage overshoot of the sytem with added pole is

  31.941511
```

3.3 Problem c

When we add a pole to the system which is close to the dominant pole of the second order system, we cannot approximate the system by that second order

system. The step response and time parameters such as rise time and percentage overshoot are affected drastically as can be seen above.
When a pole is added far away from the dominant poles of a second order system, we can approximate the system by that second order system. The step response and time parameters such as rise time and percentage overshoot are not affected much as can be seen above. The gain of the modified system system is higher in the example shown above. However, that can be taken care of by modifying the transfer function appropriately.
When we add a zero to the system, we add the derivative of the step response of the original system to the scaled version of the step response to obtain the step response of the modified system. Hence, if the zero that is added is close to the origin, then considerable deviation is observed, else it can be ignored if the zero is far away from the origin.

4 Plotting various time domain parameters as a function of ξ and ω_n

4.1 Problem a

The step response for a second order system is as follows:

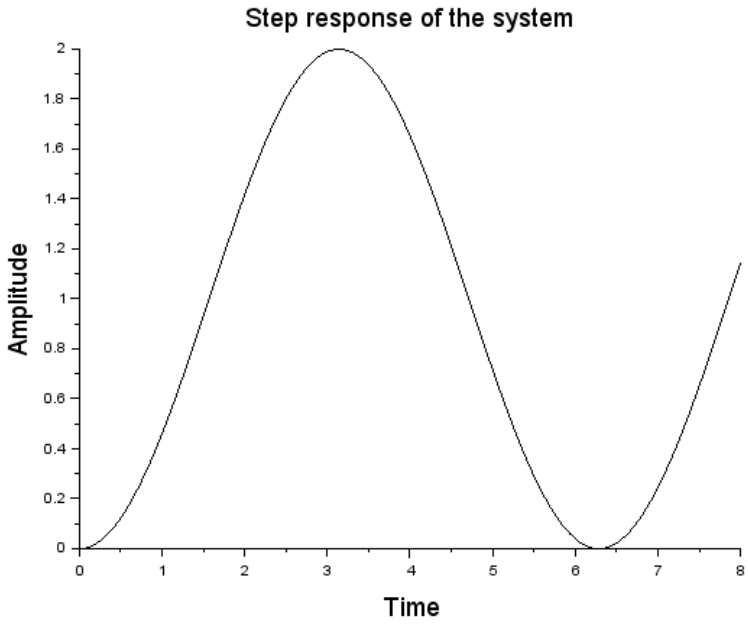
4.1.1 Undamped System

Scilab code

```
clear
close
clc

s = poly(0,'s');
g = 1/(s^2 + 1);
G = syslin('c', g);
t = 0:0.0001:8;

gs = csim('step',t,G);
plot2d(t,gs)

xlabel('Time','fontsize',4)
ylabel('Amplitude','fontsize',4)

```

```
title('Step response of the system', 'fontsize',4)


//Percentage Overshoot
g_steady = gs($);
g_max = max(gs);
per_OS = (g_max-g_steady)/g_steady *100;
disp(per_OS, 'The percentage peak overshoot is ');

//Peak time
peak_t = t(find(gs == g_max))
disp(peak_t,'The peak time is ')

//Settling time
// Infinite for the case of undamped system
disp("The settling time is infinite");

//Delay time
for i = 1:length(t)
        if (abs(gs(i)- 0.5*g_steady) < 0.001)
           t_delay = t(i);
           break;
        end
    end
//delay_t = t(find(gs == 0.5*g_max))
disp(t_delay,'The delay time is ')
```

Obtained result




```
The percentage peak overshoot is

  74.596241

The peak time is

  3.1416

The settling time is infinite

The delay time is

  1.1283
```

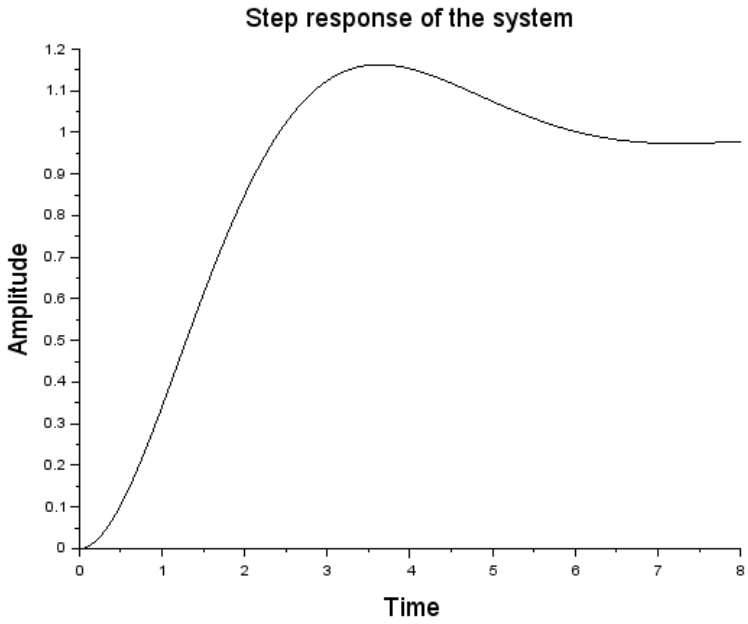
4.1.2 Underdamped system

Scilab code

```
clear
close
```

```
clc

s = poly(0,'s');
g = 1/(s^2 +s+ 1); // xi = 0.5
G = syslin('c', g);
t = 0:0.0001:8;

gs = csim('step',t,G);
plot2d(t,gs)

xlabel('Time','fontsize',4)
ylabel('Amplitude','fontsize',4)

title('Step response of the system', 'fontsize',4)


//Percentage Overshoot
g_steady = gs($);
g_max = max(gs);
per_OS = (g_max-g_steady)/g_steady *100;
disp(per_OS, 'The percentage peak overshoot is ');

//Peak time
peak_t = t(find(gs == g_max))
disp(peak_t,'The peak time is ')

//Settling time
times = find(abs(gs - .98*g_max)<0.001)
disp(t(times($)),"The settling time is ");

//Delay time
for i = 1:length(t)
        if (abs(gs(i)- 0.5*g_steady) < 0.001)
           t_delay = t(i);
           break;
        end
    end
//delay_t = t(find(gs == 0.5*g_max))
disp(t_delay,'The delay time is ')
```

Obtained result



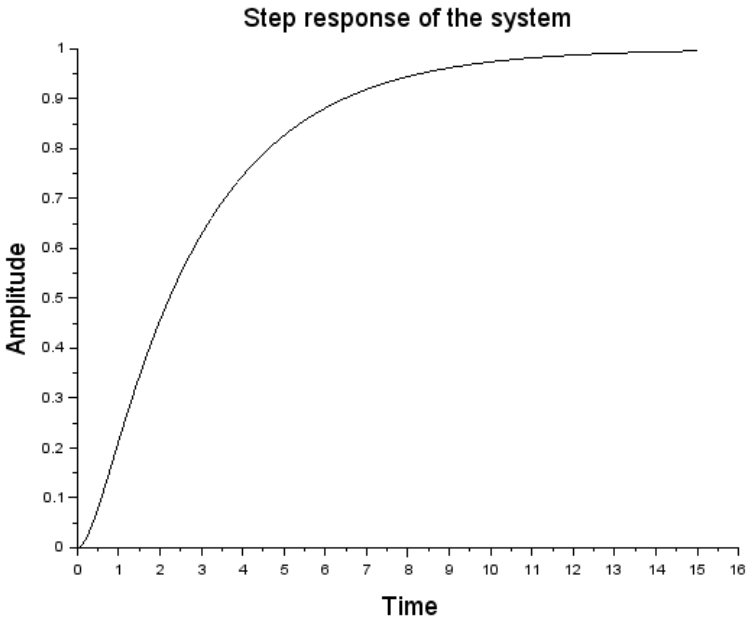

```
The percentage peak overshoot is

  18.79731

The peak time is

  3.6276

The settling time is

  4.2374

The delay time is

  1.273
```

4.1.3 Overdamped system

Scilab code

```
clear
close
clc

s = poly(0,'s');
g = 1/(s^2 +3*s+ 1); // xi = 1.5
G = syslin('c', g);
t = 0:0.0001:15;

gs = csim('step',t,G);
plot2d(t,gs)

xlabel('Time','fontsize',4)
ylabel('Amplitude','fontsize',4)

title('Step response of the system', 'fontsize',4)


//Percentage Overshoot
g_steady = gs($);
g_max = max(gs);
per_OS = (g_max-g_steady)/g_steady *100;
disp(per_OS, 'The percentage peak overshoot is ');

//Peak time
// Infinite for overdamped ssytem
disp('The peak time is infinite')

//Settling time
times = find(abs(gs - .98*g_max)<0.001)
disp(t(times($)),"The settling time is ");

//Delay time
for i = 1:length(t)
        if (abs(gs(i)- 0.5*g_steady) < 0.001)
           t_delay = t(i);
           break;
        end
    end
//delay_t = t(find(gs == 0.5*g_max))
disp(t_delay,'The delay time is ')
```

Obtained result




```
The percentage peak overshoot is

  0.

The peak time is infinite

The settling time is

  10.3199

The delay time is

  2.2097
```

i) Percentage overshoot- As we can observe in the graphs percentage overshoot decreases with increase in damping constant, as the peak deceases with ξ increasing.

ii) 2% settling time-

2% settling time decreases as we increase ξ but only till it gets critically damped

i.e. in range $\xi = [0,1)$. After that in [1,2] settling time again starts to increase as the output takes time to go to it stable state monotonically, as can be seen in the graph.
iii) Rise time-
Rise time goes on increasing as the value of ξ increases. As the output shoots up quickly for lesser values of ξ and thus 90% is reached quickly for lower zetas.
iv) Peak time-
It is clearly visible in graph that the peak time is increasing with increase in ξ.
v) Delay time-
Follows same trend as the rise time.

4.2 Problem b

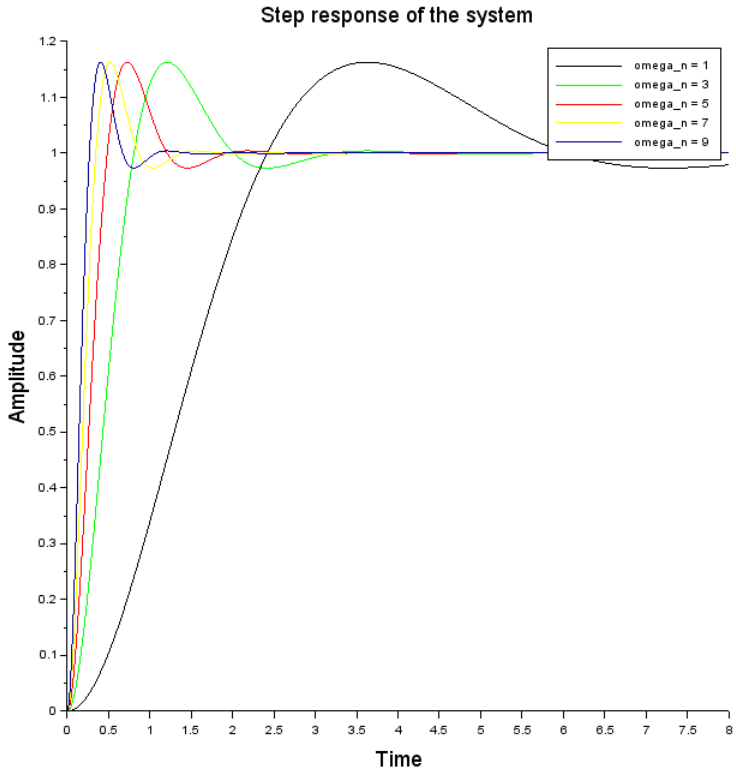
In this part we vary the values of ω_n from 1 to 9 in the steps of 2 to observe the step response

Scilab code

```
clear
close
clc

s = poly(0,'s');
w = 1:2:9; // Natural frequencies
for i = w
    g = i^2/(s^2 +2*(0.5)*i*s+ i^2); // xi = 0.5
    G = syslin('c', g);
    t = 0:0.0001:8;
    gs = csim('step',t,G);
    plot2d(t,gs, style = i)


disp(i ,'The natural frequency is ')
//Percentage Overshoot
g_steady = gs($);
g_max = max(gs);
per_OS = (g_max-g_steady)/g_steady *100;
disp(per_OS, 'The percentage peak overshoot is ');

//Peak time
// Infinite for overdamped ssytem
disp('The peak time is infinite')
```

```
25
26 //Settling time
27 times = find(abs(gs - .98*g_max)<0.001)
28 disp(t(times($)),"The settling time is ");
29
30 //Rise time
31 ten_percent_indx_g = find((abs(gs - 0.1*g_steady)<0.001))($)
32 ninety_percent_indx_g = find(abs(gs - 0.9*g_steady)<0.001)(1)
33
34 rise_time_g = t(ninety_percent_indx_g) - t(ten_percent_indx_g
    );
35
36 //Delay time
37 for i = 1:length(t)
38         if (abs(gs(i)- 0.5*g_steady) < 0.001)
39            t_delay = t(i);
40            break;
41         end
42     end
43 //delay_t = t(find(gs == 0.5*g_max))
44 disp(t_delay,'The delay time is ')
45
46
47
48 end
49
50 h = legend(['omega_n = 1','omega_n = 3','omega_n = 5','
    omega_n = 7','omega_n = 9'])
51 xlabel('Time','fontsize',4)
52 ylabel('Amplitude','fontsize',4)
53
54 title('Step response of the system', 'fontsize',4)
```

Obtained result



```
The natural frequency is

  1.

The percentage peak overshoot is

  18.79731

The peak time is infinite

The settling time is

  4.2374

The delay time is

  1.273

The natural frequency is

  3.

The percentage peak overshoot is

  16.303484

The peak time is infinite

The settling time is

  1.4124

The delay time is

  0.4308
```

```
The natural frequency is

  5.

The percentage peak overshoot is

  16.303353

The peak time is infinite

The settling time is

  0.8474

The delay time is

  0.2585

The natural frequency is

  7.

The percentage peak overshoot is

  16.303353

The peak time is infinite

The settling time is

  0.6053

The delay time is

  0.1847
```

```
The natural frequency is

  9.

The percentage peak overshoot is

  16.303353

The peak time is infinite

The settling time is

  0.4708

The delay time is

  0.1436
```

From the obtained plots and the time parameters we can see that percentage overshoot remains constant as the value of ω_n changes. Rest all the time parameters change with changing ω_n