**EE324 Control Systems Lab**

Problem sheet 3

Tarun Sai Goddu | 190070024

**Question 1**

**1.a** We Plot the step response of the transfer function for a varying from -1 to 1 in steps of 0.01 and check the zero-pole cancellation at a = 0 and also at a=1, there is a pole cancellation. The simp() function was used for rational simplification of the transfer function.

G(s) = s + 5 + a/s + 11s + 30

Scilab Code for the same:

s = poly(0,'s');

t = 0:0.01:2;

for a = -1:0.01:1

G = (s+5+a)/(s^2 + 11\*s + 30)

G\_s = simp(G);

y = csim('step',t,syslin('c', G\_s))

plot(t, y,'k-')

end

a = gca();

a.font\_size = 3;

a.x\_label.font\_size = 3;

a.y\_label.font\_size = 3;

a.data\_bounds = [0,0; 1.6,0.4];

a.title.font\_size = 4;

xlabel("t(s)")

ylabel("Step response, y(t)")

title("Step response at various ''a'' : Observing pole-zero cancellation");

for a = -1:0.2:1

G = (s+5+a)/(s^2 + 11\*s + 30)

G\_s = simp(G)

y1 = csim('step',t,syslin('c', G\_s))

plot(t, y1)

end

a = gca();

a.font\_size = 3;

a.x\_label.font\_size = 3;

a.y\_label.font\_size = 3;

a.title.font\_size = 4;

xlabel("t (s)")

ylabel("Step response, y(t)")

title("Step response at various ''a'' : Observing pole-zero cancellation")

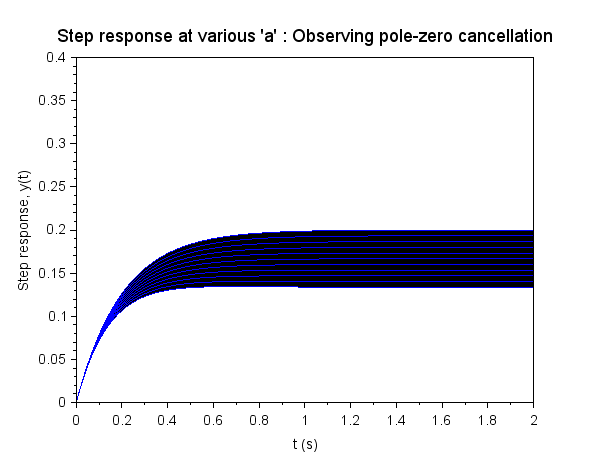


Figure 1: Step response for a = -1:0.2:1

**1.b** We have the system with transfer function:

G(s) = 1 /s^2 - s – 6

The output of the system is unstable as it does not attain stable value

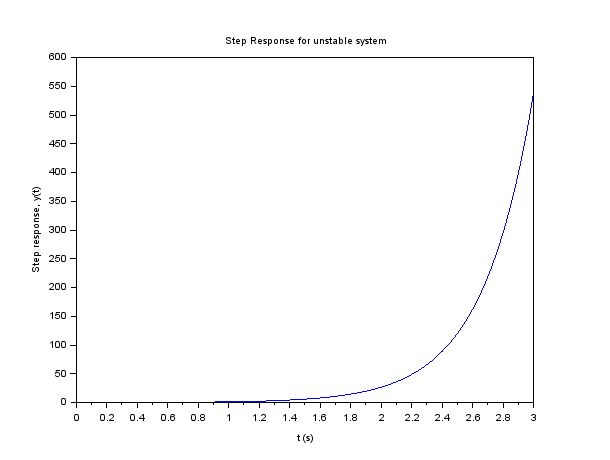


Figure 2: Step response of system (RHP pole with a zero)

This system is unstable due to pole at s = 3 in the RHP. By adding a zero at s = 3, we cancel out the only pole in the RHP, and hence making the system stable. We observe the following plot:

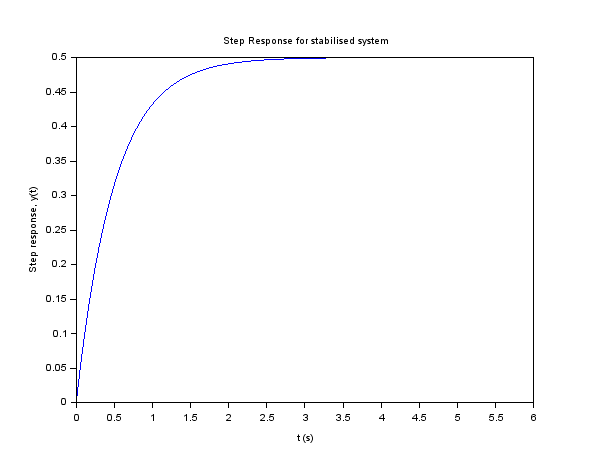


Figure 3: Step response of system after cancellation of RHP pole with a zero

The system output is observed to be stable and takes a steady state value of 0.5. Now we shift the zero by slight increments in a of 0.001 on either side upto 3 values on either side and now observe the output:

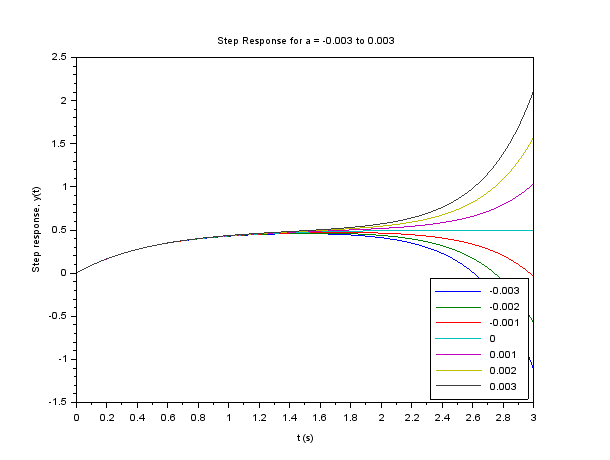


Figure 5: Step Response for a = -.0.03 : 0.01 : 0.03

Output loses it's stability on a slight movement of the zero by 0.1%. Hence, the statement an unstable plant cannot be rendered stable by cancelling unstable poles by adding zeros attempting to cancel the unstable pole is true.

Scilab Code for the Plot:

s = poly(0,'s');

sys1 = syslin('c', 1/(s^2 - s - 6));

t = 0 : 0.01 : 3;

y = csim('step', t, sys1);

plot(t', y);

xlabel("t (s)")

ylabel("Step response, y(t)")

title("Step Response for unstable system");

t = 0 : 0.01 :3 ;

sys2 = syslin('c', (s-3)/(s^2 - s - 6));

y2 = csim('step', t, sys2);

plot(t,y2);

xlabel("t (s)")

ylabel("Step response, y(t)")

title("Step Response for stabilised system");

a = -0.003 : 0.001 : 0.003;

y = zeros(length(t),length(a))

i=1;

for a1 = a

sys = syslin('c', (s - 3 + a1)/(s^2 - s - 6));

y(:,i) = csim('step', t, sys);

i=i+1;

end

plot(t, y)

h1=legend(string(a),4);

xlabel("t (s)")

ylabel("Step response, y(t)")

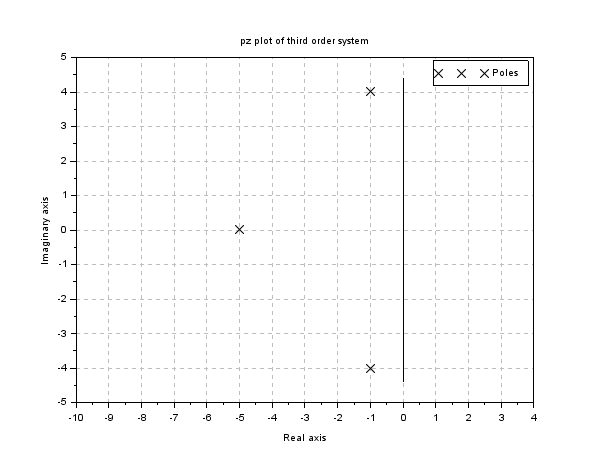
title("Step Response for a = -0.003 to 0.003");

**Question 2**

**2.a** We have the system with transfer function:

G(s) = 85/ s^3+ 7s^2 + 27s + 85

We first plot the step response of the third order system. In order to approximate it as a second order system see the locations of the system's poles, shown in the figure below:



Poles of the system G(s) are -5, -1 ± 4i.

We observe that the pole s = -5 is located far from origin as compared to s = -1 ± 4i and we can use the dominant pole approximation to get an approximate second order system G s given by:

G2 (s) = (1-4i)(1+4i)/(s+1+4i)(s+1-4i) = 17/(s^2+2s+17)

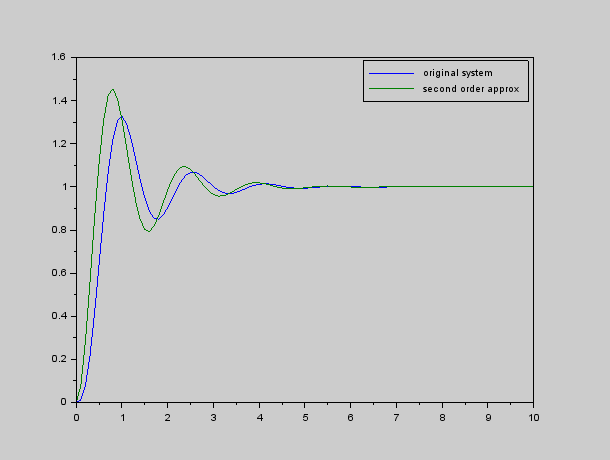


Figure6: Comparison of Third Order System and approximated Second Order System

Scilab code for the above transfer function are :

s=poly(0,'s');

sys1 = syslin('c',85/(s^3+7\*s^2+27\*s+85));

plzr(sys1); *//plots the pole zero of the system*

title('pz plot of third order system');

disp('poles of sysg1 are\n');

disp(roots(sys1.den));

t=0:0.1:10;

y1 = csim('step',t,sys1);

sys2 = syslin('c',17/(s^2+2\*s+17));

y2 = csim('step',t,sys2);

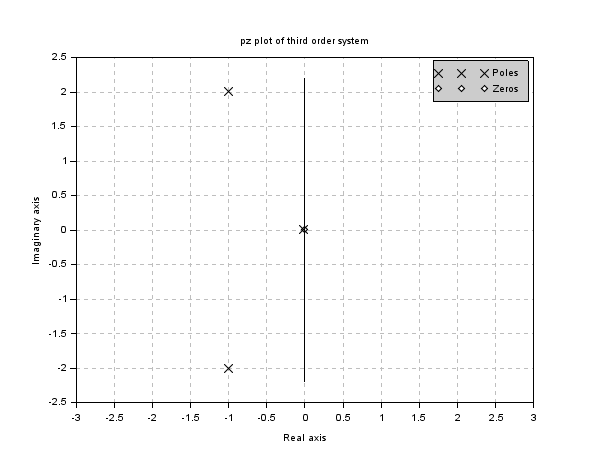
figure;plot(t,y1,t,y2);

h=legend(['original system';'second order approx'])

**2b** We have the system with transfer function:

(s+0.01)/(s^3+(101/50)\*s^2+(126/25)\*s+0.1)

We first plot the step response of the third order system. In order to approximate it as a second order system see the locations of the system's poles, shown in the figure below:



Zero of the system G(s) is at s = -0.01 and poles of the system are at s = -0.02, -1 ± 2i. As one of the zero is very close to dominant pole, pole zero cancellation takes place and we approximate the system as H(s):

0.1\*(1+2i)(1-2i)/(s+1+2i)(s+1-2i) = 0.5/(s^2+2+5)

We can also approximate the system as K(s) by matching the responses in initial instants by taking numerator as 1.

0.2\*(1+2i)(1-2i)/(s+1+2i)(s+1-2i) = 1/(s^2+2+5)

We get the responses as follows for the systems G(s), H(s) and K(s):

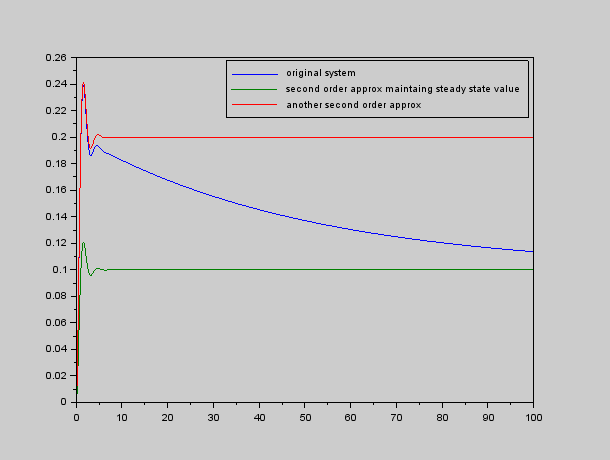


Figure 7: Step Response : Third order system and second order approximation

Scilab code for the above transfer function are :

t=0:0.1:100;

y1 = csim('step',t,sys1);

sys2 = syslin('c',0.5/(s^2+2\*s+5));

y2 = csim('step',t,sys2);

sys3 = syslin('c',1/(s^2+2\*s+5));

y3 = csim('step',t,sys3);

figure;plot(t,y1,t,y2,t,y3);

h=legend(['original system';'second order approx maintaing steady state value';'another second order approx'])

**Question 3**

**3a** We have the system with transfer function:

G1 (s) = 9 /s^2 + 2s + 9

The poles of the system are: -1 ± 2 2i. Now we add a zero to this transfer function at s = -2, thus the new transfer function is:

G2 (s) = 9 (s + 2)/ s^2 + 2s + 9

We get the following plot on comparison of G1 and G2:

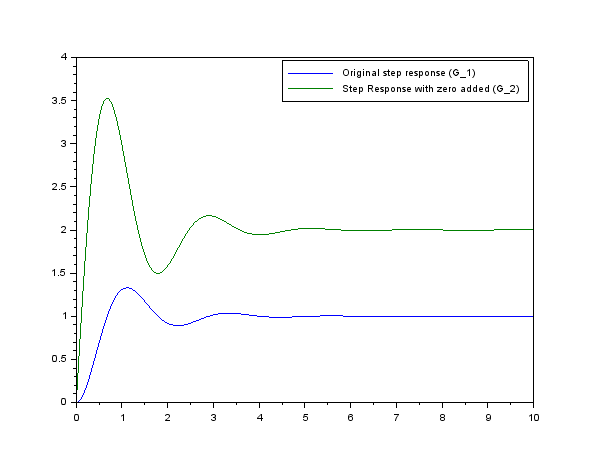


Figure 8: Effect of adding a zero to the step response

Here's how the rise time and the % overshoot fare for both the systems:

G1 (s) = 9/ s^2 + 2s + 9

Rise Time: 0.46s

%overshoot: 32.93%

G2 (s) = 9(s+2/ s^2 + 2s + 9

Rise Time: 0.19s

%overshoot: 76.32%

Scilab code for the above transfer function are :

s = poly(0,'s');

t=0:0.01:10;

G1 = 9/(s^2+2\*s+9);

sysa = syslin('c',G1);

*//plzr(sysa);*

disp("zeros of G1:");

disp(roots(sysa.num));

disp("poles of G1:");

disp(roots(sysa.den));

y1 = csim('step',t,sysa);

ss1 = 1;

t10 = t(find(y1>0.1\*ss1)(1));

t90 = t(find(y1>0.9\*ss1)(1));

tr = t90-t10;

printf("Rise Time = %.2f s\n",tr);

pov1 = (max(y1)-ss1)\*100/ss1;

printf("Percentage overshoot = %3.2f\n",pov1);

G2 = 9\*(s+2)/(s^2+2\*s+9);

sysb = syslin('c',G2);

disp("zeros of G2:");

disp(roots(sysb.num));

disp("poles of G2:");

disp(roots(sysb.den));

y2 = csim('step',t,sysb);

ss2 = 2;

t10 = t(find(y2>0.1\*ss2)(1));

t90 = t(find(y2>0.9\*ss2)(1));

tr = t90-t10;

printf("Rise Time = %.2f s\n",tr);

pov2 = (max(y2)-ss2)\*100/ss2;

printf("Percentage overshoot = %3.2f\n",pov2);

plot(t,y1,t,y2);

h1=legend(['Original step response (G\_1)','Step Response with zero added (G\_2)'],pos="ur");

**3b** In this part, we add a pole to the original transfer function in part (a), in the first case adding a nearby pole and then in the second case a faraway pole

G3(s) = 4.5 /(s^2 + 2s + 9 )(s + 0.5)

G4(s) = 180 /(s^2 + 2s + 9 )(s + 20)

The step responses for the two cases compared with the original system are shown below:

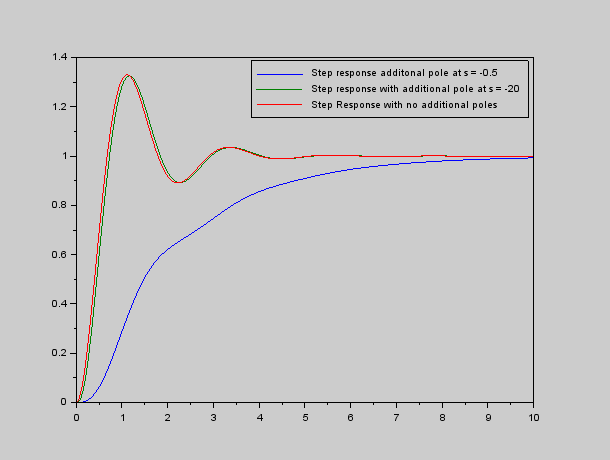


Figure 9: Step response of systems G(s), G3(s) and G4 (s)

Here's how the rise time and the % overshoot fare for both the systems:

G3(s) = 4.5/( s^2 + 2s + 9)(s+0.5)

Rise Time: 4.19s

%overshoot: 0%

G4(s) = 180/ (s^2 + 2s + 9)(s+20)

Rise Time: 0.46s

%overshoot: 32.54%

Scilab code for the above transfer function are:

s = poly(0,'s')

t=0:0.01:10;

G=9/((s^2+2\*s+9));

y=csim('step',t,(syslin('c',G)));

G3 = 4.5/((s^2+2\*s+9)\*(s+0.5))

sysa = syslin('c',G3)

disp("zeros of G3:")

disp(roots(sysa.num))

disp("poles of G3:")

disp(roots(sysa.den))

y1 = csim('step',t,sysa)

ss1 = 1;

t10 = t(find(y1>0.1\*ss1)(1));

t90 = t(find(y1>0.9\*ss1)(1));

tr = t90-t10;

printf("Rise Time = %.2f s\n",tr);

pov3 = (max(y1)-ss1)\*100/ss1; *// is 0 here in this case*

printf("Percentage overshoot = %3.2f\n",pov3);

G4 = 180/((s^2+2\*s+9)\*(s+20))

sysb = syslin('c',G4)

disp("zeros of G4:")

disp(roots(sysb.num))

disp("poles of G4:")

disp(roots(sysb.den))

y2 = csim('step',t,sysb)

ss2 =1;

t10 = t(find(y2>0.1\*ss2)(1));

t90 = t(find(y2>0.9\*ss2)(1));

tr = t90-t10;

printf("Rise Time = %.2f s\n",tr);

pov4 = (max(y2)-ss2)\*100/ss2;

printf("Percentage overshoot = %3.2f\n",pov4);

figure;plot(t,y1,t,y2,t,y)

h1=legend(['Step response additonal pole at s = -0.5','Step response with additional pole at s = -20', 'Step Response with no additional poles'],pos="ur")

**3C** Analysis of effects on response upon addition of zeros and poles

Effect of adding poles:

1. When additional pole is near to the origin than the existing poles,The system behaves approximately as a first order system. The rise time increases, whereas the % overshoot becomes 0.
2. When additional pole is far away from the origin, it has a very little (negligible) effect on the response. As observed rise time increases and the % overshoot decreases, though both changes are very small. This was expected from the dominant pole approximation

Effect of adding Zeros:

1. The rise time increases and the % overshoot decreases.
2. This is expected as adding a zero is simply taking a linear combination of the original response and it's derivative.

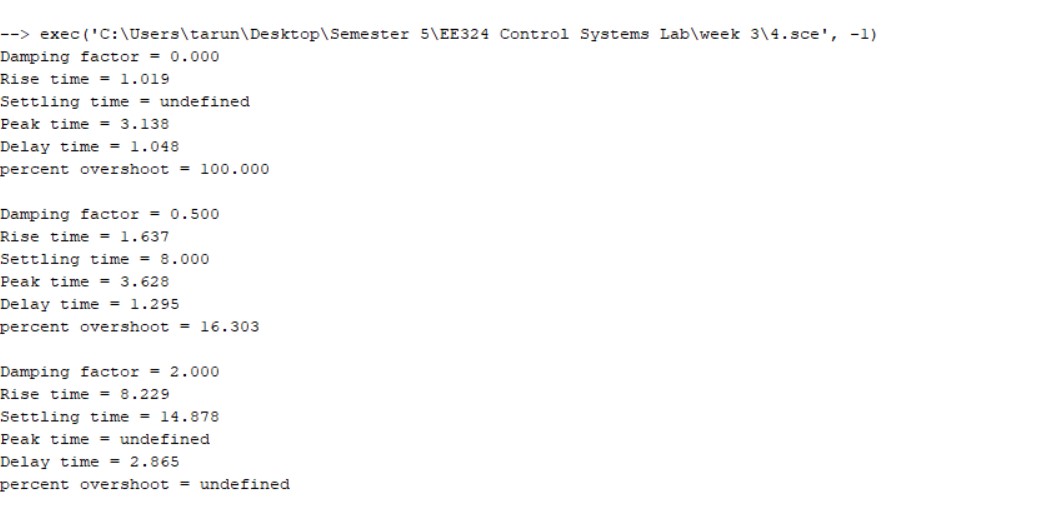
**Question 4**

The standard closed loop second order transfer function with undamped natural frequency 𝜔n is:

G(s) = 𝜔n^2 /s^2 + 2𝜁𝜔 s + 𝜔n^2 For 𝜔n = 1

the Transfer function is: G(s) = 1/s^2 + 2𝜁s + 1

the following time domain parameter variation was observed:



• % Overshoot - Decreases as damping factor increases and is defined only for 𝜁 < 1

• Peak Time - Increases as damping factor increases and is defined only for 𝜁 < 1

• Delay Time - Increases as damping factor increases.

• Rise Time - Increases as damping factor increases.

• Settling Time - Decreases as damping factor increases till 1 and increases as damping factor further increases. It is defined only for 𝜁 > 0.

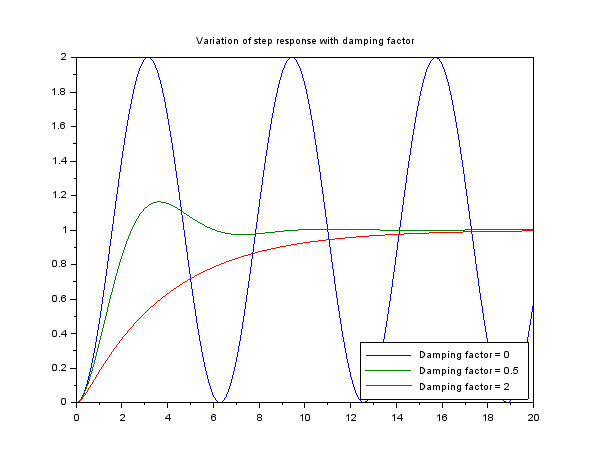


Figure 10: Variation of Step Response with Damping Factor

Scilab code :

s=poly(0,'s');

wn = 1;

t=0:0.001:20;

zeta = [0;0.5;2];

ss=1;

y1=[];

for idx=1:length(zeta)

sys = syslin('c',wn^2/(s^2+2\*zeta(idx)\*wn\*s+wn^2));

y = csim('step',t,sys);

y1=[y1;y];

t10 = t(find(y>0.1\*ss)(1));

t90 = t(find(y>0.9\*ss)(1));

tr = t90-t10;

if zeta(idx)==0 then

ov = (max(y)-ss)\*100/ss;

tp = t(find(y>=max(y)-1e-5)(1));

td = t(find(y>0.5\*ss)(1));

tss(idx)=-1;

printf("Damping factor = %1.3f\n",zeta(idx));

printf("Rise time = %.3f\n",tr)

printf("Settling time = undefined\n")

printf("Peak time = %.3f\n",tp)

printf("Delay time = %.3f\n",td)

printf("percent overshoot = %3.3f\n\n",ov)

elseif zeta(idx)<1 then

ov = (max(y)-ss)\*100/ss;

tp = t(find(y==max(y)));

td = t(find(y>0.5\*ss)(1));

ts = 4/(wn\*zeta(idx));

printf("Damping factor = %1.3f\n",zeta(idx));

printf("Rise time = %.3f\n",tr)

printf("Settling time = %.3f\n",ts)

printf("Peak time = %.3f\n",tp)

printf("Delay time = %.3f\n",td)

printf("percent overshoot = %3.3f\n\n",ov)

else

td = t(find(y>0.5\*ss)(1));

ts = t(find(y>0.98\*ss)(1));

printf("Damping factor = %1.3f\n",zeta(idx));

printf("Rise time = %.3f\n",tr)

printf("Settling time = %.3f\n",ts)

printf("Peak time = undefined\n")

printf("Delay time = %.3f\n",td)

printf("percent overshoot = undefined\n")

end;

end;

plot(t',y1');

title('Variation of step response with damping factor');

h2 = legend('Damping factor = ' + string(zeta),pos="in\_lower\_right");