EE324 Problem Sheet 3

Abhilaksh Kumar(18D070035)

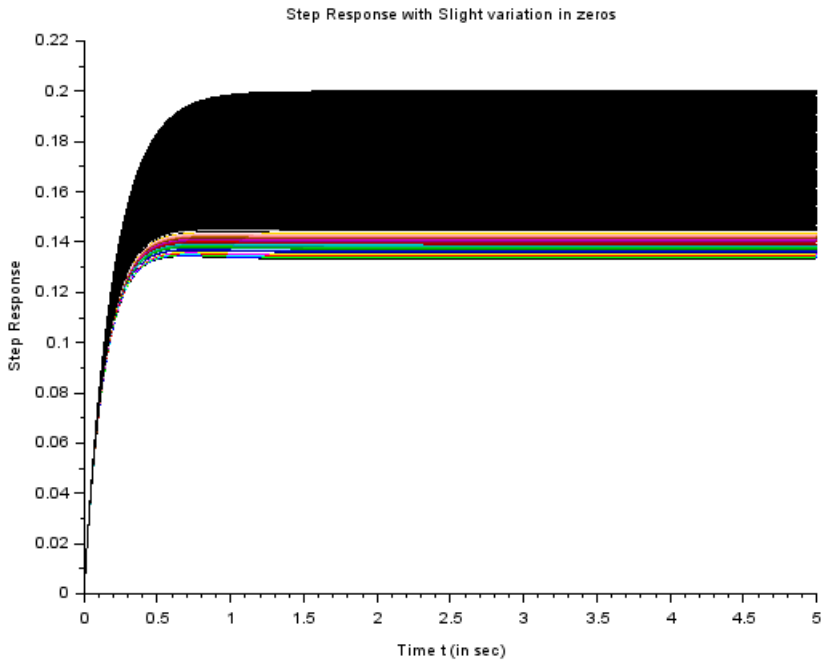
30^{th} January 2021

Question 1

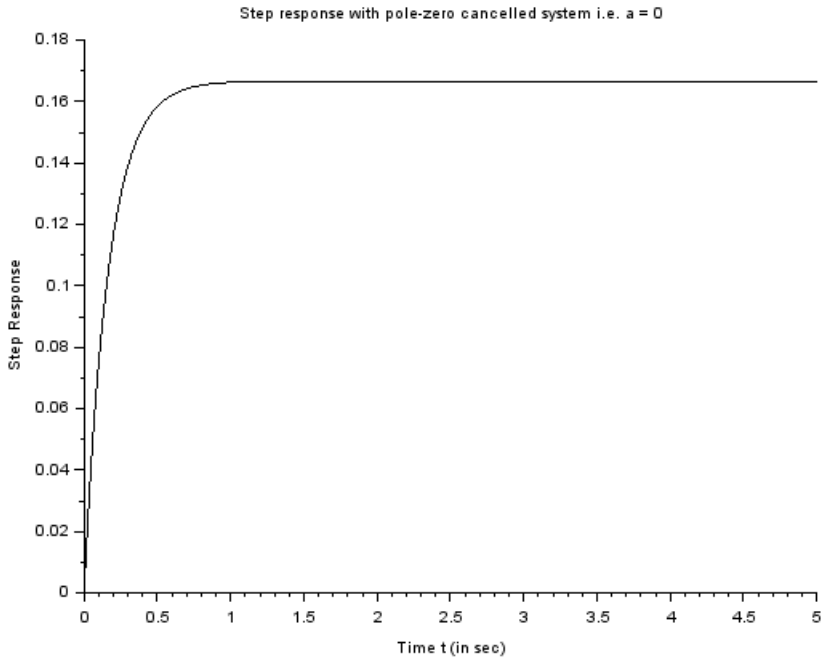
Part (a)

In this part, we plot the step responses of the transfer function for diff zeros (by varying the parameter a)

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



The shape of step responses of all the systems are identical (though different steady state values). We can also explicitly check for the condition a = 0:



Hence pole zero cancellation does indeed work for stable systems like above.

Code For Part (a)

```
s = poly(0,'s');

// PART a
a_range = -1:.01:1.001;
t = 0:.005:5;
outputs = zeros(length(t), size(a_range)(2));        // length returns
    length for either col or row vector (more convenient than size fucntion
    )
for i = 1:size(a_range)(2)
    a = a_range(i);
    G = (s+5+a)/(s^2+11*s+30);
    G = syslin('c',G);
    outputs(:,i) = csim('step', t,G);
end

// when plotting multiple y on same plot, individual y must be a col
    vector
plot2d(t,outputs);
xlabel('Time t (in sec)');
ylabel('Step Response');
title('Step Response with Slight variation in zeros');
show_window(1);
```

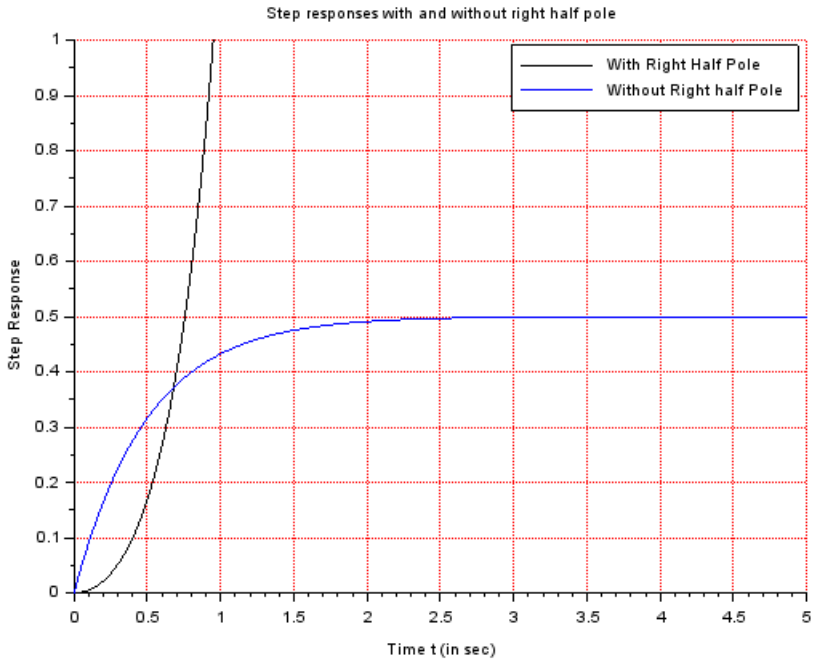
```
20
21 // Explicitly plotting for pole-zero cancellation
22 G1 = 1/(s+6);
23 G1 = syslin('c',G1);
24 plot2d(t, csim('step', t,G1));
25 xlabel('Time t (in sec)');
26 ylabel('Step Response');
27 title('Step response with pole-zero cancelled system i.e. a = 0');
28 show_window(2);
```

Part (b)

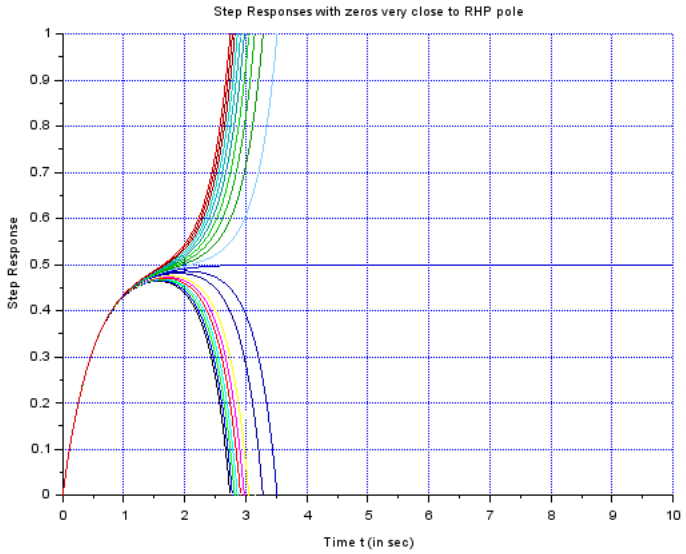
The given transfer function:

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

The output of original TF is indeed unstable as it doesn't hit a steady state value Upon adding a zero s = 3 to cancel out the pole in the right half plane, we observe that the system is stable with a steady state value of 0.5. Here's the output in both the cases:



Now we shift the zero by slight increments of 0.0002 on either side & observe the step responses:



We see that even a small variation of 0.0002 makes the step response unbounded very quickly. Thus, we can say that to cancel an unstable pole, we need to add zeroes which exactly cancels that pole. Adding zeroes which attempt to cancel that pole won't help in stabilizing the response.

Code for Part (b)

```
G = 1/(s^2-s-6);
G = syslin('c',G);
y1 = csim('step', t,G);

G_new = (s-3)/(s^2-s-6);
G_new = syslin('c',G_new);
y2 = csim('step', t,G_new);
plot2d(t,[y1',y2']);
xgrid(5, 1, 7);                          // color, thickness, style
a = gca();
a.data_bounds = [0,0;5,1];
legend(['With Right Half Pole', 'Without Right half Pole'])
xlabel('Time t (in sec)');
ylabel('Step Response');
title('Step responses with and without right half pole');
show_window(3);

t = 0:.005:10;
a_range = -.002:.0002:.002;
outputs = zeros(length(t), length(a_range));
for i = 1:length(a_range)
    a = a_range(i);
    G = (s-3+a)/(s^2-s-6);
```

```
    G = syslin('c',G);
    outputs(:,i) = csim('step', t,G);
end
plot2d(t,outputs);
a = gca();
a.data_bounds = [0,0;10,1];
xgrid(2);
xlabel('Time t (in sec)');
ylabel('Step Response');
title('Step Responses with zeros very close to RHP pole');
show_window(4);
```

Question 2

Part (a)

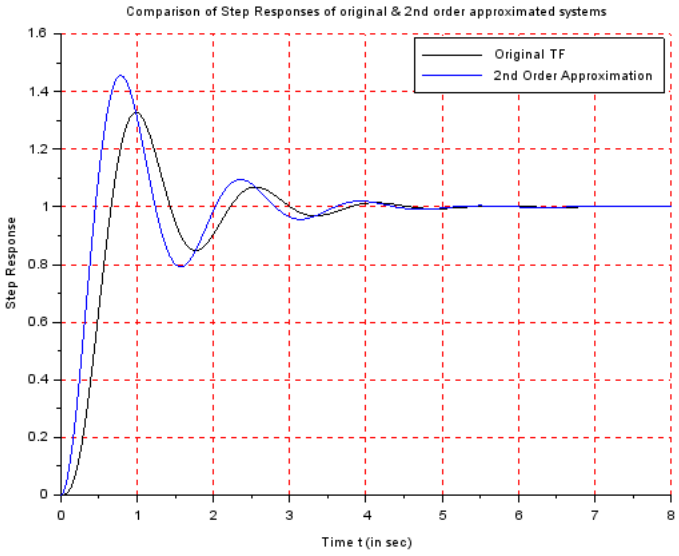
The given transfer function is

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

As a second order approximation we throw away the pole whose real part is -ve and much larger in magnitude than the other one as they will decay faster and for the most part the pole with smaller magnitude will manifest in response. We find that the poles of this transfer function are -5, -1+4i, -1-4i. Hence, we can scrap out the faraway pole at -5, and build a new transfer function as:

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

Here's the step response comparison of both:



Code for Part (a)

```
s = poly(0,'s');

// PART a
t = 0:.005:8;
G = 85/(s^3+7*s^2+27*s+85);
G = syslin('c',G);
y1 = csim('step', t,G);

G_approx = 17/(s^2+2*s+17);
G_approx = syslin('c',G_approx);
y2 = csim('step', t,G_approx);
plot2d(t,[y1',y2']);
xgrid(5,1,3);
legend(['Original TF', '2nd Order Approximation']);
xlabel('Time t (in sec)');
ylabel('Step Response');
title('Comparison of Step Responses of original & 2nd order approximated
   systems');
show_window(1);
```

Part (b)

The given transfer function is‘

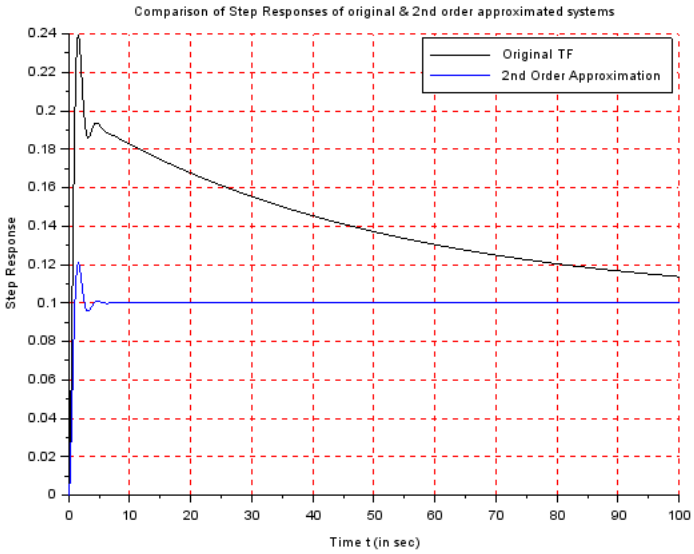
$$G(s) = \frac{s + 0.01}{s^3 + 101/50s^2 + 126/25s + 0.1}$$

The poles of this function are -0.02, -1 + 2i and -1 2i. The pole at -0.02 is closer to the zero at -0.01 and hence we can attempt to cancel it. I've also added a scaling factor to preserve the original steady state value

Preserving Steady State Value:

$$G_{approx}(s) = \frac{0.5}{s^2 + 2s + 5}$$

The responses are as shown below:



After observing the wide difference between response values for initial 100 sec or so, we can say that this isn't a good case to cancel poles and zeros due to large error.

```
t = 0:.05:100;
G = (s+.01)/(s^3+2.02*s^2+5.04*s+.1);
G = syslin('c',G);
y1 = csim('step', t,G);

G_approx = .5/(s^2+2*s+5);
G_approx = syslin('c',G_approx);
y2 = csim('step', t,G_approx);
plot2d(t,[y1',y2']);
xgrid(5,1,3);
legend(['Original TF', '2nd Order Approximation']);
xlabel('Time t (in sec)');
ylabel('Step Response');
title('Comparison of Step Responses of original & 2nd order approximated
   systems');
show_window(2);
```

Question 3

Part (a)

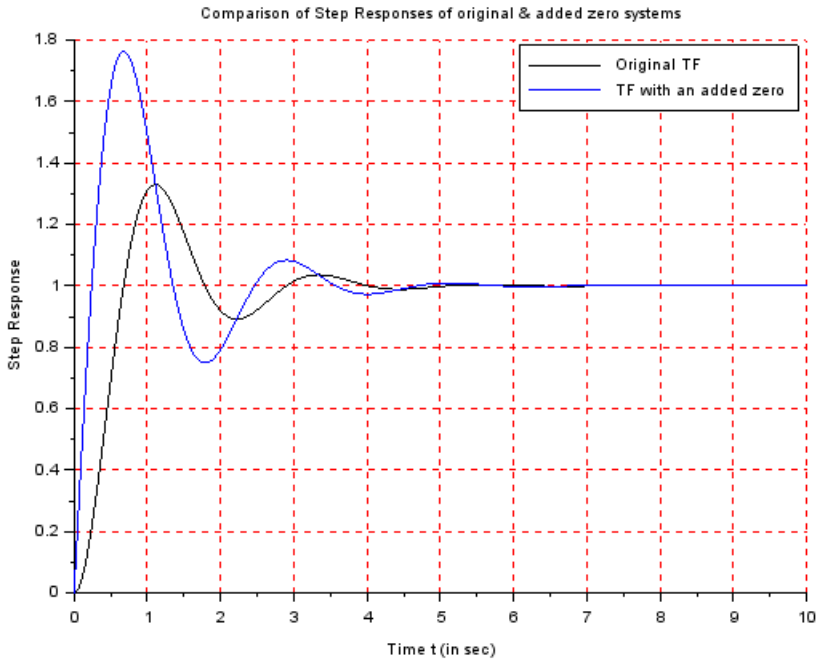
The original system function is as given below:

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

The poles of the system are: -1. + 2.8284271i and -1. - 2.8284271i. Now we add a zero to this transfer function at -2, keeping the steady state value same. The new transfer function is:

$$G_n(s) = \frac{4.5(s+2)}{s^2+2s+9}$$

Here are the step responses for both these systems:



Here's how the various parameters fare for both the systems:

System	Rise Time	% overshoot	Peak time	Settle Time
G(s)	0.46 s	32.93	1.11	.663
G_n(s)	0.19 s	76.33	.676	.235

Code for Part (a)

```
s = poly(0,'s');
t = 0:.001:10;

// PART a
G = 9/(s^2+2*s+9);
G = syslin('c',G);
[zeros_,poles,k]=tf2zp(G);          // transfer function to zeros poles

```

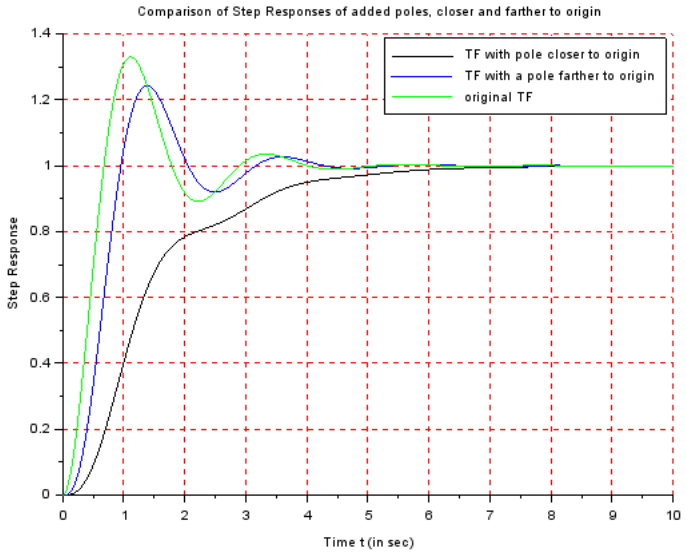
```
G1= (9/2)*(s+2)/(s^2+2*s+9);       // function is normalised st dc gain =
    1 (same as original)
G1 = syslin('c',G1);

y = csim('step',t, G);  // original step response
y1 = csim('step',t, G1);  // step response with an added zero
o = [y;y1];

steady_state_value = 1;
param = zeros(2,4);

for i = 1:2
    param(i,1) = 100*(max(o(i,:)) - steady_state_value);              // %
    overshoot
    t1 = t(find(o(i,:)>.9))(1);                  // find function returns
   array of indices of elements satisfying the criterion
    t2 = t(find(o(i,:)>.1))(1);                  // we want the first
   instance
    param(i,2) = t1-t2;                          // rise time
    param(i,3) = t(find(o(i,:) == max(o(i,:))));           // peak time
    param(i,4) =  t(find(o(i,:)>.98 & o(i,:) < 1.02))(1);          //
   settle time
end

disp(param(1,:));
disp(param(2,:));
plot2d(t, o');
xgrid(5,1,3);
legend(['Original TF', 'TF with an added zero']);
xlabel('Time t (in sec)');
ylabel('Step Response');
title('Comparison of Step Responses of original & added zero systems');
show_window(1)
```

Part (b)

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 far away pole, keeping in mind to preserve the steady state value of the system. The two transfer functions are as given below:

$$G_1(s) = \frac{.75 * 9}{(s^2 + 2s + 9)(s + .75)} \qquad G_2(S) = \frac{4 * 9}{(s^2 + 2s + 9)(s + 4)}$$

The step responses for both are given below:



Here's how the various parameters fare for both the systems:

System	Rise Time	% overshoot	Peak time	Settle Time
$G_1(s)$	2.771	0	10	5.424
G_2(s)	.575 s	24.244	1.382	.937

Code for Part (b)

```
// pole closer to origin than existing ones
G_b1 = (.75*9)/((s^2+2*s+9)*(s+.75));
G_b1 = syslin('c',G_b1);

// pole farther to origin than exisiting ones
G_b2 = (4*9)/((s^2+2*s+9)*(s+4));
G_b2 = syslin('c',G_b2);

y_b1 = csim('step',t,G_b1);
y_b2 = csim('step',t,G_b2);
o = [y_b1;y_b2];

param_b = zeros(2,4);
for i = 1:2
    param_b(i,1) = 100*(max(o(i,:)) - steady_state_value);              //
    % overshoot
    t1 = t(find(o(i,:)>.9))(1);                  // find function returns
   array of indices of elements satisfying the criterion
    t2 = t(find(o(i,:)>.1))(1);                  // we want the first
   instance
    param_b(i,2) = t1-t2;                          // rise time
```

```
    param_b(i,3) = t(find(o(i,:) == max(o(i,:))));            // peak time
    param_b(i,4) =  t(find(o(i,:)>.98 & o(i,:) < 1.02))(1);          //
   settle time
end

disp(param_b(1,:));
disp(param_b(2,:));
plot2d(t, cat(2,o',y'));                // cat function concatenates
   vectors, 2 => along col
xgrid(5,1,3);
legend(['TF with pole closer to origin', 'TF with a pole farther to origin
   ', 'original TF']);
xlabel('Time t (in sec)');
ylabel('Step Response');
title('Comparison of Step Responses of added poles, closer and farther to
   origin');
show_window(2)
```

Part (c)

- When we add a zero to the function, the percentage overshoot increases and the rise time decreases as well. Peak & settle time also decrease.
- On adding a pole, nearer than the original poles, we sort of get an overdamped situation, with 0 % overshoot and we also witness an increase in rise time.
- When we add a far away pole to the function, as the magnitude increases it approaches the original transfer function's step response.

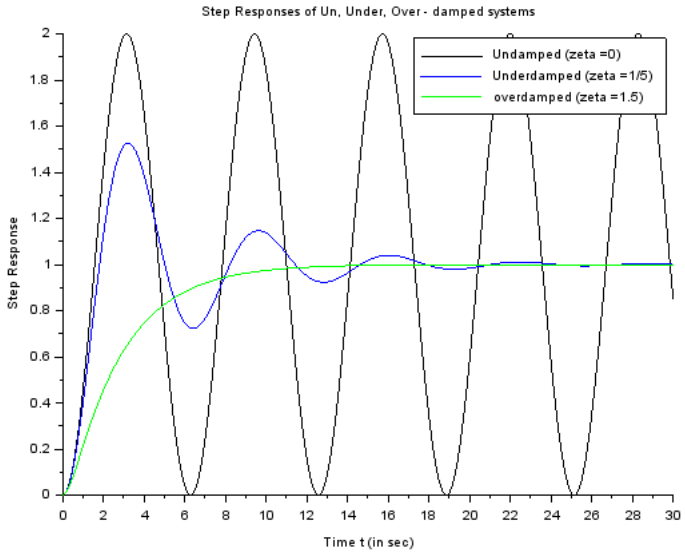
Question 4

Part (a)

In this part, we find the standard parameters of a second order system for an underdamped, overdamped and undamped condition. The 2nd order general transfer function is as given below:

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

The freq =1 for this part. The step responses are as follows:



The formulae for the three cases are given below:

- **Underdamped:**

$$t_r = \frac{1.8}{\omega_n} \quad t_p = \frac{\pi}{\omega_n\sqrt{1-\zeta^2}} \quad t_s = \frac{-ln(0.02\sqrt{1-\zeta^2})}{\zeta\omega_n} \quad t_d = \frac{1+0.7\zeta}{\omega_n} \quad \%\text{os} = e^{\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}}$$

- **Undamped:** Rise time, settling time, delay time and % overshoot are all undefined as the output doesn't hit a final steady state value

$$t_p = \frac{\pi}{\omega_n} = \pi$$

- **Overdamped:** No closed form solution for t_r, t_s, t_d, t_p and % overshoot is 0.

Here's how the various parameters fare for under-damped and over-damped systems:

System	Rise Time	% overshoot	Peak time	Settle Time	Delay time
$\text{G}_{under}(s)$	1.203	52.662	3.206	1.781	1.134
G_{over}(s)	5.859 s	0	30	10.665	2.225

Observations:

- Rise time increases as damping factor increases
- Settling time decreases first when ζ increases upto 1 and later on increases
- Peak time increases as damping factor increases

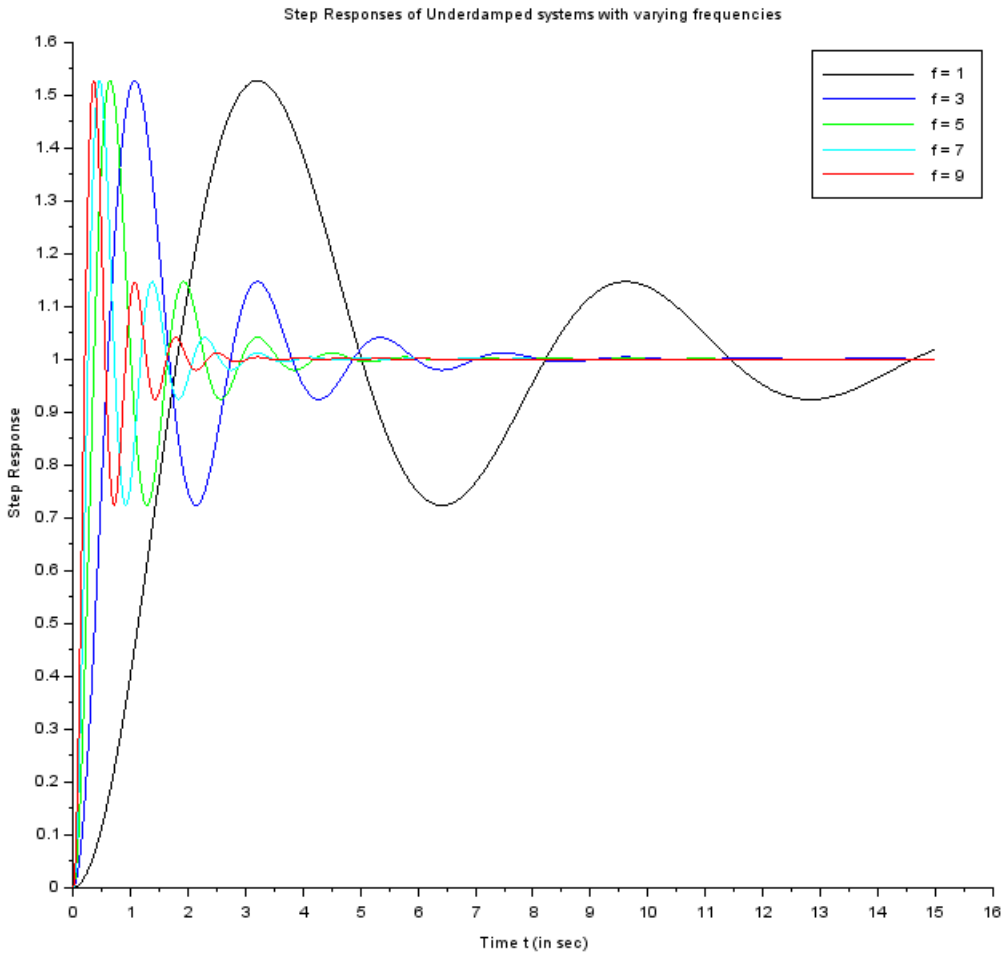
- Delay time increases as damping factor increases
- Percentage overshoot decreases as ζ increases, is 0 for overdamped
- All quantities other than peak time are undefined for undamped condition

Code for Part (a)

```
s = poly(0,'s');
t = 0:.001:30;

// PART a
G1 = 1/(s^2+1);                 // undamped
G2 = 1/(s^2+.4*s+1);          // underdamped (zeta = 1/5)
G3 = 1/(s^2+3*s+1);         // overdamped  (zeta = 1.5)

G1 = syslin('c',G1);
G2 = syslin('c',G2);
G3 = syslin('c',G3);

y1 = csim('step',t,G1);
y2 = csim('step',t,G2);
y3 = csim('step',t,G3);

plot2d(t,[y1',y2',y3']);
legend(['Undamped (zeta =0)','Underdamped (zeta =1/5)','overdamped (zeta
   =1.5)']);
xlabel('Time t (in sec)');
ylabel('Step Response');
title('Step Responses of Un, Under, Over - damped systems');
show_window(1)

param = zeros(2,5);                       // for over & underdamped only
o = [y2;y3;y1];
steady_state_value = 1                 // both over & underdamped have
   steady state value of 1
for i = 1:2
    param(i,1) = 100*(max(o(i,:)) - steady_state_value);              // %
    overshoot
    t1 = t(find(o(i,:)>.9))(1);                  // find function returns
   array of indices of elements satisfying the criterion
    t2 = t(find(o(i,:)>.1))(1);                  // we want the first
   instance
    param(i,2) = t1-t2;                          // rise time
    param(i,3) = t(find(o(i,:) == max(o(i,:))));           // peak time
    param(i,4) = t(find(o(i,:)>.98 & o(i,:) < 1.02))(1);         // settle
    time
    param(i,5) = t(find(o(i,:) >= .5))(1)                             //
   delay time (time to reach 50% in first oscillation)
end
```

Part (b)

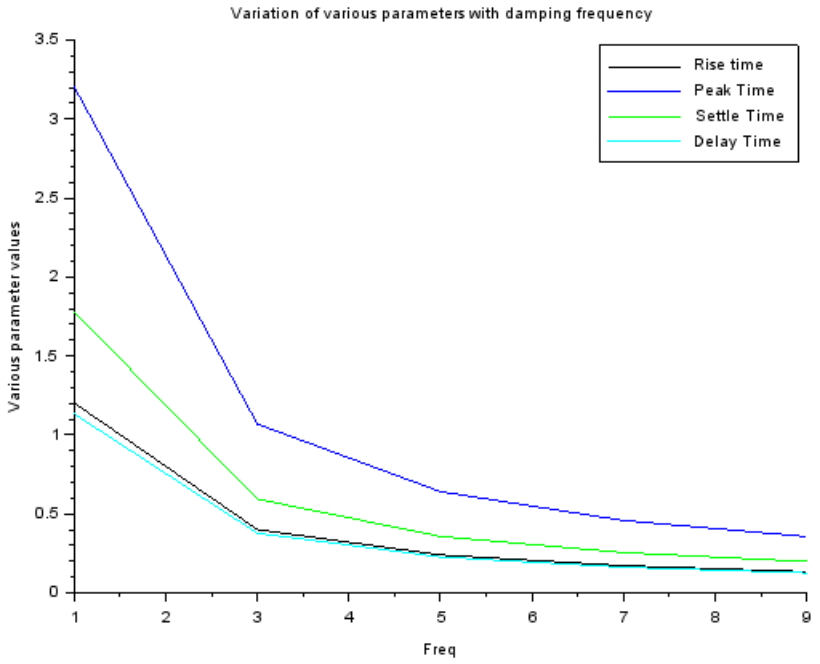
In this part we vary the frequency in steps of 2 starting from 1 for a fixed damping ratio (= 1/5 in my case) for which the system is underdamped. Here are the step responses:



The parameters were calculated using find method in scilab. These are as given below: (The OS should be exactly identical but since we are calculating via numerical methods there is some error in the 4th decimal place)

Freq	%OS	Rise Time	Peak Time	Settle Time	Delay Time
1	52.66206	1.203	3.206	1.781	1.134
3	52.66205	0.401	1.069	0.594	0.378
5	52.66201	0.24	0.641	0.357	0.227
7	52.66206	0.172	0.458	0.255	0.162
9	52.66191	0.134	0.356	0.198	0.126

One can also plot the variations of various parameters.



Observations:

- % **overshoot:** Remains unchanged as it depends only upon the damping factor
- **Rise Time:** Decreases as we increase the frequency, which is expected as faster oscillations
- **Settling Time:** Decreases as we increase the frequency, as the exponential decaying term becomes stronger and stronger
- **Peak Time:** Decreases as we increase the frequency, which is expected as faster oscillations
- **Delay Time:** Decreases as we increase the frequency, which is expected as faster oscillations

Code for Part (b)

```
// the damping ratio is still kept as 1/5
t = 0:.001:15;
o = zeros(5, length(t));
f_range = [1 3 5 7 9];                  // note that steady state value is
   independent of f and is = 1
param_b = zeros(5,5);
for i = 1:5
    f = f_range(i);
    G = f^2/(s^2+f^2+.4*f*s);
    G = syslin('c',G);
    o(i,:) = csim('step',t,G);
    param_b(i,1) = 100*(max(o(i,:)) - steady_state_value);              //
    % overshoot
    t1 = t(find(o(i,:)>.9))(1);                  // find function returns
   array of indices of elements satisfying the criterion
    t2 = t(find(o(i,:)>.1))(1);                  // we want the first
   instance
    param_b(i,2) = t1-t2;                          // rise time
    param_b(i,3) = t(find(o(i,:) == max(o(i,:))));           // peak time
    param_b(i,4) = t(find(o(i,:)>.98 & o(i,:) < 1.02))(1);         //
   settle time
    param_b(i,5) = t(find(o(i,:) >= .5))(1);
end

plot2d(t,o');
legend(['f = 1','f = 3', 'f = 5', 'f = 7', 'f = 9']);
xlabel('Time t (in sec)');
ylabel('Step Response');
title('Step Responses of Underdamped systems with varying frequencies');
show_window(2)


plot2d(f_range,param_b(:,[2 3 4 5]));
legend(['Rise time', 'Peak Time', 'Settle Time', 'Delay Time']);
xlabel('Freq');
ylabel('Various parameter values');
title('Variation of various parameters with damping frequency');
show_window(3)
```