EE 324: Problem Sheet-2

Abhilaksh Kumar, 18D070035

January 24, 2021

1 PROBLEM-1

The entire code snippet :-

```
s = poly(0,'s');
G = 35/(s+1);  // a = 35, b = 1 (Abhilaksh, 18D070035)
G = syslin('c',G);   // convert rational poly to cont. system

// Part 1(b)
t = 0:.01:10;
y_b = csim('step', t, G);  // output to step response

tau = 1;  // 1/b
settling_time = -1*tau*log(.02);
rise_time = tau*(log(.9/.1));   // overall rise time, time diff btw 90% & 10%
plot2d(t, y_b)
xlabel('time');
ylabel('system response');
title('unit step response of G');
show_window(1)

// Part 1(c)
a = 35;
range_a = a:a:100*a;
range_rise = linspace(rise_time,rise_time,100);
plot2d(range_a,range_rise);
xlabel ("a") ;
ylabel (" Rise Time (in secs )") ;
title (" Variation of Rise Time with a") ;
show_window(2)

// Part 1(d)
b = 1;
range_b = b:b:100*b;
range_rise_b = log(.9/.1)./range_b;
plot2d(range_b,range_rise_b);
xlabel ("b") ;
ylabel (" Rise Time (in secs )") ;
title (" Variation of Rise Time with b") ;
```

Part a & b

- Name is Abhilaksh Kumar
- Roll No is 18D070035

```
s = poly(0,'s');
G = 35/(s+1);  // a = 35, b = 1 (Abhilaksh, 18D070035)
G = syslin('c',G);   // convert rational poly to cont. system
```

```
// Part 1(b)
t = 0:.01:10;
y_b = csim('step', t, G);  // output to step response

tau = 1;  // 1/b
settling_time = -1*tau*log(.02);
rise_time = tau*(log(.9/.1));   // overall rise time, time diff btw 90% & 10%
plot2d(t, y_b)
xlabel('time');
ylabel('system response');
title('unit step response of G');
show_window(1)
```

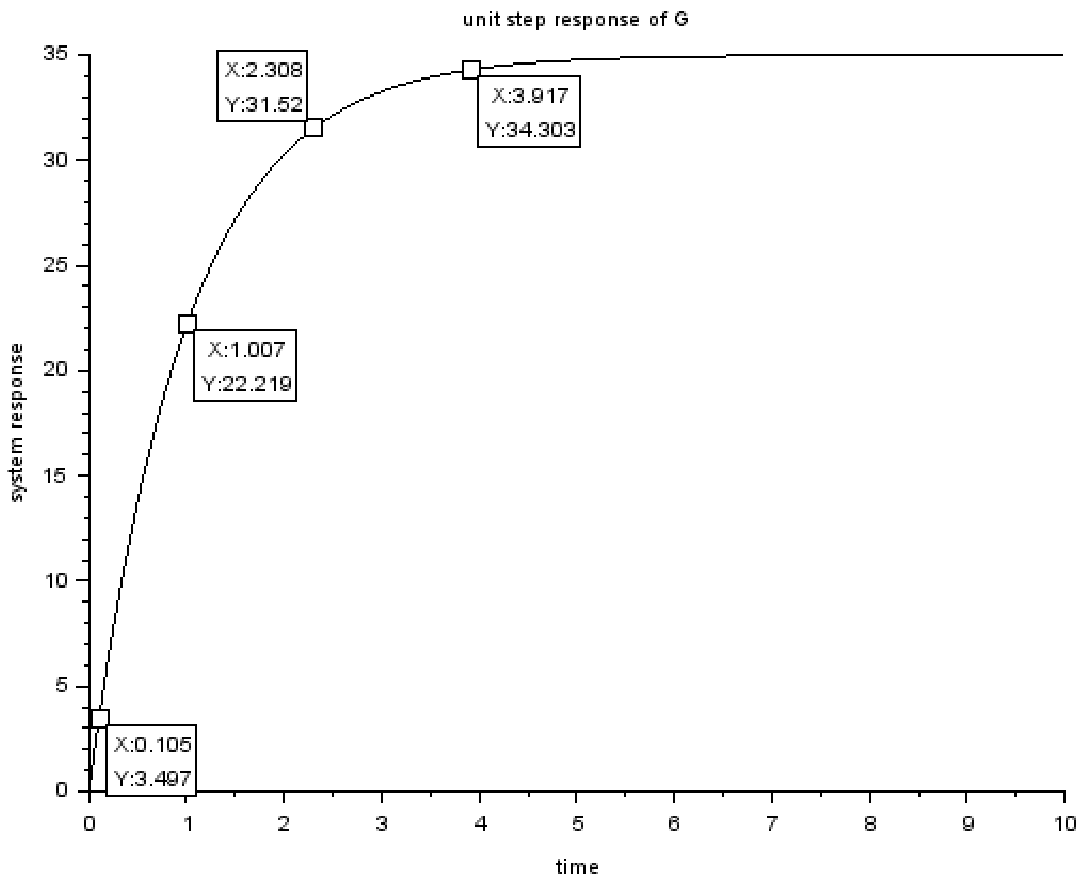


Figure 1: Step Response of G

Part c

- Rise time between 90% and 10%
- Steady state value is a/b
- Time constant = 1 unit
- For settling time, we want time such that y = .98*a/b

It remains constant with respect to variation in a The code for that is:-

```
a = 35;
range_a = a:a:100*a;
range_rise = linspace(rise_time,rise_time,100);
plot2d(range_a,range_rise);
xlabel ("a") ;
ylabel (" Rise Time (in secs )") ;
title (" Variation of Rise Time with a") ;
show_window(2)
```

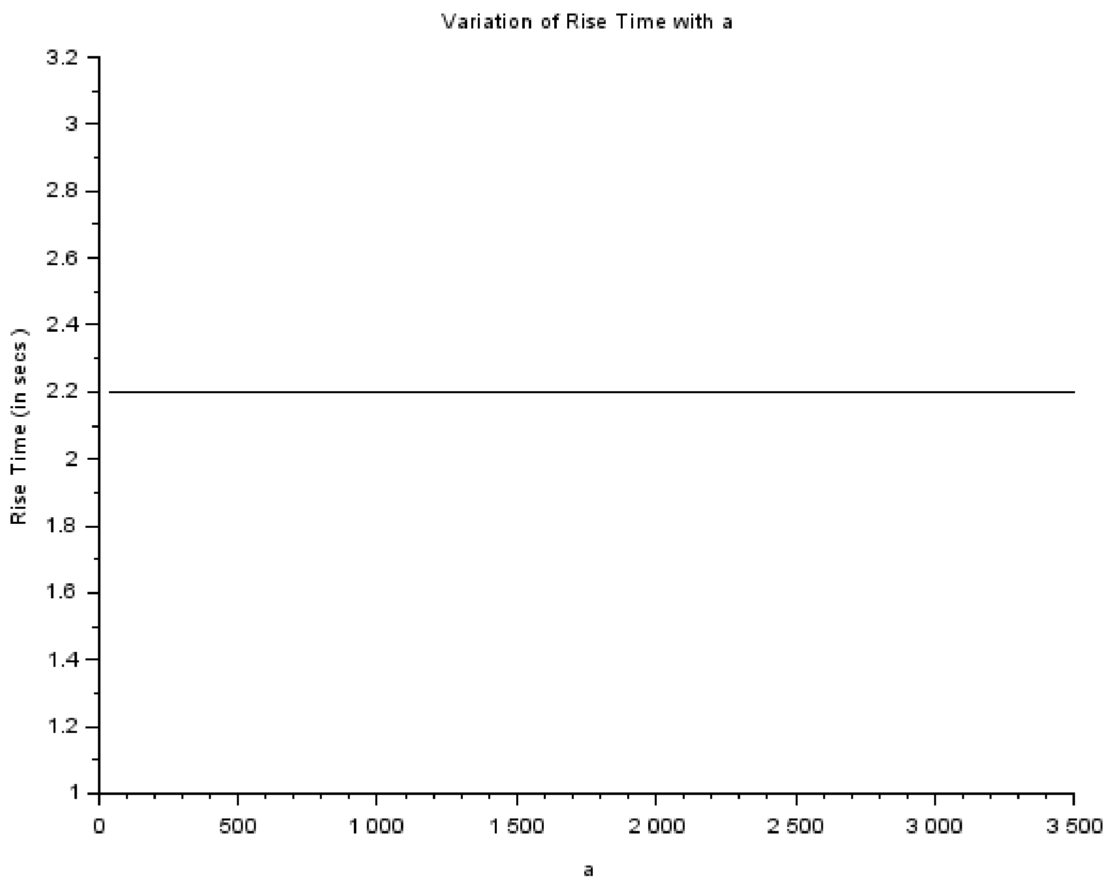


Figure 2: Rise Time vs 'a'

Part d

The rise time varies inversely with b, that is $RiseTime \propto \frac{1}{b}$, Its a rectangular hyperbola:-

```
b = 1;
range_b = b:b:100*b;
range_rise_b = log(.9/.1)./range_b;
plot2d(range_b,range_rise_b);
xlabel ("b") ;
ylabel (" Rise Time (in secs )") ;
title (" Variation of Rise Time with b") ;
```

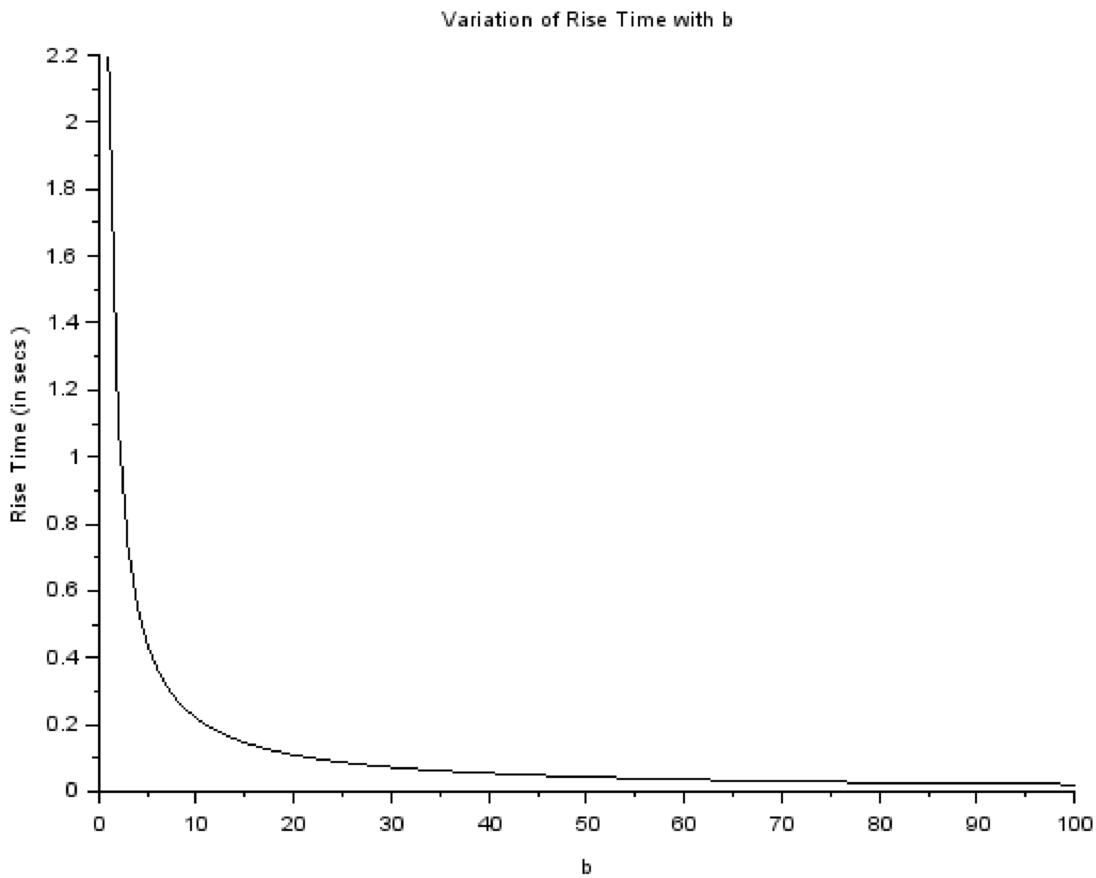


Figure 3: Rise Time vs 'b'

2 PROBLEM-2

The entire code snippet :-

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

dr = 1/5; // damping ratio
w = 1; // frequency of oscillation

G = w^2 /( s ^2 + 2* dr * w * s + w ^2) ;
G = syslin('c',G);

t = 0:.01:50;
plot2d(t,csim('step',t,G));
xlabel (" time (in secs )") ;
ylabel (" system response ") ;
title (" Step Response ");
show_window(1);

dr_range = 0:.25:2.1;     // There are 9 values of dr to check (inclusive of 2)
//settle_time = zeros(1,9);
//rise_time = zeros(1,9);
//os = zeros(1,9);                // percentage overshoot
// peak_time = zeros(1,9);
outputs_matrix = zeros(size(t)(2),9);    //size returns a tuple rxc
for i = 1:9;
    zeta = dr_range(i);
    G = w^2 /( s ^2 + 2* zeta * w * s + w ^2) ;
    G = syslin('c',G);
    outputs_matrix(:,i) = csim('step',t,G);
end

plot2d(t,outputs_matrix);
legend(['0','.25','.5','.75','1','1.25','1.5','1.75','2']);
xlabel (" time (in secs )") ;
ylabel (" system response ") ;
title (" Step Responses for varying damping ratios");
show_window(2);
```

Part a

We know that such functions have the form:-

$$G = \frac{\omega_0^2}{s^2 + 2\zeta\omega_0 s + \omega_0^2}$$

Where ζ is the damping ratio, which needs to be between 0 and 1 for an underdamped system.

- $\zeta = 1/5$ for my system
- w = 1 Hz since its standard

```
s = poly(0,'s');
dr = 1/5; // damping ratio
w = 1; // frequency of oscillation

G = w^2 /( s ^2 + 2* dr * w * s + w ^2) ;
G = syslin('c',G);

t = 0:.01:50;
plot2d(t,csim('step',t,G));
xlabel (" time (in secs )") ;
ylabel (" system response ") ;
title (" Step Response ");
show_window(1);
```

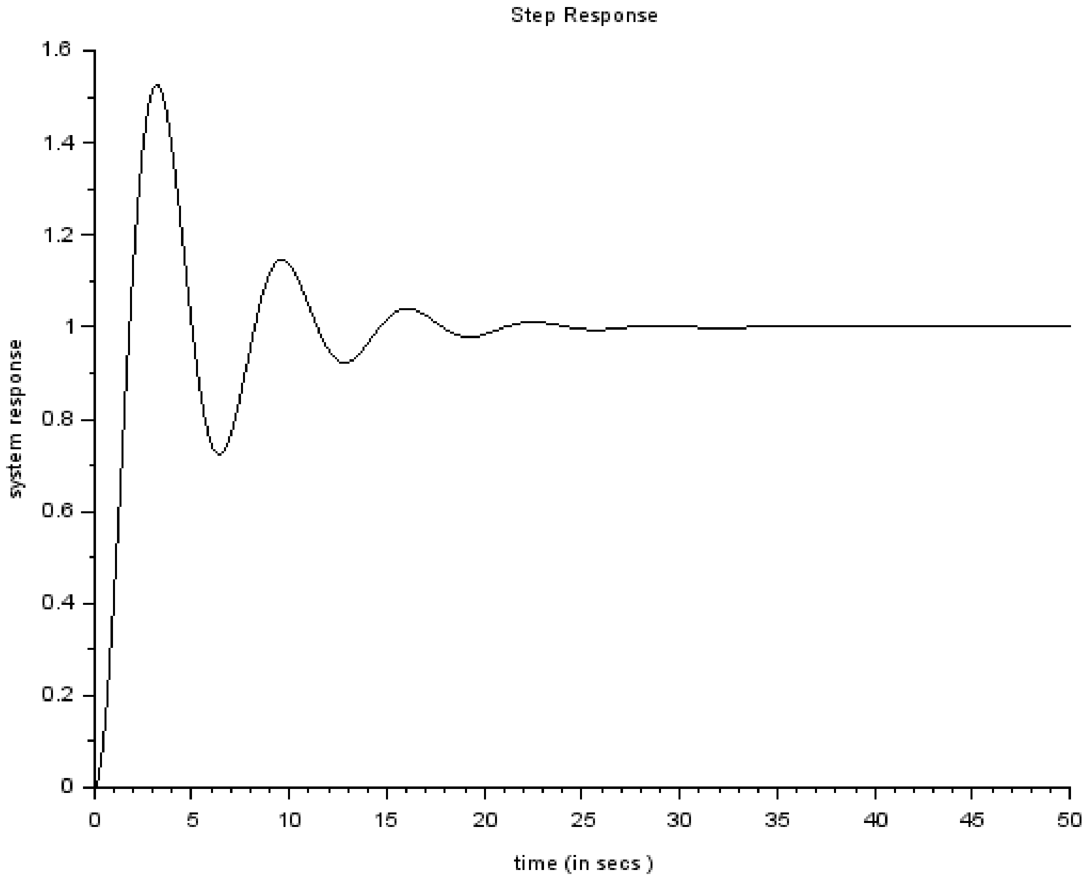


Figure 4: Step Response of Underdamped System with damping ratio=0.2

Part b

The step response of the systems with damping factor varying from 0.25 to 2 on the same plot are as follows:- The code for the above:-

```
dr_range = 0:.25:2.1;    // There are 9 values of dr to check (inclusive of 2)
//settle_time = zeros(1,9);
//rise_time = zeros(1,9);
//os = zeros(1,9);               // percentage overshoot
// peak_time = zeros(1,9);
```

```
outputs_matrix = zeros(size(t)(2),9);   //size returns a tuple rxc
for i = 1:9;
    zeta = dr_range(i);
    G = w^2 /( s ^2 + 2* zeta * w * s + w ^2) ;
    G = syslin('c',G);
    outputs_matrix(:,i) = csim('step',t,G);
end

plot2d(t,outputs_matrix);
legend(['0','.25','.5','.75','1','1.25','1.5','1.75','2']);
xlabel (" time (in secs )") ;
ylabel (" system response ") ;
title (" Step Responses for varying damping ratios");
show_window(2);
```

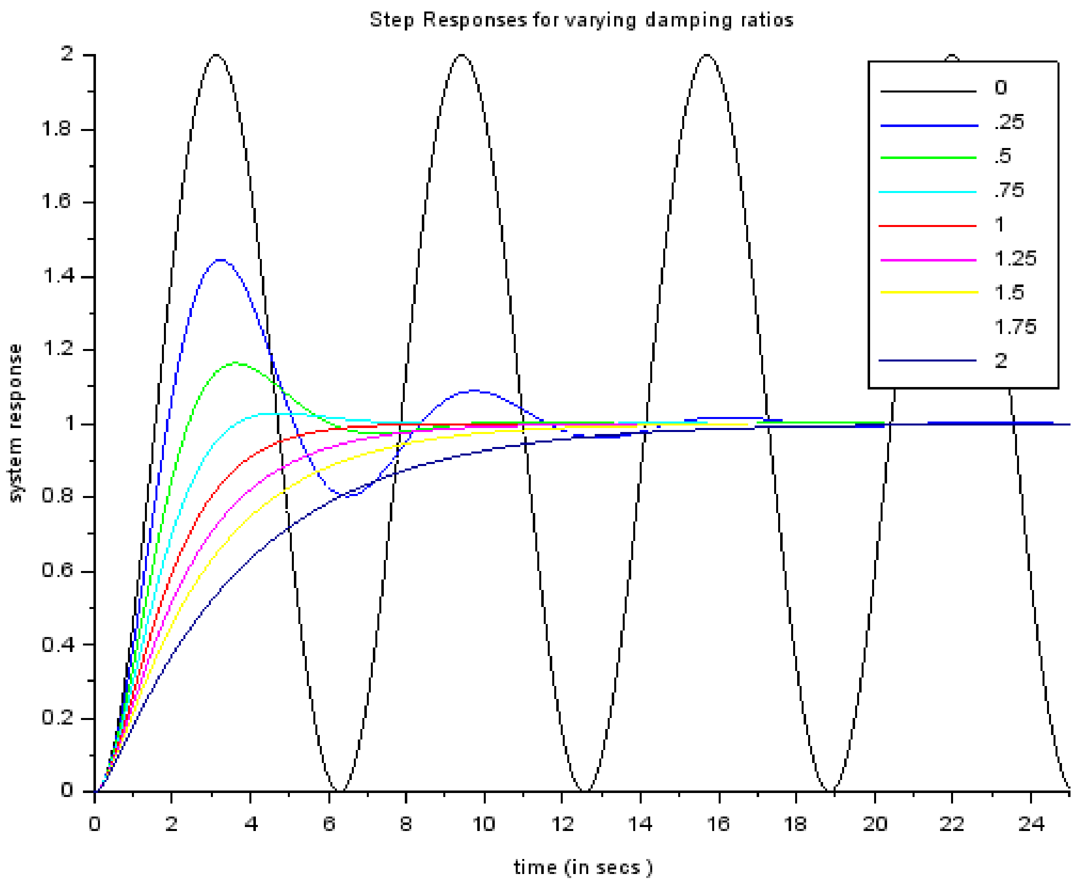


Figure 5: Varying Damping Ratio from 0 to 2

- Rise time and peak time remain approximately same for increasing damping ratio
- Overshoot percentage decreases with increasing zeta
- 2% settle time decreases with increasing zeta in underdamped regime

3 PROBLEM-3

The entire code snippet :-

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

G1 = 2/( s +2) ;
G1 = syslin ('c', G1 ) ;

G2 = 1/( s ^2 + 10* s + 1) ;
G2 = syslin ('c', G2 ) ;

t = 0:0.003:45;

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

plot2d (t , [y1',y2']) ;
legend([ 'first order'; 'second order'] ) ;
xlabel ('Time (in secs )') ;
ylabel ('Step Response') ;
title ('Step response') ;
show_window(1)

// Third case (repeated zeros, 2nd order)
G3 = 4/( s +2) ^2;
G3 = syslin ('c', G3 ) ;
y3 = csim('step', t , G3 ) ;

plot2d (t ,y3) ;
xlabel (" Time (in secs )") ;
ylabel (" System response ") ;
title (" Step Response for hird case (repeated zeros, 2nd order) Case
show_window(2)
```

The chosen first order and second order system are:-

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

$$G_2(s) = \frac{1}{s^2 + 10s + 1}$$

The code for this part is:-

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

```
G1 = 2/( s +2) ;
G1 = syslin ('c', G1 ) ;

G2 = 1/( s ^2 + 10* s + 1) ;
G2 = syslin ('c', G2 ) ;

t = 0:0.003:45;


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

plot2d (t , [y1',y2']) ;
legend([ 'first order'; 'second order'] ) ;
xlabel ('Time (in secs )') ;
ylabel ('Step Response') ;
title ('Step response') ;
show_window(1)
```

The step response obtained is:-

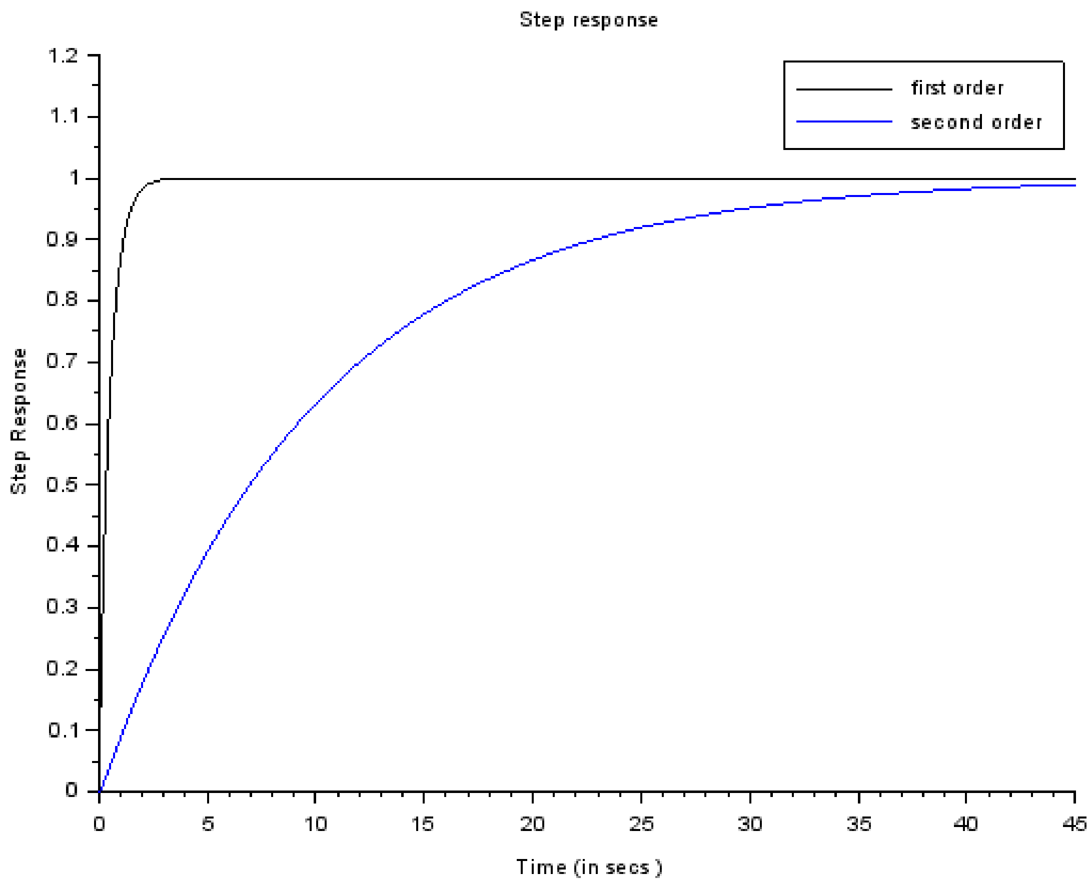


Figure 6: First Order vs Second Order step response

The differences between the two were:-

- The first order response had a smaller Rise Time
- The first order response had a smaller Settling Time
- The first order response approaches the equilibrium more quickly than the second order response.
- Derivative is 0 for 2nd order response at t =0 wherease first order has a non-zero derivative

In case of repeated roots, transfer function was:-

$$G_3(s) = \frac{4}{s^2 + 4s + 4}$$

The step response was indeed monotonic as can be seen below:-

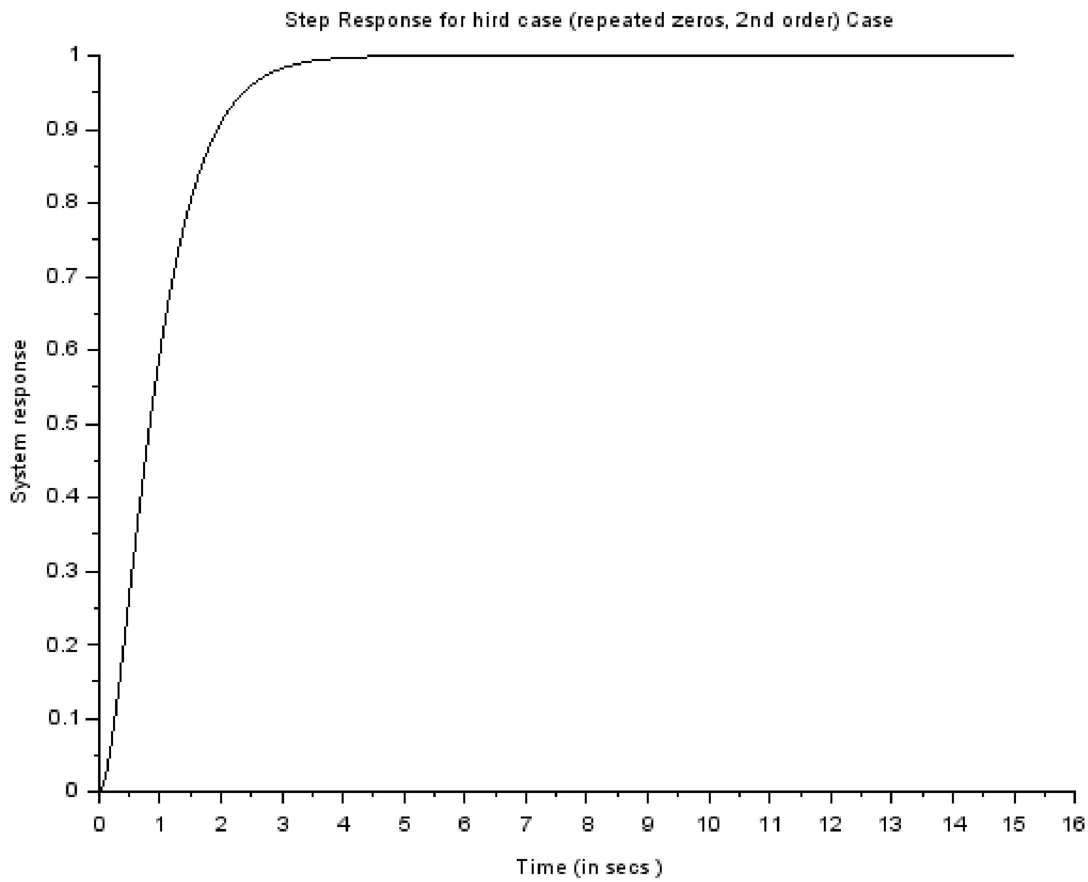


Figure 7: Repeated Pole Second Order Step Response

The code for this question was as follows:-

```
G3 = 4/( s +2) ^2;
G3 = syslin ('c', G3 ) ;
```

```
y3 = csim('step', t , G3 ) ;

plot2d (t ,y3) ;
xlabel (" Time (in secs )") ;
ylabel (" System response ") ;
title (" Step Response for hird case (repeated zeros, 2nd order) Case ") ;
show_window(2)
```

4 PROBLEM-4

The entire code snippet :-

```
s = poly(0,'s');
G = 1/s;
G = syslin('c',G);

t = 0:.01:5
plot2d(t, csim('step',t,G));
xlabel('time');
ylabel('system output');
title('Response to unit step function');
show_window(1);

z = poly(0,'z');
D = 1/z;  // naming it D since it is discrete
D = syslin('d',D);
D = tf2ss(D);

t_d = 0:1:15;
u = ones(t_d);
y_d = dsimul(D,u);
scatter(t_d,y_d);
a = gca();
a.data_bounds= [-.5,0;16,2];
xlabel('Time Steps [n]');
ylabel('y[n]');
title('Step response for discrete system D(z) = 1/z');
show_window(2);

p = poly(0,'p');
G1 = 1/p;
plot2d(t, csim('step',t,G1));
xlabel('time');
ylabel('system output');
title('Response to unit step function with polynomial given to csim');
show_window(2);
```

The code for a part is :-

```
s = poly(0,'s');
G = 1/s;
G = syslin('c',G);

t = 0:.01:5
plot2d(t, csim('step',t,G));
```

```
xlabel('time');
ylabel('system output');
title('Response to unit step function');
show_window(1);
```

The step response obtained is:-

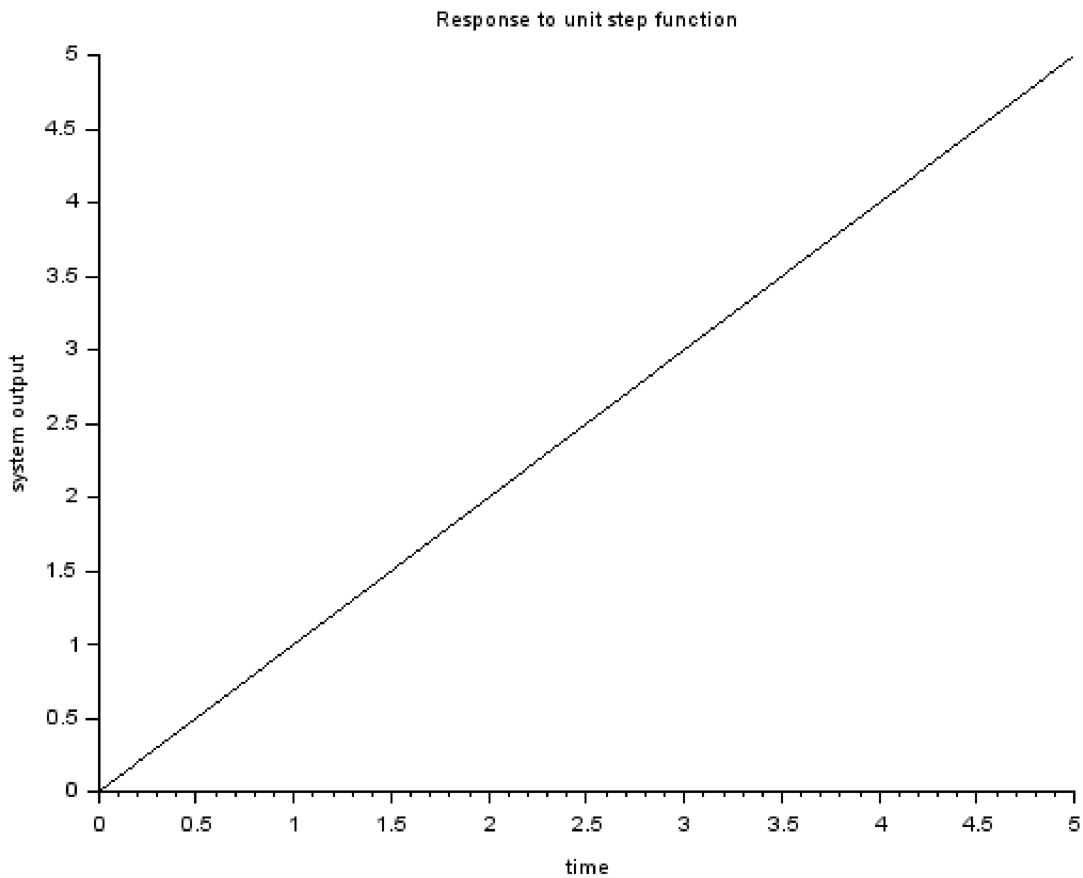


Figure 8: Continuous step response for 1/s

We see that while in continuous time, a transfer function of the form $\frac{1}{s}$ acts as an integrator

The code for b part is :-

```
z = poly(0,'z');
D = 1/z;  // naming it D since it is discrete
D = syslin('d',D);
D = tf2ss(D);

t_d = 0:1:15;
u = ones(t_d);
y_d = dsimul(D,u);
scatter(t_d,y_d);
a = gca();
a.data_bounds= [-.5,0;16,2];
xlabel('Time Steps [n]');
ylabel('y[n]');
title('Step response for discrete system D(z) = 1/z');
show_window(2);
```

The step response obtained is:-

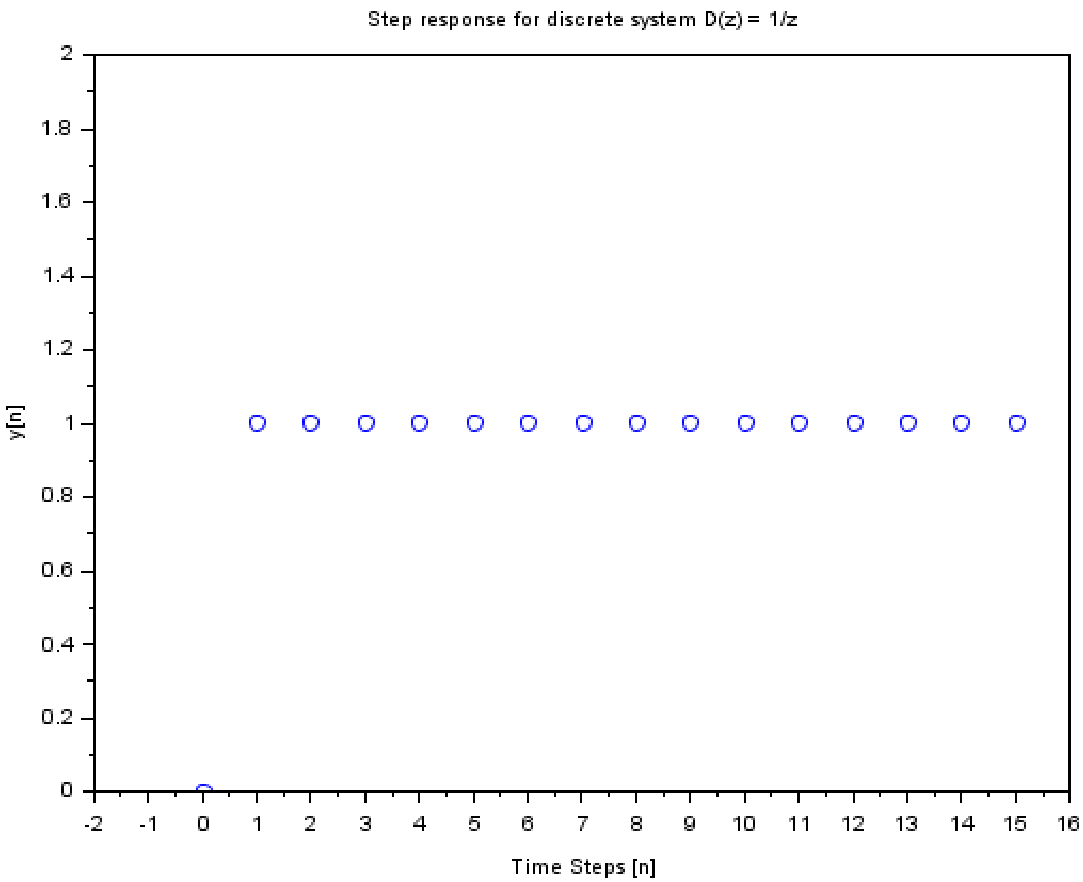


Figure 9: Discrete time step response for 1/z

in discrete time, the transfer function $\frac{1}{z}$ acts as a time delayer since in the Z-domain, a multiplication with $\frac{1}{z}$ indicates a time delay of 1 unit

The code for c part is :-

```
p = poly(0,'p');
G1 = 1/p;
plot2d(t, csim('step',t,G1));
xlabel('time');
ylabel('system output');
```

```
title('Response to unit step function with polynomial given to csim');
show_window(2);
```

The step response obtained is:-

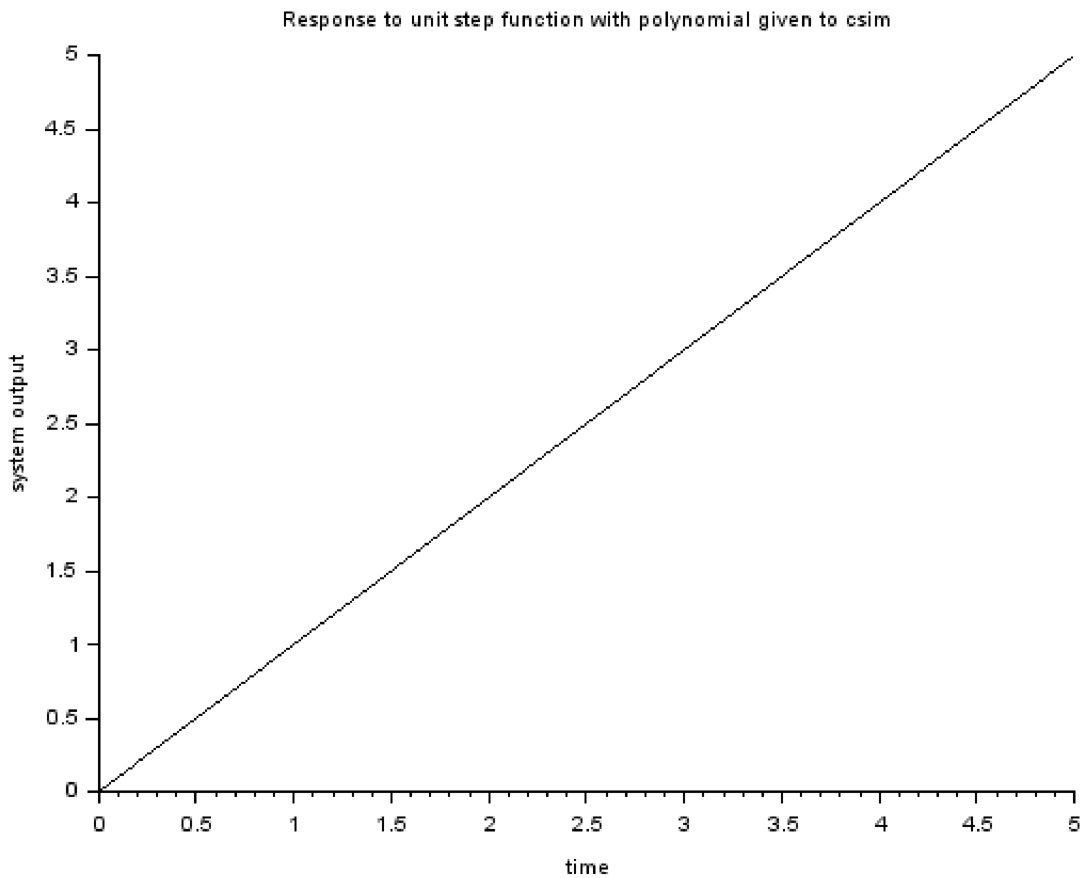


Figure 10: step response for polynomial 1/p

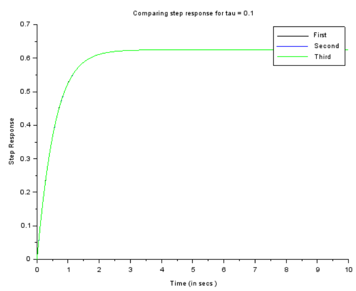
I personally didn't get any error after passing a polynomial to csim function however got a warning — WARNING: csim: Input argument #1 is assumed continuous time.

5 PROBLEM-5

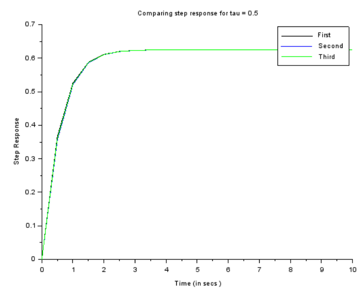
The entire code snippet :-

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

G1 = (s+5)/((s+4)*(s+2));
G1 = syslin('c',G1);

G2 = (s+5)/(s+4);
G2 = syslin('c',G2);

G3 = 1/(s+2);
G3 = syslin('c',G3);

tau = [0.1 , 0.5 , 2];

for i = 1:3
    t = 0:tau(i):10;
    y1 = csim('step', t, G1);
    y2 = csim(csim('step', t, G2),t,G3);
    y3 = csim(csim('step', t, G3),t,G2);   // to create cascade you feed pr
p into i/p of csim command
    plot2d(t, [y1',y2',y3'])
    legend(['First';'Second';'Third']);
    xlabel (" Time (in secs )") ;
    ylabel (" Step Response ") ;
    title (" Comparing step response for tau = " + string (tau(i))) ;
    show_window(i);
end
```

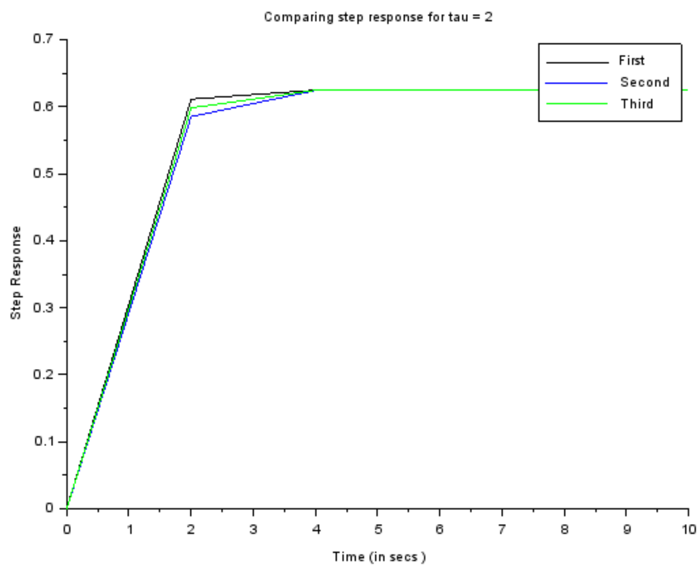
On plotting, we observe the following 3 plots:-



(a) τ=0.1s



(b) τ=0.2s



(c) τ=2s

Figure 11: The Three output graphs with varying sampling period(τ)

As one can clearly see, no observable error pops up when $\tau = 0.1s$ and when $\tau = 0.2s$. However, an error does appear when we change to $\tau = 2s$.