

# EE324 Lab-1 Report

Yash Sanjeev  
180070068

January 2021

## Contents

<b>1</b>	<b>Problem 1</b>	<b>2</b>
1.1	Code . . . . .	2
1.2	Part 1(a) . . . . .	3
1.3	Part 1(b) . . . . .	3
1.4	Part 1(c) . . . . .	4
1.5	Part 1(d) . . . . .	5
<b>2</b>	<b>Problem 2</b>	<b>7</b>
2.1	Code . . . . .	7
2.2	Part 2(a) . . . . .	7
2.3	Part 2(b) . . . . .	9
<b>3</b>	<b>Problem 3</b>	<b>11</b>
3.1	Code . . . . .	11
3.2	Part 3(a) . . . . .	11
3.3	Part 3(b) . . . . .	13
<b>4</b>	<b>Problem 4</b>	<b>15</b>
4.1	Code . . . . .	15
4.2	Part 4(a) . . . . .	16
4.3	Part 4(b) . . . . .	17
4.4	Part 4(c) . . . . .	18
<b>5</b>	<b>Problem 5</b>	<b>20</b>
5.1	Code . . . . .	20
5.2	Plots . . . . .	21

# 1 Problem 1

## 1.1 Code

```
1 a = 68; // Roll Number = 180070068
2 b = 25; // Name = Yash (Y = 25)
3
4 // build the LTI System
5 s = poly(0, 's');
6 G = a/(s+b);
7 sys = syslin('c', G);
8
9 // plot the unit step response
10 time = 0:0.00001:1;
11 out = csim("step", time, sys);
12
13 tau = 1/b; // time constant
14 settle = log(1/0.02)/b; // 2% settling time
15 start = log(1/(1-0.10))/b; // rising starts at 10%
16 fin = log(1/(1-0.90))/b; // rising finishes at 90%
17 rise = fin - start; // overall rise time
18
19 disp("Time constant =", tau);
20 disp("2% Settling time =", settle);
21 disp("Rise time", rise);
22
23 scf(0);
24 plot(time, out);
25 xlabel("time (in secs)");
26 ylabel("system response");
27 title("Step Response");
28
29 // variation in a
30 a_range = a:a:100*a;
31 a_rise = linspace(rise, rise, 100);
32
33 scf(1);
34 plot(a_range, a_rise);
35 xlabel("a");
36 ylabel("Rise Time (in secs)");
37 title("Variation of Rise Time with a");
38
39 // variation in b
40 b_range = b:b:100*b;
41 b_start = log(1/(1-0.10))./b_range;
42 b_fin = log(1/(1-0.90))./b_range;
```

```

43 b_rise = b_fin - b_start;
44
45 scf(2);
46 plot(b_range, b_rise);
47 xlabel("b");
48 ylabel("Rise Time (in secs)");
49 title("Variation of Rise Time with b");

```

## 1.2 Part 1(a)

```

1 a = 68; // Roll Number = 180070068
2 b = 25; // Name = Yash (Y = 25)
3
4 // build the LTI System
5 s = poly(0, 's');
6 G = a / (s + b);
7 sys = syslin('c', G);

```

- Roll Number = 180070068, hence  $a = 68$ .
- First Name = Yash, hence  $b = 25$ .
- According to these values, a continuous time LTI has been built.

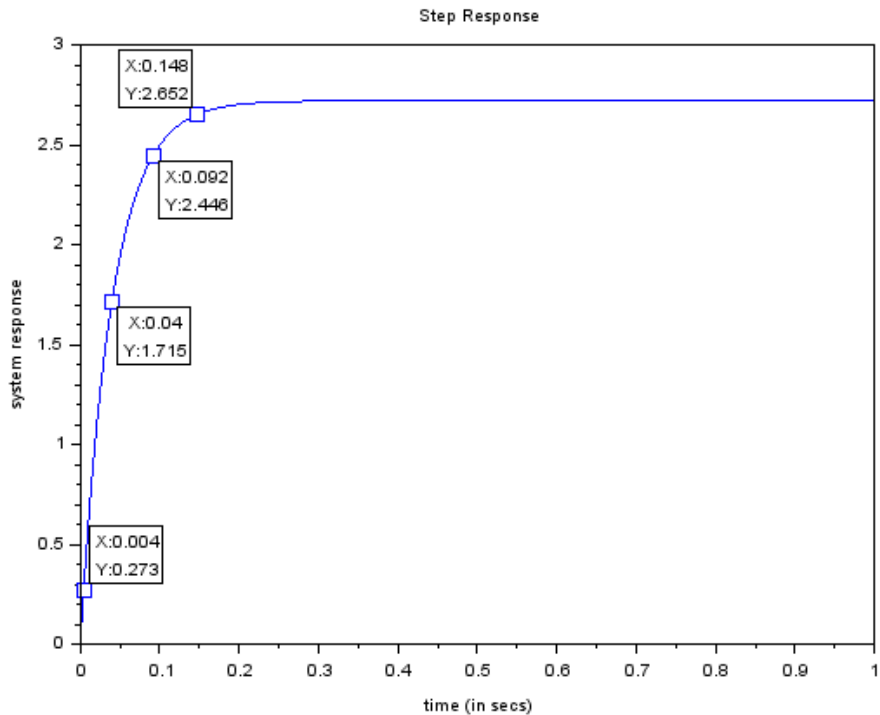
## 1.3 Part 1(b)

```

1 // plot the unit step response
2 time = 0:0.00001:1;
3 out = csim("step", time, sys);
4
5 tau = 1/b; // time constant
6 settle = log(1/0.02)/b; // 2% settling time
7 start = log(1/(1-0.10))/b; // rising starts at 10%
8 fin = log(1/(1-0.90))/b; // rising finishes at 90%
9 rise = fin - start; // overall rise time
10
11 disp("Time constant =", tau);
12 disp("2% Settling time =", settle);
13 disp("Rise time", rise);
14
15 scf(0);
16 plot(time, out);
17 xlabel("time (in secs)");
18 ylabel("system response");
19 title("Step Response");

```

- Rising time taken between 10% and 90% of steady-state value.
- Response =  $\frac{a}{b}(1 - e^{-bt})$ , hence steady state value is 2.72.
- Rise starts when  $y = 0.272$ , and stops at  $y = 2.448$ , for 2% settling time,  $y = 0.98 \times 2.72 = 2.665$  and for  $\tau$ , we need  $x = \frac{1}{25} = 0.04$ .



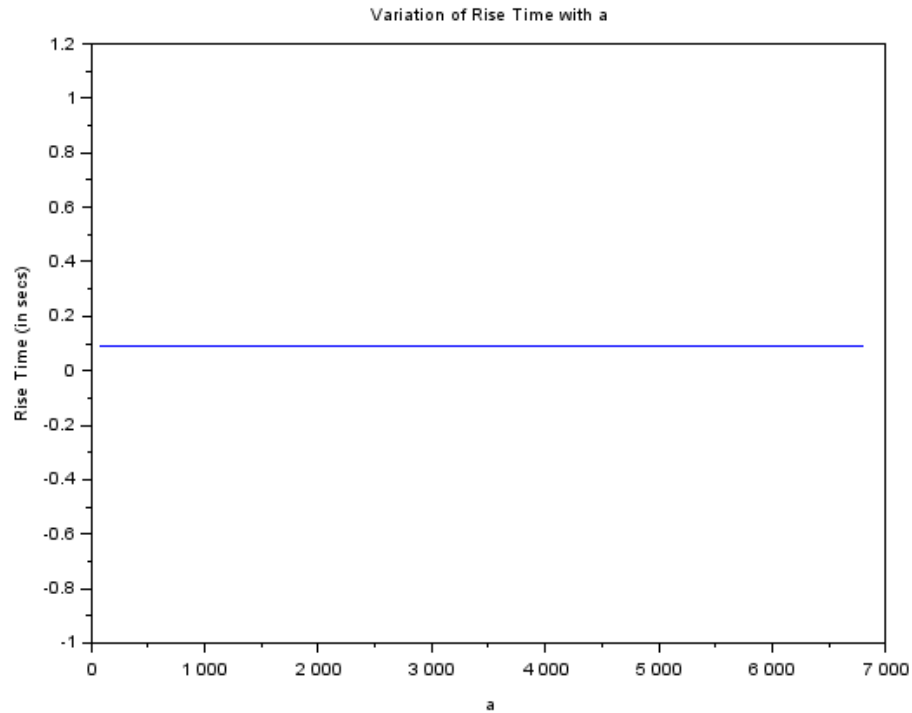
#### 1.4 Part 1(c)

```

1 // variation in a
2 a_range = a:a:100*a;
3 a_rise = linspace(rise, rise, 100);
4
5 scf(1);
6 plot(a_range, a_rise);
7 xlabel("a");
8 ylabel("Rise Time (in secs)");
9 title("Variation of Rise Time with a");

```

- Since values only depend on  $b$ , variation in  $a$  produces no effect in rise time.



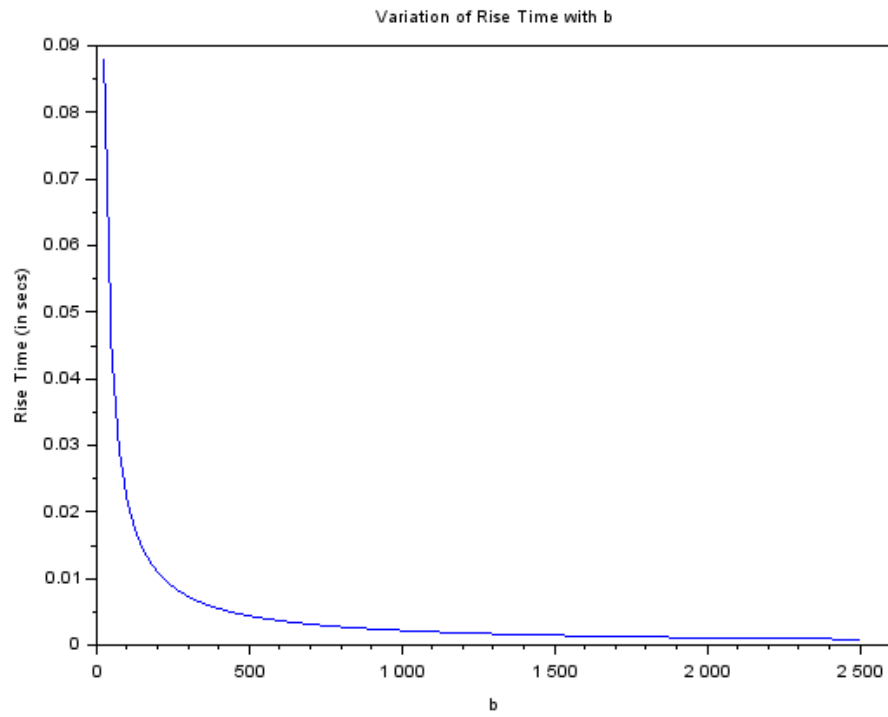
### 1.5 Part 1(d)

```

1 // variation in b
2 b_range = b:b:100*b;
3 b_start = log(1/(1-0.10))./b_range;
4 b_fin = log(1/(1-0.90))./b_range;
5 b_rise = b_fin - b_start;
6
7 scf(2);
8 plot(b_range, b_rise);
9 xlabel("b");
10 ylabel("Rise Time (in secs)");
11 title("Variation of Rise Time with b");

```

- Rise time  $\propto \frac{1}{b}$ , the graph drawn is a rectangular hyperbola.



## 2 Problem 2

### 2.1 Code

```
1 s = poly(0, 's');
2 dr = 1/3; // damping ratio
3 f = 3; // frequency of oscillation
4
5 G = f/(s^2 + 2*dr*f*s + f^2);
6 sys = syslin('c', G);
7
8 // plotting the step response
9 t = 0:0.002:10;
10 out = csim("step", t, sys);
11
12 scf(0);
13 plot(t, out);
14 xlabel("time (in secs)");
15 ylabel("system response");
16 title("Step Response");
17
18 // variation with damping ratio
19 dr_range = 0:0.25:2;
20 out_range = zeros(5001, 9);
21
22 for i = 1:size(dr_range)(2)
23     dr = dr_range(i);
24     G = f/(s^2 + 2*dr*f*s + f^2);
25     sys = syslin('c', G);
26     out_range(:, i) = csim("step", t, sys);
27 end
28
29 scf(1);
30 plot2d(t, out_range, 1:9);
31 legends(['0'; '0.25'; '0.5'; '0.75'; '1'; '1.25'; '1.5';
32         '1.75'; '2'], 1:9, opt = "?");
33 xlabel('Time (in secs)');
34 ylabel('Step Response');
35 title('Variation of step response with damping ratio')
36 ;
```

### 2.2 Part 2(a)

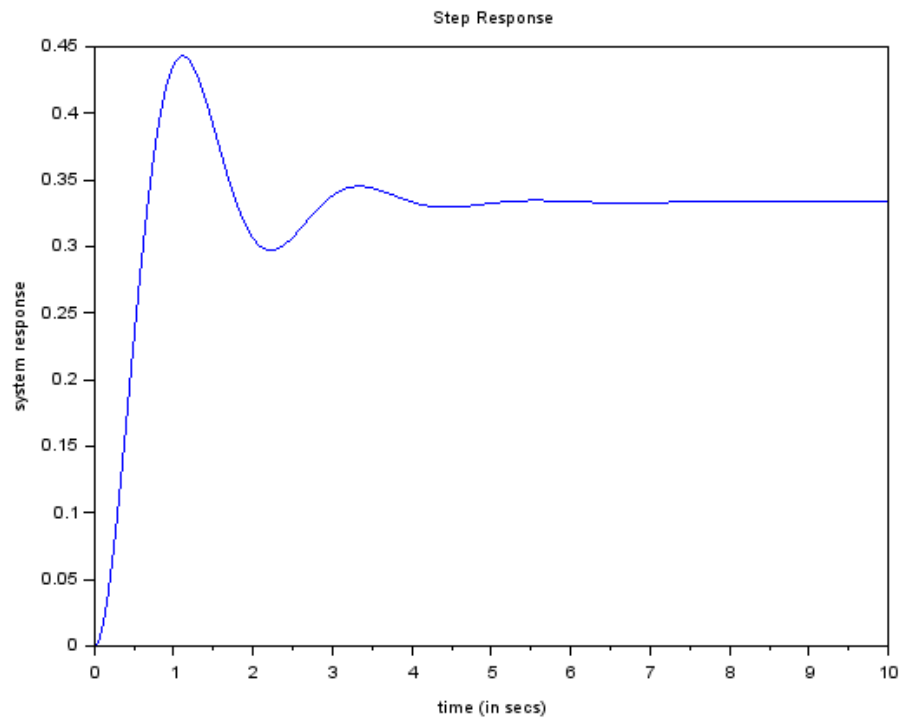
```
1 s = poly(0, 's');
```

```

2 dr = 1/3; // damping ratio
3 f = 3; // frequency of oscillation
4
5 G = f/(s^2 + 2*dr*f*s + f^2);
6 sys = syslin('c', G);
7
8 // plotting the step response
9 t = 0:0.002:10;
10 out = csim("step", t, sys);
11
12 scf(0);
13 plot(t, out);
14 xlabel("time (in secs)");
15 ylabel("system response");
16 title("Step Response");

```

- Since system is underdamped, the graph oscillates about the equilibrium, with frequency as in the code.
- Damping ratio used is  $\frac{1}{3}$  and frequency is 3 Hz.

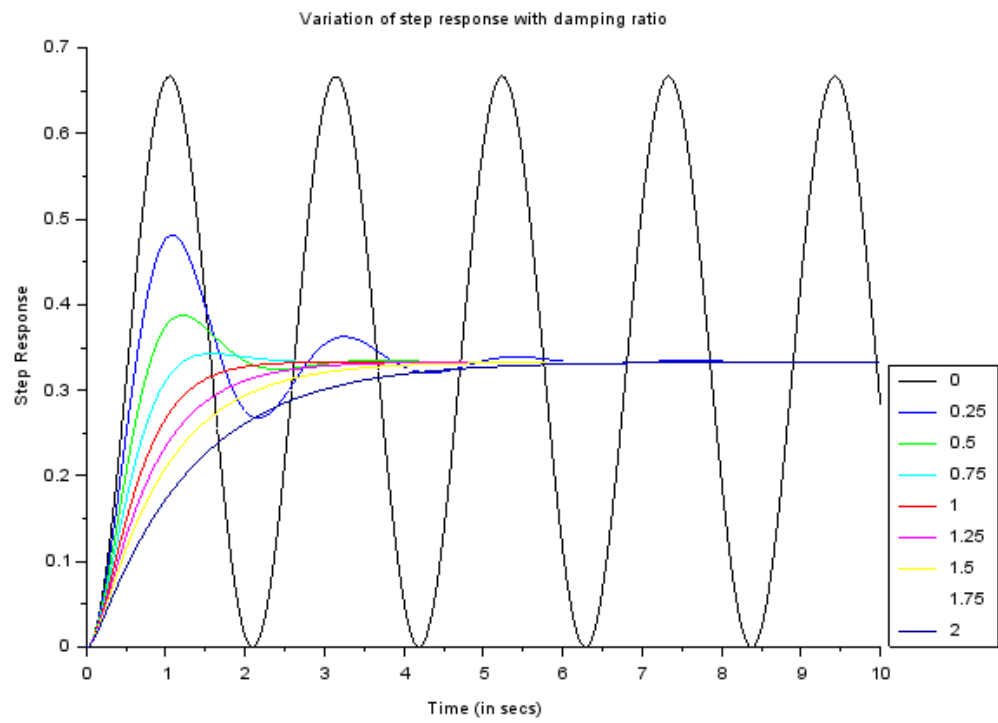




## 2.3 Part 2(b)

```
1 // variation with damping ratio
2 dr_range = 0:0.25:2;
3 out_range = zeros(5001, 9);
4
5 for i = 1:size(dr_range)(2)
6     dr = dr_range(i);
7     G = f/(s^2 + 2*dr*f*s + f^2);
8     sys = syslin('c', G);
9     out_range(:, i) = csim("step", t, sys);
10 end
11
12 scf(1);
13 plot2d(t, out_range, 1:9);
14 legends(['0'; '0.25'; '0.5'; '0.75'; '1'; '1.25'; '1.5';
15         '1.75'; '2'], 1:9, opt = "?");
16 xlabel('Time (in secs)');
17 ylabel('Step Response');
18 title('Variation of step response with damping ratio')
19 ;
```

- With increasing damping factor, percentage overshoot decreases. Rise time and peak time remain approximately same.
- The 2% settling time reaches a minimum at critically damped state, and increases when damping factor is less than or greater than 1.



## 3 Problem 3

### 3.1 Code

```
1 s = poly(0, 's');
2
3 G1 = 1/(s+1);
4 sys1 = syslin('c', G1);
5
6 G2 = 2/(s^2 + 3*s + 2);
7 sys2 = syslin('c', G2);
8
9 t = 0:0.0003:15;
10 outs = zeros(50001, 2);
11
12 outs(:, 1) = csim('step', t, sys1);
13 outs(:, 2) = csim('step', t, sys2);
14
15 // plotting the step responses
16 scf(0);
17 plot2d(t, outs, 1:2);
18 legends(['first order'; 'second order'], 1:2, opt = "?
19 ");
20 xlabel('Time (in secs)');
21 ylabel('Step Response');
22 title('Step response for first and second order
23 systems');
24
25 // critically damped case
26 G3 = 1/(s+1)^2;
27 sys3 = syslin('c', G3);
28 out = csim('step', t, sys3);
29
30 scf(1);
31 plot(t, out);
32 xlabel("Time (in secs)");
33 ylabel("System response");
34 title("Step Response for Critically Damped Case");
```

### 3.2 Part 3(a)

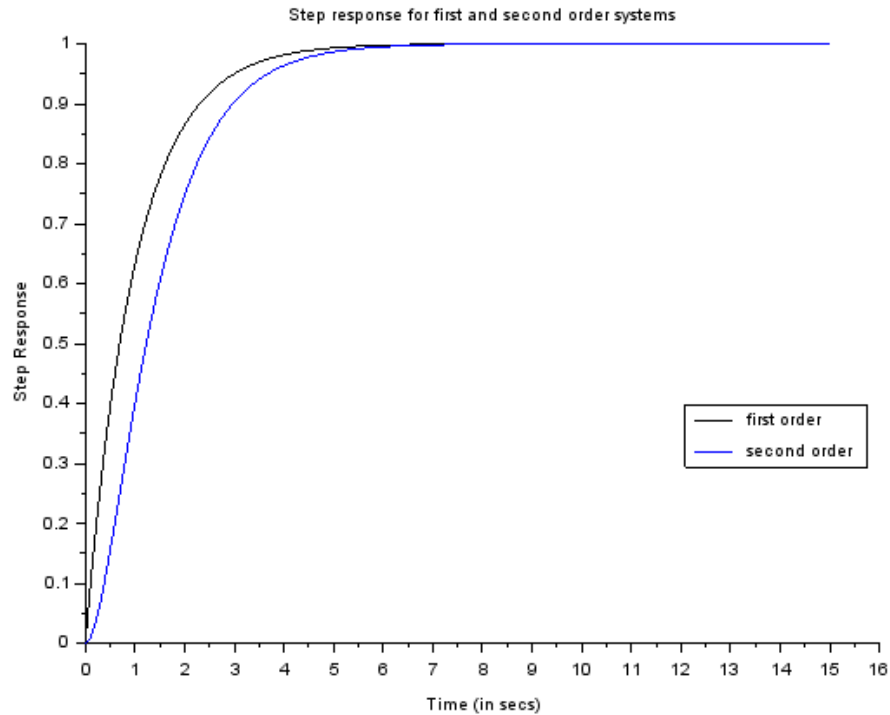
```
1 s = poly(0, 's');
2
3 G1 = 1/(s+1);
```

```

4 sys1 = syslin('c', G1);
5
6 G2 = 2/(s^2 + 3*s + 2);
7 sys2 = syslin('c', G2);
8
9 t = 0:0.0003:15;
10 outs = zeros(50001, 2);
11
12 outs(:, 1) = csim('step', t, sys1);
13 outs(:, 2) = csim('step', t, sys2);
14
15 // plotting the step responses
16 scf(0);
17 plot2d(t, outs, 1:2);
18 legends(['first order'; 'second order'], 1:2, opt = "?
19 ");
20 xlabel('Time (in secs)');
21 ylabel('Step Response');
22 title('Step response for first and second order
23 systems');

```

- The first order response has a nonzero derivative at 0, but the second order response has derivative 0.
- The first order response approaches the equilibrium more quickly than the second order response.
- The first order response is concave throughout but the second order response transitions from convex to concave.



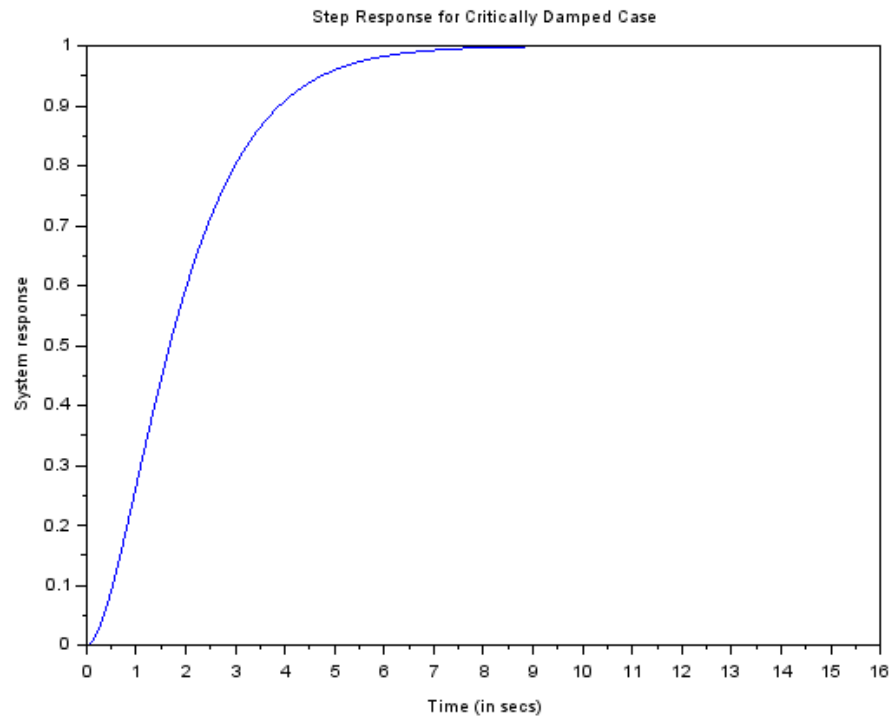
### 3.3 Part 3(b)

```

1 // critically damped case
2 G3 = 1/(s+1)^2;
3 sys3 = syslin('c', G3);
4 out = csim('step', t, sys3);
5
6 scf(1);
7 plot(t, out);
8 xlabel("Time (in secs)");
9 ylabel("System response");
10 title("Step Response for Critically Damped Case");

```

- For the critically damped case (repeated poles), the response is indeed monotonic.



## 4 Problem 4

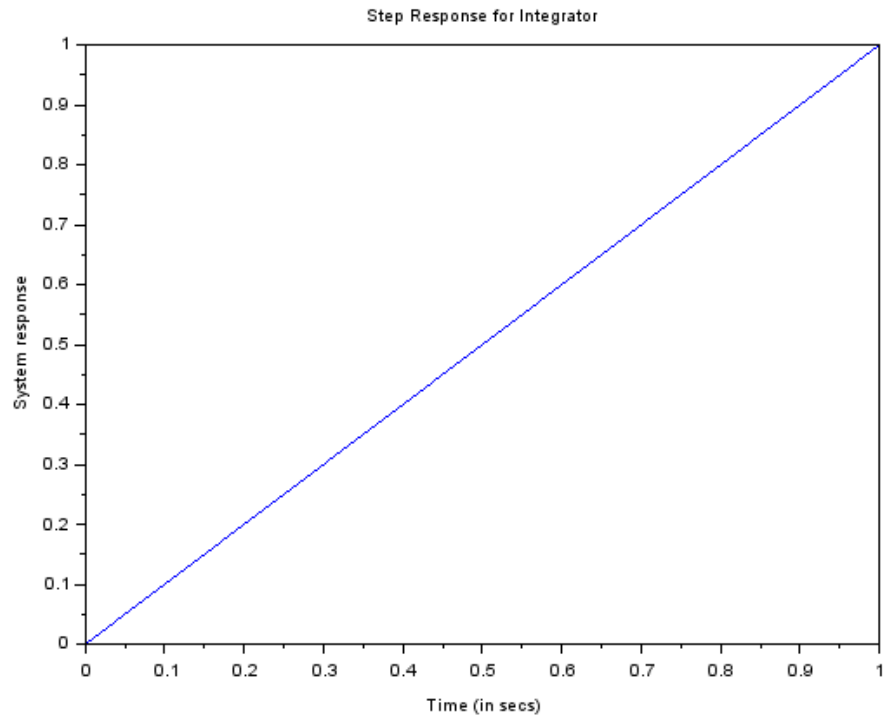
### 4.1 Code

```
1 // step response of integrator
2 s = poly(0, 's');
3 G1 = 1/s;
4 sys1 = syslin('c', G1);
5
6 t = 0:0.0001:1;
7 out1 = csim('step', t, sys1);
8
9 scf(0);
10 plot(t, out1);
11 xlabel("Time (in secs)");
12 ylabel("System response");
13 title("Step Response for Integrator");
14
15 // discrete system step response
16 z = poly(0, 'z');
17 G2 = 1/z;
18 sys2 = syslin('d', G2);
19 state = tf2ss(sys2);
20
21 n = 0:10;
22 step = ones(1, 11);
23 step(1, 1) = 0;
24 out2 = dsimul(state, step);
25
26 scf(1);
27 plot2d(n, out2, -1, rect = [-1, -0.2, 12, 1.2]);
28 xlabel("Time (in secs)");
29 ylabel("Discrete system response");
30 title("Step Response for Delay System");
31
32 // plotting a polynomial directly
33 p = 1/s;
34 out3 = csim('step', t, p);
35
36 scf(2);
37 plot(t, out3);
38 xlabel("Time (in secs)");
39 ylabel("Polynomial response");
40 title("Step Response for polynomial 1/s");
```

## 4.2 Part 4(a)

```
1 // step response of integrator
2 s = poly(0, 's');
3 G1 = 1/s;
4 sys1 = syslin('c', G1);
5
6 t = 0:0.0001:1;
7 out1 = csim('step', t, sys1);
8
9 scf(0);
10 plot(t, out1);
11 xlabel("Time (in secs)");
12 ylabel("System response");
13 title("Step Response for Integrator");
```

- The step response of the integrator is just a clipped linear response.

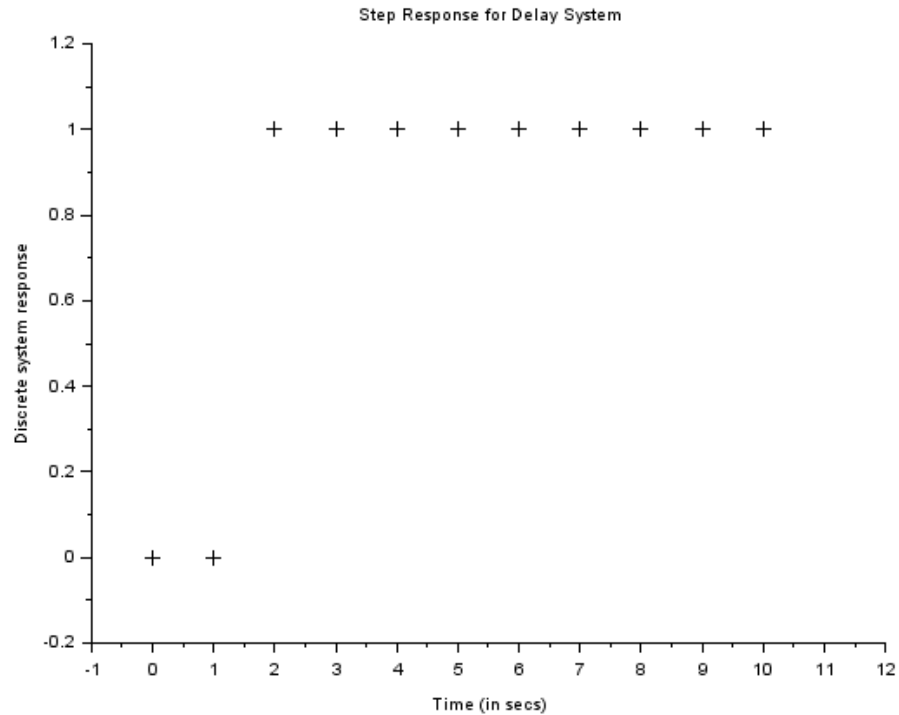




### 4.3 Part 4(b)

```
1 // discrete system step response
2 z = poly(0, 'z');
3 G2 = 1/z;
4 sys2 = syslin('d', G2);
5 state = tf2ss(sys2);
6
7 n = 0:10;
8 step = ones(1, 11);
9 step(1, 1) = 0;
10 out2 = dsimul(state, step);
11
12 scf(1);
13 plot2d(n, out2, -1, rect = [-1, -0.2, 12, 1.2]);
14 xlabel("Time (in secs)");
15 ylabel("Discrete system response");
16 title("Step Response for Delay System");
```

- The discrete system with transfer function  $\frac{1}{z}$  acts as a delay of one unit, and hence the response is a delayed step.



#### 4.4 Part 4(c)

```

1 // plotting a polynomial directly
2 p = 1/s;
3 out3 = csim('step', t, p);
4
5 scf(2);
6 plot(t, out3);
7 xlabel("Time (in secs)");
8 ylabel("Polynomial response");
9 title("Step Response for polynomial 1/s");

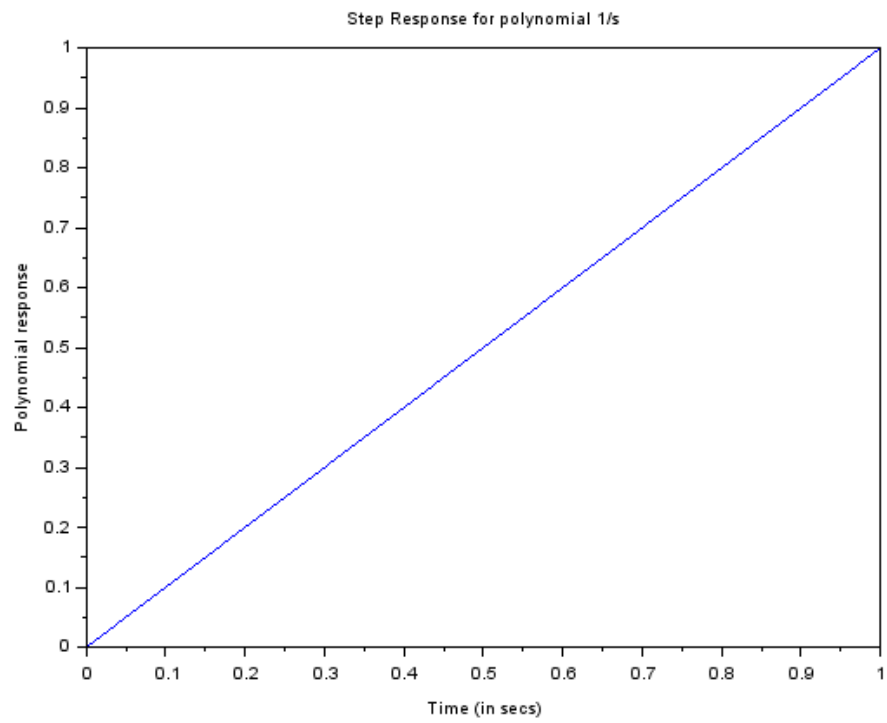
```

- The response is similar to the continuous case, but we end up with a warning in the console.

```

1 WARNING: csim: Input argument #1 is assumed continuous
   time.

```



## 5 Problem 5

### 5.1 Code

```
1 s = poly(0, 's');
2
3 G1 = (s+5)/((s+4)*(s+2));
4 sys1 = syslin('c', G1);
5
6 G2 = (s+5)/(s+4);
7 sys2 = syslin('c', G2);
8
9 G3 = 1/(s+2);
10 sys3 = syslin('c', G3);
11
12 taus = [0.1, 0.5, 2];
13
14 for i = 1:3
15     tau = taus(i);
16     t = 0:tau:10;
17
18     out1 = csim('step', t, sys1);
19
20     y1 = csim('step', t, sys2);
21     out2 = csim(y1, t, sys3);
22
23     y2 = csim('step', t, sys3);
24     out3 = csim(y2, t, sys2);
25
26     scf(i);
27     plot2d(t, [out1', out2', out3'], 1:3);
28     legends(["original", "series 1", "series 2"], 1:3,
29             opt="?");
30     xlabel("Time (in secs)");
31     ylabel("Step Response");
32     title("Comparing step response for tau = " +
33           string(tau));
34 end
```

## 5.2 Plots

