EE324 - Control Systems Lab

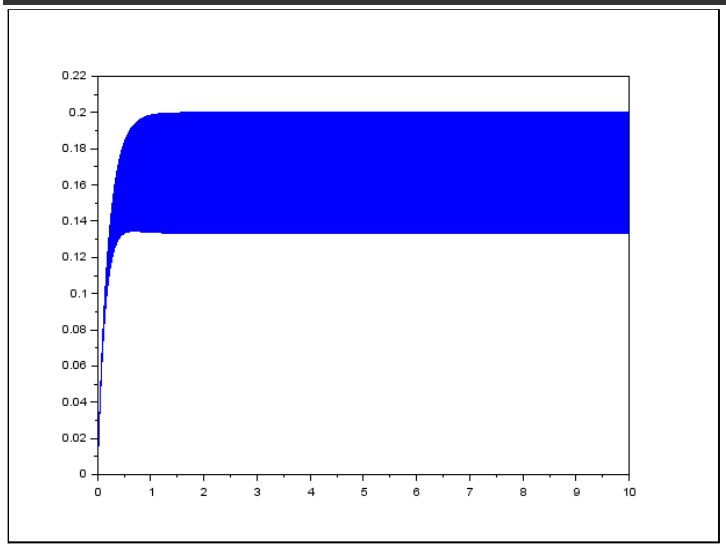
# PROBLEM SHEET 3

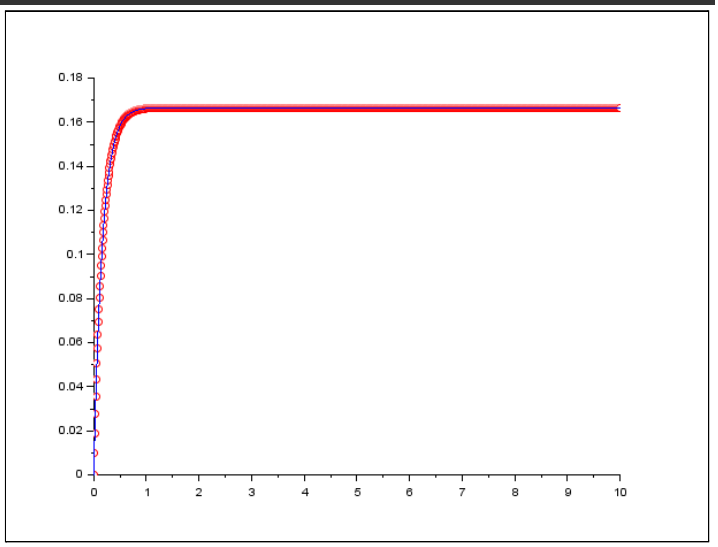
Prasann Viswanathan | 190070047

## Question 1:

**a.** Here is the code:

```
--> s = poly(0, 's'); A = -1:0.01:1; t = 0:0.01:10;
--> for i=1:size(A, 2)
  > G = (s+5+A(i))/(s^2+11*s+30);
  > sys = syslin('c', G);
  > y = csim('step', t, sys);
  > plot(t, y);
  > end
--> xs2png(gcf(), "1a_1.jpg");

--> [N, D] = simp(s+5, s^2+11*s+30);
--> G1 = N/D; sysG1 = syslin('c', G1);
--> G2 = (s+5)/(s^2+11*s+30); sysG2 = syslin('c', G2);
--> y1 = csim('step', t, sysG1); y2 = csim('step', t, sysG2);
--> scf()
--> plot(t, y1, 'r-o');
--> plot(t, y2, 'b');
--> xs2png(gcf(), "1a_2.jpg");
```



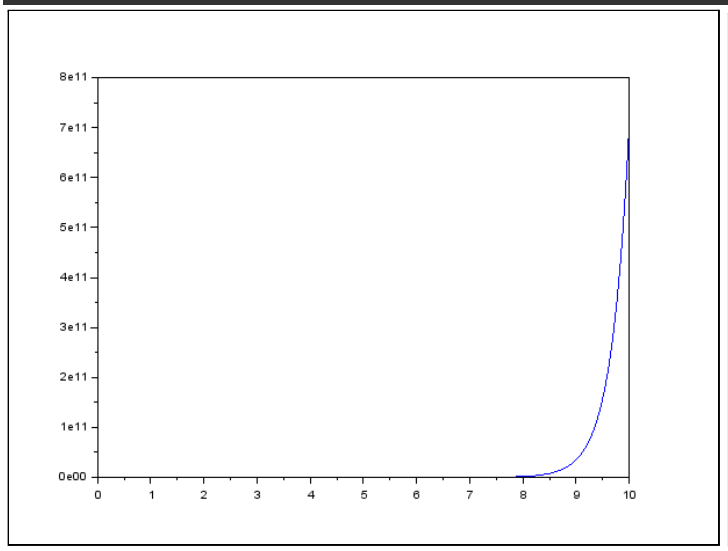



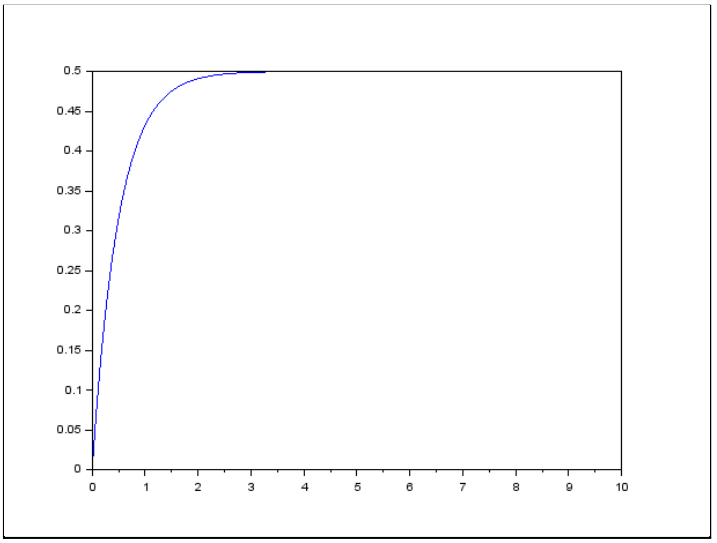

We can see in the second plot that for a=0, the step response remains unchanged even after the pole cancellation. (Both responses are plotted together on second plot)

**b.** SciLab code:

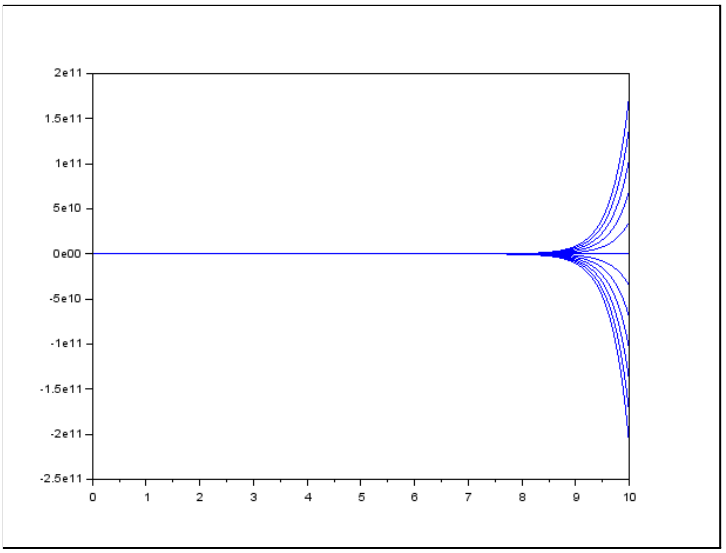
```
--> s = poly(0, 's');
--> t = 0:0.01:10;
--> G = 1/(s^2-s-6); sys = syslin('c', G);
--> y = csim('step', t, sys);
--> plot(t, y);
--> xs2png(gcf(), "1b_1.jpg");
```

```
--> G1 = (s-3)/(s^2-s-6); sys1 = syslin('c', G1);
--> y1 = csim('step', t, sys1);
--> scf();
--> plot(t, y1);
--> xs2png(gcf(), "1b_2.jpg");

--> A = -0.3:0.05:0.3;
--> for i=1:size(A, 2)
  > G2 = (s-3+A(i))/(s^2-s-6);
  > sys2 = syslin('c', G2);
  > y2 = csim('step', t, sys2);
  > plot(t, y2);
  > end
--> xs2png(gcf(), "1b_3.jpg");
```




Without addition of zero, the system is unstable (left) but after addition, the system on the right is stable.



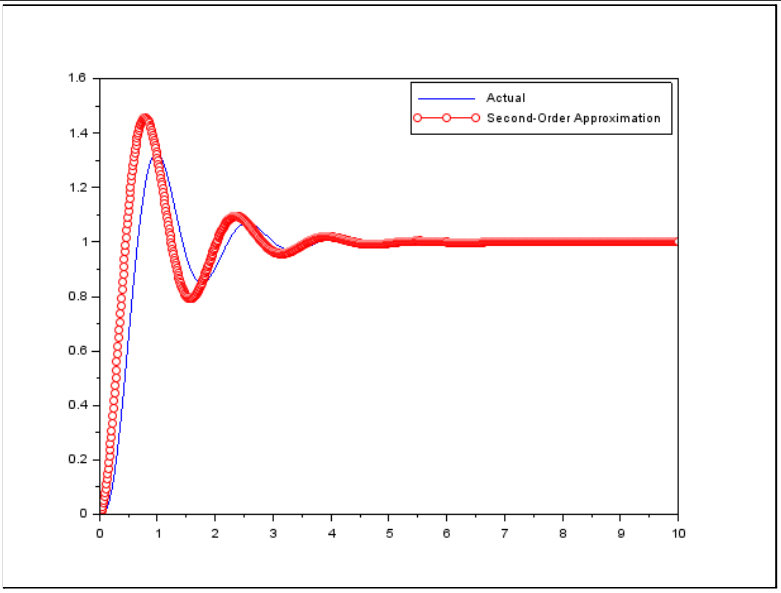

Shifting the zero slightly by using a small variable parameter A, we see that on shifting the zero even by the slightest amount, the system becomes unstable. Only the system with zero=3 (i.e. A=0) is stable.

Practically, if we add a zero to the system, it is inevitable that some errors will creep in, and the zero will not be exactly at 3, but very close to 3. Even still, the system would remain unstable. Hence, an unstable plant cannot be rendered stable by cancelling unstable poles by adding zeros attempting to cancel the pole.

# Question 2:

**a.** Here is the code:

```
--> s = poly(0, 's');
--> t = 0:0.01:10;
--> G = 85/(s^3+7*s^2+27*s+85); sys = syslin('c', G);
--> y = csim('step', t, sys);
--> plot(t, y);

--> [z, p, k] = tf2zp(sys);
--> G1 = 17/(s^2+2*s+17); sys1 = syslin('c', G1);
--> y1 = csim('step', t, sys1);
--> plot(t, y1, 'r-o');
--> legend(["Actual", "Second-Order Approximation"]);
--> xs2png(gcf(), "2a.jpg");
```



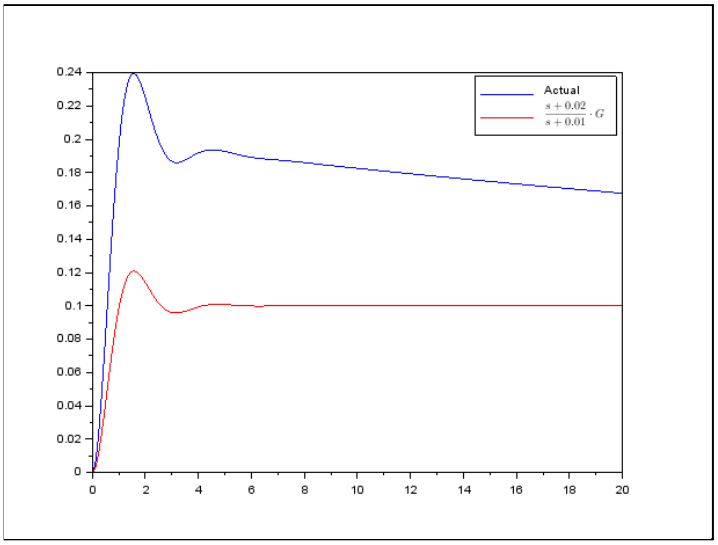

We see that the poles of the original system are: -5, -1+4j & -1-4j. Out of these, the two complex poles are dominant (closer to the imaginary axis), and hence we can ignore the third (-5) pole.

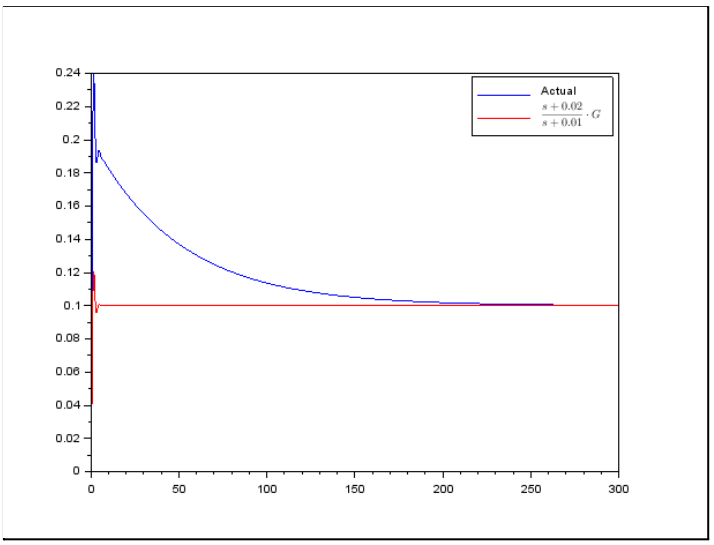
**b.** Scilab code for this part is as follows:

```
--> s = poly(0, 's');
--> t = 0:0.01:20;
--> G = (s+0.01)/(s^3+(101/50)*s^2+(126/25)*s+0.1);
--> sys = syslin('c', G);
--> y = csim('step', t, sys);
--> plot(t, y);

--> [z, p, k] = tf2zp(sys)
 z  =
  -0.01
 p  =
  -1.   + 2.i
  -1.   - 2.i
  -0.02 + 0.i

--> G1 = (s+0.02)*G/(s+0.01); sys1 = syslin('c', G1);
--> y1 = csim('step', t, sys1);
```

```
--> plot(t, y1, 'r');
--> legend(["Actual", "$\dfrac{s+0.02}{s+0.01} \cdot G$"]);
--> xs2png(gcf(), "2b.jpg");
```

We see that the poles of the system are = {-0.02, -1+2j, -1-2j} and the zero is = {-0.01}.
Since of the poles, the pole at (-0.02) is closest to the imaginary axis, it is the dominant pole, and removing two complex poles wouldn't make much difference. But we need to also cancel a zero, such that the resulting system would be a second order approximation. Which is why we cancel the pole at (-0.02) and zero at (-0.01). However, the results show that our approximations are not good:



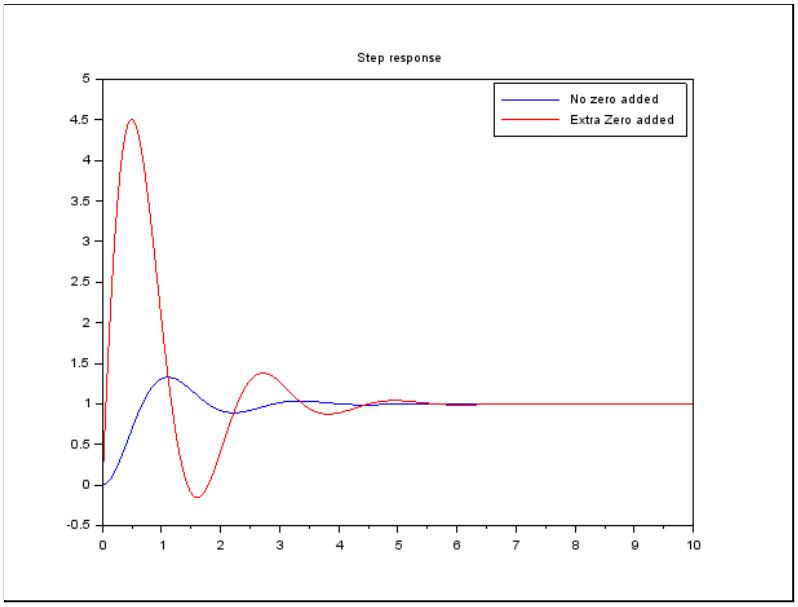




# Question 3:

**a.** Scilab code is as follows:

```
--> s = poly(0, 's'); t = 0:0.01:10;
--> G1 = syslin('c', 9/(s^2+2*s+9)); y1 = csim('step', t, G1);
--> poles = roots(G1.den)
 poles  =
  -1. + 2.8284271i
  -1. - 2.8284271i
--> G2 = syslin('c', G1*(s + 0.5)*2); y2 = csim('step', t, G2);
--> r = 1;
--> while y1(r) <= 0.1
  >     r = r + 1;
  > end
--> ten_t1 = t(r-1);
--> r = 1;
--> while y1(r) <= 0.9
  >     r = r + 1;
  > end
--> ninety_t1 = t(r-1);
--> rt_1 = ninety_t1 - ten_t1
 rt_1  =
   0.4600000
--> r = 1;
--> while y2(r) <= 0.1
  >     r = r + 1;
  > end
--> ten_t2 = t(r-1);
```

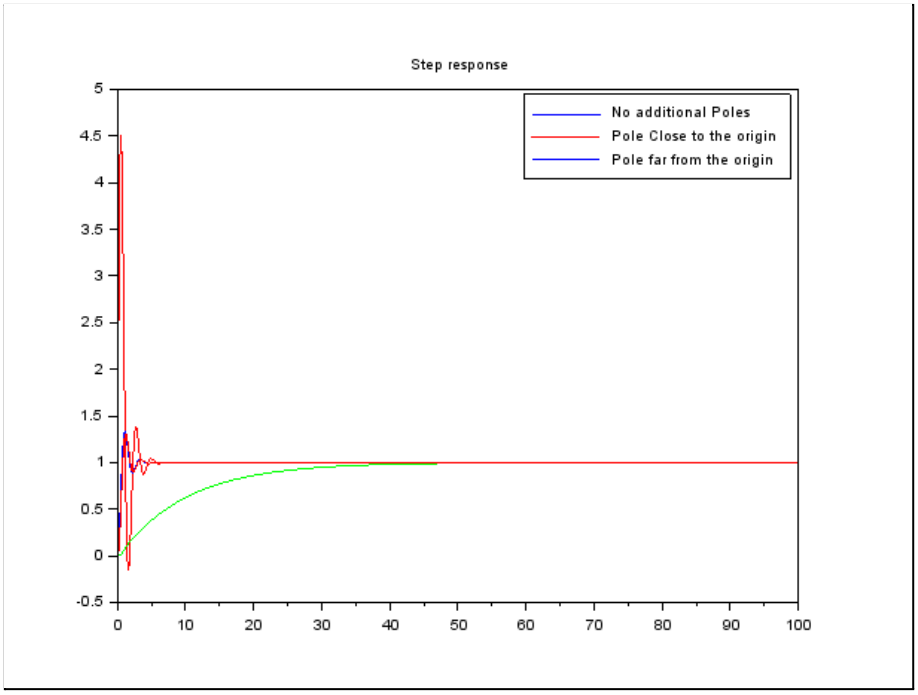
```
--> r = 1;
--> while y2(r) <= 0.9
  >     r = r + 1;
  > end
--> ninety_t2 = t(r-1);
--> rt_2 = ninety_t2 - ten_t2
 rt_2  =
   0.05
--> [g1max, tp1] = max(y1); [g2max, tp2] = max(y2);
--> os1 = (g1max - 1)*100
 os1  =
   32.932075
--> os2 = (g2max - 1)*100
 os2  =
   350.67829
--> plot(t,y1); plot(t,y2, 'r');
--> xtitle("Step response");
--> legend('No zero added', 'Extra Zero added');
--> xs2png(gcf(), "3a.jpg");
```




**b.** Code for part b is as follows:

```
--> s = poly(0, 's');
--> t = 0:0.1:100;
--> G3 = syslin('c', (G1/(s + 0.1))*0.1);
--> y3 = csim('step', t, G3);
--> G4 = syslin('c', (G1/(s + 10))*10);
--> y4 = csim('step', t, G4);
--> r = 1;
--> while y3(r) <= 0.1
  >     r = r + 1;
  > end
--> ten_t3 = t(r-1);
--> r = 1;
--> while y3(r) <= 0.9
  >     r = r + 1;
  > end
```

```
--> ninety_t3 = t(r-1);
--> rt_3 = ninety_t3 - ten_t3
 rt_3  =
   22.000000
--> r = 1;
--> while y4(r) <= 0.1
  >     r = r + 1;
  > end
--> ten_t4 = t(r-1);
--> r = 1;
--> while y4(r) <= 0.9
  >     r = r + 1;
  > end
--> ninety_t4 = t(r-1);
--> rt_4 = ninety_t4 - ten_t4
 rt_4  =
   0.5
--> [g3max, tp3] = max(y3); [g4max, tp4] = max(y4);


--> os1 = (g3max - 1)*100
 os1  =
  -0.0046379
--> os2 = ((g4max - 1))*100
 os2  =
   31.296913
--> y1 = csim('step', t, G1);
--> plot(t, y1); plot(t, y3, 'g'); plot(t, y4, 'r');
--> legend('No additional Poles', 'Pole Close to the origin', 'Pole far from the
origin');
--> xtitle("Step response");
--> xs2png(gcf(), "Q3b.jpg");
```



**c.** The following are my observations:

-> When a zero that is considerably further away from the origin than the dominant poles of the system is added, it constructively adds a scaled version of the derivative of the step response to itself. The result is that the step response tilts towards the left, giving a lower rise time and higher %OS.

-> When a pole much closer to the origin is added, it now becomes the dominant pole of the system and governs the time domain parameters. However, when a pole much further away from the origin is added, it does not affect the time domain parameters as much as the dominant poles of the system remain unchanged.

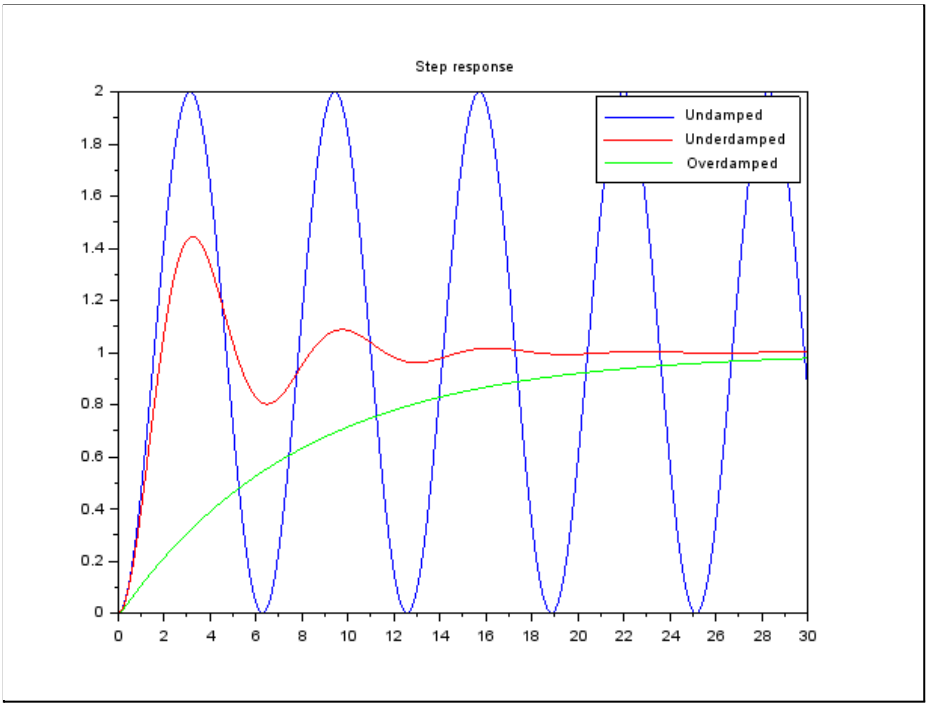
-> Also, note that on adding the pole much closer to the origin, there is only one dominant pole in the system, and hence the step response resembles that of a first order system, giving us no %OS.

# Question 4:

**a.** The entire code for this question is in part (a) itself:

```
--> s = poly(0, 's'); t = 0:0.01:30;
--> G1 = syslin('c', 1/(s^2+1)); G2 = syslin('c', 1/(s^2 + 0.5*s + 1)); G3 =
syslin('c', 1/(s^2 + 8*s + 1));
--> y1 = csim('step', t, G1); y2 = csim('step', t, G2); y3 = csim('step', t, G3);
--> plot(t, y1); plot(t, y2, 'r'); plot(t, y3, 'g');
--> legend('Undamped', 'Underdamped', 'Overdamped');
--> xtitle("Step response");

--> r = 1;
--> while y1(r) <= 0.1
  >     r = r + 1;
  > end
--> ten_t1 = t(r-1);
--> r = 1;
--> while y1(r) <= 0.9
  >     r = r + 1;
  > end
--> ninety_t1 = t(r-1);
--> rt_1 = ninety_t1 - ten_t1
 rt_1  =
   1.02
--> rt_2 = ninety_t2 - ten_t2
 rt_2  =
   1.26
--> rt_3 = ninety_t3 - ten_t3
 rt_3  =
   17.3

--> [g1max, tp1] = max(y1); [g2max, tp2] = max(y2); [g3max, tp3] = max(y3);
--> os1 = (g1max - 1)*100
 os1  =
   99.999934
--> os2 = (g2max - 1)*100
 os2  =
   44.433947
--> os3 = (g3max - 1)*100
 os3  =
  -2.2500116
```

```
--> tpeak1 = t(tp1)
 tpeak1  =
   21.990000
--> tpeak2 = t(tp2)
 tpeak2  =
   3.24

--> r = length(t) - 1;
--> while y2(r)>0.98 & y2(r)<1.02
  >     r = r - 1;
  > end
--> ts_2 = t(r+1)
 ts_2  =
   14.120000
--> ts_3 = t(r+1)
 ts_3  =
   30.

--> r=1;
--> while y1(r) <= 0.5
  >     r = r + 1;
  > end
--> td_1 = t(r-1)
 td_1  =
   1.04
--> td_2 = t(r-1)
 td_2  =
   1.1500000
--> td_3 = t(r-1)
 td_3  =
   5.58

--> xs2png(gcf(), "4.jpg");
```



The observations of different parameters have been printed in the code, for better clarity

-> Since the undamped case does not achieve steady state, Rise time, settling time, delay time & %OS for it are undefined. So in Scilab, we get the value as 30 (final t point of our plot)
-> Notice that the overdamped case has no overshoot, while the underdamped case does have it.
-> Even though the overdamped case has no %OS, it has more rise time, settling time and delay time than the underdamped case.