

Scilab Manual for  
Control and Instrumentation  
by Prof A. George Ansfer  
Electrical Engineering  
St. Xavier's Catholic College of Engineering<sup>1</sup>

Solutions provided by  
Prof A.George Ansfer  
Electrical Engineering  
St.Xavier's Catholic College of Engineering

May 19, 2026

<sup>1</sup>Funded by a grant from the National Mission on Education through ICT, <http://spoken-tutorial.org/NMEICT-Intro>. This Scilab Manual and Scilab codes written in it can be downloaded from the "Migrated Labs" section at the website <http://scilab.in>



# Contents

List of Scilab Solutions	3
1 Digital Simulation of P, PI, PD & PID Controller	5
2 Digital Simulation of Linear System	8
3 Simulation of Control Systems	10

# List of Experiments

Solution 1.1	Exp1 . . . . .	5
Solution 2.2	exp2 . . . . .	8
Solution 3.3	exp3 . . . . .	10
Solution 3.4	exp4 . . . . .	12

# List of Figures

1.1	Exp1	7
2.1	exp2	9
3.1	exp3	11
3.2	exp4	13

# Experiment: 1

## Digital Simulation of P, PI, PD & PID Controller

Scilab code Solution 1.1 Exp1

```
1 //Digital Simulation of P, PI, PD, and PID
   controllers
2 // 1- Open Loop Program
3 num=poly([10], 's', 'coeff');//Numerator input
4 den=poly([20 10 1], 's', 'coeff');//Denominator input
5 q=syslin('c',num/den)//Ratio of the numerator to the
   denominator
6 t=0:0.05:2.5;//time interval
7 p=csim('step',t,q);
8 subplot(321)
9 plot2d(t,p);
10 xtitle(['Open Loop '], 'Time(Second)', 'Amplitude' );
11
12 // 2- P Control Program
13 kp=300;
14 num=poly([kp], 's', 'coeff');
15 den=poly([20+kp 10 1], 's', 'coeff');
16 q=syslin('c',num/den)
17 t=0:0.01:2;
```

```

18 p=csim('step',t,q);
19 subplot(322)
20 plot2d(t,p);
21 xtitle(['P Control'], 'Time(Second)', 'Amplitude' );
22
23 // 3      PI Control Program
24 kp=30;
25 ki=70;
26 num=poly([ki kp], 's', 'coeff');
27 den=poly([ki 20+kp 10 1], 's', 'coeff');
28 q=syslin('c', num/den)
29 t=0:0.01:2;
30 p=csim('step',t,q);
31 subplot(323)
32 plot2d(t,p);
33 xtitle(['PI Control'], 'Time(Second)', 'Amplitude' )
    ;
34
35 // 4      PD Control Program
36 kp=300;
37 kd=10;
38 num=poly([kp kd], 's', 'coeff');
39 den=poly([20+kp 10+kd 1], 's', 'coeff');
40 q=syslin('c', num/den)
41 t=0:0.01:2;
42 p=csim('step',t,q);
43 subplot(324)
44 plot2d(t,p);
45 xtitle(['PD Control'], 'Time(Second)', 'Amplitude' )
    ;
46
47 // 5      PID Control Program
48 kp=350;
49 kd=50;
50 ki=300;
51 num=poly([ki kp kd], 's', 'coeff');
52 den=poly([ki 20+kp 10+kd 1], 's', 'coeff');
53 q=syslin('c', num/den)

```

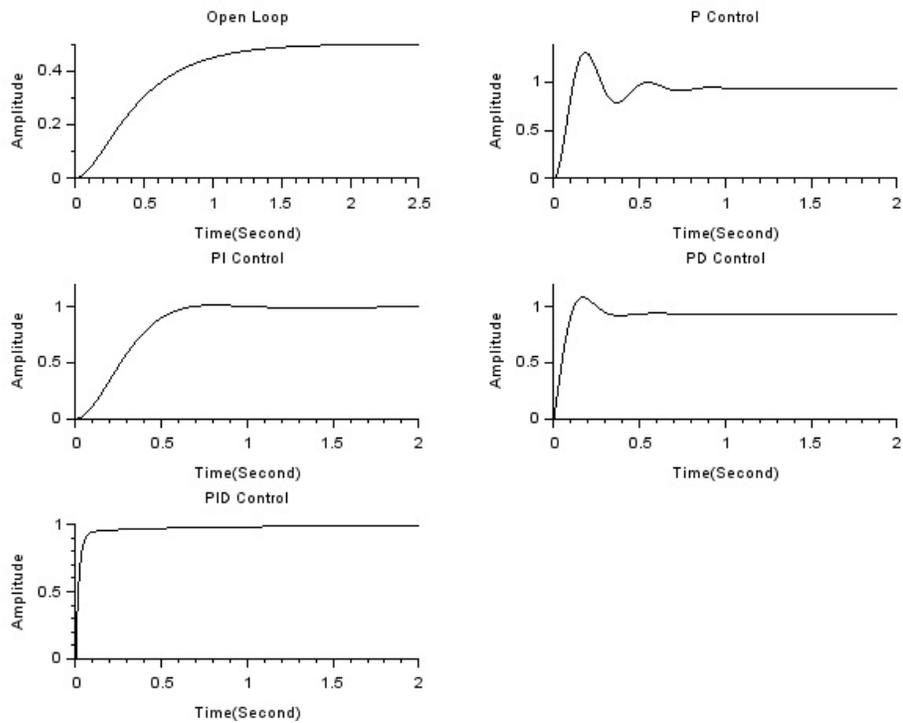


Figure 1.1: Exp1

```

54 t=0:0.01:2;
55 p=csim('step',t,q);
56 subplot(3,2,5)
57 plot2d(t,p);
58 xtitle(['PID Control'], 'Time(Second)', 'Amplitude'
);

```

---

## Experiment: 2

# Digital Simulation of Linear System

Scilab code Solution 2.2 exp2

```
1 //Second order system step response for damping
   conditions
2 t=0:0.0000001:0.0002;
3 d=[0.5 1 1.5]; //Entering the damping conditions
   values
4 cv=[1 2 3];
5 s=%s;
6 for n=1:3
7 num = 10^10;
8 den = s^2 + 2*d(n)*100000*s +10^10;
9 P = syslin('c',num,den);
10 Ps=csim('step',t,P);
11 plot2d(t,Ps,style=cv(n));
12 end;
13 xgrid(6);
14 xtitle(['Second order step response ','Time(Second)
   ','Amplitude']);
15 legends(['d=0.5(underdamped)';'d=1(critically damped)
   ');'d=1.5(overdamped)'],[1,2,3],opt=4);
```

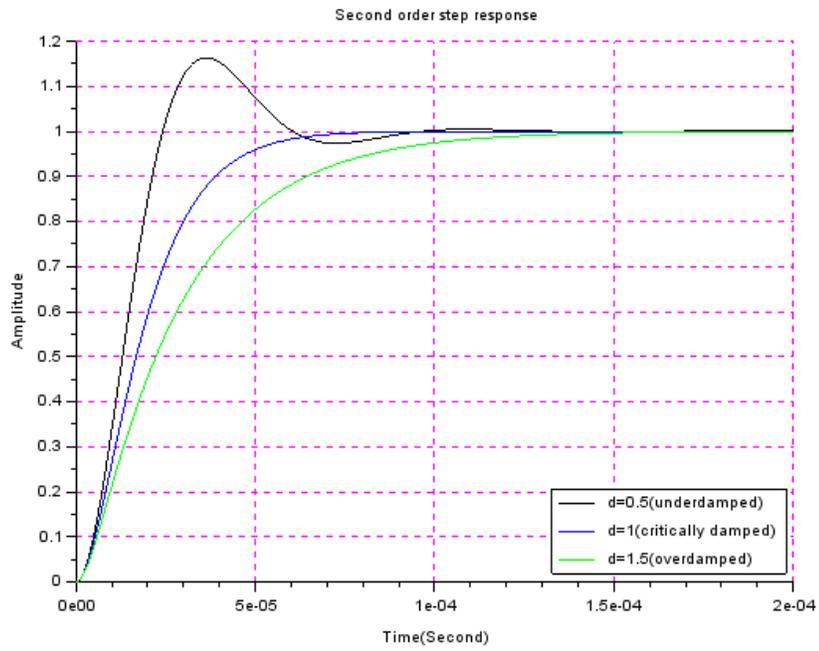


Figure 2.1: exp2

# Experiment: 3

## Simulation of Control Systems

Scilab code Solution 3.3 exp3

```
1 // Effects on Stability.
2 //  $G(s) = a^2 / s(s+2b a)$  ,  $H(s) = C$  ,  $T(s) = C a^2$ 
   //  $/ ( s(s+2b a) + C a^2 )$ 
3 s=%s;
4 t=0:0.01:10;
5 a=1;b=1;
6 C=[1,2,5,10]
7 for n=1:4
8     T=syslin('c', C(n)*a^2 , s*(s + 2*b*a) + C(n)*a
           ^2 );
9     Ts=csim('step',t,T);
10    xset("line style",n);
11    plot2d(t,Ts);
12    xgrid(5);
13 end
14 xtitle('Effects on Stability .', 'Time(sec)', 'C(t)');
15 legends(['C=1'; 'C=2'; 'C=5'; 'C=10'
           ;], [[1;1],[1;2],[1;3],[1;4]], opt=4);
```

---

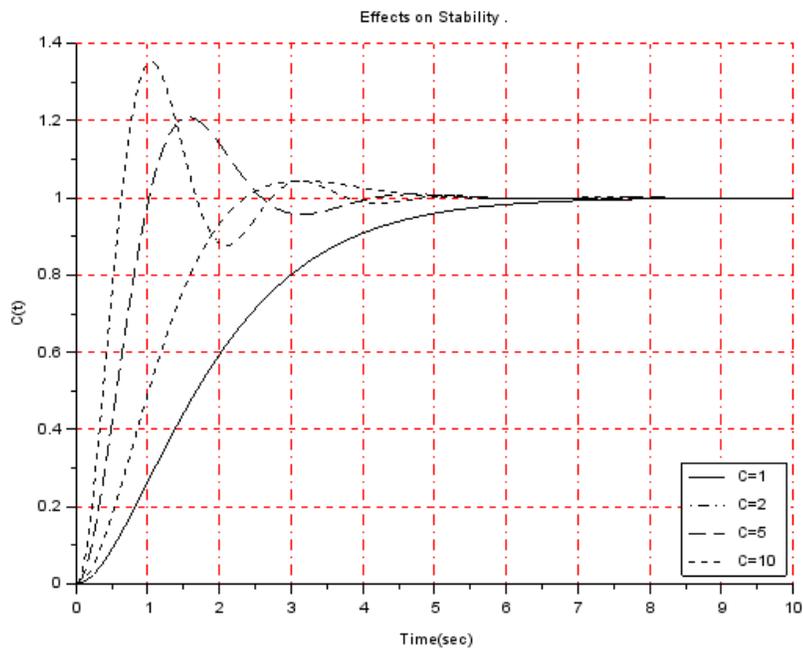


Figure 3.1: exp3

### Scilab code Solution 3.4 exp4

```
1 //Frequency Domain Methods for controller design
2
3 // 1- Normalized bandwidth vs Damping factor (second
   order system)
4 defff(" [a]=f1(b)", "a=sqrt(1-2*b^2+sqrt(2-4*b^2+4*b^4))")
5 b=[0:0.01:0.9];
6 subplot(311)
7 fplot2d(b,f1,[1])
8 xgrid(2)
9 xtitle(['Normalized bandwidth vs Damping Factor'], '
   Damping ratio', 'Normalized bandwidth');
10
11 // 2- Peak overshoot vs Resonance Peak (second
   order system)
12 defff(" [C]=f2(b)", "C=exp((-pi*b)/sqrt(1-b^2))")
13 defff(" [D]=f3(b)", "D=1/(2*b*sqrt(1-b^2))")
14 b=[0.05:0.01:0.9];
15 subplot(312)
16 xset("line style",4);
17 fplot2d(b,f2,[1])
18 xset("line style",1);
19 fplot2d(b,f3,[1])
20 xgrid(3)
21 xtitle(['Peak overshoot vs Resonance Peak'], '
   Damping ratio', 'Peak Gain, Resonance Gain');
22 legends(['Peak Gain'; 'Resonance Gain'
   ], [[1;4], [1;1]], opt=1);
23
24 // 3- Resonant Frequency vs Damping Frequency (
   Second order system)
25 defff(" [e]=f4(b)", "e=sqrt(1-2*b^2)/sqrt(1-b^2)")
```

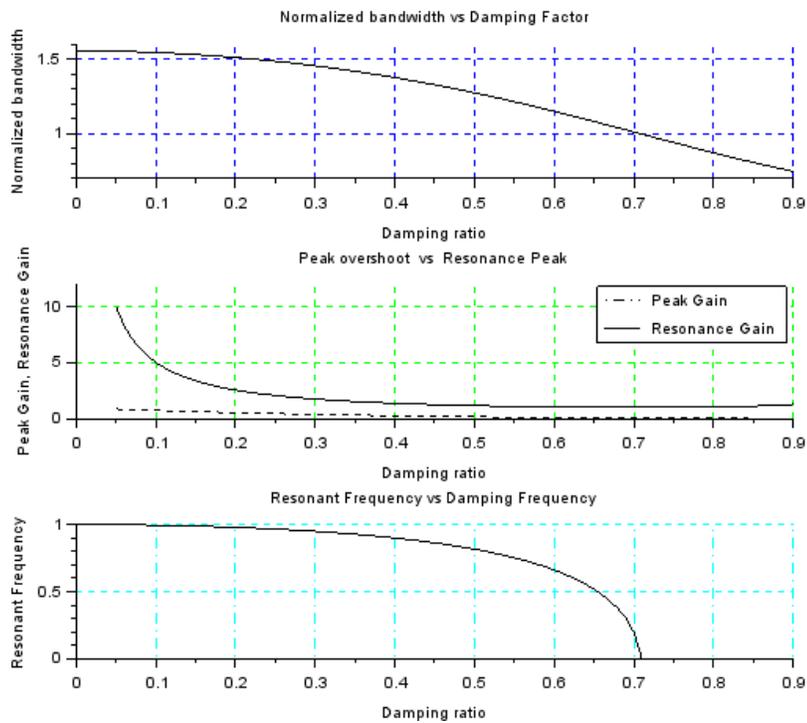


Figure 3.2: exp4

```

26 b=[0:0.01:0.9]; // don't end with 1 bec, division by
    0 error
27 subplot(313)
28 fplot2d(b,f4,[1])
29 xgrid(4)
30 xtitle(['Resonant Frequency vs Damping Frequency'],'
    Damping ratio', ' Resonant Frequency ');

```

---