Scilab Textbook Companion for Feedback Control of Dynamic Systems by G. F. Franklin, J. D. Powell and A. Emami-Naeini[1]

Created by
Mrs. Deepti Khimani
Feeback Control System, Control System Design
Instrumentation Engineering
Mumbai University
College Teacher
None
Cross-Checked by
Spandana

July 31, 2019

[1]Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Textbook Companion and Scilab codes written in it can be downloaded from the "Textbook Companion Project" section at the website http://scilab.in

Book Description

Title: Feedback Control of Dynamic Systems

Author: G. F. Franklin, J. D. Powell and A. Emami-Naeini

Publisher: Pearson Education and Dorling Kindersley

Edition: 5

Year: 2010

ISBN: 978-81-317-2142-1

Scilab numbering policy used in this document and the relation to the above book.

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

AP Appendix to Example(Scilab Code that is an Appednix to a particular Example of the above book)

For example, Exa 3.51 means solved example 3.51 of this book. Sec 2.3 means a scilab code whose theory is explained in Section 2.3 of the book.

Contents

List of Scilab Codes 4

2 Dynamic Models 5

3 Dynamic Responses 8

4 Basic properties of feedback 32

5 The Root Locus Design method 43

6 The Frequency Response Design Method 67

7 State Space Design 104

8 Digital Control 167

9 Nonlinear Systems 173

List of Scilab Codes

Exa 2.1.b	step response of Cruise control system	5
Exa 2.5.b	step response of pendulum	6
Exa 3.4	Frequency response	8
Exa 3.8	Partial fraction expansion for distinct real roots	10
Exa 3.9	Final value theorem	11
Exa 3.10	Incorrect use of final value theorem	12
Exa 3.11	DC gain of the system	13
Exa 3.14	Partial fraction expansion for distinct real roots	14
Exa 3.15	Cruise Control Transfer Function	15
Exa 3.16	DC Motor Transfer Function	16
Exa 3.17	Transformations	17
Exa 3.18	Satellite Transfer Function	18
Exa 3.21	Transfer function of a simple system	20
Exa 3.22	Response Versus Pole Locations Real Roots	22
Exa 3.23	Oscillatory Time Response	24
Exa 3.25	Aircraft Response	25
Exa 3.29	Stability versus parameter range	27
Exa 3.30	Stability versus two parameter ranges	29
Exa 4.6	PID Control of DC Motor Speed	32
Exa 4.7	Discrete Equivalent	36
Exa 4.8	Equivalent discrete controller for DC motor speed control	37
Exa 5.1	Root locus of a Motor Position Control	43
Exa 5.2	Root locus with respect to a plant open loop pole	44
Exa 5.3	Root locus for satellite attitude control with PD control	45

Exa 5.4	Root locus for satellite attitude control with modified PD control or Lead compensator	46
Exa 5.5	Root locus for satellite control with Lead compensator	47
Exa 5.6	Root locus for satellite attitude control with a Transition value for the pole	48
Exa 5.7	Root locus for satellite control with a Collocated Flexibility	49
Exa 5.8	Root locus for noncollocated case	53
Exa 5.9	Root locus for the system having complex multiple roots	57
Exa 5.10	Design using Lead compensator	60
Exa 5.11	A second Lead compensation Design	62
Exa 5.12	Negative root Locus for an Airplane	65
Exa 6.2.b	Frequency response characteristics of Lead compensator	67
Exa 6.3	Bode Plot for Real Poles and Zeros	68
Exa 6.4	Bode Plot with Complex Poles	71
Exa 6.6	Bode Plot for Complex Poles and Zeros	72
Exa 6.7	Computation of Kv	74
Exa 6.8	Nyquist plot for a second order system	76
Exa 6.9	Nyquist plot for a third order system	78
Exa 6.10	Nyquist plot for an Open loop unstable system	80
Exa 6.11	Stability properties for a conditionally stable system	82
Exa 6.12	Nyquist plot for a system with Multiple Crossover frequencies	83
Exa 6.13	Use of simple design criterion for spacecraft attitude control	85
Exa 6.14	Lead compensation for DC motor	88
Exa 6.15	Lead compensation for Temperature Control System	91
Exa 6.16	Lead compensation for Servomechanism System	94
Exa 6.17	Lag compensation for Temperature Control System	96
Exa 6.18	Lag compensation for DC motor	98

Exa 6.19	PID compensation design for spacecraft attitude control	100
Exa 7.2.b	Cruise control system step response	104
Exa 7.7	Analog computer Implementation	105
Exa 7.8	Time scaling an oscillator	107
Exa 7.9	State Equations in Modal Canonical Form	107
Exa 7.10	Transformation of Thermal System from Control to Modal Form	109
Exa 7.11	Poles and Zeros of Tape Drive System	110
Exa 7.12	Transformation of Thermal System from state description	112
Exa 7.13	Zeros for the Thermal System from a State Description	112
Exa 7.14	Analysis of state equations of Tape Drive	113
Exa 7.15	Control law for a pendulum	116
Exa 7.16	Ackermanns formula for undamped oscillator	118
Exa 7.17	How zero location affect control law	119
Exa 7.18	Introducing the reference input	120
Exa 7.19	Reference input to Type1 control system DC Motor	122
Exa 7.20	Pole Placement as a Dominant Second Order System	125
Exa 7.21	Symmetric root locus for servo speed control	128
Exa 7.22	SRL design for satellite attitude control	130
Exa 7.23	SRL design for an inverted pendulum	131
Exa 7.24	LQR Design for a Tape Drive	134
Exa 7.25	An estimator design for a simple pendulum	138
Exa 7.26	A reduced order estimator design for pendulum	140
Exa 7.27	SRL estimator design for a simple pendulum	142
Exa 7.28	Full order compensator design for satellite attitude control	144
Exa 7.29	A reduced order compensator design for a satellite attitude control	146
Exa 7.30	Full Order Compensator Design for DC Servo	149
Exa 7.31	Reduced Order Estimatortor Design for DC Servo	151

Exa 7.32	Redesign of the Dc servo compensator using SRL	154
Exa 7.33	DC servo system redesign with modified with dominant second order pole locations	157
Exa 7.34	Servomechanism increasing the velocity constant through zero assignment	159
Exa 7.35	Integral Control of a Motor Speed System	163
Exa 8.1	Digital Controller using tustin approximation	167
Exa 8.2	Design of a Space Station Attitude Digital Controller using Discrete Equivalents	169
Exa 9.5	Changing Overshoot and Saturation nonlinearity	173
Exa 9.6	Stability of conditionally stable system using root locus	175
Exa 9.7	Analysis and design of the system with limit cycle using the root locus	177
Exa 9.8	Antiwindup compensation for a PI controller	180
Exa 9.9	Describing Function for a saturation nonlinearity	182
Exa 9.11	Describing Function for a relay with hysteresis non linearity	184
Exa 9.12	Conditionally stable system	186
Exa 9.13	Determination of stability with a hysteresis nonlinearity	189

Chapter 2

Dynamic Models

Scilab code Exa 2.1.b step response of Cruise control system

```
1 //Example 2.1
2 //(b) step response of Cruise control system
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7
8 //
   ____________________________________________________________________________

9 //Cruise control parameters
10 m=1000;
11 b=50;
12 u=500;
13
14 // Transfer function
15 s=%s; // or
16 s=poly(0,'s');
17 sys=syslin('c',(1/m)/(s+b/m))
18
19 //step response to u=500;
```

```
t=0:0.5:100;
v=csim('step',t,u*sys);
plot2d(t,v,2)

//Title, labels and grid to the figure
exec .\fig_settings.sci; //custom script for setting
    figure properties
title('Responses of car velocity to a step in u','
   fontsize',3)
xlabel('Time t (sec.)','fontsize',2)
ylabel('Amplitude','fontsize',2)

//
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 2.5.b `step response of pendulum`

```
//Example 2.5
//(b) step response of pendulum

xdel(winsid())//close all graphics Windows
clear;
clc;

//

//Pendulum parameters
m=0.5;
```

```
11 l=1;
12 g=9.81;
13
14 // Transfer function
15 s=%s;
16 sys=syslin('c',(1/(m*l^2))/(s^2+g/l));
17
18 //step response to u=500;
19 t=0:0.02:10;
20 theta=csim('step',t,sys);
21 plot(t,theta*57.3);
22
23 //Title, labels and grid to the figure
24 exec .\fig_settings.sci; // custom script to set
      figure properties
25 title('Response of pendulum to a step input in the
      applied torque',...
26 'fontsize',3);
27 xlabel('Time t (sec.)','fontsize',2);
28 ylabel('Pendulum angle (degree)','fontsize',2);
29
30 //
   ____________________________________________________________________
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Chapter 3

Dynamic Responses

check Appendix AP 1 for dependency:

```
fig_settings.sci
```

Scilab code Exa 3.4 Frequency response

```
1 //Example 3.4
2 //Frequency response
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
  ------------------------------------------------------------------------
8 //(a) Frequency response of 1/(s+k)
9 k=1;
10 fmin=1e-2;
11 fmax=1e2;
12 // Transfer function
13 s=poly(0,'s');
14 sysH=syslin('c',1/(s+k))
15
```

```
16 //Frequency response for k=1
17 //Note that - magnitude plot semilog plot unlike log
     -log plot in the book.
18 bode(sysH,fmin,fmax)
19 title('Frequency response for k=1','fontsize',3)
20
21 //
     ------------------------------------------------------------------------


22 //(b) Response to u=sin(10*t);
23 t=0:0.02:10;
24 u=sin(10*t);
25 y=csim(u,t,sysH);
26 figure, plot(t,y)
27
28 //Title, labels and grid to the figure
29 exec .\fig_settings.sci; // custom script for
     setting figure properties
30 title('Complete transient response','fontsize',3)
31 xlabel('Time (sec.)','fontsize',2)
32 ylabel('Output','fontsize',2)
33
34 //phase lag
35 figure, plot(t,y)
36 plot(t,u,'r')
37 zoom_rect([9 -1 10 1])
38 exec .\fig_settings.sci; // custom script for
     setting figure properties
39 title('Phase lag between output and input','fontsize
     ',3)
40 xlabel('Time (sec.)','fontsize',2)
41 ylabel('Output, Input','fontsize',2)
42 h=legend('y(t)','u(t)')
43 h.legend_location = "in_upper_right"
44 h.fill_mode='off'
45
46 // time lag
47 w=find(t>=9.4 & t<=10);
```

```
48 T=t(w);
49 Y=y(w);
50 U=u(w);
51 wu=find(U==max(U))
52 wy=find(Y==max(Y))
53
54 //Responses
55 plot2d3(T(wy),Y(wy))
56 plot2d3(T(wu),U(wu))
57 delta_t=T(wu)-T(wy); //time lag sec.
58 xstring(9.64,-0.1,"$\delta t$",0,0)
59 xarrows([9.58;9.72], [0;0], 0.7, 1)
60 xarrows([9.72;9.58], [0;0], 0.7, 1)
61 t=get("hdl")
62 disp(abs(delta_t), "Time lag of output in sec. is")
63 disp(abs(delta_t)*10, "Phase lag of output in
     radians is")
64
65 //
```

Scilab code Exa 3.8 Partial fraction expansion for distinct real roots

```
1 //Example 3.8
2 //Partial fraction expansion for distinct real roots
    .
3
4 clear;
5 clc;
6 //
```

```
7  //Partial fraction expansion for distinct real roots
8  // Transfer function
9  s=%s;
10 num=(s+2)*(s+4)
11 p1=s;
12 p2=(s+1);
13 p3=(s+3);
14 sys=syslin('c',num/(p1*p2*p3))
15 //
   ---------------------------------------------------------------------

16 //Partial fraction expansion is: sys= r1/p1 + r2/p2
      + r3/p3
17 //residue calculation
18 r1=residu(num,p1,(p2*p3))
19 r2=residu(num,p2,(p1*p3))
20 r3=residu(num,p3,(p1*p2))
21
22 disp([r1 r2 r3]',"Residues of the poles p1, p2 and
      p3 are")
23 disp([roots(p1), roots(p2), roots(p3)]',"Poles p1,
      p2 and p3 are at")
24 disp('k=[]')
25
26 //
   ---------------------------------------------------------------------
```

Scilab code Exa 3.9 Final value theorem

```
1  //Example 3.9
2  //Computing final value (use of final value theorem)
      .
```

```
3
4 clear;
5 clc;
6
7 //
    ------------------------------------------------------------------------------


8
9 //Computing final value (use of final value theorem)
10 // Output of the system
11 s=poly(0,'s');
12 num=3*(s+2);
13 den=s*(s^2+2*s+10);
14 Ys=syslin('c',num/den);
15
16
17 //final value theorem, lim s-->0 in s*Y(s)
18
19 Y_final=horner(s*Ys,0)
20 disp(Y_final,"The final value of the output y is:")
21
22 //
    ------------------------------------------------------------------------------
```

Scilab code Exa 3.10 Incorrect use of final value theorem

```
1 //Example 3.10
2
3 //Computing final value for unstable system to show
    the incorrect
4 // use of final value theorem.
5 clear;
6 clc;
7 //
```

```
s=poly(0,'s');
num=3;
den=s*(s-2);
Ys=syslin('c',num/den);

//final value theorem, lim s-->0 in s*Y(s)
Y_final=horner(s*Ys,0);
disp(Y_final,"The final value of the output y is:");
disp('The final value computed is incorrect as the
   system...
 response is unbounded');
//
```

Scilab code Exa 3.11 DC gain of the system

```
//Example 3.11
//Computing DC gain of the system.

clear;
clc;
//

//Transfer Function
s=poly(0,'s');
num=3*(s+2);
den=(s^2+2*s+10);
Ys=syslin('c',num/den);

//The DC gain of the system Y(s) as s-->0 is
DC_Gain=horner(Ys,0)
```

```
15 disp(DC_Gain,"The DC gain of the system is:")
16 //
```

Scilab code Exa 3.14 Partial fraction expansion for distinct real roots

```
1 //Example 3.14
2 //Partial fraction expansion for distinct real roots
3 clear;
4 clc;
5 //
```

```
6 // Transfer function
7 s=%s;
8 num=2;
9 p1=(s+1);
10 p2=(s+2);
11 p3=(s+4);
12 sys=syslin('c',num/(p1*p2*p3))
13
14 //Partial fraction expansion is: sys= r1/p1 + r2/p2
   + r3/p3
15 //residue calculation
16 r1=residu(num,p1,(p2*p3))
17 r2=residu(num,p2,(p1*p3))
18 r3=residu(num,p3,(p1*p2))
19
20 disp([r1 r2 r3]',"Residues of the poles p1, p2 and
   p3 are")
21 disp([roots(p1), roots(p2), roots(p3)]',"Poles p1,
   p2 and p3 are at")
22 disp('k=[]')
23 //
```

Scilab code Exa 3.15 Cruise Control Transfer Function

```
1 //Example 3.15 Cruise Control Transfer Function.
2 //Coefficients of numerator and denominator of the
     transfer function
3
4 clear;
5 clc;
6 //
```

```
7 // Transfer function coefficients
8 num=[0.001 0];
9 den=[0 0.05 1];
10
11 // Transfer function
12 Ns=poly(num,'s','coeff');
13 Ds=poly(den,'s','coeff');
14 sys=syslin('c',Ns/Ds);
15
16 //gain (K) pole (P) and zeros (Z) of the system
17 temp=polfact(Ns);
18 Z=roots(Ns); //locations of zeros
19 P=roots(Ds); //locations of poles
20 K=temp(1); //first entry is always gain
21 disp( K,"Gain", P, "Poles",Z,"Zeros",)
22
23 //
```

check Appendix AP 1 for dependency:

fig_settings.sci

Scilab code Exa 3.16 DC Motor Transfer Function

```
1 //Example 3.16 DC Motor Transfer Function.
2
3 xdel(winsid())//close all graphics Windows
4 clear;
5 clc;
6 //
    ------------------------------------------------------------------------

7 //Coefficients of numerator and denominator of the
    transfer function
8 numb=[100];
9 denb=[0 101 10.1 1];
10
11 // Transfer function
12 Ns=poly(numb,'s','coeff');
13 Ds=poly(denb,'s','coeff');
14 sysb=syslin('c',Ns/Ds);
15
16 //gain (K) pole (P) and zeros (Z) of the system
17 temp=polfact(Ns);
18 Z=roots(Ns); //locations of zeros
19 P=roots(Ds); //locations of poles
20 K=temp(1); //first entry is always gain
21 disp( K,"Gain", P, "Poles",Z,"Zeros",)
22
23 //Transient response of DC Motor (consider velocity
    as output)
24 s=%s;
25 t=linspace(0,5,501);
26 y=csim('step',t,sysb*s)
27 plot(t,y)
```

```
28 exec .\fig_settings.sci; //custom script for setting
       figure properties
29 title('Transient response of DC Motor','fontsize',3)
30 xlabel('$Time\,\, t(sec.)$','fontsize',3)
31 ylabel('$\omega\,\,(rad/sec)$','fontsize',3)
32 //
```

Scilab code Exa 3.17 Transformations

```
1 //Example 3.17 Transformations
2
3 clear;
4 clc;
5 //
```

```
6 //Coefficients of numerator and denominator of the
     transfer function
7 numG=[9 3];
8 denG=[25 6 1];
9
10 // Transfer function
11 Ns=poly(numG,'s','coeff');
12 Ds=poly(denG,'s','coeff');
13 sysG=syslin('c',Ns/Ds);
14
15 //gain (K) pole (P) and zeros (Z) of the system
16 temp=polfact(Ns);
17 Z=roots(Ns); //locations of zeros
18 P=roots(Ds); //locations of poles
19 K=temp(1); //first entry is always gain
```

```
20 disp( K,"Gain", P, "Poles",Z,"Zeros",)
21 //
```

Scilab code Exa 3.18 Satellite Transfer Function

```
1 //Example 3.18 Satellite Transfer Function
2
3 xdel(winsid())//close all graphics Windows
4 clear;
5 clc;
6 //
```

```
7 //(a)
8 //Given
9 d=1 //meters
10 I=5000 //Kg-meter^2
11
12 //Coefficients of numerator and denominator of the
   transfer function
13 // of satellite
14 numG=[d/I 0];
15 denG=[0 0 1];
16
17 // Transfer function
18 Ns=poly(numG,'s','coeff');
19 Ds=poly(denG,'s','coeff');
20 sysG=syslin('c',Ns/Ds);
21 t=0:0.01:10;
22 [i j]=size(t);
23
24 //
```

```
25 //(b)
26 // Thrust input after 5 sec.
27 u=zeros(1,j);
28 w=find(t>=5 & t<=5+0.1);
29 u(w)=25;
30 plot(t,u);
31 exec .\fig_settings.sci; //custom script for setting
       figure properties
32 title("Transient response of the satellite...
33  (a) Thrust input",'fontsize',3);
34 xlabel('Time t (sec.)','fontsize',2)
35 ylabel('Fc','fontsize',2)
36
37 //Transient response of the satellite to the thrust
     input as a pulse
38 sysd=dscr(sysG,0.01); //sample data system model
39 y=flts(u,sysd);          //impulse response
40 figure, plot(t,y*180/%pi);
41 exec .\fig_settings.sci; //custom script for setting
       figure properties
42 title("Transient response of the satellite(double-
     pulse)...
43  (b) satellite attitude",'fontsize',3);
44 xlabel('Time t (sec.)','fontsize',2)
45 ylabel('$\theta(deg)$','fontsize',2)
46 //
     ------------------------------------------------------------------------


47 // Thrust input double-pulse.
48 u=zeros(1,j);
49 w1=find(t>=5 & t<=5+0.1);
50 u(w1)=25;
51 w2=find(t>=6.1 & t<=6.1+0.1);
52 u(w2)=-25;
53 figure,
54 plot(t,u);
55 exec .\fig_settings.sci; //custom script for setting
```

```
    figure properties
title("Transient response of the satellite (double-
    pulse)...
 (a) Thrust input",'fontsize',3);
xlabel('Time t (sec.)','fontsize',2)
ylabel('Fc','fontsize',2)

//Transient response of the satellite to the thrust
    input as a pulse
sysd=dscr(sysG,0.01); //sample data system model
y=flts(u,sysd);         //impulse response
figure, plot(t,y*180/%pi);
exec .\fig_settings.sci; //custom script for setting
    figure properties
title("Transient response of the satellite(double-
    pulse)...
 (b) satellite attitude",'fontsize',3);
xlabel('Time t (sec.)','fontsize',2)
ylabel('$\theta(deg)$','fontsize',2)

//
```

Scilab code Exa 3.21 Transfer function of a simple system

```
//Example 3.21
//Series, Parallel and Feedback connections of TF
    blocks
//to get effective TF.

```

```
5  clear;
6  clc;
7  //
   ------------------------------------------------------------------------

8  //Transfer function block G1
9  num1=[2];
10 den1=[1];
11 Ns=poly(num1,'s','coeff');
12 Ds=poly(den1,'s','coeff');
13 sysG1=syslin('c',Ns/Ds);
14
15 //Transfer function block G2
16 num2=[4];
17 den2=[0 1];
18 Ns=poly(num2,'s','coeff');
19 Ds=poly(den2,'s','coeff');
20 sysG2=syslin('c',Ns/Ds);
21
22 //Transfer function block G4
23 num4=[1];
24 den4=[0 1];
25 Ns=poly(num4,'s','coeff');
26 Ds=poly(den4,'s','coeff');
27 sysG4=syslin('c',Ns/Ds);
28
29 //Transfer function block G6
30 num6=[1];
31 den6=[1];
32 Ns=poly(num6,'s','coeff');
33 Ds=poly(den6,'s','coeff');
34 sysG6=syslin('c',Ns/Ds);
35
36 //Effective transfer function
37 // (+) operator for paralle connection,
38 // (*) operator for series connection
39 // (/.)operator for feedback connection
40 sysG=(sysG1 + sysG2) * sysG4 /. sysG6
```

```
41 disp(sysG, "The effective transfer function is")
42 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 3.22 Response Versus Pole Locations Real Roots

```
1 //Example 3.22 Response Versus Pole Locations, Real
    Roots
2
3 xdel(winsid())//close all graphics Windows
4 clear;
5 clc;
6 //
```

```
7 //Transfer function
8 numH=[1 2];
9 denH=[2 3 1];
10 Ns=poly(numH,'s','coeff');
11 Ds=poly(denH,'s','coeff');
12 sysH=syslin('c',Ns/Ds);
13
14 //Pole-zero locations
15 //Partial fraction method to see the effect of
    sperated poles
16 temp=polfact(Ds);
17 p1s=temp(2);
18 p2s=temp(3);
19
20 //residues at poles
21 r1=residu(Ns,p1s,p2s);
```

```
22 r2=residu(Ns,p2s,p1s);
23
24 //Note that - H1(s)+H2(s)=H(s)
25 H1s=syslin('c',r1/p1s);
26 H2s=syslin('c',r2/p2s);
27
28 //impulse response of the H1(s), H2(s) and H(s)
29 t=0:0.02:10;
30 h1=csim('impuls',t,H1s);
31 h2=csim('impuls',t,H2s);
32 h=csim('impuls',t,sysH);
33 figure,
34 plot(t,h1,'r--',t,h2,'m-.', t, h, 'b')
35 plot(t,h2,'m-.')
36 plot(t,h)
37
38 exec .\fig_settings.sci; //custom script for setting
      figure properties
39 title(['impulse response of the system and
     subsystems with...
40  independent poles.';'(h1(t) is faster than h2(t))'
     ],'fontsize',3)
41 xlabel('Time t (sec.)','fontsize',2)
42 ylabel('h(t), h1(t), h2(t)','fontsize',2)
43 h=legend('h1(t) with pole at -2','h2(t) with pole at
     -1'...
44 ,'h(t)=h1(t)+h2(t)')
45 h.legend_location = "in_upper_right"
46 h.fill_mode='off'
47 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 3.23 Oscillatory Time Response

```
1 //Example 3.23 Oscillatory Time Response
2
3 xdel(winsid())//close all graphics Windows
4 clear;
5 clc;
6 //
    ------------------------------------------------------------------------

7 //Transfer function of second order underdamped
    system
8 numH=[1 2];
9 denH=[5 2 1];
10 Ns=poly(numH,'s','coeff');
11 Ds=poly(denH,'s','coeff');
12 sysH=syslin('c',Ns/Ds);
13
14 //damping factor (xi) and natural frequency (wn)
15 [wn xi]=damp(sysH);
16 wn=wn(1);
17 xi=xi(1);
18 sigma=xi*wn;
19 wd=wn*sqrt(1-xi^2);
20
21 //denominator in sigma-wn form H(s)=H1(s)+H2(s)
22 s=%s;
23 p=(s+sigma)^2+wd^2
24 temp=polfact(Ns);
25 k=temp(1),zr=temp(2);
26 h1=(s+sigma)/p;
27 h2=-((s+sigma)-temp(2))*wd/p;
28 H1s=syslin('c',k*h1);
29 H2s=syslin('c',k*h2/wd);
```

```
30
31 // responses with exponential envelope
32 Env=syslin('c',k/(s+sigma));
33 t=0:0.02:10;
34 //impulse response
35 ht=csim('impuls',t,sysH);
36 envt=csim('impuls',t,Env);
37 envt_neg=csim('impuls',t,-Env);
38
39 plot(t,ht)
40 plot(t,envt,'r--')
41 plot(t,envt_neg,'r--')
42 exec .\fig_settings.sci; //custom script for setting
     figure properties
43 title('Impulse response of the underdamped system','
    fontsize',3)
44 xlabel('Time t (sec.)','fontsize',2)
45 ylabel('h(t)','fontsize',2)
46 xset("font",1,2)
47 xstring(1,0.75,"$e^{-\sigma t}$",0,0)
48 xstring(1,-0.85,"$-e^{-\sigma t}$",0,0)
49 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 3.25 `Aircraft Response`

```
1 //Example 3.25 Aircraft Response
2 xdel(winsid())//close all graphics Windows
3 clear;
```

```
4  clc;
5  //
   ------------------------------------------------------------------------

6  //(a)impulse response of aircraft
7
8  //Transfer function of aircraft
9  numG=[-6 1];
10 denG=[0 13 4 1];
11 Ns=30*poly(numG,'s','coeff');
12 Ds=poly(denG,'s','coeff');
13 u=-1 //impulsive elevator input of 1 degree
14 sysG=syslin('c',u*Ns/Ds);
15
16 //impulse response
17 t=0:0.02:10;
18 gt=csim('impuls',t,sysG);
19 plot(t,gt)
20 exec .\fig_settings.sci; //custom script for setting
       figure properties
21 title('Response of an airplanes altitude to an
      impulsive elevator input','fontsize',3)
22 xlabel('Time (sec.)','fontsize',2)
23 ylabel('Altitude (ft)','fontsize',2)
24
25 //final value theorem, lim s-->0 in s*G(s)
26 s=%s;
27 gt_final=horner(s*sysG,0)
28 disp(gt_final,"The final value of the output
      altitude is:")
29 //
   ------------------------------------------------------------------------

30 //(b)response specifications
31
32 //damping factor (xi) and natural frequency (wn)
33 [wn xi]=damp(sysG);
34 wn=wn(2);//natural frequency (wn)
```

```
35 xi=xi(2);//damping factor
36 disp(wn,xi,"Damping factor and natural frequency (
     rad)...
37  of the response are:")
38
39 tr=1.8/wn; //rise time
40 disp(tr,"Rise time (sec) of the response is:")
41
42 sigma=xi*wn
43 ts=4.6/sigma; //settling time
44 disp(ts,"Settling time (sec) of the response is:")
45
46 Mp=exp(-xi*%pi/sqrt(1-xi^2))
47 wd=wn*sqrt(1-xi^2);
48 tp=%pi/wd;
49 disp(tp, Mp,"Overshoot and time of overshoot (sec)
     ...
50  in the response are:")
51
52 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 3.29 `Stability versus parameter range`

```
1 //Example 3.29
2 //Stability versus parameter range
3
4 xdel(winsid())//close all graphics Windows
5 clear;
```

```
6  clc;
7  //
     ____________________________________________________________________

8  //Stability versus parameter range
9
10 numT=[-1];//zeros
11 denT=[1 0 -6];//poles
12 Ns=poly(numT,'s','roots');
13 Ds=poly(denT,'s','roots');
14 Gfs=syslin('c',Ns/Ds); //forward transfer function
     block
15
16 num=[1];
17 den=[1 0];
18 Ns=poly(num,'s','coeff');
19 Ds=poly(den,'s','coeff');
20 Hs=syslin('c',Ns/Ds); //feedback transfer function
     block
21
22 //check the step responses with the forward path
     gain K=7.5, 13, 25
23 t=0:0.02:12;
24 i=1;
25
26 for K=[7.5, 13, 25]
27     sysT= (K * Gfs) /. Hs;
28     yt(i,:)=csim('step',t,sysT);
29     i=i+1;
30 end
31 //Step response
32 plot(t',yt')
33 exec .\fig_settings.sci; //custom script for setting
     figure properties
34 title("Transient response for different values of K"
     ,'fontsize',3);
35 xlabel('Time t (sec.)','fontsize',2)
36 ylabel('y(t)','fontsize',2)
```

```
37 h=legend('K=7.5','K=13', 'K=25')
38 h.legend_location = "in_upper_right"
39 h.fill_mode='off'
40 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 3.30 `Stability versus two parameter ranges`

```
1 //Example 3.30
2 //Stability versus two parameter ranges
3 xdel(winsid())//close all graphics Windows
4 clear;
5 clc;
6 //
```

```
7 //Stability versus parameter ranges
8
9 num=[1 0];//zeros
10 den=[-1 -2];//poles
11 Ns=poly(num,'s','coeff');
12 Ds=poly(den,'s','roots');
13 Gfs=syslin('c',Ns/Ds); //forward transfer function
    block
14
15 num=[1];
16 den=[1 0];
17 Ns=poly(num,'s','coeff');
18 Ds=poly(den,'s','coeff');
```

```
19 Hs=syslin('c',Ns/Ds); //feedback transfer function
      block
20
21 //check the step responses with the forward, path
      gain K=7.5, 13, 25
22 t=0:0.02:12;
23 i=1;
24 num=[5 10;1 1;0 1];
25
26 for i=1:3
27     den=[0 1];
28     Ns=poly(num(i,:),'s','coeff');
29     Ds=poly(den,'s','coeff');
30     Gcs=syslin('c',Ns/Ds); //Controller transfer
          function block
31     sysT= Gcs * Gfs /. Hs;
32     yt(i,:)=csim('step',t,sysT);
33     i=i+1;
34 end
35
36 //Transient response for different values of K and
      Ki
37 plot(t',yt')
38 exec .\fig_settings.sci; //custom script for setting
      figure properties
39 title("Transient response for the system",'fontsize'
      ,3);
40 xlabel('Time t (sec.)','fontsize',2)
41 ylabel('y(t)','fontsize',2)
42 xset("font",1,1)
43 xstring(1.4,1.05,'$K=10,K_I=5$');
44 xstring(3.3,0.8,'$K=1,K_I=1$');
45 xstring(5.5,0.35,'$K=1,K_I=0$')
46 //
```

Chapter 4

Basic properties of feedback

Scilab code Exa 4.6 PID Control of DC Motor Speed

```
1 //Example 4.6
2 //PID Control of DC Motor Speed.
3
4 //
    ------------------------------------------------------------------------

5 //NOTE THAT---
6
7 //The model as given in matlab program for this
    example in the book is
8
9 //num=Ra*s + La*s^2 ;
10 //den=Ke*ki + (Ra*Ke*Ke+Ke*kp)*s + (Ra*b+Ke*Ke+Ke*kd
    )*s^2 + Jm*La*s^3;
11
12 //this does not match to the model of DC motor given
     on page 43.
13 //Also, if we assume this model, disturbance
    response given
14 //in figure 4.13 (a)
15 //is different from expected.
```

```
16 //For instance, with P control, output should
      asymptotically go to 0
17 //for disturbance step input, because numerator is s
      (Ra + La*s)
18 //and system is type 0 (no pole at origin).
19 //i.e. y(inf)=lim s->0 s*Y(s)= s*[s(Ra + La*s)/den
      ]*1/s=0;
20
21 //In following code, we have considered correct
      model of DC motor as
22 //given on page 43. Note that, this model must have
      been used
23 //by authors of the book for
24 //step reference tracking as it is correctly shown
      in figure 4.13 (b)
25
26 //
     ------------------------------------------------------------------------

27 xdel(winsid())//close all graphics Windows
28 clear;
29 clc;
30
31 //
     ------------------------------------------------------------------------

32 // System parameters
33 Jm=0.0113; // N-m-s^2/rad
34 b=0.028;   // N-m-s/rad
35 La=0.1;    // henry
36 Ra=0.45;   // ohms
37 Kt=0.067   // n-m/amp
38 Ke=0.067;  // V-sec/amp
39
40 // Controller parameters
41 kp=3;
42 ki=15;     // sec^-1
43 kd=0.3;    // sec
```

```
44
45 // DC Motor Transfer function as given on page 43 of
       book (edition 5)
46 //G=Kt/[Jm*La s^2 + (Jm*Ra + La*b)s +(Ra*b +Kt*Ke)]
47 s=%s;
48 num=[Kt];
49 den=[(Ra*b +Kt*Ke) (Jm*Ra + La*b) Jm*La];
50 Ns=poly(num,'s','coeff');
51 Ds=poly(den,'s','coeff');
52 G=syslin('c',Ns/Ds)
53
54 //PID controller, Gc=(kd s^2 + kp s + ki)/s
55 num=[ki kp kd;ki kp 0;0 kp 0]; //numerator
      parameters of controller)
56                                    //(row wise for PID
                                        , PI and P)
57 den=[0 1];                        //denominator
      parameters of controller
58 Ds=poly(den,'s','coeff');     //denominator
      polynomial of controller
59 t=0:0.005:10;                  // Simulation time
60 //
     ------------------------------------------------------------------

61 //Step disturbance response with P, PI and PID
      controller.
62
63 for i=1:3
64 Ns=poly(num(i,:),'s','coeff');//numerator polynomial
       of controller
65 sysG=syslin('c',Ns/Ds);
66 sysD=G/. sysG;
67 v(i,:)=csim('step',t,sysD);
68 end
69 plot(t',v');
70 //Title, labels and grid to the figure
71 exec .\fig_settings.sci; //custom script to set the
      figure properties
```

```
72 title('Responses of P,PI and PID control to step
      disturbance...
73  input','fontsize',3)
74 xlabel('Time t (sec.)','fontsize',2)
75 ylabel('Amplitude','fontsize',2)
76 hl=legend(['PID','PI','P']);
77
78 //
   ------------------------------------------------------------------------

79 //Reference step response
80
81 figure
82 for i=1:3
83 Ns=poly(num(i,:),'s','coeff');
84 Gc=syslin('c',Ns/Ds);
85 // Step reference response with P, PI and PID
      controller.
86 sysR=G*Gc/(1+G*Gc);
87 v(i,:)=csim('step',t,sysR);
88 end
89 plot(t',v')
90 //Title, labels and grid to the figure
91 exec .\fig_settings.sci; //custom script to set the
      figure properties
92 title('Responses of PID control to step reference
      input','fontsize',3)
93 xlabel('Time t (sec.)','fontsize',2)
94 ylabel('Amplitude','fontsize',2)
95 hl=legend(['PID','PI','P']);
96
97 //
   ------------------------------------------------------------------------
```

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 4.7 `Discrete Equivalent`

```
1 //Example 4.7
2 //Discrete Equivalent.
3 //
  ------------------------------------------------------------------------

4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7
8 // Transfer function
9 s=%s;
10 num=[1 11];
11 den=[1 3]
12 Us=poly(num,'s','coeff');
13 Es=poly(den,'s','coeff');
14 Ds=syslin('c',Us/Es);
15 sysc=tf2ss(Ds)
16
17 //Discretize the system using sampling time Ts=1 and
       Bilinear Transform
18 Ts=1;
19 sysd=cls2dls(sysc,Ts);
20
21 //Pulse transfer function
22 Dd=ss2tf(sysd)
```

```
23 disp(Dd,"Dd=")
24 disp("Note that, multiply numerator and denomintor
     each by 7...
25  will give the result as in book.")
26 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 4.8 Equivalent discrete controller for DC motor speed control

```
1 //Example 4.8
2 //Equivalent discrete controller for DC motor speed
     control.
3 //

4 //NOTE THAT— The system response (continuous) to
     sampled control
5 //input depends on
6 //the sampling time set for continuous signal in
     SIMULATION.
7 //In this example we consider sampling period of
     0.009 sec
8 //to represent continuous time signal.
9 //

10
11 xdel(winsid())//close all graphics Windows
12 clear;
13 clc;
```

```
14 //
   ____________________________________________________________________

15 // Continuous time system and controller
16 // System transfer function
17 s=%s;
18 num=[45 0];
19 den=[45 14 1]
20 Nms=poly(num,'s','coeff');
21 Dns=poly(den,'s','coeff');
22 Gp=syslin('c',Nms/Dns); //system transfer function
23
24 // Controller
25
26 numDa=[6 1];
27 denDa=[0 1]
28 Nms=poly(numDa,'s','coeff');
29 Dns=poly(denDa,'s','coeff');
30 sysD=syslin('c',1.4*Nms/Dns); //controller transfer
      function
31
32 //Closed loop responses
33
34 num=[1 0];
35 den=[1 0];
36 Nms=poly(num,'s','coeff');
37 Dns=poly(den,'s','coeff');
38 H=syslin('c',Nms/Dns)
39
40 sysDa=Gp*sysD/.H;
41
42 //step response and control input
43 t=0:0.009:5;
44 yt=csim('step',t,sysDa); //step response
45 figure(0)
46 plot2d(t,yt,1)
47 Gu=sysD/(1+Gp*sysD);
48 ut=csim('step',t,Gu); //control input
```

```
49 figure(1)
50 plot2d(t,ut,1)
51 //
   ------------------------------------------------------------------------


52
53 sys=tf2ss(Gp); //state space model of the system
54 con=tf2ss(sysD); //controller state space model
55
56 // discrete-time time system and controller
57
58 //Discretize the system and control with sampling
      time Ts=0.07
59 // using Bilinear Transform
60 Ts=0.07;
61 sysDd=cls2dls(sys,Ts); // discrete-time system state
       space model
62 conDd=cls2dls(con,Ts); // discrete-time controller
      state space model
63
64 //Pulse transfer function of system
65 Gpz=ss2tf(sysDd);
66 //Pulse transfer function of controller
67 Gcz=ss2tf(conDd);
68 //Closed loop response
69 Gz=Gpz*Gcz/(1+Gpz*Gcz)
70 //Control input pulse transfer function
71 Guz=Gcz/(1+Gpz*Gcz)
72 T=0:Ts:5;
73 r=ones(1,length(T));
74 yd=flts(r,Gz);.....//Discrete respnse to
      discrete input
75 ud=flts(r,Guz);           //Discrete Control input
76 //continuous response for digital input
77 t=0:0.009:5;
78 k=0;
79
80 for i=1:length(yd)
```

```
81      for j=1:8
82          if (k+j)>length(t) then
83              break
84              else
85              YD(1,k+j)=yd(i);
86          end
87          end
88      k=k+j;
89  end
90
91  yt=csim(1-YD,t,Gp*sysD);
92  scf(0)
93  plot2d(t,yt,5);
94  scf(1)
95  plot2d2(T,ud,5);
96  //
    ____________________________________________________________________________

97  //Discretize the system and control with sampling
       time Ts=0.035
98  // using Bilinear Transform
99  Ts=0.035;
100 sysDd=cls2dls(sys,Ts); // discrete-time system state
        space model
101 conDd=cls2dls(con,Ts); // discrete-time controller
       state space model
102
103 Gpz=ss2tf(sysDd);       //Pulse transfer function of
       system
104 Gcz=ss2tf(conDd);      //Pulse transfer function of
       controller
105
106 //Closed loop response
107 Gz=Gpz*Gcz/(1+Gpz*Gcz)
108 //Control input pulse transfer function
109 Guz=Gcz/(1+Gpz*Gcz)
110 T=0:Ts:5;
111 r=ones(1,length(T));
```

```
112 yd=flts(r,Gz);.....//Discrete respnse to
      discrete input
113 ud=flts(r,Guz);          //Discrete Control input
114 t=0:0.009:5;
115 k=0;
116
117 for i=1:length(yd)
118     for j=1:4
119        if (k+j)>length(t) then
120           break
121           else
122           YD(1,k+j)=yd(i);
123        end
124        end
125     k=k+j;
126 end
127
128 yt=csim(1-YD,t,Gp*sysD);
129 scf(0)
130 plot2d(t,yt,2);
131 scf(1)
132 plot2d2(T,ud,2);
133
134 scf(0)
135 //Title, labels and grid to the figure
136 exec .\fig_settings.sci; //custom script to set the
      figure properties
137 title('Comparision plots of Speed-control system
      with continuous...
138  and discrete controllers','fontsize',3)
139 xlabel('Time t (sec.)','fontsize',2)
140 hl=legend(['Continuous time','Discrete-time, Ts=0.07
       s'...
141 ,'Discrete-time, Ts=0.035 s'],4);
142 scf(1)
143 //Title, labels and grid to the figure
144 exec .\fig_settings.sci; //custom script to set the
      figure properties
```

```
145 title('Comparision plots of Speed-control system
      with continuous...
146  and discrete controllers','fontsize',3)
147 xlabel('Time t (sec.)','fontsize',2)
148 hl=legend(['Continuous time','Discrete-time, Ts=0.07
      s',...
149 'Discrete-time, Ts=0.035 s']);
150 //
```

Chapter 5

The Root Locus Design method

check Appendix AP 1 for dependency:

fig_settings.sci

Scilab code Exa 5.1 Root locus of a Motor Position Control

```
1 //Example 5.1
2 //Root locus of a Motor Position Control.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7
8 //
   ------------------------------------------------------------------------
9
10 //System transfer function and its root locus
11
12 s=poly(0,'s');
13 Ls=1/(s*(s+1));
14
15 //Title, labels and grid to the figure
```

```
16 exec .\fig_settings.sci; //custom script for setting
       figure properties
17 evans(Ls)
18 title(['Root locus for', '$L(s)=1/[s(s+1)]$'],'
       fontsize',3)
19 zoom_rect([-2 -1.5 2 1.5])
20 sgrid([0.5],1,5)
21 xset("font",1,1.5)
22 xstring(-1.2,1.1,'$\theta=sin^{-1} \xi$",0,0)
23 h=legend('');
24 h.visible = "off"
25
26 //-------------------------------------------------------------------------
   ------------------------------------------------------------------
```

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 5.2 `Root locus with respect to a plant open loop pole`

```
1 //Example 5.2
2 //Root locus with respect to a plant open loop pole.
3 xdel(winsid())//close all graphics Windows
4 clear;
5 clc;
6
7 //
   -------------------------------------------------------------------------
   ------------------------------------------------------------------
8 //System transfer function and its root locus
9 s=poly(0,'s');
10 Gs=s/(s*s+1);
11
```

```
12 //Title, labels and grid to the figure
13 exec .\fig_settings.sci; //custom script for setting
       figure properties
14 evans(Gs,100)
15 title(['Root locus vs. damping factor','$c$',...
16 'for','$1+G(s)=1+1/[s(s+c)]=0$'],'fontsize',3)
17 zoom_rect([-2 -1.5 2 1.5])
18 h=legend('');
19 h.visible = "off"
20
21 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 5.3 `Root locus for satellite attitude control with PD control`

```
1 //Example 5.3
2 //Root locus for satellite attitude control with PD
     control.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7
8 //
```

```
9 //System transfer function and its root locus
10
11 s=poly(0,'s');
```

```
12 sysS=(s+1)/(s^2);
13 evans(sysS,100)
14
15 //Title, labels and grid to the figure
16 exec .\fig_settings.sci; // custom script for
      setting figure properties
17 title(['Root locus for','$L(s)=G(s)=(s+1)/s^2$'],'
      fontsize',3)
18 zoom_rect([-6 -3 2 3])
19 h=legend('');
20 h.visible = "off"
21
22 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 5.4 Root locus for satellite attitude control with modified PD con

```
1 //Example 5.4
2 //Root locus for satellite attitude control with
      modified
3 //PD control or Lead compensator.
4
5 xdel(winsid())//close all graphics Windows
6 clear;
7 clc;
8
9 //
```

```
10 //System transfer function and its root locus
11
12 s=poly(0,'s');
13 sysL=(s+1)/(s^2*(s+12));
14 evans(sysL,100)
15
16 //Title, labels and grid to the figure
17 exec .\fig_settings.sci; // custom script for
      setting figure properties
18 title(['Root locus for', '$L(s)=(s+1)/s^2(s+12)$'],'
      fontsize',3)
19 zoom_rect([-6 -3 2 3])
20 h=legend('');
21 h.visible = "off"
22
23 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 5.5 `Root locus for satellite control with Lead compensator`

```
1 //Example 5.5
2 //Root locus for satellite control with Lead
      compensator.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7
```

```
8 //
   ____________________________________________________________________

9 //System transfer function and its root locus
10
11 s=poly(0,'s');
12 sysL=(s+1)/(s^2*(s+4));
13 evans(sysL)
14
15 //Title, labels and grid to the figure
16 exec .\fig_settings.sci; // custom script for
      setting figure properties
17 title(['Root locus for', '$L(s)=(s+1)/s^2(s+4)$'],'
      fontsize',3)
18 zoom_rect([-6 -3 2 3])
19 h=legend('');
20 h.visible = "off"
21
22 //
   ____________________________________________________________________
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 5.6 `Root locus for satellite attitude control with a Transition va`

```
1 //Example 5.6
2 //Root locus for satellite attitude control with a
3 //Transition value for the pole.
4
5 xdel(winsid())//close all graphics Windows
6 clear;
```

```
clc;

//
    ------------------------------------------------------------------------

//System transfer function and its root locus

s=poly(0,'s');
sysL=(s+1)/(s^2*(s+9));
evans(sysL)

//Title, labels and grid to the figure
exec .\fig_settings.sci; // custom script for
    setting figure properties
title(['Root locus for', '$L(s)=(s+1)/(s^2(s+9))$'],
    'fontsize',3)
zoom_rect([-6 -3 2 3])
h=legend('');
h.visible = "off"

//
    ------------------------------------------------------------------------
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 5.7 `Root locus for satellite control with a Collocated Flexibility`

```
//Example 5.7
//Root locus for satellite control with a Collocated
    Flexibility.

```

```
4  xdel(winsid())//close all graphics Windows
5  clear;
6  clc;
7
8  //
     ---------------------------------------------------------------------------

9  //System transfer function with controller.
10
11 s=poly(0,'s');
12 NumD=(s+1);
13 DenD=(s+12);
14 D=NumD/DenD;
15
16 NumG=(s+0.1)^2+36
17 DenG=s^2*((s+0.1)^2+(6.6)^2)
18
19 G=NumG/DenG;
20
21 NumL=NumD*NumG;
22 DenL=DenD*DenG;
23
24 L=NumL/DenL;
25
26 zr=roots(NumL);
27 pl=roots(DenL);
28
29 //
     ---------------------------------------------------------------------------

30 //Angle of departure.
31 //Find angle of departure from pole at phi1=-0.1+
       6.6i
32 //(real poles don't have angle of departure,
33 //they move along real axis only)
34 //psi1=angle[(Departing pole)-(zero at -0.1+6.6i
     )]
35 [Mpsi1, psi1] = polar(pl(2)-zr(1))
```

```
36 psi1=real(psi1)*180/%pi;                    //angle in degree
37
38 //psi2=angle[(Departing pole)- (zero at - 0.1 - 6.6i
      )]
39 [Mpsi2, psi2] = polar(pl(2)-zr(2))
40 psi2=real(psi2)*180/%pi;                    //angle in degree
41
42 //psi3=angle[(Departing pole)- (zero at - 1)]
43 [Mpsi3, psi3] = polar(pl(2)-zr(3))
44 psi3=real(psi3)*180/%pi;                    //angle in degree
45
46 //phi2=angle[(Departing pole)- (pole at 0)]
47 [Mphi2, phi2] = polar(pl(2)-pl(4))
48 phi2=real(phi2)*180/%pi;                    //angle in degree
49
50 //phi3 is same as phi2, as pole is repeated at 0.
51 phi3=phi2;
52
53 //phi4=angle[(Departing pole)-(pole at - 0.1 - 6.6i
      )]
54 [Mphi4, phi4] = polar(pl(2)-pl(3))
55 phi4=real(phi4)*180/%pi;                    //angle in degree
56
57 //phi5=angle[(Departing pole)- (pole at - 12 )]
58 [Mphi5, phi5] = polar(pl(2)-pl(1))
59 phi5=real(phi5)*180/%pi;                    //angle in degree
60
61 //Therefore angle of departure phi1 at - 0.1 + 6.6i
      is
62 //phi1 = 180 + sum(angle to zeros) - sum(angle to
      poles)
63
64 phi1 = 180 + sum(psi1+psi2+psi3) - sum(phi2+phi3+
      phi4+phi5)
65
66 //angle contributions in figure
67 figure(0)
68 plzr(L)
```

```
xset('font size',1.5)
xarrows([real(pl(1));real(pl(2))],[imag(pl(1));imag(
   pl(2))],0,2)
xarc(-13,1,2,2,0,phi5*64)
xstring(-11,0.05,"$\phi_5$")


xarrows([real(zr(3));real(pl(2))],[imag(zr(3));imag(
   pl(2))],0,4)
xarc(-2,1,2,2,0,psi3*64)
xstring(-0.7,1,"$\psi_3$")

xarrows([real(pl(4));real(pl(2))],[imag(pl(4));imag(
   pl(2))],0,5)
xarc(-1,1,2,2,0,phi2*64)
xstring(0.8,0.5,"$\phi_2,\,\phi_3$")


xarrows([real(pl(3));real(pl(2))],[imag(pl(3));imag(
   pl(2))],0,3)
xarc(-1,-6.6,2,2,0,phi4*64)
xstring(0.8,-7,"$\phi_4$")

xarrows([real(zr(2));real(pl(2))],[imag(zr(2));imag(
   pl(2))],0,6)
xarc(-1,-5,2,2,0,psi2*64)
xstring(0.8,-5.5,"$\psi_2$")

xarrows([real(zr(1));real(pl(2))],[imag(zr(1));imag(
   pl(2))],0,24)
xstring(0.3,5.5,"$\psi_1$")
xstring(0.3,6.5,"$\phi_1$")

exec .\fig_settings.sci; //custom script for setting
    figure properties
title(['Figure for computing a departure angle for'
   ,...
'$L(s)=\frac{s+1}{s+12}\frac{(s+0.1)^2+6^2}{s^2[(s
```

```
    +0.1)^2+6.6^2]}$'],...
99  'fontsize',3)
100 zoom_rect([-15 -8 5 8])
101 h=legend('');
102 h.visible = "off"
103
104 //
    ------------------------------------------------------------------------

105 //Root locus system transfer function with
    controller.
106 figure(1)
107 evans(L)
108 //Title, labels and grid to the figure
109 exec .\fig_settings.sci; //custom script for setting
    figure properties
110 title(['Root locus for','$L(s)=\frac{s+1}{s+12}\frac
    {(s+0.1)^2+6^2}...
111 {s^2[(s+0.1)^2+6.6^2]}$'],'fontsize',3)
112 zoom_rect([-15 -8 5 8])
113 h=legend('');
114 h.visible = "off"
115
116 //
    ------------------------------------------------------------------------
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 5.8 `Root locus for noncollocated case`

```
1  //Example 5.8
2  //Root locus for noncollocated case.
3
4  xdel(winsid())//close all graphics Windows
5  clear;
6  clc;
7
8  //
     ---------------------------------------------------------------------------

9  //System transfer function with controller
10
11 s=poly(0,'s');
12 NumD=(s+1);
13 DenD=(s+12);
14 sysD=NumD/DenD;
15
16 NumG=1
17 DenG=s^2*((s+0.1)^2+(6.6)^2)
18
19 sysG=NumG/DenG;
20
21 NumL=NumD*NumG;
22 DenL=DenD*DenG;
23
24 sysL=NumL/DenL;
25
26 zr=roots(NumL);
27 pl=roots(DenL);
28
29 //
     ---------------------------------------------------------------------------

30 //Angle of departure.
31 //Find angle of departure from pole at phi1=-0.1+
       6.6i
32 //(real poles don't have angle of departure,
33 //they move along real axis only)
```

```
34
35 //psi1=angle[(Departing pole)- (zero at - 1]
36 [Mpsi1, psi1] = polar(pl(2)-zr(1))
37 psi1=real(psi1)*180/%pi;            //angle in degree
38
39 //phi2=angle[(Departing pole)- (pole at 0)]
40 [Mphi2, phi2] = polar(pl(2)-pl(4))
41 phi2=real(phi2)*180/%pi;            //angle in degree
42
43 //phi3 is same as phi2, as pole is repeated at 0.
44 phi3=phi2;
45
46 //phi4=angle[(Departing pole)-(pole at - 0.1 - 6.6i
      )]
47 [Mphi4, phi4] = polar(pl(2)-pl(3))
48 phi4=real(phi4)*180/%pi;            //angle in degree
49
50 //phi5=angle[(Departing pole)- (pole at - 12 )]
51 [Mphi5, phi5] = polar(pl(2)-pl(1))
52 phi5=real(phi5)*180/%pi;            //angle in degree
53
54 //Therefore angle of departure phi1 at - 0.1 + 6.6i
      is
55 //phi1 = 180 + sum(angle to zeros) - sum(angle to
      poles)
56
57 phi1 = 180 + sum(psi1) - sum(phi2+phi3+phi4+phi5)
58
59 //angle contributions in figure
60 figure(0)
61 plzr(sysL)
62 xset('font size',1.5)
63 xarrows([real(pl(1));real(pl(2))],[imag(pl(1));imag(
      pl(2))],0,2)
64 xarrows([real(pl(1)); -10],[0;0],0,2)
65 xarc(-13,1,2,2,0,phi5*64)
66 xstring(-11,0.05,"$\phi_5$")
67
```

```
68 xarrows([real(zr(1));real(pl(2))],[imag(zr(1));imag(
      pl(2))],0,6)
69 xarrows([real(zr(1)); -0.3],[0;0],0,6)
70 xarc(-2,1,2,2,0,psi1*64)
71 xstring(-0.7,1,"$\psi_1$")
72
73 xarrows([real(pl(4));real(pl(2))],[imag(pl(4));imag(
      pl(2))],0,5)
74 xarrows([real(pl(4)); 1],[0;0],0,5)
75 xarc(-1,1,2,2,0,phi2*64)
76 xstring(0.8,0.5,"$\phi_2,\,\phi_3$")
77
78 xarrows([real(pl(3));real(pl(2))],[imag(pl(3));imag(
      pl(2))],0,17)
79 xarrows([real(pl(3)); 2],[imag(pl(3));imag(pl(3))
      ],0,17)
80 xarc(-1.1,-5.6,2,2,0,phi4*64)
81 xstring(0.8,-5.5,"$\phi_4$")
82
83 xstring(0.3,6.5,"$\phi_1$")
84
85 exec .\fig_settings.sci; //custom script for setting
       figure properties
86 title(['Figure to compute a departure angle for',...
87 '$L(s)=\frac{s+1}{s+12}\frac{1}{s^2[(s+0.1)
      ^2+6.6^2]}$'],...
88 'fontsize',3)
89 zoom_rect([-15 -8 5 8])
90 h=legend('');
91 h.visible = "off"
92
93 //
      ____________________________________________________________________

94 //Root locus of system transfer function with
      controller
95 figure(1)
96 evans(sysL)
```

```
97 //Title, labels and grid to the figure
98 exec .\fig_settings.sci; //custom script for setting
       figure properties
99 title(['Root locus for','$L(s)=\frac{s+1}{s+12}\frac
       {1}...
100 {s^2[(s+0.1)^2+6.6^2]}$'],'fontsize',3)
101 zoom_rect([-15 -8 5 8])
102 h=legend('');
103 h.visible = "off"
104
105 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 5.9 `Root locus for the system having complex multiple roots`

```
1 //Example 5.9
2 //Root locus for the system having complex multiple
      roots.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
```

```
8 //System transfer function
9
```

```
10 s=poly(0,'s');
11
12 NumL=1;
13 DenL=s*(s+2)*[(s+1)^2+4];
14
15 L=NumL/DenL;
16
17 zr=roots(NumL);
18 pl=roots(DenL);
19
20 //
    ------------------------------------------------------------------------

21 //Angle of departure.
22 //Find angle of departure from pole at phi1=-1+2
     i
23 //(real poles don't have angle of departure,
24 // they move along real axis only)
25
26 //phi2=angle[(Departing pole)-(pole at 0)]
27 [Mphi1, phi1] = polar(pl(1)-pl(4))
28 phi1=real(phi1)*180/%pi;            //angle in degree
29
30 //phi2=angle[(Departing pole)-(pole at -2)]
31 [Mphi2, phi2] = polar(pl(1)-pl(3))
32 phi2=real(phi2)*180/%pi;            //angle in degree
33
34 //phi2=angle[(Departing pole)-(pole at -1-2i)]
35 [Mphi4, phi4] = polar(pl(1)-pl(2))
36 phi4=real(phi4)*180/%pi;            //angle in degree
37
38 //Therefore angle of departure phi1 at -1+2i is
39 //phi3 = 180 + sum(angle to zeros) - sum(angle to
     poles)
40
41 phi3 = 180 - sum(phi1+phi2+phi4)
42
43 //angle contributions in figure
```

```
44 figure(0)
45 plzr(L)
46 xset('font size',1.5)
47 xarrows([real(pl(4));real(pl(1))],[imag(pl(4));imag(
      pl(1))],0,2)
48 xarrows([real(pl(4)); 1],[0;0],0,2)
49 xarc(-0.5,0.5,1,1,0,phi1*64)
50 xstring(0.5,0.25,"$\phi_1$")
51
52 xarrows([real(pl(3));real(pl(1))],[imag(pl(3));imag(
      pl(1))],0,5)
53 xarrows([real(pl(3)); -1.3],[0;0],0,5)
54 xarc(-2.5,0.5,1,1,0,phi2*64)
55 xstring(-1.5,0.25,"$\phi_2$")
56
57 xarrows([real(pl(2));real(pl(1))],[imag(pl(2));imag(
      pl(1))],0,17)
58 xarrows([real(pl(2)); -0.3],[-2;-2],0,17)
59 xarc(-1.5,-1.5,1,1,0,phi4*64)
60 xstring(-0.5,-1.7,"$\phi_4$")
61
62 xstring(-0.8,2,"$\phi_1$")
63
64 exec .\fig_settings.sci; //custom script for setting
       figure properties
65 title(['Figure to computing a departure angle for'
      ,...
66 '$L(s)=\frac{1}{s(s+2)[(s+1)^2+4]}$'],'fontsize',3)
67 zoom_rect([-4 -3 4 3])
68 h=legend('');
69 h.visible = "off"
70 //
   ____________________________________________________________

71 //Root locus of system transfer function with
      controller
72 figure(1)
73 evans(L)
```

```
74 //Title, labels and grid to the figure
75 exec .\fig_settings.sci; // custom script for
      setting figure properties
76 title(['Root locus for','$L(s)=\frac{1}{s(s+2)[(s+1)
      ^2+4]}$'],...
77 ,'fontsize',3)
78 zoom_rect([-4 -3 4 3])
79 h=legend('');
80 h.visible = "off"
81 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 5.10 Design using Lead compensator

```
1 //Example 5.10
2 //Design using Lead compensator.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
```

```
8 //System transfer function and its root locus
9
10 s=poly(0,'s');
11
```

```
12 NumG=1;
13 DenG=s*(s+1);
14 NumD=(s+2);
15 DenD=(s+10);
16
17 G=NumG/DenG;
18 D=NumD/DenD;
19
20 L=G*D; //open loop transfer function
21
22 figure(0)
23 evans(L)
24 sgrid(0.5,7,6);
25
26 xstring(-2,4,"Damping=0.5",0,0)
27 xstring(-7,4,"w=7",0,0)
28 //Title, labels and grid to the figure
29 exec .\fig_settings.sci; // custom script for
      setting figure properties
30 title('Root locus for lead design','fontsize',3)
31 zoom_rect([-14 -8 4 8])
32 h=legend('');
33 h.visible = "off"
34 //
     ------------------------------------------------------------------------


35 // Unit step response
36 //closed loop system
37 K=70;
38 sysc=K*L/(1+K*L);
39 sysc=syslin('c',sysc);
40 t=linspace(0,10,1000);
41 y=csim('step',t,sysc);
42 figure(1)
43 plot(t,y);
44 title('Step response for the system with lead
      compensator','fontsize',3)
45 xlabel('Time (sec)','fontsize',2)
```

```
46 ylabel('Amplitude','fontsize',2)
47 set(gca(),"grid",[0.3 0.3])
48 zoom_rect([0 0 1.8 1.4])
49 exec .\fig_settings.sci;
50
51 scf(0)
52 pl=roots(DenG*DenD+K*NumG*NumD)      //closed loop
     poles at K=70;
53 plot(real(pl),imag(pl),'ro')         //closed loop
     pole-locations at K=70;
54 xstring(-5.8,6,"K=70",0,0)
55 //
```

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 5.11 A second Lead compensation Design

```
1 //Example 5.11
2 //A second  Lead compensation Design.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
```

```
8 //System transfer function and its root locus
9
```

```
10 s=poly(0,'s');
11
12 NumG=1;
13 DenG=s*(s+1);
14 NumD=(s+5.4);
15 DenD=(s+20);
16
17 Gs=NumG/DenG;
18 Ds=NumD/DenD;
19
20 Ls=Gs*Ds;    //open loop transfer function
21
22 zr=roots(NumD*NumG); //open loop system zeros
23 pl=roots(DenD*DenG); //open loop system poles
24 pd=-3.5+3.5*sqrt(3)*%i; //desired pole
25
26 //Construction for placing a specific point on the
      root locus.
27 figure(0)
28 plzr(Ls)
29 plot(real(pd),imag(pd),'ro')
30 xarrows([real(pl(1));real(pd)],[imag(pl(1));imag(pd)
      ],0,2)
31 xarrows([real(pl(2));real(pd)],[imag(pl(2));imag(pd)
      ],0,2)
32 xarrows([real(pl(3));real(pd)],[imag(pl(3));imag(pd)
      ],0,2)
33 xarrows([real(zr);real(pd)],[imag(zr);imag(pd)],0,6)
34 xarrows([real(zr);-3],[0;0],0,6)
35 xarc(-6.4,1,2,2,0,72.6*64)
36 xset('font size',1.5);
37 xstring(-4.7,0.5,"$\psi$")
38 exec .\fig_settings.sci; //custom script for setting
      figure properties
39 title('Construction for placing a specific point on
      the root locus',...
40 'fontsize',3)
41 h=legend('');
```

```
42 h.visible = "off"
43 //
   ------------------------------------------------------------------------

44 //Root locus of system transfer function with
      controller
45 figure(1)
46 evans(Ls)
47 sgrid(0.5,7,6)
48 //Title, labels and grid to the figure
49 exec .\fig_settings.sci; //custom script for setting
       figure properties
50 title(['Root locus for','$ L(s)=\frac {s+5.4}{s(s+1)
      (s+20)}$'],...
51 'fontsize',3)
52 zoom_rect([-20 -8 5 8])
53 h=legend('');
54 h.visible = "off"
55 //
   ------------------------------------------------------------------------

56 // Unit step response
57 //closed loop system
58
59 K=127;  // from root locus gain  is computed
60 sysc=K*Ls/(1+K*Ls)
61 sysc=syslin('c',sysc);
62 t=linspace(0,10,1000);
63 y=csim('step',t,sysc);
64 figure(2)
65 plot(t,y);
66 exec .\fig_settings.sci; //custom script for setting
       figure properties
67 title(['Step response for K=127', 'and',...
68 '$ L(s)=\frac {s+5.4}{s(s+1)(s+20)}$']...
69 ,'fontsize',3)
70 xlabel('Time (sec)','fontsize',2)
71 ylabel('Amplitude','fontsize',2)
```

```
72 zoom_rect([0 0 1.8 1.4])
73
74 pl=roots(DenG*DenD+K*NumG*NumD) //closed loop poles
     at K=127;
75 scf(1)
76 plot(real(pl),imag(pl),'ro') //closed loop pole-
     locations at K=127;
77 //
```

---

check Appendix AP 1 for dependency:

```
fig_settings.sci
```

Scilab code Exa 5.12 Negative root Locus for an Airplane

```
1 //Example 5.12
2 //Negative root Locus for an Airplane.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7
8
9 //
```

```
10 //System transfer function and its root locus
11
12 s=poly(0,'s');
13 Ls=-(s-6)/(s*(s^2+4*s+13));
```

```
14 evans(Ls)
15
16 //Title, labels and grid to the figure
17 exec .\fig_settings.sci; //custom script for setting
       figure properties
18 title(['Negative root locus for','$L(s)=\frac{s-6}{s
     (s^2+4s+13)}$'],...
19 'fontsize',3)
20 zoom_rect([-5 -6 10 6])
21 h=legend('');
22 h.visible = "off"
23
24 //
```

Chapter 6

The Frequency Response Design Method

Scilab code Exa 6.2.b Frequency response characteristics of Lead compensator

```
1 //Example 6.2
2 //Frequency response characteristics of Lead
     compensator.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
     ____________________________________________________________________


8 //System transfer function and its bode plot
9 K=1, T=1, alpha=0.1
10 s=poly(0,'s');
11 sysD=syslin('c',K*(T*s+1)/(alpha*T*s+1));
12
13 //The bode plot of the system
14
15 fmin=0.1/2/%pi; //mininmum frq. in Hz for response
     (0.1 rad/sec)
```

```
16 fmax=100/2/%pi; //maximum frq. in Hz for response
     (100 read/sec)
17 //
     ------------------------------------------------------------------------

18 //Bode plot for frequency in Hz (scilab ver. 5.4.1)
19 //bode(g,fmin,fmax);
20 //OR
21 //Bode plot for frequency in rad/sec (scilab ver.
     5.5.1)
22 bode(sysD,fmin,fmax,"rad")
23
24 //
     ------------------------------------------------------------------------

25 title('(a) Magnitude and  (b) phase for the lead
     compensator',...
26 'fontsize',3)
27 exec .\fig_settings.sci; //custom script for setting
      figure properties
28 //
     ------------------------------------------------------------------------
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 6.3 `Bode Plot for Real Poles and Zeros`

```
1  //Example 6.3
2  //Bode Plot for Real Poles and Zeros.
3
4  xdel(winsid())//close all graphics Windows
5  clear;
6  clc;
7  //
     ------------------------------------------------------------------------

8  //System transfer function and its bode plot
9  K=2000;
10 s=poly(0,'s');
11 Gs=syslin('c',(K*(s+0.5))/(s*(s+10)*(s+50)));
12
13 //The bode plot of the system
14 wmin=0.1;          // mininmum frq. in rad/sec for
     response
15 wmax=100;          // maximum frq. in red/sec for
     response
16 fmin=wmin/2/%pi    // mininmum frq. in Hz for
     response
17 fmax=wmax/2/%pi    // maximum frq. in Hz for response
18 //
     ------------------------------------------------------------------------

19 //Bode plot for frequency in Hz (scilab ver. 5.4.1)
20 //bode(g,fmin,fmax)
21 //OR
22 //(Only for scilab ver. 5.5.1)
23 //Bode (frequency scale in rad/sec)
24 // or gainplot or phaseplot plot with asymptotes
25 figure(0)
26 gainplot(Gs,fmin,fmax);
27 bode_asymp(Gs,wmin,wmax);
28 xstring(0.03,22,"slope=-1(-20db/dec)",0,0);
29 xstring(0.2,9,"slope=0",0,0);
30 xstring(3,7,"slope=-1(-20db/dec)",0,0)
31 xstring(0.9,-8,"slope=-2(-40db/dec)",0,0)
```

```
32 title('Composit plots (a) magnitude plot','fontsize'
     ,3);
33 h=legend('');
34 exec .\fig_settings.sci; //custom script for setting
     figure properties
35 h.visible = "off"
36 //
   ------------------------------------------------------------------

37
38 //phase plot for poles and zeros
39 zr=((s/0.5)+1)/s //infact this is zero and pole at
     origin.
40 zr=syslin('c', zr);
41 pl1=1/((s/10)+1)
42 pl1=syslin('c', pl1);
43 pl2=1/((s/50)+1)
44 pl2=syslin('c', pl2);
45 figure(1)
46 phaseplot([Gs;zr;pl1;pl2],fmin,fmax);
47 xstring(5.5,-14,"$\frac {1}{s/0.5+1}$",0,0);
48 xstring(2.8,-22,"$\frac{1}{s/50+1}$",0,0);
49 xstring(2.5,-60,"$\frac{1}{s/10+1}$",0,0);
50 xstring(1.2,-100,["Composite";"(Actual)"],0,0);
51 title('Composit plots (b) Phase','fontsize',3);
52 exec .\fig_settings.sci; //custom script for setting
     figure properties
53
54 //
   ------------------------------------------------------------------

55 figure(2)
56 bode(Gs,fmin,fmax,"rad"); //frequency scale n
     radians
57 bode_asymp(Gs,wmin,wmax);
58 exec .\fig_settings.sci; //custom script for setting
     figure properties
59 title('(c) magnitude plot and phase plot approximate
```

```
    and actual...
','fontsize',3)
xstring(2.8,-22,"$\frac{1}{s/50+1}$",0,0);
h=legend('');
h.visible = "off"

//
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 6.4 Bode Plot with Complex Poles

```
//Example 6.4
//Bode Plot with Complex Poles.

xdel(winsid())//close all graphics Windows
clear;
clc;
//
```

```
//System transfer function and its bode plot
K=10;
s=poly(0,'s');
Gs=syslin('c',(K)/(s*(s^2+0.4*s+4)));
//The bode plot of the system

```

```
14 fmin=0.1/2/%pi; //mininmum frq. in Hz for response
      (0.1 rad/sec)
15 fmax=10/2/%pi; //maximum frq. in Hz for response
      (100 read/sec)
16 //
      ------------------------------------------------------------------------

17 //Bode plot for frequency in Hz (scilab ver. 5.4.1)
18 //bode(g,fmin,fmax);
19 //OR
20 //Bode plot for frequency in rad/sec (scilab ver.
      5.5.1)
21 bode(Gs,fmin,fmax,0.01,"rad")
22
23 //
      ------------------------------------------------------------------------

24 title(['Bode plot for a transfer function with
      complex poles';...
25  '(a) magnitude...
26  (b) phase'],'fontsize',3)
27
28 //
      ------------------------------------------------------------------------
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 6.6 `Bode Plot for Complex Poles and Zeros`

```
1 //Example 6.6
2 //Bode Plot for Complex Poles and Zeros:
```

```
3  //Satellite with Flexible appendages.
4
5  xdel(winsid())//close all graphics Windows
6  clear;
7  clc;
8  //
   ------------------------------------------------------------------------

9  //System transfer function and its bode plot
10 K=0.01;
11 s=poly(0,'s');
12 NumG=K*(s^2+0.01*s+1);
13 DenG=s^2*((s^2/4)+0.02*(s/2)+1)
14 sysG=syslin('c',NumG/DenG);
15
16 fmin=0.09/2/%pi; //mininmum frq. in Hz for response
      (0.1 rad/sec)
17 fmax=11/2/%pi; //maximum frq. in Hz for response
      (100 read/sec)
18 //
   ------------------------------------------------------------------------

19 //Bode plot for frequency in Hz (scilab ver. 5.4.1)
20 //bode(g,fmin,fmax);
21 //OR
22 //Bode plot for frequency in rad/sec (scilab ver.
      5.5.1)
23 bode(sysG,fmin,fmax,0.01,"rad")
24
25 //
   ------------------------------------------------------------------------

26 title(["Bode plot for a transfer function with
      complex...
27 poles and zeros"; "(a) magnitude (b) phase"],'
      fontsize',3)
28 //
   ------------------------------------------------------------------------
```

```
29
30 disp('NOTE : Result of the above example can be
       verified by checking the figures shown in example
       6.5')
```

Scilab code Exa 6.7 Computation of Kv

```
1 //Example 6.7
2 //Computation of velocity error constant Kv from
      Bode plot
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
    ------------------------------------------------------------------------

8 //System transfer function and its bode plot
9 K=10;
10 s=poly(0,'s');
11 Gs=syslin('c',(K)/(s*(s+1)));
12 //The bode plot of the system
13
14 fmin=0.01/2/%pi; //mininmum frq. in Hz for response
      (0.1 rad/sec)
15 fmax=10/2/%pi; //maximum frq. in Hz for response
      (100 read/sec)
16 //
    ------------------------------------------------------------------------
```

```
17 //Bode plot for frequency in Hz (scilab ver. 5.4.1)
18 //bode(g,fmin,fmax);
19 //OR
20 //Bode plot for frequency in rad/sec (scilab ver.
     5.5.1)
21 bode(Gs,fmin,fmax,0.01,"rad")
22 title(['Determination of Kv from the Bode plot for
     the system',...
23 '$10/[s(s+1)]$'],'fontsize',3)
24 //choose frequency (rad) and magnitude from bode
     plot and calculate Kv
25 //Here at w=0.01, magngitude in db is M=60
26 //i.e actual magnitude of the reponse is |A|=10^(M
     /20)
27 w=0.01;        // in rad
28 M=60           // in db
29 A=10^(M/20)   //actual gain
30
31 //Velocity error constant Kv=w*|A(w)|
32 Kv=w*A;
33 disp(Kv,"The Velocity error Constant from bode plot
     is: ")
34 //
   ------------------------------------------------------------------------

35 // Computation of the Kv
36 [frq repf]=repfreq(Gs,fmin,fmax);
37 //frq in Hz, repf is freq. response in rectangular
     form.
38 //From bode plot, Kv=w*|A(w)|
39 //i.e Kv=2*pi*f*|A(2*pi*f)|
40
41 idx=1;//selecting the frequency and response at that
      frequency from arrays
42 Kv=2*%pi*frq(idx)*abs(repf(idx))
43 disp(Kv,"The Velocity error Constant is computed at
     0.0015915 Hz (0.01 rad/sec) : ")
44 //
```

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 6.8 `Nyquist plot for a second order system`

```
1 //Example 6.8
2 // Nyquist plot for a second order system.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
  ----------------------------------------------------------------------

8 //System transfer function and its root locus
9 s=poly(0,'s');
10 g=1/(s+1)^2;
11 sysG=syslin('c',g);
12
13 evans(sysG);
14 exec .\fig_settings.sci; //custom script for setting
      figure properties
15 f=gca();
16 f.x_location = "origin"
17 f.y_location = "origin"
18 title(['Root locus of','$G(s)=1/(s+1)^2$','with
     respect to K'],...
19 'fontsize',3)
20 zoom_rect([-3,-2,2,3])
21 h=legend('');
22 h.visible = "off"
```

```
23 //
   ------------------------------------------------------------------------

24 figure(1)
25 //The bode plot of the system
26 fmin=0.01/2/%pi; //mininmum frq. in Hz for response
      (0.1 rad/sec)
27 fmax=100/2/%pi; //maximum frq. in Hz for response
      (100 read/sec)
28
29 //Bode plot for frequency in Hz (scilab ver. 5.4.1)
30 //bode(g,fmin,fmax);
31 //OR
32 //Bode plot for frequency in rad/sec (scilab ver.
      5.5.1)
33 bode(sysG,fmin,fmax,0.01,"rad")
34 title(['Open loop bode plot for', '$G(s)=1/(s+1)^2$'
      ],'fontsize',3);
35 exec .\fig_settings.sci; //custom script for setting
       figure properties
36 //
   ------------------------------------------------------------------------

37
38 figure(2)
39 //The nyquist plot of the system
40 nyquist(sysG);
41 title('Nyquist plot of the evaluation of K G(s) for
      s=C1 and K=1'...
42 ,'fontsize',3);
43 exec .\fig_settings.sci; //custom script for setting
       figure properties
44 f=gca();
45 f.x_location = "origin"
46 f.y_location = "origin"
47 xset('color',2)
48 //
   ------------------------------------------------------------------------
```

check Appendix AP 1 for dependency:

fig_settings.sci

Scilab code Exa 6.9 Nyquist plot for a third order system

```
1 //Example 6.9
2 // Nyquist plot for a third order system.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
    ------------------------------------------------------------------------

8 //System transfer function
9 s=poly(0,'s');
10 g=syslin('c',1/(s*(s+1)^2));
11
12 //The bode plot of the system
13 fmin=0.01/2/%pi;
14 fmax=100/2/%pi;
15 //[frq,repf]=repfreq(g1,fmin,fmax,0.01);
16 bode(g,fmin,fmax,"rad");
17 frq=[1,10]/2/%pi;
18 [frq, repf]=repfreq(g,frq);
19 [db, phi]=dbphi(repf);
20 plot(frq*2*%pi,db,'ro');
21 exec .\fig_settings.sci; //custom script for setting
    figure properties
```

```
22 title(["Bode plot for","$G(s)=1/[s(s+1)^2]$"],'
     fontsize',3)
23 //zoom_rect([[0.1 0] -70 [12 -180] 20])
24 xset("font size", 3);
25
26 xstring(1,0,"$C\,\, (\omega=1)$",0,0);
27 xstring(2,-75,"$E\,\, (\omega=10)$",0,0);
28 f=gca();
29
30 //
     ------------------------------------------------------------------------


31 //The nyquist plot of the system
32 figure;
33 nyquist(g,0.8/2/%pi,10/2/%pi,0.02)
34
35 exec .\fig_settings.sci; //custom script for setting
     figure properties
36 title(["Nyquist plot for","$G(s)=1/[s(s+1)]^2$"],'
     fontsize',3)
37 f=gca();
38 f.x_location = "origin";
39 f.y_location = "origin";
40 zoom_rect([-1 -0.2 0.5 0.2]);
41 xset("clipping", -1.2, 0.2, 1.4,0.4);
42 xset("font size", 3);
43 xset("color",2);
44 xstring(-0.6,0.1,"${\fgcolor{blue}{\omega<0}}$",0,0)
     ;
45 xstring(-0.6,-0.1,"${\fgcolor{blue}{\omega>0}}$"
     ,0,0);
46 xstring(-0.7,0.005,"${\fgcolor{blue}{\omega=\pm 1}}$
     ",0,0);
47 xstring(-1,-0.2,...
48 "${\fgcolor{blue}{\text{From \infty at \omega=0^+}}$
     ",0,0);
49  xstring(-0.7,0.15,"${\fgcolor{blue}...
50  {\text{Towards \infty at \omega=0^-}}$",0,0);
```

```
51 xstring(-0.525,-0.04,"C",0,0);
52 xstring(-0.075,0,"E",0,0);
53 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 6.10 `Nyquist plot for an Open loop unstable system`

```
1 //Example 6.10
2 // Nyquist plot for an Open-loop unstable system.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
```

```
8 //System transfer function
9 s=poly(0,'s');
10 sysG=(s+1)/(s*(s/10-1));
11 evans(sysG,50)
12 exec .\fig_settings.sci; //custom script for setting
     figure properties
13 title(["Root Locus for","$G(s)=(s+1)/[s(s/10-1)]$"],
    'fontsize',3)
14 zoom_rect([-5 -4 5 4])
15 f=gca();
16 f.x_location = "origin"
```

```
17 f.y_location = "origin"
18 h=legend('');
19 h.visible = "off"
20
21 g1=syslin('c',(s+1)/(s*(s/10-1)));
22 //
     ---------------------------------------------------------------------------

23 //The bode plot of the system
24 figure;
25 bode(g1,0.1/2/%pi,100/2/%pi,"rad")
26 exec .\fig_settings.sci; //custom script for setting
       figure properties
27 title(["Bode plot for","$G(s)=(s+1)/[s(s/10-1)]$"],'
     fontsize',3)
28 //bode(g,2*%pi*0.1,2*%pi*100)
29 //
     ---------------------------------------------------------------------------

30 figure;
31 //The nyquist plot of the system
32 nyquist(g1,0.5/2/%pi,100/2/%pi,0.05)
33 exec .\fig_settings.sci; //custom script for setting
       figure properties
34 title(["Nyquist plot for","$G(s)=(s+1)/[s(s/10-1)]$"
     ],'fontsize',3)
35 f=gca();
36 f.x_location = "origin";
37 f.y_location = "origin";
38 zoom_rect([-2 -2 1 2]);
39 xset("color",2);
40 xset("font size", 3);
41 xstring(-1,1.5,"${\fgcolor{blue}{\omega>0}}$",0,0);
42 xstring(-1,-1.5,"${\fgcolor{blue}{\omega<0}}$",0,0);
43 xstring(-1.5,0,"${\fgcolor{blue}{\omega=\pm \sqrt
     {10}}}$",0,0);
44 xstring(-0.5,0.1,"${\fgcolor{blue}{\omega=\infty}}$"
     ,0,0);
```

```
45 xarrows([-0.2;0],[0.2;0],-1,2)
46 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 6.11 Stability properties for a conditionally stable system

```
1 //Example 6.11
2 // Stability properties for a conditionally stable
     system.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
```

```
8 //System transfer function
9 s=poly(0,'s');
10 Gs=(s+10)^2/(s^3);
11 evans(Gs,100)
12 exec .\fig_settings.sci; //custom script for setting
       figure properties
13 zoom_rect([-40 -20 5 20])
14 title(["Root locus for","$G(s)=(s+10)^2/s^3$"],'
     fontsize',3)
15 h=legend('');
16 h.visible = "off"
```

```
17 Gs1=syslin('c',(s+10)^2/(s^3));
18 //
   ------------------------------------------------------------------------

19 //The nyquist plot of the system
20 figure;
21 nyquist(7*Gs1,8/2/%pi,100/2/%pi,0.005)
22 exec .\fig_settings.sci; //custom script for setting
       figure properties
23 title(["Nyquist plot for","$G(s)=(s+10)^2/s^3$"],'
     fontsize',3)
24 f=gca();
25 f.x_location = "origin";
26 f.y_location = "origin";
27 xset("color",2);
28 //
   ------------------------------------------------------------------------
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 6.12 Nyquist plot for a system with Multiple Crossover frequencies

```
1 //Example 6.12
2 // Nyquist plot for a system with Multiple Crossover
      frequencies
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
```

```
7  //
   ------------------------------------------------------------------------

8  //System transfer function
9  s=poly(0,'s');
10 K=85;
11 g1=K*(s+1)/(s^2*(s^2+2*s+82));
12 g2=(s^2+2*s+43.25)/(s^2+2*s+101);
13
14 Gs=syslin('c',g2*g1);
15 //
   ------------------------------------------------------------------------

16 figure;
17 //The nyquist plot of the system
18 nyquist(Gs,0.5/2/%pi,100/2/%pi,0.005)
19 title(["Nyquist plot for the complex system";...
20 "$G(s)=85(s+1)(s^2+2s+43.25)/[((s^2+2s+82)(s^2+2s
      +101)]$"],...
21 'fontsize',3)
22 exec .\fig_settings.sci; //custom script for setting
       figure properties
23 zoom_rect([-2 -1 0.6 1])
24 f=gca();
25 f.x_location = "origin";
26 f.y_location = "origin";
27 xset("color",2);
28 //
   ------------------------------------------------------------------------

29 //The bode plot of the system
30 gm=g_margin(Gs);
31 pm=p_margin(Gs)
32 disp(pm,"Phase margin",gm,"Gain margin")
33 figure(1)
34 bode(Gs,0.01/2/%pi,100/2/%pi,0.01)
35 title(["Bode plot for";...
36 "$G(s)=85(s+1)(s^2+2s+43.25)/[((s^2+2s+82)(s^2+2s
```

```
    +101)]$"],...
37 'fontsize',3)
38 exec .\fig_settings.sci; //custom script for setting
       figure properties
39 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 6.13 Use of simple design criterion for spacecraft attitude contro

```
1 //Example 6.13
2 // Use of simple design criterion for spacecraft
     attitude control.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
```

```
8 //System transfer function
9 s=poly(0,'s');
10 G=1/s^2;
11 g1=syslin('c',G);
12
13 //The bode plot of the system
14 zoom_rect([0.01 -20 100 60])
15 bode(g1,0.05/2/%pi,2/2/%pi,"rad")
```

```
16 exec .\fig_settings.sci; //custom script for setting
       figure properties
17 title('Magnitude of the spacecrafts frequency','
       fontsize',3)
18 //
     ------------------------------------------------------------------


19
20 K=1;
21 Td=20;
22 Ds=(Td*s+1);
23 gd1=syslin('c',K*Ds*G);
24
25 ////The bode plot of compnenstaed open loop system
26 figure
27 bode(gd1,0.01/2/%pi,1/2/%pi,"rad")
28 exec .\fig_settings.sci; //custom script for setting
       figure properties
29 title('Bode plot for compensated open-loop transfer
       function'...
30 ,'fontsize',3)
31 xstring(0.02,70,"-40db/decade",0,0);
32 xstring(0.2,40,"-20db/decade",0,0);
33
34 //The bode plot of compnenstaed closed loop system
35 K=0.01;
36 gc1=K*gd1/(1+K*gd1);
37 gcl1=syslin('c',gc1);
38 figure
39 bode(gcl1,0.01/2/%pi,10/2/%pi,"rad")
40 title('Closesd loop frequency response','fontsize'
       ,3)
41 exec .\fig_settings.sci; //custom script for setting
       figure properties
42
43 //Bandwidth
44 [frq, repf,splitf]=repfreq(gc1,[0.01/2/%pi
       :0.001:10/2/%pi]);
```

```
45 [db, phi]=dbphi(repf);
46 w=find(db<=db(1)-3);
47 wc=w(1);
48 frqc=frq(wc)*2*%pi;
49
50 plot2d3(frqc,db(wc),5)
51
52 [r c]=size(frq(1:w(1)));
53 magn=db(wc)*ones(r,c)
54 plot(frq(1:w(1))*2*%pi,magn,"b--")
55 temp_db=db(w);
56 [r c]=size(db(w));
57 temp_w=frqc*ones(r,c);
58 plot(temp_w,temp_db,"b--")
59 xset("font size", 3);
60 xstring(0.04,-16,"$\omega_{BW}$");
61 xstring(frqc,-4,"-3db");
62 xset("line style",4)
63 xarrows([0.01;frqc],[-10;-10],-0.2,5)
64 xarrows([frqc;0.01],[-10;-10],-0.2,5)
65 //
   ------------------------------------------------------------------------

66 //Step response of PD compnensation
67 figure
68 t=0:0.5:100;
69 v=csim('step',t,gcl1);
70 plot2d(t,v)
71
72 //Title, labels and grid to the figure
73 exec .\fig_settings.sci; //custom script for setting
       figure properties
74 title('Step response for PD compensation','fontsize'
     ,3)
75 xlabel('Time t (sec.)','fontsize',2)
76 ylabel('$\theta$','fontsize',2)
77 //
   ------------------------------------------------------------------------
```

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 6.14 `Lead compensation for DC motor`

```
1 //Example 6.14
2 //Lead compensation for DC motor.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
  ------------------------------------------------------------------------

8 //System transfer function
9 s=poly(0,'s');
10 g=1/s/(s+1);
11 K=10; //DC gain
12 KGs=syslin('c',K*g);
13
14 //Lead compensator
15 numD=s/2+1;
16 denD=s/10+1;
17 D=numD/denD;
18 Ds=syslin('c',D);
19
20 KGDs=Ds*KGs; //compensated system
21 //
  ------------------------------------------------------------------------
```

```
22 //(a) The bode plot of the system
23 bode([KGs;KGDs],0.1/2/%pi,100/2/%pi,['KG(s)';'D(s)G(
     s)'],"rad");
24 exec .\fig_settings.sci; //custom script for setting
      figure properties
25 title('Frequency response of lead compensation
     design','fontsize',3)
26
27 //root locus
28 figure(1)
29 evans(KGDs/K)
30 xset("font size", 3);
31 xstring(-10,4,"$KD(s)=\frac{s/2+1}{s/10+1}$",0,0)
32 xstring(-10,2,"$G(s)=\frac{1}{s(s+1)}$",0,0)
33
34 //Title, labels and grid to the figure
35 exec .\fig_settings.sci; // custom script for
     setting figure properties
36 title('Root locus for lead compensation design','
     fontsize',3)
37 zoom_rect([-14 -8 4 8])
38 f=gca();
39 f.x_location = "origin";
40 f.y_location = "origin";
41 h=legend('');
42 h.visible = "off"
43 //
     ------------------------------------------------------------------------


44 //(b) digital version of lead compensator
45 //Discretize the system using sampling time Ts=0.05
     and Bilinear Transform
46 Ts=0.05;          //in book its 0.005, which may not
     give expected responses
47 D=tf2ss(KGDs/K/g);
48 sysD=cls2dls(D,Ts);
49
50 //Pulse transfer function
```

```
51 Ddz=ss2tf(sysD)
52 disp(Ddz,"Ddz=")
53
54 //
    ------------------------------------------------------------------------

55 //(c) Compare step and ramp responses.
56 //step response switch sw=1 and for ramp response sw
    =0
57 //
    ------------------------------------------------------------------------

58
59 //step response
60 sw=1;
61 importXcosDiagram(".\Ex6_14_model.xcos")
62
63 xcos_simulate(scs_m,4);
64 scs_m.props.context
65 figure,
66 a1=newaxes();
67 a1.axes_bounds=[0,0,1.0,0.5];
68 plot(time_resp.time,time_resp.values)
69
70 xlabel('time');
71 ylabel('y');
72 title(["Lead-compensation design (a) step Response
    ...
73  (b) ramp response"],'fontsize',3)
74 exec .\fig_settings.sci; //custom script for setting
     figure properties
75 legend("continuous controller","digital controller"
    ,4)
76 //
    ------------------------------------------------------------------------

77 //ramp response
78 sw=0;
```

```
79 importXcosDiagram(".\Ex6_14_model.xcos")
80
81 xcos_simulate(scs_m,4);
82 scs_m.props.context
83
84 a2=newaxes();
85 a2.axes_bounds=[0,0.5,1.0,0.5];
86 plot(time_resp.time,time_resp.values)
87
88 xlabel('time');
89 ylabel('y');
90 title("(b)",'fontsize',3)
91 exec .\fig_settings.sci; //custom script for setting
     figure properties
92 legend("continuous controller","digital controller"
   ,4)
93 //
```

This code can be downloaded from the website wwww.scilab.in check Appendix AP 1 for dependency:

```
fig_settings.sci
```

Scilab code Exa 6.15 Lead compensation for Temperature Control System

```
1 //Example 6.15
2 //Lead compensation for Temperature Control System.
3
```

```
4  xdel(winsid())//close all graphics Windows
5  clear;
6  clc;
7  //
   ------------------------------------------------------------------------

8  //System transfer function
9  s=poly(0,'s');
10 numG=1;
11 denG=(s/0.5+1)*(s+1)*(s/2+1);
12 sysG=numG/denG;
13 //Dc gain
14 K=9;
15
16 KGs=syslin('c',K*sysG);
17
18 //Lead compensator 1
19 numD=s+1;
20 denD=s/3+1;
21 D1=numD/denD;
22 D1s=syslin('c',D1);
23
24 KGD1s=D1s*KGs; //compensated system
25
26 //Lead compensator 2
27 numD=s/1.5+1;
28 denD=s/15+1;
29 D2=numD/denD;
30 D2s=syslin('c',D2);
31
32 KGD2s=D2s*KGs; //compensated system
33
34 //The bode plot of the system with K
35 bode([KGs;KGD1s;KGD2s],0.1/2/%pi,10/2/%pi,['KG';'
      KGD1';'KGD2'],"rad");
36 exec .\fig_settings.sci; // custom script for
      setting figure properties
37 title('Bode plot for lead compensation design','
```

```
    fontsize',3)
38 //
   ------------------------------------------------------------------------

39 //Margins of uncompensated and compensated systems
40 [gm1,wcg1]=g_margin(KGs);
41 [pm1,wcp1]=p_margin(KGs);
42 disp(wcp1*2*%pi,"Wcp",wcg1*2*%pi,"Wcg",pm1,...
43 "Phase margin",gm1,"Gain margin",...
44 "Uncompensated system :")
45
46 [gm2,wcg2]=g_margin(KGD1s);
47 [pm2,wcp2]=p_margin(KGD1s);
48 disp(wcp2*2*%pi,"Wcp",wcg2*2*%pi,"Wcg",pm2,...
49 "Phase margin",gm2,"Gain margin",...
50 "System with D1 compensator :")
51
52 [gm3,wcg3]=g_margin(KGD2s);
53 [pm3,wcp3]=p_margin(KGD2s);
54 disp(wcp3*2*%pi,"Wcp",wcg3*2*%pi,"Wcg",pm3,...
55 "Phase margin",gm3,"Gain margin",...
56 "System with D2 compensator :")
57 //
   ------------------------------------------------------------------------

58 //step response comparison
59 //closed loop system
60 Gc1=KGD1s/(KGD1s+1);
61 Gc2=KGD2s/(KGD2s+1);
62 figure;
63 t=0:0.05:20;
64 v1=csim('step',t,Gc1);
65 v2=csim('step',t,Gc2);
66 plot2d([t',t'],[v1',v2'])
67
68 //Title, labels and grid to the figure
69 exec .\fig_settings.sci; //custom script for setting
       figure properties
```

```
70 title('Step response for lead compensation design','
   fontsize',3)
71 xlabel('Time t (sec.)','fontsize',2)
72 ylabel('y','fontsize',2)
73
74 xset("font size", 3);
75 xarrows([2.5;1.5],[1.3;1.2],-1,1)
76 xstring(2.5,1.3,"D2",0,0)
77 xstring(4,1.2,"D1",0,0)
78 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 6.16 Lead compensation for Servomechanism System

```
1 //Example 6.16
2 //Lead compensation for Servomechanism System.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
```

```
8 //System transfer function
9 s=poly(0,'s');
10 numG=10;
11 denG=s*(s/2.5+1)*(s/6+1);
```

```
12 G=numG/denG;
13 //Dc gain
14 K=1;
15
16 KGs=syslin('c',K*G);
17
18 //Lead compensator 1
19 numD=s/2+1;
20 denD=s/20+1;
21 D1=numD/denD;
22 D1s=syslin('c',D1);
23
24 KGD1s=D1s*KGs; //compensated system
25
26 //Lead compensator 2
27 numD=s/4+1;
28 denD=s/40+1;
29 D2=D1*numD/denD; //double compensator
30 D2s=syslin('c',D2);
31
32
33 KGD2s=D2s*KGs; //compensated system
34
35 //The bode plot of the system with K
36 bode([KGs;KGD1s;KGD2s],0.1/2/%pi,100/2/%pi,['KG';'
     KGD1';'KGD2'],"rad");
37 exec .\fig_settings.sci; //custom script for setting
      figure properties
38 title('Bode plot for lead compensation design','
     fontsize',3)
39 //
    ------------------------------------------------------------------------

40 //Margins of uncompensated and compensated systems
41 [gm1,wcg1]=g_margin(KGs);
42 [pm1,wcp1]=p_margin(KGs);
43 disp(wcp1*2*%pi,"Wcp",wcg1*2*%pi,"Wcg",pm1,...
44 "Phase margin",gm1,"Gain margin","Uncompensated
```

```
    system :")
45
46 [gm2,wcg2]=g_margin(KGD1s);
47 [pm2,wcp2]=p_margin(KGD1s);
48 disp(wcp2*2*%pi,"Wcp",wcg2*2*%pi,"Wcg",pm2,...
49 "Phase margin",gm2,"Gain margin","System with D1
    compensator :")
50
51 [gm3,wcg3]=g_margin(KGD2s);
52 [pm3,wcp3]=p_margin(KGD2s);
53 disp(wcp3*2*%pi,"Wcp",wcg3*2*%pi,"Wcg",pm3,...
54 "Phase margin",gm3,"Gain margin","System with D2
    compensator :")
55 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 6.17 Lag compensation for Temperature Control System

```
1 //Example 6.17
2 //Lag compensation for Temperature Control System.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
```

```
8 //System transfer function
9 s=poly(0,'s');
```

```
10 numG=1;
11 denG=(s/0.5+1)*(s+1)*(s/2+1);
12 G=numG/denG;
13 //Dc gain
14 K=3; //to set phase requirement
15
16 KGs=syslin('c',K*G);
17
18 //Lag compensator
19 numD=5*s+1;
20 denD=15*s+1;
21 D=3*numD/denD;
22 Ds=syslin('c',D);
23
24 KGDs=Ds*KGs; //compensated system
25
26 //The bode plot of the system with K
27 bode([KGs;KGDs],0.01/2/%pi,10/2/%pi,['KG';'KGD'],"
      rad");
28 exec .\fig_settings.sci; //custom script for setting
       figure properties
29 title('Frequency response of lag-compensation design
      ','fontsize',3)
30
31 //
     ------------------------------------------------------------------------

32 //Margins of uncompensated and compensated systems
33 [gm1,wcg1]=g_margin(KGs);
34 [pm1,wcp1]=p_margin(KGs);
35 disp(wcp1*2*%pi,"Wcp",wcg1*2*%pi,"Wcg",pm1,"Phase
      margin",...
36 gm1,"Gain margin","Uncompensated system :")
37
38 [gm2,wcg2]=g_margin(KGDs);
39 [pm2,wcp2]=p_margin(KGDs);
40 disp(wcp2*2*%pi,"Wcp",wcg2*2*%pi,"Wcg",pm2,"Phase
      margin",...
```

```
gm2,"Gain margin","Compensated system :")

//
```

```
//step response
//closed loop system
Gc=KGDs/(KGDs+1);
figure;
t=0:0.05:20;
v=csim('step',t,Gc);
plot2d(t,v)

//Title, labels and grid to the figure
exec .\fig_settings.sci; //custom script for setting
    figure properties
title('Step response for lag compensation design','
   fontsize',3)
xlabel('Time t (sec.)','fontsize',2)
ylabel('y','fontsize',2)
//
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 6.18 `Lag compensation for DC motor`

```
//Example 6.18
//Lag compensation for DC motor.
```

```
3
4  xdel(winsid())//close all graphics Windows
5  clear;
6  clc;
7  //
     ---------------------------------------------------------------------------

8  //System transfer function
9  s=poly(0,'s');
10 g=1/s/(s+1);
11 K=10; //DC gain
12 KGs=syslin('c',K*g);
13
14 //Lag compensator
15 numD=10*s+1; //0.1
16 denD=100*s+1; //0.01
17 D=numD/denD;
18 Ds=syslin('c',D);
19
20 KGDs=Ds*KGs; //compensated system
21
22 //The bode plot of the system
23 bode([KGs;KGDs],0.001/2/%pi,10/2/%pi,['KG(s)';'D(s)G
     (s)'],"rad");
24 exec .\fig_settings.sci; // custom script for
     setting figure properties
25 title('Frequency response of lag-compensation design
     ...
26  of DC motor','fontsize',3)
27 //
     ---------------------------------------------------------------------------

28 //step response
29 //closed loop system
30 Gc=KGDs/(KGDs+1);
31 figure;
32 t=0:0.05:50;
33 v=csim('step',t,Gc);
```

```
34 plot(t,v,2)
35
36 //Title, labels and grid to the figure
37 exec .\fig_settings.sci; // custom script for
      setting figure properties
38 title('Step response for Lag-compensation design...
39  of DC motor','fontsize',3)
40 xlabel('Time t (sec.)','fontsize',2)
41 ylabel('y','fontsize',2)
42 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 6.19 PID compensation design for spacecraft attitude control

```
1 //Example 6.19
2 //PID compensation design for spacecraft attitude
     control.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
```

```
8 //System transfer function
9 s=poly(0,'s');
10 G1=(0.9/s^2);
```

```
11 G2=(2/(s+2));
12 G=G1*G2;
13 Gs=syslin('c',G);
14
15 // PID controller parameters
16 Td_inv=0.1;  //  Td_inv=1/Td=0.1
17 Kd=1/Td_inv; //Kd=Td=Td_inv (derivative gain)
18
19 Ti_inv=0.005;  //  Ti_inv=1/Ti=0.005
20 Ki=Ti_inv; //Ki=Ti_inv (integral gain)
21
22 Kp=0.05 //Kp (Proportional gain)
23
24 D=Kp*(Kd*s+1)*(Ki/s+1); //PID Compensator
25
26 Dsc=syslin('c',D);
27
28 Ds=syslin('c',D/Kp); //PID Compensator with Kp=1
29 // Compensated system with Kp=1
30 GDs=Gs*Ds;
31 //PID compensated system Kp=0.05;
32 GDsc=Gs*Dsc;
33 //
    ------------------------------------------------------------------------

34 //The bode plots
35 bode([Gs;GDs;GDsc],0.01/2/%pi,100/2/%pi,...
36 ['G(s)';'D(s)G(s) with (Kp=1)';'D(s)G(s) with (Kp
     =0.05)'],"rad");
37 exec .\fig_settings.sci; //custom script for setting
      figure properties
38 title('Compensation for PID design','fontsize',3)
39
40 //Phase margin of pid compensated system with Kp
     =0.05;
41 [pm wcp]=p_margin(GDsc);
42
43 //
```

```
44 //closed loop system
45 //step response
46 Gc=GDsc/(GDsc+1);
47 figure;
48 t=0:0.05:40;
49 y=csim('step',t,Gc);
50 plot(t,y,2)
51
52 //Title, labels and grid to the figure
53 exec .\fig_settings.sci; //custom script for setting
       figure properties
54 title('Step response for PID compensation of
      spacecraft'...
55 ,'fontsize',3)
56 xlabel('Time t (sec.)','fontsize',2)
57 ylabel('$theta$','fontsize',2)
58 //
```

```
59 //step disturbance response
60 Gc=G1/((G1*G2*D)+1);
61 Gcs=syslin('c',Gc);
62 figure;
63 t=0:0.5:1000;
64 u=0.1*ones(1,length(t));
65 y=csim(u,t,Gcs)
66 plot(t,y,2)
67
68 //Title, labels and grid to the figure
69 exec .\fig_settings.sci; // custom script for
      setting figure properties
70 title('Step disturbance response for PID
      compensation...
71  of spacecraft','fontsize',3)
72 xlabel('Time t (sec.)','fontsize',2)
73 ylabel('$theta$','fontsize',2)
```

```
74  //
```

Chapter 7

State Space Design

Scilab code Exa 7.2.b Cruise control system step response

```
1 //Example 7.2
2 //Cruise control system step response.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 clc;
8 //
   ------------------------------------------------------------------------

9 //Cruise control system parameters
10 m=1000;
11 b=50;
12 u=500;
13
14 // Transfer function
15 s=%s; // or
16 s=poly(0,'s');
17 sys1=syslin('c',(1/m)/(s+b/m));
18 disp(sys1)
19 //
```

```
F=[0 1; 0 -b/m];
G=[0;1/m];
H=[0 1];
J=0;
sys=syslin('c',F,G,H,J);
//
```

```
//step response to u=500;
t=0:0.5:100;
v=csim('step',t,u*sys);
plot(t,v,2)

//Title, labels and grid to the figure
exec .\fig_settings.sci; // custom script for
   setting figure properties
title('Responses of car velocity to a step in u','
   fontsize',3)
xlabel('Time t (sec.)','fontsize',2)
ylabel('Amplitude','fontsize',2)
//
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 7.7 `Analog computer Implementation`

```
//Example 7.7
//Analog computer Implementation.
```

```
3
4  xdel(winsid())//close all graphics Windows
5  clear;
6  clc;
7  //
     ---------------------------------------------------------------------------

8  // State space model of the given system
9  F=[-6 -11 -6; 1 0 0; 0 1 0];
10 G=[6; 0; 0];
11 H=[0 0 1];
12 J=0;
13 sys_ss=syslin('c',F,G,H,J)
14 disp(sys_ss)
15 //
     ---------------------------------------------------------------------------

16 //Transfer function form
17 [d,Ns,Ds]=ss2tf(sys_ss)
18 Ns=clean(Ns);
19 G=syslin('c',Ns/Ds);
20 disp(G)
21 //
     ---------------------------------------------------------------------------

22 // convert numerator - denominator to pole - zero
     form
23 //gain (K) pole (P) and zeros (Z) of the system
24 temp=polfact(Ns);
25 Z=roots(Ns); //locations of zeros
26 P=roots(Ds); //locations of poles
27 K=temp(1); //first entry is always gain
28 disp( K,"Gain", P, "Poles",Z,"Zeros",)
29 //
     ---------------------------------------------------------------------------
```

Scilab code Exa 7.8 Time scaling an oscillator

```
1 //Example 7.8
2 //Time scaling an oscillator.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
    ------------------------------------------------------------------------

8 // State space model of an oscillator
9 wn=15000 // rad/sec
10 F=[0 1;wn^2 0];
11 G=[0;10^6];
12 disp(G,"G",F,"F","Given system");
13
14 //
    ------------------------------------------------------------------------

15 // State space model of the time-scaled system for
16 // a millisecond scale w0=1e3;
17 w0=1e3; //rad/sec
18 F1=F/w0;
19 G1=G/w0;
20 disp(G1,"G1",F1,"F1","Time scaled system in mm");
21 //
    ------------------------------------------------------------------------
```

Scilab code Exa 7.9 State Equations in Modal Canonical Form

```
1  //Example 7.9
2  //State Equations in Modal Canonical Form.
3
4  xdel(winsid())//close all graphics Windows
5  clear;
6  clc;
7  //
   ------------------------------------------------------------------------

8  //System transfer function
9  s=poly(0,'s');
10 g1=1/s^2;
11 g2=-1/(s^2+2*s+4);
12 Gs=g1+g2;
13 //
   ------------------------------------------------------------------------

14 // State space representation in modal canonical
      form
15 sys1=tf2ss(g1);
16 sys2=tf2ss(g2);
17 [F1,G1,T1]=canon(sys1.A, sys1.B)
18 H1=sys1.C*T1;
19
20 [F2,G2,T2]=canon(sys2.A, sys2.B)
21 H2=sys2.C*T2;
22
23 F=[F1 zeros(2,2);zeros(2,2) F2];
24 G=[G1;G2];
25 H=[H1,H2];
26 J=0;
27 disp(J,"J",H,"H",G,"G",F,"F","System in modal
      canonical form")
28 //
   ------------------------------------------------------------------------

29  //As Y=G*U;  consatnts k1 and k2  are taken out
      from G1 and G2 will be
```

```
30  //multiplied to H1 and H2
31
32 // So alternately, it can be reprsented as
33 k1=-1;k2=-2;
34 F=[F1 zeros(2,2);zeros(2,2) F2];
35 G=[G1/k1;G2/k2];
36 H=[H1*k1,H2*k2];
37 J=0;
38 disp(J,"J",H,"H",G,"G",F,"F","System in modal
      canonical form")
39 //
   ---------------------------------------------------------------------
   ---------------------------------------------------------------------
```

Scilab code Exa 7.10 Transformation of Thermal System from Control to Modal Form

```
1  //Example 7.10
2  //Transformation of Thermal System from Control to
      Modal Form
3
4  xdel(winsid())//close all graphics Windows
5  clear;
6  clc;
7  //
   ---------------------------------------------------------------------

8  // State space matrices of the given system
9  Ac=[-7 1; -12 0];
10 Bc=[1;2];
11 Cc=[1 0];
12 Dc=0;
13 //
   ---------------------------------------------------------------------

14 // State space representation in modal canonical
```

```
    form
T=[4 -3;-1 1]
Am=T\Ac*T;
Bm=T\Bc;
Cm=Cc*T;
Dm=Dc;
disp(Dm,"Dm",Cm,"Cm", Bm,"Bm",Am,"Am","Thermal
    System in modal canonical form")
//
```

Scilab code Exa 7.11 Poles and Zeros of Tape Drive System

```
//Example 7.11
//Poles and Zeros of Tape Drive System.
//Also, Transform the system into modal form

xdel(winsid())//close all graphics Windows
clear;
clc;
//
```

```
// State space matrices of Tape Drive System

F=[0 2 0 0 0;
-0.1 -0.35 0.1 0.1 0.75;
0 0 0 2 0;
0.4 0.4 -0.4 -1.4 0;
0 -0.03 0 0 -1];
G=[0 0 0 0 1]';
H2=[0 0 1 0 0];
H3=[0.5 0 0.5 0 0];
Ht=[-0.2 -0.2 0.2 0.2 0];
```

```
20 //
   ------------------------------------------------------------------------

21 // Poles (eigen values) of the system
22 p=clean(spec(F));
23 disp(p,"Poles of Tape Drive System are")
24
25 //It requires complete state-space model.
26 sys=syslin('c',F,G,[Ht;H2;H3],[0;0;0])
27
28 // zeros of the system
29 [tr]=trzeros(sys)
30 disp(tr,"Transmission zeros of Tape Drive System are
      ")
31 //
   ------------------------------------------------------------------------

32 // State space representation in modal canonical
      form with H3 output only.
33
34 [m Am1]=spec(F)
35 T1=[1/2 -%i/2;1/2 %i/2];
36 //transformation for a complex pair of eigen values.
37 temp=eye(5,5);
38 T=[T1 zeros(2,3);zeros(3,2) eye(3,3)];
39 temp(1,1)=-1; temp(2,2)=-1; //for change in input
      output signs as desired
40 M=m*T*temp //real Modal transformation
41
42 Am=clean(M\F*M);
43 Bm=clean(M\G);
44 Cm=clean(H3*M);
45 Dm=0;
46
47 disp(Dm,"Dm",Cm,"Cm", Bm,"Bm",Am,"Am","Tape Drive
      System in modal canonical form")
48 //
   ------------------------------------------------------------------------
```

Scilab code Exa 7.12 Transformation of Thermal System from state description

```
1 //Example 7.12
2 //Transformation of Thermal System from state
     description
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
     ------------------------------------------------------------------------

8 // State space model of Thermal System
9 s=%s;
10 F=[-7 -12; 1 0];
11 G=[1;0];
12 H=[1 2];
13 J=0;
14 sys=syslin('c',F,G,H,J)
15 //
     ------------------------------------------------------------------------

16 //Transfer function model of Thermal System
17 [ch num den]=ss2tf(sys);
18 disp(num/den, "G=","Transfer function model of
     Thermal System")
19 //
     ------------------------------------------------------------------------
```

Scilab code Exa 7.13 Zeros for the Thermal System from a State Description

```
1 //Example 7.13
2 //Zeros for the Thermal System from a State
     Description
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
     ------------------------------------------------------------------------

8 // State space model of the given system
9 F=[-7 -12; 1 0];
10 G=[1;0];
11 H=[1 2];
12 J=0;
13 sysG=syslin('c',F,G,H,J)
14 //
     ------------------------------------------------------------------------

15 //Transfer function model
16 [d num den]=ss2tf(sysG);
17 zr=roots(num);
18 disp(zr,'zr=');
19 //Alternately, it can be obtained as
20 zr=trzeros(sysG);
21 disp(zr,'zr=');
22 //
     ------------------------------------------------------------------------
```

Scilab code Exa 7.14 Analysis of state equations of Tape Drive

```
1 //Example 7.14
2 //Analysis of state equations of Tape Drive.
3 //compute the poles, zeros  and transfer function of
```

```
    Tape Drive System.
4
5  xdel(winsid())//close all graphics Windows
6  clear;
7  clc;
8  //
   ---------------------------------------------------------------------

9  // State space matrices of Tape Drive System
10
11 F=[0 2 0 0 0;
12 -0.1 -0.35 0.1 0.1 0.75;
13 0 0 0 2 0;
14 0.4 0.4 -0.4 -1.4 0;
15 0 -0.03 0 0 -1];
16 G=[0 0 0 0 1]';
17 H2=[0 0 1 0 0];
18 H3=[0.5 0 0.5 0 0];
19 Ht=[-0.2 -0.2 0.2 0.2 0];
20 //
   ---------------------------------------------------------------------

21 //Poles (eigen values) of the system
22 p=clean(spec(F));
23
24 disp(p,"P","Poles of Tape Drive System are (for any
      output)")
25 disp("
      ************************************************************
      ")
26
27
28 disp("pole and zero polynomials and transfer
      function...
29  for a system with output H2")
30 sys2=syslin('c',F,G,H2,0);
31 [d2 num2 den2]=ss2tf(sys2);
32 N2=coeff(num2);
```

```
33 D2=coeff(den2);
34 disp(D2,"D2",N2,"N2")
35 // zeros of the system with output H2
36 [zer2]=trzeros(sys2)
37 disp(zer2,"ZER2","zeros are")
38 // transfer function of the system with output H2
39 G2=clean(num2/den2);
40 disp(G2,"G2(s)=N2(s)/D2(s)=")
41 disp("
      ************************************************************
      ")
42
43 disp("pole and zero polynomials and transfer
      function for a...
44  system with output H3")
45 sys3=syslin('c',F,G,H3,0);
46 [d3 num3 den3]=ss2tf(sys3);
47 N3=coeff(num3);
48 D3=coeff(den3);
49 disp(D3,"D3",N3,"N3")
50 // zeros of the system with output H3
51 [zer3]=trzeros(sys3)
52 disp(zer3,"ZER3","zeros are")
53 // transfer function of the system with output H3
54 G3=clean(num2/den2);
55 disp(G3,"G3(s)=N3(s)/D3(s)=")
56 disp("
      ************************************************************
      ")
57
58
59 disp("pole and zero polynomials and transfer
      function for a...
60  system with output Ht")
61 syst=syslin('c',F,G,Ht,0);
62 [dt numt dent]=ss2tf(syst);
63 Nt=coeff(numt);
64 Dt=coeff(dent);
```

```
disp(Dt,"Dt",Nt,"Nt","zeros are")
// zeros of the system with output Ht
[zert]=trzeros(syst)
disp(zert,"ZERT")
// transfer function of the system with output Ht
Gt=clean(numt/dent);
disp(Gt,"G(s)=Nt(s)/Dt(s)=")
disp("
   ************************************************************
   ")
//
   ____________________________________________________________
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 7.15 `Control law for a pendulum`

```
//Example 7.15
//Control law for a pendulum.

xdel(winsid())//close all graphics Windows
clear;
clc;
//
   ____________________________________________________________

//Pendulum state model;
w0=1;

F=[0 1;-w0^2 0];
G=[0 1]';
H=eye(2,2); //representing x1 and x2 states as
   outputs
```

```
14 J=[0 0]';
15
16 sys=syslin('c',F,G,H,J); //open loop system
17
18 x0=[1 0]' //initial condition
19 t=0:0.2:7;
20 y=csim('impulse',t,sys); //open loop response
21 //
     ------------------------------------------------------------------------


22
23 //simulation for closed loop system
24 x0=[1 0]' //initial condition
25
26 //control law u=-Kx;
27 K=[3*w0^2 4*w0];
28 syscl=syslin('c',(F-G*K),G,H,J); //closed loop
     system
29
30
31 t=0:0.1:7;
32 u=zeros(1,length(t));
33 [x z]=csim(u,t,syscl,x0); //closed loop response
34 plot(t',x');
35
36 u=-K*x;
37 plot(t',u'/4,'r--'); //control law u plot (scaled to
     1/4 in figure);
38 legend("x1","x2","u/4")
39
40 //Title, labels and grid to the figure
41 exec .\fig_settings.sci; //custom script for setting
     figure properties
42 title('Impulse response of undamped oscillator with
     full-state...
43  feedback(w0=1)','fontsize',3)
44 xlabel('Time t (sec.)','fontsize',2)
45 ylabel('Amplitude','fontsize',2)
```

```
46 //
```

---

check Appendix AP 2 for dependency:

`acker_dk.sci`

Scilab code Exa 7.16 `Ackermanns formula for undamped oscillator`

```
1  //Example 7.16
2  //Ackermann's formula for undamped oscillator.
3
4  xdel(winsid())//close all graphics Windows
5  clear;
6  clc;
7  //
```

```
8  //undamped oscillator (Pendulum) state model;
9  w0=1;
10
11 F=[0 1;-w0^2 0];
12 G=[0 1]';
13 H=eye(2,2); //representing x1 and x2 states as
       outputs
14 J=[0 0]';
15 //
```

```
16 //Ackermann's formula for feedback gain computation
17
18 pc=[-2 -2];    //desired poles
19 exec('./acker_dk.sci', -1);
```

```
20 [K,eig]=acker_dk(F,G,pc)
21 disp(K,"Feedback gain K=")
22 disp(eig,"Closed loop eigen values are ")
23 //
```

---

check Appendix AP 2 for dependency:

`acker_dk.sci`

Scilab code Exa 7.17 How zero location affect control law

```
1 //Example 7.17 How zero location affect control law
2 // Obtain state feedback gain matrix for the given
    system
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
```

```
8 //(a) state feedback gain matrix for zero at 2.
9 //Location of system Zero
10 z0=2;
11
12 // State space representation
13 Ao=[-7 1;-12 0];
14 Bo=[1 -z0]';
15 Co=[1 0];
16 Do=0;
17
18 // Desired poles
19 Pd=[1 2 4];
20 Pc=roots(Pd);
```

```
21
22
23 // State feedback gain matrix for system zero at
      -2.0
24 K=ppol(Ao,Bo,Pc)
25 disp(K,"K=","State feeback gain for a system with
      zero at 2" )
26 //
   ---------------------------------------------------------------------------

27 //Location of system Zero
28 z0=-2.99
29 B=[1 -z0]';
30 // State feedback gain matrix for system zero...
31 // at -2.99 (by ackermann's formula)
32 exec('./acker_dk.sci', -1);
33 K1=acker_dk(Ao,B,Pc)
34 disp(K1,"K1","State feeback gain for a system with
      zero at -2.99")
35 //
   ---------------------------------------------------------------------------
```

---

check Appendix AP 2 for dependency:

`acker_dk.sci`

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 7.18 `Introducing the reference input`

```
1 //Example 7.18
2 //Introducing the reference input.
3
4 xdel(winsid())//close all graphics Windows
```

```
5  clear;
6  clc;
7  //
     ------------------------------------------------------------------------

8  //Pendulum state model;
9  w0=1;
10
11 F=[0 1;-w0^2 0];
12 G=[0 1]';
13 H=[1 0]; //representing x1 and x2 states as outputs
14 J=0;
15 n=sqrt(length(F));
16
17 //computing state feedback matrix to place poles at
     [-2 -2]
18 exec('./acker_dk.sci', -1);
19 K=acker_dk(F,G,[-2, -2]);
20 //
     ------------------------------------------------------------------------

21 //augmented matrix for tracking the reference
22 A=[F G;H J];
23 N=A\[zeros(1,n) 1]';
24 Nx=N(1:n);
25 Nu=N(n+1);
26
27 //feedforward gain (input weight)
28 Ntilde=Nu+K*Nx;
29
30 //-----------------------------------------
31 //Alternately, it can be computed as /
32 Ntilde1=-inv(H*inv(F-G*K)*G); //     /
33 //-----------------------------------------
34
35 //Closed loop system and step response
36 syscl=syslin('c',(F-G*K),G*Ntilde,H,J); //closed
     loop system
```

```
37
38 t=0:0.1:7;
39 [y x]=csim('step',t,syscl); //closed loop response
40 plot(t',x');
41
42 u=-K*x+Ntilde;
43 plot(t',u'/4,'r--'); //control law u plot (scaled to
       1/4 in figure);
44 legend("x1","x2","u/4");
45 xset('font size',3);
46 xstring(5,0.93,"$x_{ss}$")
47 xstring(5,0.25,"$u_{ss}$")
48
49 //Title, labels and grid to the figure
50 exec .\fig_settings.sci; //custom script for setting
       figure properties
51 title('Step response of undamped oscillator to
     reference input',...
52 'fontsize',3);
53 xlabel('Time t (sec.)','fontsize',2);
54 ylabel('Amplitude','fontsize',2);
55
56 //
```

---

check Appendix AP 2 for dependency:

`acker_dk.sci`

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 7.19 `Reference input to Type1 control system DC Motor`

```
1  //Example 7.19
2  //Reference input to Type-1 control system: DC Motor
3
4  xdel(winsid())//close all graphics Windows
5  clear;
6  clc;
7  //
   ---------------------------------------------------------------------------


8
9  //Location of system Zero
10 z0=2;
11
12 // State space representation
13 F=[0 1;0 -1];
14 G=[0 1]';
15 H=[1 0];
16 J=0;
17 n=sqrt(length(F)); //order of the system
18
19 //computing state feedback matrix to place poles at
      assumed location [-1 -2]
20 exec('./acker_dk.sci', -1);
21 K=acker_dk(F,G,[-1, -2]); //assume pd=[-1 -2]
22 //
   ---------------------------------------------------------------------------


23 //augmented matrix for tracking the reference
24 A=[F G;H J];
25 N=A\[zeros(1,n) 1]';
26 Nx=N(1:n);
27 Nu=N(n+1);
28 disp(Nx,"Nx",Nu,"Nu")
29
30 //feedforward gain (input weight)
31 Ntilde=Nu+K*Nx;
32 disp(Ntilde,"N_tilde","Input gain: N_tilde =Nu+K Nx"
      )
```

```
33 //
   ------------------------------------------------------------------------

34 // Verify if ||y-r|| -> 0;
35
36 syscl=syslin('c',(F-G*K),G*Ntilde,H,J); //closed
      loop system
37
38 t=0:0.1:10;
39 r=ones(1,length(t));//reference input
40 [y x]=csim('step',t,syscl); //closed loop response
41
42 e=sqrt((r-y).^2) //norm of error
43 plot(t,y);
44 plot(t,r,'m:'); //reference input
45 plot(t,e,'r-.'); //norm of error
46 xset('font size',3);
47 xstring(3,0.83,"y")
48 xstring(2,1,"r")
49 xstring(3,0.1,"$\|e\|$")
50 //Title, labels and grid to the figure
51 exec .\fig_settings.sci; // custom script for
      setting figure properties
52 title('Step response of undamped oscillator to
      reference input','fontsize',3);
53 xlabel('Time t (sec.)','fontsize',2);
54 ylabel('Amplitude','fontsize',2);
55 zoom_rect([0 -0.1 10 1.1])
56 //
   ------------------------------------------------------------------------
```

---

check Appendix AP 2 for dependency:

`acker_dk.sci`

check Appendix AP 1 for dependency:

fig_settings.sci

Scilab code Exa 7.20 Pole Placement as a Dominant Second Order System

```
1 //Example 7.20
2 // Pole Placement as a Dominant Second-Order System
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
  ------------------------------------------------------------------------
8
9 clc;
10 clear all;
11
12 // State space representation
13 F=[0 2 0 0 0;-0.10 -0.35 0.1 0.1 0.75; 0 0 0 2 0;...
14  0.4 0.4 -0.4 -1.4 0; 0 -0.03 0 0 -1];
15 G=[0 0 0 0 1]';
16 H=[0.5 0 0.5 0 0]; //Tape position at the head
17 Ht=[-0.2 -0.2 0.2 0.2 0]; //Tension output
18 J=0;
19 n=sqrt(length(F))
20 // Desired poles
21 Pc=[-0.707+0.707*%i -0.707-0.707*%i -4 -4 -4]/1.5;
22 //
  ------------------------------------------------------------------------
23 // State feedback gain matrix via LQR (riccati
   equation)
24 Q = eye(5,5);
25 R =1
26 // Riccati equation
```

```
27 P=riccati(F, G*inv(R)*G', Q, 'c')
28 K1=inv(R)*G'*P
29 //
     ---------------------------------------------------------------------------

30 // State feedback gain matrix via pole-placement
31 exec('./acker_dk.sci', -1);
32 K2=acker_dk(F,G,Pc);
33 disp(K2,'K2=',"Gain by ackermans formula" );
34 //
     ---------------------------------------------------------------------------

35 Ntilde1=-inv(H*inv(F-G*K1)*G); //input gain for LQR
     feedback gain.
36 Ntilde2=-inv(H*inv(F-G*K2)*G); //input gain for
     Ackerman's feedback gain.
37
38 syscl1=syslin('c',(F-G*K1),G*Ntilde1,H,J); //closed
     loop system with K1
39 syscl2=syslin('c',(F-G*K2),G*Ntilde2,H,J); //closed
     loop system with K2
40
41 t=0:0.1:12;
42 [y1 x1]=csim('step',t,syscl1); //response of
     position head with K1
43 [y2 x2]=csim('step',t,syscl2); //response of
     position head with K2
44
45 //plot of a position of read write head
46 plot(t,y1,"m-."); //Design via LQR
47 plot(t,y2,2); //Design via Ackerman's Formula
48
49 //Title, labels and grid to the figure
50 exec .\fig_settings.sci; // custom script for
     setting figure properties
51 title('Step response of tape servomotor designs','
     fontsize',3);
52 xlabel('Time t (sec.)','fontsize',2);
```

```
53 ylabel('Tape Posotion','fontsize',2);
54
55 xstring(2.5,1.1,"LQR")
56 xarrows([3;4],[1.1;0.95],-1,1)
57 xstring(5,0.7,["Dominant";"second order"])
58 xarrows([5;4.2],[0.8;0.9],-1.5,1)
59 //
   ---------------------------------------------------------------------------


60
61 //response as a tape tension
62 yt1=Ht*x1;
63 yt2=Ht*x2;
64
65 figure(1)
66 plot(t,yt1,"m-."); //Design via LQR
67 plot(t,yt2,2); //Design via Ackerman's Formula
68
69 //Title, labels and grid to the figure
70 exec .\fig_settings.sci; // custom script for
      setting figure properties
71 title('Tension plots for tape servomotor step
      responses','fontsize',3);
72 xlabel('Time t (sec.)','fontsize',2);
73 ylabel('Tape Tension','fontsize',2);
74
75 xstring(3.5,0,"LQR")
76 xarrows([3.7;4.7],[0;0],-1)
77 xstring(6.1,-0.015,["Dominant";"second order"])
78 xarrows([6;6],[-0.013;-0.002],-1)
79 //
   ---------------------------------------------------------------------------
```

check Appendix AP 1 for dependency:

fig_settings.sci

Scilab code Exa 7.21 Symmetric root locus for servo speed control

```
1 //Example 7.21
2 // Symmetric root locus (SRL) for servo speed
    control
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
    ------------------------------------------------------------------------

8 //Transfer function model of the given system
9 a=1.5;//assume
10 s=poly(0,'s');
11 nums=1;
12 dens=s+a;
13 num_s=1;
14 den_s=-s+a;
15 G0s=syslin('c',nums/dens); //G0(s)
16 G0_s=syslin('c',num_s/den_s); //G0(-s)
17
18 evans(G0s)
19 evans(G0_s)
20 zoom_rect([-3 -0.1 3 0.1])
21 f=gca();
22 f.x_location = "origin"
```

```
23 f.y_location = "origin"
24 xset("color",2);
25 h=legend('');
26 h.visible = "off"
27
28 //Title, labels and grid to the figure
29 exec .\fig_settings.sci; // custom script for
      setting figure properties
30 title('Symmetric root locus for a first order system
      ','fontsize',3);
31 //
   ------------------------------------------------------------------------

32 //Root locus design
33 //rho>0; choose rho=2
34 rho=2;
35 //optimal pole p=-sqrt(a^2+rho)
36 p=-sqrt(a^2+rho)
37 sig=real(p);
38 omega=imag(p);
39 plot(sig,omega,'ro')
40 xstring(-2.5,0.02,["pole location at";"$\rho=2$"])
41 xarrows([-2.2;-2.07],[0.02;0.002],-1.5,1)
42 //
   ------------------------------------------------------------------------
```

---

check Appendix AP 2 for dependency:

`acker_dk.sci`

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 7.22 SRL design for satellite attitude control

```
1 //Example 7.22
2 // SRL design for satellite attitude control
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
    ------------------------------------------------------------------------

8 //Transfer function for satellite attitude control
    system
9 s=poly(0,'s');
10 nums=1;
11 dens=s^2;
12 num_s=1;
13 den_s=(-s)^2;
14 G0s=syslin('c',nums/dens); //G0(s)
15 G0_s=syslin('c',num_s/den_s); //G0(-s)
16 //evans(G0s*G0_s)
17 evans(1/s^4)
18 zoom_rect([-3 -3 3 3])
19 f=gca();
20 f.x_location = "origin"
21 f.y_location = "origin"
22 xset("color",2);
23 h=legend('');
24 h.visible = "off"
25
26 //Title, labels and grid to the figure
27 exec .\fig_settings.sci; //custom script for setting
    figure properties
28 title('Symmetric root locus for the satellite','
    fontsize',3);
29 //
    ------------------------------------------------------------------------
```

```
30 //Root locus design
31 //choose rho=4.07 that places pole at -1+-j
32 rho=4.07;
33 chr_eqn=(1+rho*G0s*G0_s)
34 p=[-1+%i, -1-%i];
35 sig=real(p);
36 omega=imag(p);
37 plot(sig,omega,'ro')
38 xstring(-2.2,0.5,["pole locations at";"$\rho=4.07$"
   ])
39 //
   ---------------------------------------------------------------------

40 //pole-placement design;
41 sys=tf2ss(G0s);
42 exec('./acker_dk.sci', -1);
43 K=acker_dk(sys.A,sys.B,p);
44 syscl=syslin('c',(sys.A-sys.B*K),sys.B, sys.C, sys.D
   )
45 disp(spec(syscl.A),"Closed loop eigen values");
46 //
   ---------------------------------------------------------------------
```

---

check Appendix AP 2 for dependency:

`acker_dk.sci`

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 7.23 `SRL design for an inverted pendulum`

```
1  //Example 7.23
2  // SRL Design for an Inverted Pendulum
3
4  xdel(winsid())//close all graphics Windows
5  clear;
6  clc;
7  //
   ____________________________________________________________________________


8
9  //Transfer function model of Inverted Pendulum.
10 s=poly(0,'s');
11
12 nums=-(s+2);
13 dens=(s^2-1)
14 num_s=-(-s+2);
15 den_s=((-s)^2-1)
16 G0s=syslin('c',nums/dens); //G0(s)
17 G0_s=syslin('c',num_s/den_s); //G0(-s)
18 sysGG=G0s*G0_s;
19 evans(sysGG)
20 title('Symmetric root locus for Inverted Pendulum')
21 zoom_rect([-3 -2 3 2])
22 f=gca();
23 f.x_location = "origin"
24 f.y_location = "origin"
25 xset("color",2);
26 h=legend('');
27 h.visible = "off"
28
29 //Title, labels and grid to the figure
30 exec .\fig_settings.sci; // custom script for
      setting figure properties
31 title('Symmetric root locus for the inverted
      pendulum','fontsize',3);
32 //
   ____________________________________________________________________________
```

```
33 //Root locus design
34 //choose rho=1 that places pole at -1.36+-j0.606
35 rho=1;
36 p=[-1.36+0.606*%i, -1.36-0.606*%i];
37 sig=real(p);
38 omega=imag(p);
39 plot(sig,omega,'ro')
40 xstring(-1.25,0.5,["pole locations at";"$\rho=1$"])
41 //
   ------------------------------------------------------------------------

42 //pole-placement design;
43 Ac=[0 1;1 0];Bc=[0 -1]'; Cc=[2 1];Dc=0;
44 exec('./acker_dk.sci', -1);
45 K=acker_dk(Ac,Bc,p);
46 disp(K,"K=",spec(Ac-Bc*K),"Closed loop eigen values"
      );
47
48 //input gain calculation
49 n=sqrt(length(Ac));
50 A=[Ac Bc;Cc Dc];
51 N=A\[zeros(1,n) 1]';
52 Nx=N(1:n);
53 Nu=N(n+1);
54
55 //feedforward gain (input gain)
56 Ntilde=Nu+K*Nx;
57
58 //Step respose
59 t=0:0.1:4.5;
60 syscl=syslin('c',(Ac-Bc*K),Bc*Ntilde, Cc, Dc)
61 [y x]=csim('step',t,syscl); //closed loop response
62 figure,
63 plot(t,y);
64
65 //Title, labels and grid to the figure
66 exec .\fig_settings.sci; // custom script for
      setting figure properties
```

```
67 title('Step response for inverted pendulum','
     fontsize',3);
68 xlabel('Time t (sec.)','fontsize',2);
69 ylabel(["Position","$x_1$"],'fontsize',2);
70 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 7.24 LQR Design for a Tape Drive

```
1 //Example 7.24
2 // LQR Design for a Tape Drive
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
```

```
8 // State space model for a Tape Drive
9 F=[0 2 0 0 0;-0.10 -0.35 0.1 0.1 0.75; 0 0 0 2 0;
     0.4 0.4 -0.4 -1.4 0; 0 -0.03 0 0 -1];
10 G=[0 0 0 0 1]';
11 H3=[0.5 0 0.5 0 0];
12 //
```

```
13 // State feedback gain matrix via LQR (riccati
     equation)
14 // (a) Continuous LQR for rho=1
15 rho=1;
16 R1=1;
17 Q1=rho*H3'*H3;
18 // Riccati equation
19 P1=riccati(F, G*inv(R1)*G', Q1, 'c')
20 K1=inv(R1)*G'*P1
21 disp(K1,'K1')
22 //
     ---------------------------------------------------------------------

23 // State feedback gain matrix via LQR (riccati
     equation)
24 // (a) Comparision in step response with rho
     =0.1,1,10.
25 rho=0.1;
26 R2=1;
27 Q2=rho*H3'*H3;
28 // Riccati equation
29 P2=riccati(F, G*inv(R2)*G', Q2, 'c')
30 K2=inv(R2)*G'*P2
31
32 rho=10;
33 R3=1;
34 Q3=rho*H3'*H3;
35 // Riccati equation
36 P3=riccati(F, G*inv(R3)*G', Q3, 'c')
37 K3=inv(R3)*G'*P3
38 //
     ---------------------------------------------------------------------

39 //input gains for step reference with rho=0.1,1,10.
40 Ntilde1=-inv(H3*inv(F-G*K1)*G);
41 Ntilde2=-inv(H3*inv(F-G*K2)*G);
42 Ntilde3=-inv(H3*inv(F-G*K3)*G);
43
```

```
44 //Closed loop system with rho=0.1,1,10.
45 syscl1=syslin('c',(F-G*K1),G*Ntilde1,H3,0);
46 syscl2=syslin('c',(F-G*K2),G*Ntilde2,H3,0);
47 syscl3=syslin('c',(F-G*K3),G*Ntilde3,H3,0);
48
49 //step response with rho=0.1,1,10.
50 t=0:0.1:12;
51 [y1 x1]=csim('step',t,syscl1); //closed loop
      response
52 [y2 x2]=csim('step',t,syscl2); //closed loop
      response
53 [y3 x3]=csim('step',t,syscl3); //closed loop
      response
54
55 figure,
56 a1=newaxes();
57 a1.axes_bounds=[0,0,1.0,0.5];
58 plot(t,y1);
59 plot(t,y2,'r-.');
60 plot(t,y3,'m:');
61
62 //Title, labels and grid to the figure
63 exec .\fig_settings.sci; // custom script for
      setting figure properties
64 title('(a)Step response of step servo motor for LQR
      Design','fontsize',3);
65 xlabel('Time t (sec.)','fontsize',2);
66 ylabel(["Tape Position","$x_3$"],'fontsize',2);
67
68 xstring(4.1,0.85,"$\rho=1$")
69 xstring(5.5,0.75,"$\rho=0.1$")
70 xstring(2.1,1.05,"$\rho=10$")
71 //
     ---------------------------------------------------------------------------

72 //Tensions for the Tape
73 //For tape output is Ht=[-0.2 -0.2 0.2 0.2 0];
74 Ht=[-0.2 -0.2 0.2 0.2 0];
```

```
75 H3=Ht;
76 //input gains can not be computed because of
      singularity. so set it 1;
77 Ntilde1=1;
78 Ntilde2=1;
79 Ntilde3=1;
80
81 //Closed loop system with rho=0.1,1,10.
82 syscl1=syslin('c',(F-G*K1),G*Ntilde1,H3,0);
83 syscl2=syslin('c',(F-G*K2),G*Ntilde2,H3,0);
84 syscl3=syslin('c',(F-G*K3),G*Ntilde3,H3,0);
85
86 //step response with rho=0.1,1,10.
87 t=0:0.1:12;
88 [y1 x1]=csim('step',t,syscl1); //closed loop
      response
89 [y2 x2]=csim('step',t,syscl2); //closed loop
      response
90 [y3 x3]=csim('step',t,syscl3); //closed loop
      response
91
92 a2=newaxes();
93 a2.axes_bounds=[0,0.5,1.0,0.5];
94 plot(t,y1);
95 plot(t,y2,'r-.');
96 plot(t,y3,'m:');
97
98 //Title, labels and grid to the figure
99 exec .\fig_settings.sci; // custom script for
      setting figure properties
100 title('(b)Corresponding tension for Tape servomotor
      step response','fontsize',3);
101 xlabel('Time t (sec.)','fontsize',2);
102 ylabel(["Tape Tension","T"],'fontsize',2);
103
104
105 xstring(4.3,-0.05,"$\rho=1$")
106 xstring(6,-0.1,"$\rho=0.1$")
```

```
107 xstring(1.5,-0.03,"$\rho=10$")
108 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 7.25 An estimator design for a simple pendulum

```
1  //Example 7.25
2  // An estimator design for a simple pendulum
3
4  xdel(winsid())//close all graphics Windows
5  clear;
6  clc;
7  //
```

```
8  // State space representation
9  w0=1;
10 F=[0 1; -w0^2 0];
11 G=[0 1]';
12 H=[1 0];
13 J=0;
14 n=sqrt(length(F));//order of the system
15 // Desired estimator poles
16 Pe=[-10*w0 -10*w0];
17 // Observer gain matrix for system
18 Lt=ppol(F',H',Pe);
19 L=Lt';
20 disp(L,"L=" );
```

```
21 //
   ------------------------------------------------------------------------

22 //simulation for closed loop system
23 x0=[1 0]' //initial condition
24
25 //State feedback control law u=-Kx; (from Ex7_15)
26 K=[3*w0^2 4*w0];
27 //
   ------------------------------------------------------------------------

28 //Augmented plant and observer
29 Faug=[F-G*K, zeros(n,n); L*H, F-L*H];
30 Gaug=[0 0 0 0]';
31 Haug=[H -H];
32 Jaug=0;
33
34 sys_aug=syslin('c',Faug,Gaug,Haug,Jaug);
35 t=0:0.1:4;
36 u=zeros(1,length(t));
37 x0=[1 0 0 0]';
38 [x z]=csim(u,t,sys_aug,x0); //closed loop response
39 plot(t,z(1,:));
40 plot(t,z(2,:),'m');
41 plot(t,z(3,:),'b:');
42 plot(t,z(4,:),'m:');
43
44 //Title, labels and grid to the figure
45 exec .\fig_settings.sci; // custom script for
      setting figure properties
46 title(['Initial condition response of oscillator
      showing',...
47 '$\mathbf{x}$','and','$\hat{\mathbf{x}}$'],'fontsize
      ',3)
48 xlabel('Time t (sec.)','fontsize',2)
49 ylabel('Amplitude','fontsize',2)
50 legend('$x_1$','$x_2$','$\hat{x}_1$','$\hat{x}_2$')
51 xset('font size',2)
```

```
52 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 7.26 A reduced order estimator design for pendulum

```
1  //Example 7.26
2  // A reduced order estimator design for pendulum
3
4  xdel(winsid())//close all graphics Windows
5  clear;
6  clc;
7  //
8  // State space representation
9  w0=1;
10 F=[0 1; -w0^2 0];
11 G=[0 1]';
12 H=[1 0];
13 J=0;
14 n=sqrt(length(F));//order of the system
15
16 //partioned system
17 Faa=F(1,1); Fab=F(1,2);
18 Fba=F(2,1); Fbb=F(2,2);
19
20 // Desired estimator poles
21 Pe=[-10];
22 // Observer gain matrix for system
```

```
23 L=ppol(Fbb',Fab',Pe);
24 L=L';
25 disp(L,"L=" );
26 //
   ---------------------------------------------------------------------------

27 //simulation for closed loop system
28 x0=[1 0 10]' //initial condition
29
30 //State feedback control law u=-Kx; (from Ex7_15)
31 K=[3*w0^2 4*w0];
32 //
   ---------------------------------------------------------------------------

33 //Augmented plant and observer
34 Faug=[F-G*K, zeros(n,1); Fab, L*Fab, Fbb-L*Fab];
35 Gaug=[0 0 0]';
36 Haug=[H 0];
37 J=0;
38
39 sys_aug=syslin('c',Faug,Gaug,Haug,J);
40 t=0:0.1:4;
41 u=zeros(1,length(t));
42 [x z]=csim(u,t,sys_aug,x0); //closed loop response
43 plot(t,z(1,:),'b');
44 plot(t,z(2,:),'r');
45 plot(t,z(3,:),'r--');
46
47
48 //Title, labels and grid to the figure
49 exec .\fig_settings.sci; // custom script for
      setting figure properties
50 title('Initial condition response of the reduced
      order estimator','fontsize',3)
51 xlabel('Time t (sec.)','fontsize',2)
52 ylabel('Amplitude','fontsize',2)
53 legend('$x_1$','$x_2$','$\hat{x}_2$')
54 xset('font size',2)
```

```
55 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 7.27 SRL estimator design for a simple pendulum

```
1  //Example 7.27
2  // SRL estimator design for a simple pendulum
3
4  xdel(winsid())//close all graphics Windows
5  clear;
6  clc;
7  //
```

```
8  // State space representation
9  F=[0 1; -1 0];
10 G=[0 1]';
11 H=[1 0];
12 J=0;
13
14 //Transfer function
15 sys=syslin('c',F,G,H,J)
16 sysGG=ss2tf(sys)
17
18 //Symmetric root locus for the inverted pendulum
      estimator design
19 //
```

```
20 //Root locus design
21 evans(sysGG*sysGG)
22 zoom_rect([-5 -5 5 5])
23 f=gca();
24 f.x_location = "origin"
25 f.y_location = "origin"
26 xset("color",2);
27 h=legend('');
28 h.visible = "off"
29 //Title, labels and grid to the figure
30 exec .\fig_settings.sci; // custom script for
      setting figure properties
31 title('Symmetric root locus for inverted the
      pendulum estimator design',...
32 'fontsize',3);
33 //
   ------------------------------------------------------------------------

34 //pole locations for q=365; p=-3+-j3.18
35 p=[-3+3.18*%i -3-3.18*%i]
36 sig=real(p);
37 omega=imag(p);
38 plot(sig,omega,'ro')
39 xstring(-4,1,["pole location at";"q=365"])
40 xarrows([-3.5;-3.05],[2;3.1],-1.5,1)
41 //
   ------------------------------------------------------------------------
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

check Appendix AP 3 for dependency:

`zpk_dk.sci`

Scilab code Exa 7.28 Full order compensator design for satellite attitude control

```
1 //Example 7.28
2 // Full order compensator design for satellite
     attitude control.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
     ------------------------------------------------------------------------


8
9 // State space representation
10 A=[0 1; 0 0];
11 B=[0 1]';
12 C=[1 0];
13 D=0;
14 n=sqrt(length(A));
15 //Desired poles for the satellite attitude control
     system.
16 Pc=[-0.707+0.707*%i -0.707-0.707*%i ]
17
18 // State feedback gain
19 K=ppol(A,B,Pc)
20 disp(K,'K=',"State feedback gain")
21
22 //Estimator - error roots are at
23 Pe=[-2.5+4.3*%i -2.5-4.3*%i]
24 L=ppol(A',C',Pe);
25 L=L';
26 disp(L,'L=',"Observer gain")
27 //
     ------------------------------------------------------------------------
```

```
28 //Compensator Design
29 sys1=syslin('c',A,B,C,D);
30 G=ss2tf(sys1);
31 s=poly(0,'s');
32
33 Ds=-K*inv(s*eye(n,n)-A+B*K+L*C)*L;
34
35 exec('./zpk_dk.sci', -1);
36 [pl,zr Kp]=zpk_dk(Ds);
37 D=poly(zr,'s','roots')/poly(pl,'s','roots')
38
39 evans(G*D)
40 zoom_rect([-8 -6 8 6])
41
42 f=gca();
43 f.x_location = "origin"
44 f.y_location = "origin"
45 xset("color",2);
46 h=legend('');
47 h.visible = "off"
48
49 //Title, labels and grid to the figure
50 exec .\fig_settings.sci; //custom script for setting
       figure properties
51 title('Root locus for combined control and estimator
     ,...
52  with process gain as the parameter','fontsize',3);
53 //
     ------------------------------------------------------------------------


54 //Frequnecy response for 1/s^2 and compensated
55
56 figure,
57 bode([-Ds*G;G],0.01/2/%pi,100/2/%pi,"rad");
58 title(["Frequency response for","$G(s)=1/s^2$"],'
     fontsize',3)
59 legend('Compensated','Uncompensated')
```

```
60 exec .\fig_settings.sci; //custom script for setting
       figure properties
61 //
   ____________________________________________________________________
   ____________________________________________________________________
```

check Appendix AP 1 for dependency:

`fig_settings.sci`

check Appendix AP 3 for dependency:

`zpk_dk.sci`

Scilab code Exa 7.29 A reduced order compensator design for a satellite attitude c

```
1  //Example 7.29
2  // A reduced order compensator design for a
       satellite attitude control
3
4  xdel(winsid())//close all graphics Windows
5  clear;
6  clc;
7  //
   ____________________________________________________________________

8  // State space representation
9  F=[0 1;0 0];
10 G=[0 1]';
11 H=[1 0];
12 J=0;
13 n=sqrt(length(F));//order of the system
14
```

```
15 //partioned system
16 Faa=F(1,1); Fab=F(1,2);
17 Fba=F(2,1); Fbb=F(2,2);
18 Ga=G(1);Gb=G(2);
19
20 // Desired estimator poles
21 Pe=[-5];
22 // Observer gain matrix for system
23 L=ppol(Fbb',Fab',Pe);
24 L=L';
25 disp(L,"L=" );
26 //
   ------------------------------------------------------------------------

27 //State feedback control law u=-Kx=-(K+[L*k2 0])[y
   xc]';
28 k1=1; k2=sqrt(2);
29 K=[k1 k2];
30 Kc=K+[L*k2 0];
31 //
   ------------------------------------------------------------------------

32 //compensator differential equation
33 //xc_dot=(Fbb-L*Fab)*xb_hat + (Fba - L*Faa)*y + (Gb
   - L*Ga)*u
34 //xc_dot=((Fbb-L*Fab)-k2)*xc + [(Fba - L*Faa)-(Gb -
   L*Ga)*(k1+L*k2)+L*(Fbb-L*Fab)]*y
35 Fc=(Fbb-L*Fab)-Gb*k2
36 Fy=(Fba - L*Faa)-(Gb - L*Ga)*(k1+k2*L)+(Fbb-L*Fab)*L
37 //compensator transfer function
38 s=poly(0,'s');
39 Gest=syslin('c',Fy/(s-Fc))//estimator transfer
   function
40 Dcr=-[k1+L*k2+k2*Gest]
41 disp(Dcr,'Dcr','compensator transfer function')
42 //
   ------------------------------------------------------------------------
```

```
43 //Root locus with reduced order compensator
44 G=1/s^2;
45 G=syslin('c',G);
46 exec('./zpk_dk.sci', -1);
47 [pl,zr Kp]=zpk_dk(Dcr);
48
49 Dcr=poly(zr,'s','roots')/poly(pl,'s','roots')
50 Dcr=syslin('c',Dcr);
51 evans(G*Dcr)
52 zoom_rect([-8 -4  2 4])
53
54 f=gca();
55 f.x_location = "origin"
56 f.y_location = "origin"
57 xset("color",2);
58 h=legend('');
59 h.visible = "off"
60
61 //Title, labels and grid to the figure
62 exec .\fig_settings.sci; //custom script for setting
       figure properties
63 title(['Root locus of a reduced order controller and
     ',"$1/s^2$",...
64  "process"],'fontsize',3);
65 //
     ---------------------------------------------------------------------

66 //Frequnecy response for 1/s^2 and compensated
67
68 figure,
69 bode([-Kp*G*Dcr;G],0.01/2/%pi,100/2/%pi,"rad");
70 title(["Frequency response","$G(s)=1/s^2$", "with a
     reduced...
71  order estimator"],'fontsize',3)
72 exec .\fig_settings.sci; //custom script for setting
       figure properties
73 legend('Compensated','Uncompensated')
74 //
```

check Appendix AP 1 for dependency:

`fig_settings.sci`

check Appendix AP 3 for dependency:

`zpk_dk.sci`

Scilab code Exa 7.30 `Full Order Compensator Design for DC Servo`

```
1 //Example 7.30
2 // Full-Order Compensator Design for DC Servo.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
8
9 // State space representation
10 //Transfer function model for DC Servo
11 s=poly(0,'s');
12 num=10;
13 den=s*(s+2)*(s+8);
14 Gs=syslin('c',num/den);
15
16 // State space representation
17 F=[-10 1 0;-16 0 1;0 0 0];
18 G=[0 0 10]';
```

```
19 H=[1 0 0];
20 J=0;
21 n=sqrt(length(F));
22 //Desired poles for the DC Servo system.
23 Pc=[-1.42 -1.04+2.14*%i -1.04-2.14*%i ]
24
25
26 // State feedback gain
27 K=ppol(F,G,Pc)
28 disp(K,'K=',"State feedback gain")
29
30 //Estimator - error roots are at
31 Pe=[-4.25 -3.13+6.41*%i -3.13-6.41*%i]
32 L=ppol(F',H',Pe);
33 L=L';
34 disp(L,'L=',"Observer gain")
35 //
   ------------------------------------------------------------------------

36 //Compensator Design
37 DK=-K*inv(s*eye(n,n)-F+G*K+L*H)*L;
38
39 exec('./zpk_dk.sci', -1);
40 [p,z]=zpk_dk(DK);
41 D=poly(z,'s','roots')/poly(p,'s','roots')
42
43 evans(Gs*D)
44 zoom_rect([-8 -9 3 9])
45
46 f=gca();
47 f.x_location = "origin"
48 f.y_location = "origin"
49 xset("color",2);
50 h=legend('');
51 h.visible = "off"
52
53 //Title, labels and grid to the figure
54 exec .\fig_settings.sci; // custom script for
```

```
    setting figure properties
title('Root locus for DC servo pole assignment','
    fontsize',3);
//
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 7.31 Reduced Order Estimatortor Design for DC Servo

```
//Example 7.31
// Reduced-Order Estimator Design for DC Servo.

xdel(winsid())//close all graphics Windows
clear;
clc;
//


// State space representation
//Transfer function model for DC Servo
s=poly(0,'s');
num=10;
den=s*(s+2)*(s+8);
Gs=syslin('c',num/den);
// State space representation
F=[-10 1 0;-16 0 1;0 0 0]
G=[0 0 10]';
H=[1 0 0];
J=0;
```

```
20 n=sqrt(length(F));
21 //Desired poles for the DC Servo system.
22 Pc=[-1.42 -1.04+2.14*%i -1.04-2.14*%i ]
23 // State feedback gain
24 K=ppol(F,G,Pc)
25 disp(K,'K=',"State feedback gain")
26
27 //
    ---------------------------------------------------------------------------

28 //Estimator - error roots are at
29 //partioned system
30 Faa=F(1,1); Fab=F(1,2:3);
31 Fba=F(3,1); Fbb=F(2:3,2:3);
32 Ga=G(1);Gb=G(2:3);
33
34 Pe=[-4.24+4.24*%i, -4.24-4.24*%i]
35 // Observer gain matrix for system
36 L=ppol(Fbb',Fab',Pe);
37 L=L';
38 disp(L,"L=" );
39 //
    ---------------------------------------------------------------------------

40
41 //State feedback control law u=-Kx=-(K+[L*k2 0])[y
      xc]';
42 k1=K(1); k2=K(2:3);
43
44 //
    ---------------------------------------------------------------------------

45 //compensator transfer function
46 s=poly(0,'s');
47 num=(-0.735+s)*(1.871+s);
48 den=poly([-0.990 + 6.12* %i, -0.990 - 6.12* %i] ,'s'
      ,'roots')
49 Dcr=syslin('c',num/den);
```

```
50 disp(Dcr,'Dcr','compensator transfer function')
51 //
   ---------------------------------------------------------------------

52 //Root locus with reduced order compensator
53 evans(-Dcr*Gs)
54 zoom_rect([-8 -9 3 9])
55
56 f=gca();
57 f.x_location = "origin"
58 f.y_location = "origin"
59 xset("color",2);
60 h=legend('');
61 h.visible = "off"
62
63 //Title, labels and grid to the figure
64 exec .\fig_settings.sci; // custom script for
      setting figure properties
65 title('Root locus for DC servo reduced order
      controller','fontsize',3);
66 //
   ---------------------------------------------------------------------
```

---

check Appendix AP 2 for dependency:

`acker_dk.sci`

check Appendix AP 1 for dependency:

`fig_settings.sci`

check Appendix AP 3 for dependency:

`zpk_dk.sci`

Scilab code Exa 7.32 Redesign of the Dc servo compensator using SRL

```
1 //Example 7.32
2 // Redesign of the Dc servo compensator using SRL
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
    ------------------------------------------------------------------------


8
9 // State space representation
10 //Transfer function model for DC Servo
11 s=poly(0,'s');
12 num=10;
13 den=s*(s+2)*(s+8);
14 Gs=syslin('c',num/den);
15
16 // State space representation
17 F=[-10 1 0;-16 0 1;0 0 0]
18 G=[0 0 10]';
19 H=[1 0 0];
20 J=0;
21 n=sqrt(length(F));
22 //Desired poles for the DC Servo system.
23 Pc=[-2+1.56*%i -2-1.56*%i -8.04]
24
25
26 // State feedback gain
27 K=ppol(F,G,Pc)
28 disp(K,'K=',"State feedback gain")
29
30 //Estimator - error roots are at
31 Pe=[-4+4.49*%i -4-4.49*%i -9.169]
32 exec .\acker_dk.sci;
33 Lt=ppol(F',H',Pe);
34 L=clean(Lt');
```

```
disp(L,'L=',"Observer gain")
//Error in book, Gain values are different in book.
//
```

```
//Compensator Design
DK=-K*inv(s*eye(n,n)-F+G*K+L*H)*L;
DK=syslin('c',DK)
exec('./zpk_dk.sci', -1);
[pl,zr,Kp]=zpk_dk(DK);
Dc=poly(zr,'s','roots')/poly(pl,'s','roots')
//
```

```
//symmetric root locus
G_s=horner(Gs,-s);
evans(Gs*G_s)
zoom_rect([-10 -5 10 5])
f=gca();
f.x_location = "origin"
f.y_location = "origin"
xset("color",2);
h=legend('');
h.visible = "off"
//Title, labels and grid to the figure
exec .\fig_settings.sci; //custom script for setting
    figure properties
title('Symmetric root locus','fontsize',3);
//
```

```
//root locus
figure,
evans(Gs*Dc) //Correct root locus
zoom_rect([-11 -6 1 6])
f=gca();
f.x_location = "origin"
f.y_location = "origin"
```

```
66 xset("color",2);
67 h=legend('');
68 h.visible = "off"
69 //Title, labels and grid to the figure
70 exec .\fig_settings.sci; // custom script for
      setting figure properties
71 title('Root locus for pole assignment from the SRL',
      'fontsize',3);
72 //
    ---------------------------------------------------------------------

73 //Discrete-time controller
74 nc=94.5*conv([7.98 1],[2.52 1])
75 dc=conv([59.5348 8.56 1],[10.6 1])
76 sysDc=poly(nc,'s','coeff')/poly(dc,'s','coeff');
77 sysDc_ss=syslin('c',tf2ss(sysDc));
78 ts=0.1;
79 sysDd=dscr(sysDc_ss,ts)
80 Gdz=ss2tf(sysDd);
81
82 disp(sysDc,"Continuous-time compensator")
83 disp(Gdz,"Discrete-time compensator")
84 //
    ---------------------------------------------------------------------

85 //step responses
86 importXcosDiagram(".\Ex7_32_model.xcos")
87
88 xcos_simulate(scs_m,4);
89 scs_m.props.context
90 figure,
91 plot(yt.time,yt.values(:,1),2)
92 plot(yt.time,yt.values(:,2),'r--')
93 xlabel('Time (sec)');
94 ylabel('y');
95 title("Comaprison of step responses for continuous
      and discrete...
96  controllers",'fontsize',3)
```

```
97 exec .\fig_settings.sci; //custom script for setting
       figure properties
98 legend("continuous controller","digital controller"
      ,4)
99
100 //Control inputs
101 figure,
102 plot(ut.time,ut.values(:,1),2)
103 plot(ut.time,ut.values(:,2),'r--')
104 xlabel('Time (sec)');
105 ylabel('u');
106 title("Comaprison of control signals for continuous
      and discrete...
107  controllers",'fontsize',3)
108 exec .\fig_settings.sci; //custom script for setting
       figure properties
109 legend("continuous controller","digital controller")
110 //
```

This code can be downloaded from the website wwww.scilab.in check Appendix AP 2 for dependency:

`acker_dk.sci`

check Appendix AP 3 for dependency:

`zpk_dk.sci`

Scilab code Exa 7.33 `DC servo system redesign with modified with dominant second o`

```
1  //Example 7.33
2  // DC servo system redesign with modified with
       dominant second
3  // order pole locations.
4
5  xdel(winsid())//close all graphics Windows
6  clear;
7  clc;
8  //
     ____________________________________________________________________


9
10 // State space representation
11 //Transfer function model for DC Servo
12 s=poly(0,'s');
13 num=10;
14 den=s*(s+2)*(s+8);
15 Gs=syslin('c',num/den);
16
17 // State space representation
18 F=[-10 1 0;-16 0 1;0 0 0]
19 G=[0 0 10]';
20 H=[1 0 0];
21 J=0;
22 n=sqrt(length(F));
23 //Desired poles for the DC Servo system.
24 Pc=[-1.41+1.41*%i -1.41-1.41*%i -8]
25
26
27 // State feedback gain
28 K=ppol(F,G,Pc)
29 disp(K,'K=',"State feedback gain")
30
31 //Estimator - error roots are at
32 Pe=[-4.24+4.24*%i -4.24-4.24*%i -8]
33 exec .\acker_dk.sci;
34 Lt=ppol(F',H',Pe);
35 L=clean(Lt');
```

```
disp(L,'L=',"Observer gain")
//Error in book, Gain values are different in book.
//
```

```
//Compensator Design
DK=-K*inv(s*eye(n,n)-F+G*K+L*H)*L;
DK=syslin('c',DK)
exec('./zpk_dk.sci', -1);
[pl,zr,Kp]=zpk_dk(DK*10);
disp(zr,"zeros",pl,"Poles",Kp*10,"Gain(includung
    system gain)")
Dcs=poly(zr,'s','roots')/poly(pl,'s','roots')
disp(Dcs,'Dcs=',"Compensator transfer function")
//
```

---

check Appendix AP 2 for dependency:

`acker_dk.sci`

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 7.34 `Servomechanism increasing the velocity constant through zero`

```
//Example 7.34
// Servomechanism, increasing the velocity constant
    through
// zero assignment.

xdel(winsid())//close all graphics Windows
clear;
clc;
```

```
8  //
   ------------------------------------------------------------------------

9
10 // State space representation
11 //Transfer function model for DC Servo
12 s=poly(0,'s');
13 num=1;
14 den=s*(s+1);
15 Gs=syslin('c',num/den);
16
17 // State space representation
18 F=[0 1;0 -1]
19 G=[0 1]';
20 H=[1 0];
21 J=0;
22 n=sqrt(length(F));
23 //Desired poles for the DC Servo system.
24 Pc=[-2 -2]
25
26 // State feedback gain
27 exec .\acker_dk.sci;
28 K=acker_dk(F,G,Pc)//Gain computed in book is
      incorrect.
29 disp(K,'K=',"State feedback gain")
30 //
   ------------------------------------------------------------------------

31 //Overall transfer function with reduced order
      estimator.
32 Gred=8.32*(0.096+s)/(0.1 +s)/(8 + 4*s+s^2)
33 Gred=syslin('c',Gred)
34 disp(Gred,'Ys/Rs',"Overall transfer function with
      reduced...
35  order estimator")
36
37 //Compensator
38 D=(0.096+s)*(s+1)/(4.08 +s)/(0.0196+s)
```

```
39 Ds=syslin('c',D*8.32)
40 disp(Ds,'Ds=',"Compensator transfer function")
41 //
   ------------------------------------------------------------------------


42 //root locus
43 figure(0)
44 evans(D*Gs,100) //Correct root locus
45 zoom_rect([-0.2 -0.1 0.1 0.1])
46 f=gca();
47 f.x_location = "origin"
48 f.y_location = "origin"
49 xset("color",2);
50 h=legend('');
51 h.visible = "off"
52 //Title, labels and grid to the figure
53 exec .\fig_settings.sci; // custom script for
      setting figure properties
54 title('Root locus of lag-lead compensation','
      fontsize',3);
55 //
   ------------------------------------------------------------------------


56 //Bode plot
57 figure(1)
58 bode(Ds*Gs,0.01/2/%pi,100/2/%pi,"rad") //Correct
      root locus
59
60 f=gca();
61 h=legend('');
62 h.visible = "off"
63 //Title, labels and grid to the figure
64 exec .\fig_settings.sci; //custom script for setting
       figure properties
65 title('Frequency response of lag-lead compensation',
      'fontsize',3);
66 //
   ------------------------------------------------------------------------
```

```
67 //step response of the system with lag compensation
68 t=0:0.1:5;
69 ylag=csim('step',t,8.32*Gs*D/(1+8.32*Gs*D));
70 figure
71 plot(t,ylag,2);
72 xlabel('Time (sec)');
73 ylabel('y');
74 title("Step response of the system with lag
      compensation",'fontsize',3)
75 exec .\fig_settings.sci; //custom script for setting
       figure properties
76 //
    ---------------------------------------------------------------------------

77 //Discrete-time controller
78 sysDc_ss=syslin('c',tf2ss(Ds));
79 ts=0.1;
80 sysDd=dscr(sysDc_ss,ts)
81 Gdz=ss2tf(sysDd)
82
83 disp(Gdz,"Discrete-time compensator")
84 //
    ---------------------------------------------------------------------------

85 //step responses comparision
86 importXcosDiagram(".\Ex7_34_model.xcos")
87
88 xcos_simulate(scs_m,4);
89 scs_m.props.context
90 figure,
91 plot(yt.time,yt.values(:,1),2)
92 plot(yt.time,yt.values(:,2),'r--')
93 xlabel('Time (sec)');
94 ylabel('y');
95 title("Comaprison of step responses for continuous
      and discrete...
96  controllers",'fontsize',3)
```

```
97 exec .\fig_settings.sci; //custom script for setting
       figure properties
98 legend("continuous controller","digital controller"
      ,4)
99
100 //Control inputs
101 figure,
102 plot(ut.time,ut.values(:,1),2)
103 plot(ut.time,ut.values(:,2),'r--')
104 xlabel('Time (sec)');
105 ylabel('u');
106 title("Comaprison of control signals for continuous
      and discrete...
107  controllers",'fontsize',3)
108 exec .\fig_settings.sci; //custom script for setting
       figure properties
109 legend("continuous controller","digital controller")
110 //
```

This code can be downloaded from the website wwww.scilab.in check Appendix AP 1 for dependency:

```
fig_settings.sci
```

Scilab code Exa 7.35 Integral Control of a Motor Speed System

```
1 //Example 7.35
2 // Integral Control of a Motor Speed System
```

```
3
4  xdel(winsid())//close all graphics Windows
5  clear;
6  clc;
7  //
   ------------------------------------------------------------------------

8
9  //Transfer function model
10 num=1;
11 s=poly(0,'s');
12 den=(s+3);
13 G=syslin('c',num/den);
14 sys=tf2ss(G)
15
16 // State space representation of augmented system
17 F=[0 1; 0 -3];
18 G=[0 1]'
19 H=[1 0];
20 J=0;
21
22 //Desired poles for augmented system
23 Pc=[-5 -5]
24
25 // State feedback gain is
26 K=ppol(F,G,Pc)
27 disp(K,'K=')
28
29 //Estimator
30 Pe=[-10]
31 L=ppol(sys.A',sys.C',Pe)
32 disp(L','L=')
33
34 //
   ------------------------------------------------------------------------

35 //(c) Compare step reference and disturbance
      response.
```

```
36 //step reference response switch r=1 and w=0;
37 r=1;w=0;
38 importXcosDiagram(".\Ex7_35_model.xcos")
39  //The diagram data structure
40 xcos_simulate(scs_m,4);
41 scs_m.props.context
42 figure(0)
43 plot(yt.time,yt.values)
44 xlabel('time');
45 ylabel('y');
46
47 figure(1)
48 plot(ut.time,ut.values)
49 xlabel('time');
50 ylabel('y');
51 //
    ------------------------------------------------------------------------

52 // Step disturbance response switch r=0 and w=1;
53 w=1;r=0;
54 importXcosDiagram(".\Ex7_35_model.xcos")
55  //The diagram data structure
56 xcos_simulate(scs_m,4);
57 scs_m.props.context
58
59 scf(0)
60 plot(yt.time,yt.values,'r--')
61 xlabel('time');
62 ylabel('y');
63 title("step Response",'fontsize',3)
64 exec .\fig_settings.sci; // custom script for
     setting figure properties
65 legend("y1","y2")
66 xset('font size',3);
67 xstring(0.9,0.9,"$y_1$");
68 xstring(0.25,0.12,"$y_2$");
69
70
```

```
71 scf(1)
72 plot(ut.time,ut.values,'r--')
73 xlabel('time');
74 ylabel('y');
75 title("Control efforts",'fontsize',3)
76 exec .\fig_settings.sci; // custom script for
      setting figure properties
77 legend("u1","u2")
78 xset('font size',3);
79 xstring(0.25,2.5,"$u_1$");
80 xstring(1,-1,"$u_2$");
81 //
```

This code can be downloaded from the website wwww.scilab.in

Chapter 8

Digital Control

check Appendix AP 1 for dependency:

```
fig_settings.sci
```

Scilab code Exa 8.1 Digital Controller using tustin approximation

```
1 ///Example 8.1
2 // Digital Controller using tustin approximation.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
  ------------------------------------------------------------------------

8 //Cntroller
9 s=poly(0,'s');
10 numD=s/2+1;
11 denD=s/10+1;
12 D=10*numD/denD;
13 Ds=syslin('c',D);
14 //sampling freq. = 25 times bandwidth
15 Wbw=10;
```

```
16 Ws=25*Wbw;
17 fs=Ws/2/%pi;
18 T=1/fs; //sampling time
19 a=1;b=-1;
20 c=1;d=1;
21 //Digital controller
22 z=poly(0,'z');
23 Dz=horner(Ds,2/T*(a*z+b)/(c*z+d));
24 disp(Dz,'Digital Controller : ')
25
26 //
    ------------------------------------------------------------------------

27 //step response and control efforts.
28 figure(0);
29 importXcosDiagram(".\Ex8_1_model.xcos")
30  //The diagram data structure
31 xcos_simulate(scs_m,4);
32 scs_m.props.context
33 plot(yt.time,yt.values(:,1),'r--')
34 plot(yt.time,yt.values(:,2),2)
35
36 xlabel('Time (sec.)');
37 ylabel('Position, y');
38 title(["Comparison between digital and continuous
      controller step...
39  response";"with a sample rate 25 times bandwidth";"
      (a) Position "],...
40  'fontsize',3);
41 exec .\fig_settings.sci; // custom script for
     setting figure properties
42
43 //control effort
44
45 figure(1);
46 plot(ut.time,ut.values(:,1),'r--')
47 plot2d2(ut.time,ut.values(:,2),2)
48
```

```
49 xlabel('Time (sec.)');
50 ylabel('Control, u');
51 title(["Comparison between digital and continuous
      controller step...
52  response";"with a sample rate 25 times bandwidth";"
      (b) Control "],...
53  'fontsize',3);
54 exec .\fig_settings.sci; // custom script for
      setting figure properties
55 //
```

This code can be downloaded from the website wwww.scilab.in check Appendix AP 1 for dependency:

```
fig_settings.sci
```

Scilab code Exa 8.2 Design of a Space Station Attitude Digital Controller using Di

```
1 //Example 8.2
2 // Design of a Space Station Attitude Digital
     Controller using
3 // Discrete Equivalents
4
5 xdel(winsid())//close all graphics Windows
6 clear;
7 clc;
8 //
```

```
// State space representation of continuous time
   system
s=poly(0,'s');
num=1;
den=(s^2);
Gs=syslin('c',num/den);
Ds=0.81*(s+0.2)/(s+2);
Ds=syslin('c',Ds);
sysc=Gs*Ds;

//Root locus
evans(sysc)
zoom_rect([-2 -0.4 0.5 0.4])
f=gca();
f.x_location = "origin"
f.y_location = "origin"
h=legend('');
h.visible = "off"
exec .\fig_settings.sci; //custom script for setting
    figure properties
title('s-plane locus with respect to K','fontsize'
   ,3)
//
   ------------------------------------------------------------------------

//Contonuous time response of the system
figure,
tc=0:0.1:30;
syscl=sysc/(1+sysc)
yc=csim("step",tc,syscl);
plot(tc,yc,'b')
//
   ------------------------------------------------------------------------

// Discretization of the system at
z=poly(0,'z')
// sampling time Ts=1 sec
Ts=1;
```

```
40 Dz1=horner(Ds,2/Ts*(z-1)/(z+1))
41 disp(Dz1,"Dz1=","Discrete-time controller with Ts=1
      sec.")
42
43 // sampling time Ts=0.5 sec
44 Ts2=0.5;
45 Dz2=horner(Ds,2/Ts2*(z-1)/(z+1))
46 disp(Dz2,"Dz2=","Discrete-time controller with Ts
      =0.5 sec.")
47
48 //discrete-time response of the system.
49
50 importXcosDiagram(".\Ex8_2_model.xcos")
51  //The diagram data structure
52 xcos_simulate(scs_m,4);
53 //scs_m.props.context
54 plot(yt1.time,yt1.values,'m-.') //with Ts=1sec.
55 plot(yt2.time,yt2.values,'r--') //with Ts=0.5 sec.
56 //
   ------------------------------------------------------------------------


57
58 title('step responses of continous and digital
      implementations','fontsize',3)
59
60 exec .\fig_settings.sci; // custom script for
      setting figure properties
61 xlabel('Time (sec)','fontsize',2)
62 ylabel('Plant output','fontsize',2)
63 legend("Continuous design","Discrete equivalent
      design, T=1 sec."...
64 ,"Discrete equivalent design, T=0.5 sec.",4)
65 //
   ------------------------------------------------------------------------
```

This code can be downloaded from the website wwww.scilab.in

Chapter 9

Nonlinear Systems

check Appendix AP 1 for dependency:

fig_settings.sci

Scilab code Exa 9.5 Changing Overshoot and Saturation nonlinearity

```
1 //Example 9.5
2 //Changing Overshoot and Saturation nonlinearity.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
  ------------------------------------------------------------------------

8 //System transfer function and its root locus
9
10 s=poly(0,'s');
11 num=(s+1)
12 den=(s^2);
13 Gs=syslin('c',num/den)
14
15 //Root locus
```

```
16 evans(Gs,5)
17 title(["Root locus of", "$(s+1)/(s^2)$","with
      saturation removed"],...
18 'fontsize',3);
19 f=gca();
20 f.x_location = "origin"
21 f.y_location = "origin"
22 h=legend('');
23 h.visible = "off"
24 exec .\fig_settings.sci; //custom script for setting
      figure properties
25 //
   ____________________________________________________________________

26 // Step response
27 K=1;
28 i=[2 4 6 8 10 12];
29 figure(1);
30 importXcosDiagram(".\Ex9_5_model.xcos")
31
32 for r=i
33 xcos_simulate(scs_m,4);
34 scs_m.props.context
35 plot(yt.time,yt.values)
36 end
37
38 xlabel('time');
39 ylabel('y');
40 title("Step response of the system for various input
      sizes",'fontsize',3);
41 exec .\fig_settings.sci; //custom script for setting
      figure properties
42
43 xset('font size',3);
44 xstring(4,2.5,"$r=2$");
45 xstring(6,5.5,"$4$");
46 xstring(8,8.7,"$6$");
47 xstring(10,12.2,"$8$");
```

```
48 xstring(12,15.4,"$10$");
49 xstring(14,18.4,"$12$");
50 //
```

This code can be downloaded from the website wwww.scilab.in check Appendix AP 1 for dependency:

```
fig_settings.sci
```

Scilab code Exa 9.6 Stability of conditionally stable system using root locus

```
1 //Example 9.6
2 //Stability of conditionally stable system using
    root locus.
3 xdel(winsid())//close all graphics Windows
4 clear;
5 clc;
6 //
```

```
7 //System transfer function and its root locus
8
9 s=poly(0,'s');
10 num=(s+1)^2
11 den=(s^3);
12 Gs=syslin('c',num/den)
13 //Root locus
14 evans(Gs,7)
```

```
15 title(["Root locus for", "$(s+1)^2/(s^3)$","for
      system"],...
16 'fontsize',3);
17 f=gca();
18 f.x_location = "origin"
19 f.y_location = "origin"
20 h=legend('');
21 h.visible = "off"
22 exec .\fig_settings.sci; //custom script for setting
       figure properties
23 //
   ------------------------------------------------------------------------

24 //Response of the system
25 K=2;
26 i=[1 2 3 3.475];
27 figure(1);
28
29 importXcosDiagram(".\Ex9_6_model.xcos")
30
31 for r=i
32 xcos_simulate(scs_m,4);
33 scs_m.props.context
34 plot(yt.time,yt.values)
35 end
36
37 xlabel('Time (sec.)');
38 ylabel('Amplitude');
39 title("Step response of the system",'fontsize',3);
40
41 exec .\fig_settings.sci; //custom script for setting
       figure properties
42 xset('font size',3);
43 xstring(3,6.5,"$r=3.475$");
44 xstring(2.5,5.2,"$3$");
45 xstring(2,3,"$2$");
46 xstring(1,1.4,"$1$");
47 //
```

This code can be downloaded from the website wwww.scilab.in check Appendix AP 1 for dependency:

```
fig_settings.sci
```

Scilab code Exa 9.7 Analysis and design of the system with limit cycle using the r

```
1 //Example 9.7
2 //Analysis and design of the system with limit cycle
      using the root locus.
3 xdel(winsid())//close all graphics Windows
4 clear;
5 clc;
6 //
    ------------------------------------------------------------------------

7 //System transfer function and its root locus
8
9 s=poly(0,'s');
10 num=0.1;
11 den=(s^2+0.2*s+1)*(s);
12 Gs=syslin('c',num/den);
13
14 //Root locus
15 evans(Gs,40)
16 title(["Root locus of", "$(0.1/s(s^2+0.2*s+1)$"],'
     fontsize',3);
```

```
17 f=gca();
18 f.x_location = "origin"
19 f.y_location = "origin"
20 h=legend('');
21 h.visible = "off"
22 exec .\fig_settings.sci; // custom script for
      setting figure properties
23 //
   ------------------------------------------------------------------------

24 //Response of the system
25 figure;
26 //Response of the system
27 K=0.5;
28 i=[1 4 8];
29 importXcosDiagram(".\Ex9_7_model.xcos")
30
31 for r=i
32 xcos_simulate(scs_m,4);
33 scs_m.props.context
34 plot(yt.time,yt.values)
35 end
36
37 xlabel('Time (sec.)');
38 ylabel('Amplitude');
39 title("Step response of the system",'fontsize',3);
40 exec .\fig_settings.sci; // custom script for
      setting figure properties
41 zoom_rect([0 0 150 9])
42
43 xset('font size',3);
44 xstring(80,1.6,"$r=1$");
45 xstring(80,4.6,"$r=4$");
46 xstring(80,8.2,"$r=8$");
47 //
   ------------------------------------------------------------------------

48 //System with notch compensation
```

```
49 D=123*(s^2+0.18*s+0.81)/(s+10)^2;
50
51 //Root locus
52 figure,
53 evans(Gs*D,40)
54 title(["Root locus including notch compensation"],'
     fontsize',3);
55 f=gca();
56 f.x_location = "origin"
57 f.y_location = "origin"
58 h=legend('');
59 h.visible = "off"
60 exec .\fig_settings.sci; //custom script for setting
     figure properties
61 zoom_rect([-14 -2 2 2])
62 //
     ------------------------------------------------------------------------

63 //Response of the system witth notch filter
64 figure;
65 K=0.5;
66 i=[2 4];
67 importXcosDiagram(".\Ex9_7_model_notch.xcos")
68
69 for r=i
70 xcos_simulate(scs_m,4);
71 scs_m.props.context
72 plot(yt.time,yt.values)
73 end
74
75 xlabel('Time (sec.)');
76 ylabel('Amplitude');
77 title("Step response of the system with notch filter
     ",'fontsize',3);
78 exec .\fig_settings.sci; //custom script for setting
     figure properties
79 xset('font size',3);
80 xstring(30,2.2,"$r=2$");
```

```
81 xstring(34,3.75,"$r=4$");
82 //
```

This code can be downloaded from the website wwww.scilab.in This code

can be downloaded from the website wwww.scilab.in check Appendix AP 1 for dependency:

```
fig_settings.sci
```

Scilab code Exa 9.8 Antiwindup compensation for a PI controller

```
1 //Example 9.8
2 //Antiwindup compensation for a PI controller.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
```

```
8 //System Model
9
10 //Response of the system
11 kp=2;
12 ki=4;
13
```

```
14 //Without antiwindup
15 ka=0;
16 importXcosDiagram(".\Ex9_8_model.xcos")
17 xcos_simulate(scs_m,4);
18 scs_m.props.context
19 figure(0)
20 plot(yt.time,yt.values,'m-.')
21 figure(1)
22 plot(ut.time,ut.values,'m-.')
23
24 //With antiwindup
25 ka=10;
26 xcos_simulate(scs_m,4);
27 scf(0)
28 plot(yt.time,yt.values)
29 exec .\fig_settings.sci; // custom script for
      setting figure properties
30 xlabel('Time (sec.)');
31 ylabel('Output');
32 title("Integrator antiwindup (a) step response.",'
      fontsize',3);
33
34
35 scf(1)
36 plot(ut.time,ut.values);
37 exec .\fig_settings.sci; // custom script for
      setting figure properties
38 xlabel('Time (sec.)');
39 ylabel('Control');
40 title("Integrator antiwindup (b) Control effort.",'
      fontsize',3);
41 zoom_rect([0 -1.2 10 1.2])
42
43 //
```

This code can be downloaded from the website wwww.scilab.in check Appendix AP 1 for dependency:

```
fig_settings.sci
```

Scilab code Exa 9.9 Describing Function for a saturation nonlinearity

```
1 //Example 9.9
2 //Describing Function for a saturation nonlinearity.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
    ------------------------------------------------------------------------

8 //Response of the saturation nonlinearity to
     sinusoidal input
9 figure;
10 importXcosDiagram(".\Ex9_9_model.xcos")
11 xcos_simulate(scs_m,4);
12 scs_m.props.context
13 plot(yt.time,yt.values(:,1),'r--')
14 plot(yt.time,yt.values(:,2),'b')
15
16 xlabel('Time (sec.)');
17 ylabel('Amplitude');
18 title("Saturation nonlinearity output to sinusoidal
     input",...
```

```
'fontsize',3);
exec .\fig_settings.sci; //custom script for setting figure properties
//-------------------------------------------------------------------------

//Describing Functin for saturation nonlinearity.
k=1;
N=1;
i=1;
Keq=[];

for a=0:0.2:10
    if k*a/N > 1 then
    Keq(i,1)=2/%pi*(k*asin(N/a/k)+N/a*sqrt(1-(N/k/a)^2))
    else
    Keq(i,1)=k
    end
    i=i+1;
end

a=0:0.2:10;
a=a';
figure,
plot(a,Keq)
xlabel('$a$');
ylabel('$K_{eq}}$');

xset('font size',3);
title("Describing Function for a saturation nonlinearity...
 with k=N=1",'fontsize',3);
exec .\fig_settings.sci; //custom script for setting figure properties
zoom_rect([0 0 10 1.1])
//-------------------------------------------------------------------------
```

This code can be downloaded from the website wwww.scilab.in check Appendix AP 1 for dependency:

```
fig_settings.sci
```

Scilab code Exa 9.11 Describing Function for a relay with hysteresis non linearity

```
1 //Example 9.11
2 //Describing Function for a relay with hysteresis
     nonlinearity.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
     ____________________________________________________________

8 //Response of the saturation noninearity to
     sinusoidal input
9 figure;
10 importXcosDiagram(".\Ex9_11_model.xcos")
11 xcos_simulate(scs_m,4);
12 scs_m.props.context
13 plot(yt.time,yt.values(:,1),'r--')
14 plot(yt.time,yt.values(:,2),'b')
15
16 xlabel('Time (sec.)');
17 ylabel('Amplitude');
```

```
18 title("Relay with hysteresis nonlinearity output to
      sinusoidal...
19  input",'fontsize',3);
20 exec .\fig_settings.sci; //custom script for setting
       figure properties
21 zoom_rect([0 -1.2 5 1.2])
22 //
     ____________________________________________________________


23 ////Describing Functin for relay with hysteresis
      nonlinearity.
24 h=0.1;
25 N=1;
26 i=1;
27
28 for a=0.1:0.025:1
29     if a<h then
30         Keq(i,1)=0;
31         ro(i,1)=0;
32         theta(i,1)=0
33     else
34         Keq(i,1)=4*N/(%pi*a)*(sqrt(1-(h/a)^2)-%i*h/a
              )
35         [r th]=polar(Keq(i,1));
36         ro(i,1)=r; //magnitude
37         theta(i,1)=clean(th); //angle in radians
38     end
39     i=i+1;
40 end
41
42 a=0.1:0.025:1
43 a=a';
44 figure,
45
46 subplot(2,1,1), plot(a,ro)
47 xlabel('$a$');
48 ylabel(['Magnitude', '$|K_{eq}|$']);
49
```

```
50 xset('font size',3);
51 exec .\fig_settings.sci; //custom script for setting
       figure properties
52 title("Describing Functin for relay with hysteresis
      nonlinearity...
53  with h=0.1 and N=1",'fontsize',3);
54
55 subplot(2,1,2), plot(a,theta*180/%pi)
56 xlabel('$a$');
57 ylabel(['Phase', '$ \angle K_{eq}$','deg.']);
58 xset('font size',3);
59 exec .\fig_settings.sci; //custom script for setting
       figure properties
60 //
```

This code can be downloaded from the website wwww.scilab.in check Appendix AP 1 for dependency:

```
fig_settings.sci
```

Scilab code Exa 9.12 `Conditionally stable system`

```
1 //Example 9.12
2 //Conditionally stable system.
3 xdel(winsid())//close all graphics Windows
4 clear;
5 clc;
```

```
6  //
   ____________________________________________________________________

7  //System transfer function and its root locus
8
9  s=poly(0,'s');
10 num=0.1;
11 den=(s^2+0.2*s+1)*(s);
12 Gs=syslin('c',num/den)
13
14 //Nyquist plot of the system
15 nyquist(Gs,0.035,10)
16 title("Nyquist plot and describing function to
      determine limit...
17  cycle",'fontsize',3);
18
19 f=gca();
20 f.x_location = "origin"
21 f.y_location = "origin"
22 h=legend('');
23 h.visible = "off"
24 xset("color",2);
25
26 // Nyquist Plot of Describing Function for
      saturation nonlinearity.
27 omegat=0.05:0.05:%pi;
28 a=sin(omegat);
29 N=0.1;
30 k=1;
31
32 Keq=2/%pi*(k*asin(N ./a /k)+N ./a .* sqrt(1-(N/k ./a
      ) .^2));
33 DF_nyq=-1 ./Keq;
34
35 plot(DF_nyq,zeros(1,length(DF_nyq)),'m-.')
36 exec .\fig_settings.sci; //custom script for setting
       figure properties
37 zoom_rect([-0.8 -0.5 0.2 0.5])
```

```
38
39 //limit cycle points
40 plot(-0.5,0,'bo');
41
42 xset('font size',3)
43 xstring(-0.78,0.08,"limit cycle point");
44 xarrows([-0.6;-0.52],[0.1;0.02],-1)
45 xstring(-0.62,-0.22,"$-\frac{1}{K_{eq}$");
46 xarrows([-0.55;-0.55],[-0.1;0],-1)
47 //
    ------------------------------------------------------------------------

48 //Describing Functin for saturation nonlinearity.
49 Keq=[]
50 i=1;
51
52 for a=0:0.2:10
53     if k*a/N > 1 then
54     Keq(i,1)=2/%pi*(k*asin(N/a/k)+N/a*sqrt(1-(N/k/a)
         ^2))
55     else
56     Keq(i,1)=k
57     end
58     i=i+1;
59 end
60
61 a=0:0.2:10;
62 a=a';
63
64 figure,
65 plot(a,Keq)
66 xlabel('$a$');
67 ylabel('$K_{eq}$');
68
69 xset('font size',3);
70 title("Describing Function for a saturation
     nonlinearity...
71  with N=0.1 and k=1",'fontsize',3);
```

```
72 exec .\fig_settings.sci; //custom script for setting
       figure properties
73 zoom_rect([0 0 10 1.1])
74 //
```

---

check Appendix AP 1 for dependency:

`fig_settings.sci`

Scilab code Exa 9.13 `Determination of stability with a hysteresis nonlinearity`

```
1 //Example 9.13
2 //Determination of stability with a hysteresis
     nonlinearity.
3
4 xdel(winsid())//close all graphics Windows
5 clear;
6 clc;
7 //
```

```
8 //System Model
9 s=poly(0,'s');
10 num=1;
11 den=(s^2+s);
12 Gs=syslin('c',num/den);
13 //
```

```
14 //Nyquist Plot of the system
```

```
15 nyquist(Gs,0.25,3)
16
17 // Nyquist Plot of Describing Function for
      hysteresis nonlinearity
18 N=1;
19 h=0.1;
20 i=1;
21
22 for omegat=0:0.05:%pi-0.1;
23     a=sin(omegat);
24     DF_nyq(i,1)=-%pi/4/N*(sqrt(a^2-h^2) + h * %i)
25     i=i+1;
26 end
27
28 plot(real(DF_nyq),imag(DF_nyq),'m-.')
29 exec .\fig_settings.sci; // custom script for
      setting figure properties
30 zoom_rect([-0.3  -0.3 0 0.3])
31 title('Nyquist plot of system and describing
      function to...
32  determine limit cycle','fontsize',3)
33
34 //limit cycle points
35 plot(-0.1714,-0.0785,'ro');
36 xstring(-0.25,0,"limit cycle point");
37 xarrows([-0.2;-0.172],[0;-0.077],-1);
38
39 //
     ------------------------------------------------------------------------

40 //Response of the system
41 K=2;
42 r=1
43 figure(1);
44 importXcosDiagram(".\Ex9_13_model.xcos")
45 xcos_simulate(scs_m,4);
46 scs_m.props.context
47 plot(yt.time,yt.values)
```

```
48
49 xlabel('Time (sec.)');
50 ylabel('Output, y');
51 title("Step response displaying limit cycle
      oscillations",'fontsize',3);
52 exec .\fig_settings.sci; //custom script for setting
      figure properties
53 //
```

This code can be downloaded from the website wwww.scilab.in

Appendix

Scilab code AP 1 figure setting file

```
1 //
    ____________________________________________________________________

2 //figure handel settings
3 f=get("current_figure"); //Current figure handle
4 f.background=8; //make the figure window background
     white
5 l=f.children(1);
6 l.background=8 ;//make the text background white
7 id=color('grey');
8 xgrid(id);
9 //
    ____________________________________________________________________
```

Scilab code AP 2 State feedback gain matrix computation

```
1 //
    ____________________________________________________________________

2 //
    ____________________________________________________________________

3 //A function written by Deepti Khimani.
4 //Usage:-
5 //[K, lambda]=acker_dk(a, b, pl)
```

```
6  //K=acker_dk(a, b, pl)
7  //a:- System matrix.
8  //b:- input matrix.
9  //p:- Desired poles.
10 //K:-State feedback gain for the control law u=-Kx.
11 //lambda:- Eigen values of (a-b*k)
12 //
   ------------------------------------------------------------------------

13 //
   ------------------------------------------------------------------------

14
15 function [K, lambda]=acker_dk(a, b, pl)
16         [lhs,rhs]=argn(0)
17
18     if rhs == 0 then
19         disp(["K=acker_dk(a, b, pl)";"[K, lambda]=
              acker_dk(a, b, pl)"]);
20         disp(["a:- System matrix";"b:- input matrix"
              ;"p:- Desired poles"]);
21         disp(["K:-State feedback gain for the
              control law u=-Kx";...
22  "lambda:- Eigen values of (a-b*k)"]);
23         return;
24     end
25 [ra ca]=size(a);
26 [rb cb]=size(b);
27 l=length(pl);
28
29 CO=cont_mat(a,b);
30
31 if ra~=l then
32     error(["Dimension error:";"number of desired
          poles must equal...
33  to order of the system"]);
34 elseif ra~=ca then
35     error(["Dimension error:";"system matrix should
```

```
        be...
36   a sqaure matrix"]);
37  elseif rb~=ra then
38      error (["Dimension error:","Input matrix should
        be...
39   as many rows as a system matrix."]);
40  elseif rank(CO)<ra then
41      error("system is not controllable");
42  end
43  //
    ---------------------------------------------------------------------

44  //controllable canonical form
45  [Ac,Bc,T,ind]=canon(a,b);
46
47  //CO=zeros(ra,cb);
48  for i=1:ra
49      CO(:,ra+1-i)=Ac^(i-1)*Bc;
50  end
51  //
    ---------------------------------------------------------------------

52  chr_eq=poly(pl,'s');
53  des_chr_coeff=coeff(chr_eq);
54
55  des_chr_coeff=des_chr_coeff(1:ra);
56  alpha_c=Ac^ra;
57
58  for k=1:ra
59      alpha_c=alpha_c + des_chr_coeff(k)*Ac^(k-1)
60  end
61  //
    ---------------------------------------------------------------------

62  //State feedback gain
63  temp=zeros(1,ra);
64  temp(1)=1;
65  K=temp*inv(CO)*alpha_c;
```

```
66 K=K/T;
67 lambda=spec(a-b*K);
68 endfunction
69 //
```

Scilab code AP 3 ZPK computation

```
1  //
   ____________________________________________________________________

2  //
   ____________________________________________________________________

3  //A function written by Deepti Khimani.
4  //Usage:-
5  //p=zpk_dk(sl)
6  //[p, z]=zpk_dk(sl)
7  //[p, z, k]=zpk_dk(sl)
8  //p:- Poles of the system
9  //z:- zeros of the system
10 //k:- DC gain of the system
11 //
   ____________________________________________________________________

12 //
   ____________________________________________________________________

13
14 function[pl,zr,k]=zpk_dk(sysmodel)
15     [lhs,rhs]=argn(0)
16
17     if rhs == 0 then
18      disp(["p=zpk_dk(sl)";"[p, z]=zpk_dk(sl)";"[p, z
           , k]=zpk_dk(sl)"]);
19      disp(["p:- Poles of the system";"z:- zeros of
           the system"]);
```

```
20       disp("k:- DC gain of the system");
21       return;
22      end
23
24      if typeof(sysmodel)=="rational" then
25          sys=tf2ss(sysmodel);
26          pl=spec(sys.A);
27          zr=trzeros(sys);
28          temp1=poly(zr,'s','roots')/poly(pl,'s','
               roots');
29          temp2=sysmodel/temp1;
30          temp3=tf2ss(temp2);
31          k=temp3.D;
32      elseif typeof(sysmodel)=="state-space" then
33          pl=spec(sysmodel.A);
34          zr=trzeros(sysmodel);
35          g=ss2tf(sysmodel);
36          temp1=poly(zr,'s','roots')/poly(pl,'s','
               roots');
37          temp2=g/temp1;
38          temp3=tf2ss(temp2);
39          k=temp3.D
40      else
41          error("Wrong type of input argument.")
42      end
43  endfunction
```