Scilab Textbook Companion for Control Engineering - Theory & Practice by M. N. Bandyopadhyay[1]

Created by
Pooja Naik
B. E.
Instrumentation Engineering
Watumaull College of Electronics
College Teacher
Prof. Ashutosh Sharma
Cross-Checked by
Chaitanya

July 17, 2017

[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: Control Engineering - Theory & Practice

Author: M. N. Bandyopadhyay

Publisher: PHI Learning Pvt. Ltd.

Edition: 1

Year: 2009

ISBN: 9788120319547

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 Review of some mathematical tools	5
3 Transient and steady state behaviour of system	16
4 State variable analysis	18
5 stability of linear control system	23
6 study of the locus of the roots of the characristic equation	31
7 Analysis of frequency response	36
8 stability in frequency response systems	38
9 compensators and controllers	41
10 Non linear control system	44
11 Digital control system	47
15 Miscellaneous solved problems	51

List of Scilab Codes

Exa 2.1	solving differential equation using scilab . .	5
Exa 2.2	inverse of laplace transform using scilab . .	6
Exa 2.3	computing initial value using scilab	6
Exa 2.4.b	eigen values using scilab	7
Exa 2.4	computing initial value of transfer function .	7
Exa 2.4.1	eigen values using scilab	8
Exa 2.5	computing initial value of function F in scilab	9
Exa 2.6	computing final value of function F using scilab	9
Exa 2.7	Inverse laplace transform using scilab	10
Exa 2.8	Z transform of the signal	10
Exa 2.9	z transform of the signal using scilab	11
Exa 2.10	Inverse of z transform using scilab	12
Exa 2.11	Inverse of z transform by power expansion series	12
Exa 2.12	inverse of Z transform by partial fraction method	14
Exa 3.1	Type of the system	16
Exa 4.1	State equation	18
Exa 4.3	Eigen values	18
Exa 4.6.a	Canonical form	19
Exa 4.6.b	Canonical form	20
Exa 4.8	Jordan canonical form	21
Exa 4.10	Controllable companion form	21
Exa 5.1	Hurwitz stability test	23
Exa 5.2	Routh array	24
Exa 5.2.2a	Routh array	25
Exa 5.2.2b	Routh array	25
Exa 5.2.2c	Routh array	26
Exa 5.2.2d	Routh array	26

Exa 5.3	Routh array	27
Exa 5.4	Routh array	28
Exa 5.5	Routh array	28
Exa 5.6	Routh array	29
Exa 6.2	Root locus in scilab	31
Exa 6.2.2	location of the root locus between poles and zeros	32
Exa 6.3	Root locus	32
Exa 6.4	root locus	33
Exa 6.5	Root locus	34
Exa 6.6	Root locus	34
Exa 6.7	Root locus	35
Exa 7.3.1a	Bode plot	36
Exa 7.3.1b	Bode plot	36
Exa 8.1	Nyquist plot	38
Exa 8.2	Nyquist plot	38
Exa 8.3	Nyquist plot	39
Exa 9.1	compensation in open loop control system	41
Exa 10.1.1	Mass dashpot and spring arrangement	44
Exa 10.3	determination of quadratic form	45
Exa 10.4	Lipunovs method	45
Exa 11.6	Jurys stability test	47
Exa 11.9.2a	stability of linear continuous system	48
Exa 11.9.2b	stability of linear continuous system	49
Exa 11.9.3	Schurcohn stability test	50
Exa 15.2	Time domain specifications of second order system	51
Exa 15.4	transfer function of gyroscope	52
Exa 15.5	Transfer function of system	52
Exa 15.6	comparison of sensitivities of two systems	53
Exa 15.7	To find bandwidth of the transfer function	53
Exa 15.8	Bandwidth of the transfer function	54
Exa 15.9	Nyquist plot	55
Exa 15.12	solution of polynolynomial equation	55
Exa 15.13	Time domain specifications	56
Exa 15.14	Position servomotor	57
Exa 15.15	steady output speed of DC motor	57
Exa 15.26	Routh array	58

Exa 15.27	To check reachability of the system	59
Exa 15.28	Determine the stability of the system	60
Exa 15.31	Time domain specifications of second order system	60
Exa 15.33	Lipunovs method	61
Exa 15.34	Find bandwidth of the transfer function	62
Exa 15.36	Impulse response of the transfer function	62
Exa 15.37	step response of the transfer function	63
Exa 15.38	Roots of characteristic equation	63
Exa 15.39	Bode plot	64
Exa 15.40	Nyquist plot	64
Exa 15.41	Nyquist plot	65
Exa 15.42	Bode plot	65
Exa 15.43	Eigen values of matrix	66
Exa 15.44	State space representation of LTI system	66
Exa 15.45	Covariant matrix of A	67
Exa 15.49	Root locus of the transfer function	67
Exa 15.50	Bode plot	68
Exa 15.53	Statespace model of the differential equation	68
Exa 15.54	Nyquist plot	69
Exa 15.57	determination of zeta and wn	69
Exa 15.59	transfer function of signal flow graph	70
Exa 15.60	root locus	71

Chapter 2

Review of some mathematical tools

Scilab code Exa 2.1 solving differential equation using scilab

```
//Example 2.1
//solving differential equation in scilab
clear;clc;
xdel(winsid());

function ydot=f(t, y)
ydot=(10/4)-(3*y/4)
endfunction
y0=1;
t0=0;
t=0:5:10;
y=ode(y0,t0,t,f)
//since"t=0;5;10"
//the answer is calculated for t=0:5:10"
//thus the value of "y"can be calculated at any
   value of "t".
```

Scilab code Exa 2.2 `inverse of laplace transform using scilab`

```
//Example 2.2
//Inverse laplace transform using scilab
clear;clc;
xdel(winsid());
s=%s;
num=(s+6);
den=(s^2+2*s+10);
F1=syslin('c',num,den)
F=pfss(F1)
//since pfss(F1) is not able to factorise F1, therefore,
//Rewriting numerator as, (s+6)=(s+1+5);
//Rewriting the denominator as, (s^2+2*s+6)=(s+1)^2+3^2;
disp("F=[((s+1)/(s+1)^2+3^2)+(5/3)*(3/(s+1)^2+3^2)]")
//From the standard formula of inverse laplace transform;
//(s+1)/(s+1)^2+3^2=%e^-t*(cos3t);
//(5/3)*(3/(s+1)^2+3^2)=(5/3)*%e^-t*(sin3t);
disp("f(t)=(%e^-t)*(cos3t)+(5/3)*(%e^-t)*(sin3t)")
```

Scilab code Exa 2.3 `computing initial value using scilab`

```
//Example 2.3
//computing initial value using scilab
```

```
clear;clc;
xdel(winsid());
s=%s
n3=(3*s+2)
d3=s*(s^2+4*s+5)
F=n3/d3
//Applying initial value theorem
//when limit "s" tends to infinity, final value of "F" becomes "0"

disp(0,"Final value=")
```

Scilab code Exa 2.4.b eigen values using scilab

```
//Example sec 2.4.2
//eigen values
clear;clc;
xdel(winsid());

A=[0 6 -5;1 0 2;3 2 4]
B=spec(A)
disp(B,"Eigen values=")
```

Scilab code Exa 2.4 computing initial value of transfer function

```
//Example 2.4
//computing f'(0+) and f''(0+) using scilab
clear;clc;
xdel(winsid());
```

```
s=%s;
n4=(4*s+1);
d4=s*(s^2+4*s+5);
F=n4/d4
// As per initial value theorem, limit "t" tends to
   zero and limit "s" tends to infinity

//for f'(0+)
F1=s*F+0

for s=%inf
    disp("f''(0+)=4")
end
//for f''(0+)
s=%s;
F2=((s*(F1))-(4))

for s=%inf
    disp("f''''(0+)=-15")
end
```

Scilab code Exa 2.4.1 eigen values using scilab

```
//Example sec 2.4. a
//eigen values
clear;clc;
xdel(winsid());

A=[1 -1;0 -1]
B=spec(A)
disp(B,"Eigen values=")
```

Scilab code Exa 2.5 computing initial value of function F in scilab

```
//Example 2.5
//computing final value of function F using scilab
clear;clc;
xdel(winsid());

s=%s;
n5=(8*s+5);
d5=s*(s+1)*(s^2+4*s+5);
F=n5/d5
F1=s*F
//for final value limit "t" tends to infinity and
   limit "s" tends to zero.
//When s=0, the value of F1 will be "(5/5)=1"
for s=0
    disp("Final value=1")
end
```

Scilab code Exa 2.6 computing final value of function F using scilab

```
//Example 2.6
//computing final value of function F using scilab
clear;clc;
xdel(winsid());
s=%s;
n6=(5);
d6=s*(s^2+49);
```

```
F=n6/d6
F1=s*F

disp("THE SYSTEM HAS POLES ON IMAGINARY AXIS.
   THEREFORE f(t) HAS NO FINAL VALUE")
```

Scilab code Exa 2.7 Inverse laplace transform using scilab

```
//Example 2.7
//Inverse laplace transform of "2/(s^2*(s+1))" using
    scilab
clear;clc;
xdel(winsid());
s=%s;
num=2;
den=(s^2)*(s+1);
F1=syslin('c',num,den)
F=pfss(F1)
//from the partial fraction decomposition, taking
   out 2 as common term.
//The result would be in the form of"F(s)=2*(1/s
   ^2-1/s+(1/s+1))"
disp("F(s)=2*((1/s^2)-(1/s)+(1/(s+1)))")
//From the standard formula of inverse laplace
   transform;
//(1/s^2)=t;(1/s)=1;(1/(s+1))=%e^-t
disp("f(t)=2*(t-1+e^-t)")
```

Scilab code Exa 2.8 Z transform of the signal

```
//Example 2.8
//Z transform of the signal x(n)=(0.5)^n*u(n)
clear;clc;
xdel(winsid());

//u(n)is unit step input
n=2;
x=(0.5)^n;
m=1;
w=1;
phi=tand(0);
a=1;
theta=tand(45);
[cxz]=czt(x,m,w,phi,a,theta)
```

Scilab code Exa 2.9 z transform of the signal using scilab

```
//Example 2.9
//Z transform of the signal x(n)=(a)^n*u(n)+(b)^n*u
    (-n-1).
clear;clc;
xdel(winsid());

//u(n)is unit step input
//a=0.5 and b=0.6
n1=2;
x1=(0.5)^n1;
m1=1;
w1=1;
phi1=tand(0);
a1=1;
theta1=tand(45);
[X1]=czt(x1,m1,w1,phi1,a1,theta1)
```

```
n2=2;
x2=(0.6)^n2;
m2=1;
w2=-1;
phi2=tand(-45);
a2=1;
theta2=tand(45);
[X2]=czt(x2,m2,w2,phi2,a2,theta2)
X=X1+X2;
disp(X,"ans=")
```

Scilab code Exa 2.10 Inverse of z transform using scilab

```
//Example 2.10
//inverse Z transform of 1/(1-a*z^-1)
clear;clc;
xdel(winsid());

// a=1
function y=f(z);
    y=z/(z-1) //upon simplification of the given
        equation
endfunction
intc(1+%i,2-%i,f)
```

Scilab code Exa 2.11 Inverse of z transform by power expansion series

```
//Example 2.11
//inverse z transform by power series expansion
```

```
clear;clc;
xdel(winsid());

z=%z;
num=2;
den=(2-(3*z^-1)+z^-2);
X=syslin('c',num/den)
//dividing the numerator and denominator by 2
num1=1;
den1=1-(1.5*(z^-1))+(0.5*(z^-2));
X1=syslin('c',(num1)/(den1))
//when mod(z)>1
//developing series expansion in negative power of z
A1=(num1)-(den1)
 // multiplying the den2 by 1 and subtracting it from num1
B1=((1.5*(z^-1))-(0.5*(z^-2)))-((1.5*(z^-1))*den1)
// multiplying the den2 by (1.5*z^-1) and subtracting it from reminder of A1
C1=((1.75*z^-2)-(0.75*z^-3))-((1.75*z^-2)*den1)
// multiplying the den2 by (1.75*z^-2) and subtracting it from reminder of A1
D1=((1.875*z^-3)-(0.875*z^-4))-((1.875*z^-3)*den1)
 // multiplying the den2 by (1.875*z^-3) and subtracting it from reminder of A1
E1=((1.9375*z^-4)-(0.9375*z^-5))-((1.9375*z^-4)*den1)
// multiplying the den2 by (1.9375*z^-4) and subtracting it from reminder of A1
disp("x1(n)=1,1.5,1.75,1.875,1.9375,........")

//when mod(z)<0.5
//developing series expansion in positive power of z
A2=(num)-((2*z^2)*den) //multiplyong the den by 2*(z^2) and subtracting it from num
B2=A2-(6*z^3*den)
//multiplyong the den by 2*(z^2) and subtracting it from A2
```

```
C2=B2-(14*z^4*den)
//multiplyong the den by 2*(z^2) and subtracting it
   from B2
D2=C2-(30*z^5*den)
//multiplyong the den by 2*(z^2) and subtracting it
   from C2
E2=D2-(62*z^6*den)
//multiplyong the den by 2*(z^2) and subtracting it
   from D2
disp("X2(z)=2*z^2+6*z^3+14*z^4+30*z^5+62*z
   ^6+........")
disp("x2(n)=.....,62,30,14,6,2,0,0")
```

Scilab code Exa 2.12 `inverse of Z transform by partial fraction method`

```
//Example 2.12
//inverse z transform by partial fraction method
clear;clc;
xdel(winsid());

z=%z;
num=1;
den=((1-z^-1)^2)*(1+z^-1);
X=syslin('c',num/den)
X1=X/z
pfss(X1)
// by partial fraction the X1 will be factorised as
   (in terms of z)
disp("X(z)=(0.25*z/(z+1))+(0.75*z/(z-1))+(0.5*z/(z
   -1)^2)")
disp("X(z)=(0.25/(1+z^-1))+(0.75/(1-z^-1))+(0.5*z/(z
   -1)^2)")
// 0.25/(1+z^-1) is the z transform of "0.25*(-1)^n*
```

```
    u(n)"
// (0.75/(1-z^-1)) is the z transform of "0.75*u(n)"
//(0.5*z/(z-1)^2) is the z transform of "0.5*n*u(n)"
disp("x(n)=0.25*((-1)^n)*u(n)+0.75*u(n)+0.5*n*u(n)")
```

Chapter 3

Transient and steady state behaviour of system

Scilab code Exa 3.1 Type of the system

```
 //Example 3.1
 //type of the system
clear; clc;
xdel(winsid());

// fig(3.14)
s=%s;
n1=(2);
d1=((s)*(s^2+2*s+2));
A=n1/d1
disp("since one integration is being observed, it is
    TYPE 1 system")

//fig(3.15)
s=%s;
n2=(5);
d2=((s+2)*(s^2+2*s+3));
B=n2/d2
disp("since no integration is being observed, it is
```

```
TYPE 0 system")

//fig(3.16)
s=%s;
n3=(s+1);
d3=((s^2)*(s+2));
C=n3/d3
disp("since two integration is being observed, it is TYPE 2 system")
```

Chapter 4

State variable analysis

Scilab code Exa 4.1 State equation

```
//Example 4.1
//state equation
clear;clc;
xdel(winsid());

A=[0 1;-2 -3]
B=[0;1]
C=[0]
[Ac Bc U ind]=canon(A,B);
disp(clean(Ac),'Ac=');
disp(clean(Bc),'Bc=');
disp(U,'transformation matrix U=');
```

Scilab code Exa 4.3 Eigen values

```
//Example 4.3
```

```
//for given matrix "A" proving eigen values of "A"="
    t^-1*A*T"
clear;clc;
xdel(winsid());
A=[0 1 0;0 0 1;-6 -11 -6]
P=bdiag(A) //eigen values of "A"

T=[1 1 1;-1 -2 -3; 1 4 9] //vandermode matrix
inv(T)

A1=inv(T)*A*T  //diagonal canonical form of A

//thus "P=A1" is proved.
```

Scilab code Exa 4.6.a Canonical form

```
//Example sec 4.6a
//example of canonical form
clear;clc;
xdel(winsid());
A=[1 2 1;0 1 3;1 1 1];
B=[1;0;1];
C=[1 0 0;0 1 0;0 0 1]
S=cont_mat(A,B)
s=%s;
D=s*C-A
det(D)

//the characteristic equation i.e. det(D)=s^3-3*s^2-
    s-3=0 is of the form of
//s^3+a2*S^2+a1*s+a0=0. therefore comparing two
    equation.

```

```
a2=-3
a1=-1
a0=-3
M=[a1 a2 1;a2 1 0;1 0 0]

P=S*M
A1=inv(P)*A*P
B1=inv(P)*B
```

Scilab code Exa 4.6.b Canonical form

```
//Example sec 4.6b
//example of canonical form
clear;clc;
xdel(winsid());
A=[1 2 1;0 1 3;1 1 1];
B=[1;0;1];
C=[1 1 0];
V=[C;C*A;C*A^2]

D=eye(3,3)
s=%s
E=s*D-A
det(E)

//the characteristic equation i.e. det(E)=s^3-3*s^2-
    s-3=0 is of the form of
//s^3+a2*S^2+a1*s+a0=0. therefore comparing two
    equation.

a2=-3
a1=-1
a0=-3
```

```
M=[a1 a2 1;a2 1 0;1 0 0]
F=M*V
Q=inv(F)
A1=inv(Q)*A*Q
B1=inv(Q)*B
C1=C*Q
```

Scilab code Exa 4.8 Jordan canonical form

```
//Example sec 4.8
//Jordan canonical form
clear;clc;
xdel(winsid());
A=[0 6 -5;1 0 2;3 2 4]
B=spec(A)
//Eigen vectors corresponding to eigen values of A
   are
p1=[2;-1;-2];
p2=[1;-0.4285;-0.7142];
p3=[1;-0.4489;-0.93877];
T=[p1 p2 p3];
A1=inv(T)*A*T
```

Scilab code Exa 4.10 Controllable companion form

```
//Example sec 4.10
//Controllable companion form
clear;clc;
xdel(winsid());
```

```
A=[1 0 0;0 2 0;0 0 3]
B=[1 0;0 1;1 1]
b1=[1;0;1]
b2=[0;1;1]
u=[B A*B A^2 B]
u1=[1 0 1;0 1 0;1 1 3]
// u1 is arranged from [b1 A^(v1-1)*b1 A^(v2-1)*b2]
// v1 and v2 are controllability indices.
u1=[b1 A*b1 b2]
v1=2;
v2=1;
inv(u1)

p1=[-0.5 -0.5 0.5]
p2=[0 1 0]
P=[p1;p1*A;p2]
A1=P*A*inv(P)
B1=P*B
C=eye(3,3)
s=%s
D=s*C-A1
E=det(D)
routh_t(E)
//to get equation E, A must be equal to
A2=[0 1 0;0 0 1;-6 -11 -6]
B2=[0 0;1 0.5;0 1]
N1=[6 1.5 4.5;-6 -11 8]
N=N1*P
```

Chapter 5

stability of linear control system

Scilab code Exa 5.1 Hurwitz stability test

```
//Example 5.1
//Hurwitz stability test in scilab
clear;clc;
xdel(winsid());

s=%s
A=s^4+8*s^3+18*s^2+16*s+4 //characteristic equation

//coefficients of characteristic equation
a0=det(coeff(A,4))
a1=det(coeff(A,3))
a2=det(coeff(A,2))
a3=det(coeff(A,1))
a4=det(coeff(A,0))

D=[a1 a0 0 0;a3 a2 a1 a0;0 a4 a3 a2;0 0 0 a4]//
   Hurwitz determinant

//minors of hurwitz determinant
```

```
D1=[a1]
det(D1)
D2=[a1 a0;a3 a2]
det(D2)
D3=[a1 a0 0;a3 a2 a1;0 a4 a3]
det(D3)
D4=[a1 a0 0 0;a3 a2 a1 a0;0 a4 a3 a2;0 0 0 a4]
det(D2)
```

Scilab code Exa 5.2 Routh array

```
//Example 5.2
//constructing Routh array in scilab
 clear; clc;
xdel(winsid());
mode(0);

s=%s;

A=s^4+4*s^3+4*s^2+3*s; // characteristic equation

k=poly(0,'k')

routh_t((1)/A,poly(0,'k'))
disp("0<k<2.4375")

//the function will automatically computes Routh array
//from the Routh array the value of "k" lies between 0 and 2.4375
```

Scilab code Exa 5.2.2a `Routh array`

```
//Example sec 5.2.2 a
//Routh array in scilab
clear;clc;
xdel(winsid());

s=poly(0,'s')
A=s^5+s^4+2*s^3+2*s^2+4*s+6
routh_t(A)
```

Scilab code Exa 5.2.2b `Routh array`

```
//Example sec 5.2.2 b
//Routh array in scilab
clear;clc;
xdel(winsid());

s=poly(0,'s')
B=s^5+2*s^4+6*s^3+12*s^2+8*s+16
routh_t(B)
// In this example a row of zero forms at s^3.
//The function automatically the derivative of the
//auxillary polynomial 2*s^4+12*s^2+16
//viz=8*s^3+24*s
```

Scilab code Exa 5.2.2c Routh array

```
//Example sec 5.2.2 c
//Routh array in scilab
clear;clc;
xdel(winsid());

s=poly(0,'s')
p=poly(0,'p')
C=s^5+s^4+2*s^3+2*s^2+3*s+5

//substituting "s=(1/p)" in B
//The resulting characteristic equation is

C1=5*p^5+3*p^4+2*p^3+2*p^2+p+1
routh_t(C1)
```

Scilab code Exa 5.2.2d Routh array

```
//Example sec 5.2.2 d
//Routh array in scilab
clear;clc;
xdel(winsid());

s=poly(0,'s')
D=2*s^6+2*s^5+3*s^4+3*s^3+2*s^2+s+1
routh_t(D)

```

```
D1=s^2+1
//dividing the main polynomial D by the auxillary
   polynomial D1
D/D1
D2=2*s^4+2*s^3+s^2+s+1
routh_t(D2)
```

Scilab code Exa 5.3 Routh array

```
// Example 5.3
 // Constructing Routh array in scilab

clear;clc
xdel(winsid());
mode(0);

s=%s;

A=s^4+4*s^3+4*s^2+3*s;  // characteristic equation
   after simplification

k=poly(0,'k')

routh_t((1)/A,poly(0,'k'))

//system will construct Routh array and
//from Routh array "k" must lie between 0&39/16 i.e
   (0<k<2.4375)

disp("0<k<39/16")
```

Scilab code Exa 5.4 Routh array

```
//example5.4
//constructing Routh array in scilab
clear;clc
xdel(winsid());//close all windows
mode(0);
s=%s;
A=s^3+8*s^2+26*s+40;

//consider p-plane is located to the left of the s-
    plane.
//distance between p-plane and s-plane is 1.
//if the origin is shifted from s-plane to the p-
    plane,then, s=p-1

z=%z
B=z^3+5*z^2+13*z+21;//substituting s=p-1 in the
    equation of A, the resulting equation will be
routh_t(B)
```

Scilab code Exa 5.5 Routh array

```
//Example 5.5 a
//constructing Routh array in scilab
clear;clc
xdel(winsid());//close all windows
mode(0);
```

```
s=%s;
A=s^3+s^2-s+1
routh_t(A)

//Example 5.5 b

s=%s;
B=s^4-s^2-2*s+2
routh_t(B)
//in this example 0 occurs in the first column of
   the array
// for which system assumes any small value "eps"
   and computes the array automatically.
```

Scilab code Exa 5.6 Routh array

```
//Example 5.6
//constructing Routh array in scilab
 clear; clc;
xdel(winsid());
mode(0);

s=%s;

A=s^4+8*s^3+24*s^2+32*s; //characteristic equation
k=poly(0,'k')

routh_t((1)/A,poly(0,'k'))
disp(k=80)

//since from the fourth row of the Routh array
// the positive value "k=80" will give roots with
   zero real part.
```

Chapter 6

study of the locus of the roots of the characristic equation

Scilab code Exa 6.2 `Root locus in scilab`

```
//Example 6.2
// Plotting root locus
clear; clc;
 xdel(winsid());
s=%s;
num=1;
den=s*(s+3)^2;
G=syslin('c',num/den);
clf();
evans(G);
axes_handle.grid=[1 1]
mtlb_axis([-5 5 -5 5]);
//form the graph it can be seen that the break away
   point is at "-1"
disp("Break away point=-1")
```

Scilab code Exa 6.2.2 location of the root locus between poles and zeros

```
//Example sec 6.2.2
// location of root locus in between poles and zeros
    .
clear; clc;
 xdel(winsid());
s=%s;
num=((s+1)*(s+2));
den=(s*(s+3)*(s+4));
G=syslin('c',num/den);
clf();
evans(G);
axes_handle.grid=[1 1]
mtlb_axis([-5 5 -5 5]);
```

Scilab code Exa 6.3 Root locus

```
//Example 6.3
// Plotting root locus
clear; clc;
 xdel(winsid());
s=%s;
num=(s+2);
```

```
den1=(s+1+(%i*sqrt(3)))*(s+1+(%i*sqrt(3)));
//upon simplification the denominator becomes
den2=(s^2+2*s+4)
G=syslin('c',num/den2);
clf();
evans(G);
axes_handle.grid=[1 1]
mtlb_axis([-5 5 -5 5]);
```

Scilab code Exa 6.4 root locus

```
//Example 6.4
// Plotting root locus
clear; clc;
 xdel(winsid());
s=%s;
num=-(s+2);
den1=(s+1+(%i*sqrt(3)))*(s+1+(%i*sqrt(3)));
//upon simplification the denominator becomes
den2=(s^2+2*s+4)
G=syslin('c',num/den2);
clf();
evans(G);
axes_handle.grid=[1 1];
mtlb_axis([-3 3 -3 3]);
```

Scilab code Exa 6.5 Root locus

```
//Example 6.5
// Plotting root locus
clear; clc;
 xdel(winsid());
s=%s;
num=1;
den=s*(s+4)*(s^2+4*s+20);
G=syslin('c',num/den);
clf;
evans(G);
axes_handle.grid=[1 1]
mtlb_axis([-5 5 -5 5]);
```

Scilab code Exa 6.6 Root locus

```
//Example 6.6
// Plotting root loci in scilab
clear; clc;
 xdel(winsid());
s=%s;
num=(s+2);
den=(s+1)^2;
```

```
t=syslin('c',num/den);
clf;
evans(t);
axes_handle.grid=[1 1]
mtlb_axis([-4 4 -4 4]);
```

Scilab code Exa 6.7 `Root locus`

```
//Example 6.7
// Plotting root locus
clear; clc;
 xdel(winsid());
 Beta=0
s=%s;
num=1;
den=s*(s+1)*(s+Beta);
G=syslin('c',num/den);
clf();
evans(G);
axes_handle.grid=[1 1]
mtlb_axis([-4 4 -4 4]);
```

Chapter 7

Analysis of frequency response

Scilab code Exa 7.3.1a Bode plot

```
//Example 7.3.1 a
// Bode plot in scilab
 clear; clc;
xdel(winsid());

s=poly(0,'s');
H=syslin('c',(10*(1+s)),s^2*(1+.25*s+0.0625*s^2));
clf();
bode(H,0.1,1000)
```

Scilab code Exa 7.3.1b Bode plot

```
//Example:(i) 7.3.1 b
// Bode plot in scilab
 clear; clc;
```

```
  xdel(winsid());

s=poly(0,'s');
G=syslin('c',(8*(1+0.5*s)),s*(1+2*s)*(1+0.05*s
   +0.0625*s^2));
clf();
bode(G,0.01,1000);
```

Chapter 8

stability in frequency response systems

Scilab code Exa 8.1 Nyquist plot

```
//Example 8.1
//Nyquist plot
clear; clc;
xdel(winsid());

 s = %s/2/%pi;
 num=(1);
 den=s*(s+1);
 G=syslin('c',num,den)
 clf();
 nyquist(G)
```

Scilab code Exa 8.2 Nyquist plot

```
//Example 8.2
//Nyquist plot
clear; clc;
xdel(winsid());

 s = %s/2/%pi;
 //since the value of "K" and "tau" in the given transfer function is constant
 // thus assuming "K=1" and "tau=1"
 //the resulting transfer function is,
 num2=(1);
 den2=(s+1)^2;
 G=syslin('c',num2,den2)
 clf();
 nyquist(G)
```

Scilab code Exa 8.3 `Nyquist plot`

```
//Example 8.3
//Nyquist plot
clear; clc;
xdel(winsid());

 s = %s /2 /%pi;
 num=(s+3);
 den=(s+1)*(s-1)
 G=syslin('c',num,den)
clf();
nyquist(G)
```

Chapter 9

compensators and controllers

Scilab code Exa 9.1 compensation in open loop control system

```
//Example sec 9.1
//compensation in open loop control system
clear;clc;
xdel(winsid());

s=%s;
disp("G=(60*k)/s*(s+1)*(s+6)")
//velocity error constant "Kv" when unit ramp input
   is applied to G is "5k".
//If "k=1",then,steady state error is 0.2.
// when "k=35/60" G becomes

num=35;
den=s*(s+2)*(s+6);
G1=syslin('c',num,den);
subplot(1,2,1);
evans(G1)
// From the figure 9.1
OA=sqrt((0.3)^2+(2.8)^2);
wn1=OA
theta=84 // analytically calculated
```

```
zeta1=cosd(theta)
Ts1=4/(zeta1*wn1) // Ts1=settling time in seconds
//For zeta to be 0.6 and settling time less than 0.4
    sec
a=acosd(0.6)
//By drawing angle "a" on the root locus
OB=1.26;
wn2=OB;
Ts2=4/(0.6*1.26) //in seconds
k=10.5/60;
//substituting "s=0" and "60k=10.5" in the equation
   for G.
Kv1=10.5/12 //Kv= velocity error coefficient
Ess1= 1/Kv1  //Ess= steady state error
//To get the required value of the zeta, steady
   state error increases and settling time improves.

//inserting one zero in the expression for "G"
disp("G2=60*k*(s+3)/s*(s+2)*(s+6)")
//considering k=1
num1=60*(s+3);
den1=(s*(s+1)*(s+6));
G3=syslin('c',num1,den1);
subplot(1,2,2);
evans(G3);
//considering "zeta=0.6" and drawing line OA at an
   angle 53.13, on the root locus.
zeta=0.6;
OA1=3.4;
wn=OA1
K=16/60
Kv=(60*K*3)/(2*6)
Ts=4/(zeta*wn) //in seconds
Ess=1/Kv
```

Chapter 10

Non linear control system

Scilab code Exa 10.1.1 Mass dashpot and spring arrangement

```
//Example sec 10.1.1
//mass, dashpot, spring arrangement.
clear;clc;
xdel(winsid());
M=1
K=2
F=2
A=[0 1;-2 -2]
C=eye(A)
s=%s
D=s*C-A
X=inv(D)*[1;1]
//taking the laplace transform of X
disp("X(t)=sqrt(5)*sin(t+inv(tan 0.5));sqrt(10)*sin(
    t+inv(tan -1/3))")
disp("The system is asymptotically stable")
```

Scilab code Exa 10.3 determination of quadratic form

```
//Example 10.3
//determination of quadratic form
clear;clc;
xdel(winsid());
//from the given euation we get the following
A=[9 1 -2;1 4 -1;-2 -1 1]
det(A)
A1=[9 1;1 4]
det(A1)
//since determinant of A and A1 is positive
//therefore W is positive definite.
disp("W is positive definite")
```

Scilab code Exa 10.4 Lipunovs method

```
//Example 10.4
//Lipunov's method
clear;clc;
xdel(winsid());

x1=poly(0,'x1');
x2=poly(0,'x2');
x11=poly(0,'x11');
x22=poly(0,'x22');
x2=x11
//assuming K1 and K2 equal to one.
disp("W=x1^2+x2^2")
//"W=x1^2+x2^2" is Liapunov's function
//W is chosen arbitrarily, since there no standard
    procedure for selecting W.
disp("dW/dt=2*x1*x11+2*x2*x22=-2*(x2^2+x2^4)")
```

```
disp("This will be negative semidefinite and therefore the system will be stable")
```

Chapter 11

Digital control system

Scilab code Exa 11.6 Jurys stability test

```
//Example 11.6
//Jury's stability test
clear;clc;
xdel(winsid());

z=%z;
F=4*z^4+6*z^3+12*z^2+5*z+1
//equating the equation F with a4*z^4+a3*z^3+a2*z^2+
   a1*z3+a0.
a0=1
a1=5
a2=12
a3=6
a4=4

b0=[a0 a4;a4 a0]
det(b0)
b1=[a0 a3;a4 a1]
det(b1)
b2=[a0 a2;a4 a2]
det(b2)
```

```
b3=[a0 a1;a4 a3]
det(b3)

c0=[det(b0) det(b3);det(b3) det(b0)]
det(c0)
c1=[det(b0) det(b2);det(b3) det(b1)]
det(c1)
c2=[det(b0) det(b1);det(b3) det(b2)]
det(c2)

disp("det(a0)<det(a4)=satisfied")
disp("det(b0)>det(b3)=satisfied")
disp("det(c0)<det(c3)=not satisfied")

disp("The system is unstable")
```

Scilab code Exa 11.9.2a stability of linear continuous system

```
//Example sec 11.9.2 a
//stability of linear continuous system
clear;clc;
xdel(winsid());

s=%s;
G=1/(s*(s+1)*(s+2))
G1=pfss(G)
//taking Z transform of G1
z=%z;
G2=(z/(2*(z+1)))-(z/(z+%e^(-1)))+(z/(2*(z+%e^(-2))))
//upon simplification we get the following characteristic equation
B=z^3-(1.3*z^2)+0.85*z-0.5
//substituting "z=(1+r/1-r)" in B
```

```
//the resultant equation is B1
r=poly(0,'r');
B1=3.65*r^3+1.95*r^2+2.35*r+0.05
routh_t(B1)
disp("The system is stable")
```

Scilab code Exa 11.9.2b stability of linear continuous system

```
//Example sec 11.9.2 b
//stability of linear continuous system
clear;clc;
xdel(winsid());

s=%s;
G=5/(s*(s+1)*(s+2))
G1=pfss(G)
//taking Z transform of G1
z=%z;
G2=5*((z/(2*(z+1)))-(z/(z+%e^(-1)))+(z/(2*(z+%e^(-2)))))
//upon simplification we get the following characteristic equation
B=z^3-(0.5*z^2)+2.49*z-0.496
//substituting "z=(1+r/1-r)" in B
//the resultant equation is B1
r=poly(0,'r')
B1=3.5*r^3-2.5*r^2+0.5*r+2.5
routh_t(B1)
disp("The system is unstable")
```

Scilab code Exa 11.9.3 Schurcohn stability test

```
//Example sec 11.9.3
//Schurcohn stability test
clear;clc;
xdel(winsid());

z=%z
G=1/(1-((7/4)*(z^-1))-((1/2)*(z^-2)))
A2=1-((7/4)*(z^-1))-((1/2)*(z^-2))
//K2=coefficient of z^-2
K2=-0.5
B2=-0.5-1.75*(z^-1)+z^-2

A1=(A2-K2*B2)/(1-K2^2)
//K1=coefficient of z^-1
K1=-3.5
//mod(K1)>1 and mod(K2)<1
disp("The sytem is unstable")
```

Chapter 15

Miscellaneous solved problems

Scilab code Exa 15.2 Time domain specifications of second order system

```
//Example 15.2
//time domain specifications of second order system
clear;clc;
xdel(winsid());
mode(0);

//converting the given differential equation in "s"
    domain
//since x and y are constants
//therefoere considering "x=y=1"

s=%s;
g=s^2+2*s;
x=roots(g)
wn=sqrt (abs(x(1))) //undamped natural frequency
zeta=(1/wn) //damping ratio
wd=wn*sqrt(1-zeta^2)//damped natural frequency
Dc=(zeta*wn) //Dc=damping coefficient
Tc=1/(zeta*wn)//Tc=time constant of the system
```

Scilab code Exa 15.4 transfer function of gyroscope

```
//Example 15.4
//Transfer function of Gyroscope
clear;clc;
xdel(winsid());
//in case of Gyroscope the equation is

disp("(J*s^2+B*s+K)theta(s)=H*w(s)")
//therefore
disp("theta(s)/w(s)=H/J*s^2+B*s+K")
```

Scilab code Exa 15.5 Transfer function of system

```
//Example 15.5  (fig 15.4)
//transfer function of the system
clear;clc;
xdel(winsid());
mode(0);

s=poly(0,'s');
//G1 and G2 are connected in series
G1=s^2/(s+4)^2
G2=(s+1)/(s^3*(s+3))
//H1 is feedback loop
H1=(s^2+s+1)/(s*(s+3))
// Tf=transfer function
Tf=(G1*G2*H1)
```

```
A=type(s);
disp(A,'Type of the system=')
```

Scilab code Exa 15.6 comparison of sensitivities of two systems

```
//Example 15.6
//comparison of sensivity of the two system
clear;clc;
xdel(winsid());
//k1 andk2 are series blocks of the transfer
   function
k1=100
k2=100
//transfer function of fig.15.5
T1=k1*k2/(1+(0.0099*k1*k2))
//transfer function of fig.15.6
T2=(k1/(1+(0.09*k1)))*(k2/(1+(0.09*k2)))
disp("both transfer function are equal")
//sensitivity of the transfer function T1 with
   respect to k1
T11=1/(1+(0.0099*k1*k2))
//sensitivity of the transfer function T2 with
   respect to k1
T12=1/(1+(0.09*k1))
disp("The system of fig 15.6 is 10 times more
   sensitive than system of fig 15.5 with respect to
    variations in k1")
```

Scilab code Exa 15.7 To find bandwidth of the transfer function

```
//Example 15.7
//find bandwidth of the transfer function
clear;clc;
xdel(winsid());

s=%s;
O=1;
R=(s+1);
tf=O/R
disp("when O/R(jw)=0.707, w=wc")

wc=(1/0.707)^2-1
//wc=bandwidth of the transfer function

disp("Hence the bandwidth is 1 rad/sec")
```

Scilab code Exa 15.8 Bandwidth of the transfer function

```
//Example 15.8
//find bandwidth of the transfer function
clear;clc;
xdel(winsid());

s=%s;
O=6;
R=(s^2+2*s+6);
tf=O/R

disp("when O/R(jw)=6/sqrt(w^4-8*w+36)")

w=[+2 -2] //after differentiation and simplification

disp("when O/R(jw)=6/sqrt(w^4-8*w+36), At w=+-2")
```

```

peak=3/sqrt(5)
```

Scilab code Exa 15.9 Nyquist plot

```
//Example 15.9
//Nyquist plot
clear; clc;
xdel(winsid());

 s = %s/2/%pi;
 num=(1);
 den=s^3*(s+1);
 G=syslin('c',num,den)
 clf()
 nyquist(G)
```

Scilab code Exa 15.12 solution of polynolynomial equation

```
//Example 15.12
//prove the solution of the equation
clear;clc;
xdel(winsid());
//assuming n=1
n=1;
z=%z;
y(n)=z^n;
```

```
y(n+1)=z^(n+1);
y(n+2)=z^(n+2);
A=y(n+2)+3*y(n+1)+2*y(n)
B=A/z
roots(z^2+3*z+2)
disp("y(n)=z^n is solution of polynomial equation (z+2)*(z+1)=0")
```

Scilab code Exa 15.13 Time domain specifications

```
//Example 15.13
//Time domain specifications
clear;clc;
xdel(winsid());

J=5.5*10^-2;
f=3.0*10^-4;
disp("wn=sqrt(k/J)=10^3*sqrt(k/5.5)")
disp("zeta=sqrt(4.9*10^-3/k)")
//at critically damped condition "zeta=1", therefore
k=4.09*10^-3
//when k=1.5*10^-2
zeta=sqrt((4.09*10^-3)/(1.5*10^-2))
wn=10^3*sqrt(1.5*10^-2/5.5)
wd=(wn/(2*%pi))*sqrt(1-zeta^2)
//wd=frequency of damped oscillation
Pwd=1/wd
//Pwd=period of damped oscillation
```

Scilab code Exa 15.14 Position servomotor

```
//Example 15.14
//position servomoter
clear;clc;
xdel(winsid());

//Mil= motor inertia referred tothe load side
Mil=20^2*0.45*10^-6  //unit= kg.m^2

//Tr= Transformation ratio of gear train between the
    loadshaft and the tachogenerator
Tr=20*2

//til= tachogenerator inertia referred to the load
    side
til=40^2*0.35*10^-6  //unit= kg.m^2

//Til= total inertia referred to the load side
Til=(20*10^-6)+(1.8*10^-4)+(5.6*10^-4)  //unit= kg.m
    ^2

//Mi= inertia referred to the motor side
Mi=(760*10^-6)/400  //unit= kg.m^2
```

Scilab code Exa 15.15 steady output speed of DC motor

```
//Example 15.15
//steady output speed of DC motor
clear;clc;
xdel(winsid());

//Jm= moment of inertia of motor
```

```
Jm=6.5*10^-2;
//Fm= friction of motor
Fm=3.5*10^-3;
//a=gear ratio
a=1/100;
//Jl= inertia of load
Jl=420;
//Fl= friction of load
Fl=220;
//J= total moment of inertia
J=Jm+(a^2*Jl)    //unit=kg.m^2
//F= total friction
F=Fm+(a^2*Fl)   //unit=kg.m^2
s=%s
//wm1=Angular velocity in frequency domain
wm1=2/(s*((J*s)+F))
t=1;
//wm2=Angular velocity in time domain
//since "t=1", wm2 is initial value of angular
   velocity
wm2=(2/F)*(1-(%e^((-5.7*10^-2)/(10.7*10^-2))*t))   //
   unit=rad/sec
//Nm1=motor speed in rps(initial speed)
Nm1=wm2/(2*%pi);
//Nm2=motor speed in rpm
Nm2=(wm2/(2*%pi))*60;   //unit=rpm
//Nl=load speed
Nl=(1/100)*((wm2/(2*%pi))*60) //unit=rpm
//Nos= steady output speed
//since Nos is steady speed, the exponential term of
    wn2 becomes 0.
Nos=(1/100)*(60/(2*%pi))*(2/(5.7*10^-2))   //unit=rpm
```

Scilab code Exa 15.26 `Routh array`

```
// Example 15.26
// Constructing Routh array in scilab

clear;clc
xdel(winsid());
mode(0);

A=[5 -6 -12;-1 1 2;5 -6 -11]
B=eye(3,3)
s=%s
C=s*B-A
D=s^3+5*s^2+5*s+1;  // characteristic equation after
    simplification
routh_t(D)
disp("No sign change in the first column, hence the
   system is asymptotically stable")
```

Scilab code Exa 15.27 `To check reachability of the system`

```
// Example 15.27
 // To check whether the system is reachable or not

clear;clc
xdel(winsid());
mode(0);
A=[1 0;0 1]
B=[1;1]
Wc=[A*B B]
disp("The rank of Wc=(1*1-1*1)=0,and not equal to 2.
    Thus the given system is not reachable ")
```

Scilab code Exa 15.28 Determine the stability of the system

```
// Example 15.28
// Determine the stability of the system.

clear;clc
xdel(winsid());
mode(0);


z=%z

D=z^3+6*z^2+8*z-0.04;   // characteristic equation
    after simplification
routh_t(D)
disp("There is sign change in the first column,
    hence the system is unstable")
```

Scilab code Exa 15.31 Time domain specifications of second order system

```
//Example 15.31
 //time domain specifications of second order system
 clear;clc;
 xdel(winsid());
 mode(0);

 //converting the given differential equation in "s"
     domain
```

```
//since x and y are constants
//therefoere considering "x=y=1"

s=%s;
g=s^2+5*s+7;
x=coeff(g)
//comparing with the standard equation of second order system.
wn=sqrt(x(:,1)) //undamped natural frequency
zeta=(5/(2*wn)) //damping ratio
wd=wn*sqrt(1-zeta^2)//damped natural frequency
Tc=1/(zeta*wn)//Tc=time constant of the system
```

Scilab code Exa 15.33 Lipunovs method

```
//Example 15.33
//Lipunov's method
clear;clc;
xdel(winsid());

x1=poly(0,'x1');
x2=poly(0,'x2');
x11=poly(0,'x11');
x22=poly(0,'x22');
x2=x11
disp("x22+x2+x2^3+x1=0")
//(x1,x2) has singular point at (0,0)
disp("V=x1^2+x2^2")
//"V=x1^2+x2^2" is Liapunov's function
//V is positive for all values of x1 and x2, except at x1=x2=0
disp("dV/dt=2*x1*x2-2*x1*x2-2*x2^2-2*x2^4=-2*x2^2-2*x2^4")
```

```
disp("dV/dt will never be positive hence origin is
    stable")
```

Scilab code Exa 15.34 Find bandwidth of the transfer function

```
//Example 15.34
//find bandwidth of the transfer function
clear;clc;
xdel(winsid());

s=%s
A=1
B=(s+1)
tf=A/B

disp("when A/B(jw)=1/sqrt(2), w=w1")

w1=(1/0.707)^2-1
//w1=bandwidth of the transfer function

disp("Hence the bandwidth is 1 rad/sec")
```

Scilab code Exa 15.36 Impulse response of the transfer function

```
//Example 15.36
//impulse response transfer function
clear;clc;
xdel(winsid());

```

```
s=%s;
G=syslin('c',25,s^2+4*s+25);
t=0:0.05:5;
y=csim('impuls',t,G);
plot(t,y)
xtitle('Impulse response 25/(s^2+4*s+25)','t sec','
    Response');
```

Scilab code Exa 15.37 `step response of the transfer function`

```
//Example 15.37
//step response transfer function
clear;clc;
xdel(winsid());

s=%s;
G=syslin('c',25,s^2+4*s+25);
t=0:0.05:5;
y=csim('step',t,G);
plot(t,y)
xtitle('step response 25/(s^2+4*s+25)','t sec','
    Response');
```

Scilab code Exa 15.38 `Roots of characteristic equation`

```
//Example 15.38
//find roots of characteristic equation

clear;clc;
xdel(winsid());
s=poly(0,'s')
G=s^4+2*s^3+s^2-2*s-1
roots(G)
```

Scilab code Exa 15.39 Bode plot

```
//Example:15.39
// Bode plot in scilab
 clear; clc;
  xdel(winsid());

s=poly(0,'s');
G=syslin('c',(25),s^2+4*s+25);
clf();
bode(G,0.01,1000);
```

Scilab code Exa 15.40 Nyquist plot

```
//Example 15.40
//Nyquist plot
clear; clc;
xdel(winsid());
```

```

s = %s/2/%pi;
num=(1);
den=(s^2+0.8*s+1);
G=syslin('c',num,den)
clf();
nyquist(G)
```

Scilab code Exa 15.41 Nyquist plot

```
//Example 15.41
//Nyquist plot
clear; clc;
xdel(winsid());

s = %s/2/%pi;
num=(s+2);
den=(s+1)*(s+1);
G=syslin('c',num,den)
clf();
nyquist(G)
```

Scilab code Exa 15.42 Bode plot

```
//Example:15.42
```

```
// Bode plot in scilab
 clear; clc;
  xdel(winsid());

s=poly(0,'s');
G=syslin('c',(64*(s+2)),(s*(s+0.5)*(s^2+3.2*s+64)));
clf();
bode(G,0.01,1000);
```

Scilab code Exa 15.43 Eigen values of matrix

```
//Example:15.43
//eigen values of matrix A
clear;clc;
xdel(winsid());

A=[0 6 -5;1 0 2;3 2 4];
spec(A)
```

Scilab code Exa 15.44 State space representation of LTI system

```
//Example 15.44
//state space representation of LTI system
clear;clc;
xdel(winsid());

A=[0 1;-2 -3];
```

```
B=[0;1];
C=[1 1];
D=[0];
E=[0];

H=syslin('c',A,B,C);
s=%s;
g=eye(2,2);
P=(-s*g)-A
sm=[P B;C D];
H1=sm2ss(sm)
```

Scilab code Exa 15.45 Covariant matrix of A

```
//Example 15.45
// Covariant matrix of "A"
clear;clc;
xdel(winsid());
A=[1 0 0;0 2 0;0 0 3]
mvvacov(A)
```

Scilab code Exa 15.49 Root locus of the transfer function

```
//Example 15.49
// Plotting root loci of the transfer function k/s*(
    s+4)*(s^2+4*s+20)
clear; clc;
 xdel(winsid());
s=%s;
```

```
num=(1);
den=s*(s+3)*(s^2+2*s+2);
G=syslin('c',num/den);
clf;
evans(G);
mtlb_axis([-5 5 -5 5]);
```

Scilab code Exa 15.50 Bode plot

```
//Example:15.50
// Bode plot in scilab
 clear; clc;
  xdel(winsid());

s=poly(0,'s');
G=syslin('c',(16*(s+2)),(s*(s+0.5)*(s^2+3.2*s+64)));
clf();
bode(G,0.01,1000);
```

Scilab code Exa 15.53 Statespace model of the differential equation

```
//Example 15 53
//state space model of differential equation.
clear;clc;
xdel(winsid());
```

```

// converting the differential equation in terms of
   transfer function.
s=%s
//transfer function
A=1/(s^3+6*s^2+11*s+6)
B=tf2ss(A)
```

Scilab code Exa 15.54 Nyquist plot

```
//Example 15.54
//Nyquist plot
clear; clc;
xdel(winsid());

 s = %s/2/%pi;
 num=(2);
 den=s*(s^2+2*s+2);
 G=syslin('c',num,den)
 clf();
 nyquist(G)
```

Scilab code Exa 15.57 determination of zeta and wn

```
//Example 15.57
//determination of zeta & wn
clear;clc;
```

```
xdel(winsid());

s=%s
num=10;
den=s^2+2*s+10;   //since k=0
G=num/den;
B=coeff(den)
//wn= undamped natural ftrquency
wn=sqrt(B(:,1))
// zeta= damping ratio
zeta=2/(2*sqrt(wn))
// when time t tends to infinity, static error viz.
   ess tends to 0.
ess=0
// when "zeta=0.65" i.e.(zeta1=0.65)
zeta1=0.65
k0=2*zeta1*wn-2
```

Scilab code Exa 15.59 `transfer function of signal flow graph`

```
//Example 15.59
// transfer function of signal flow graph
clear;clc;
xdel(winsid());

k1=1;
k2=5;
k3=5;
s=%s;
// From the graph the transfer function is
T=(k3*k1)/(s^3+s^2+(k3*k1)+(k1*k2*s^2)+5)
// substitutins "s=0"in the equation of T
// and differentiating and simplifying the equation
```

```
// the following value of T will appear
T1=1/(1+k1)
```

Scilab code Exa 15.60 root locus

```
//Example 15.60
//root locus
clear;clc;
xdel(winsid());

s=%s;
//substituting "a=15" in the numerator
num=2*(s+15);
den=s*(s+2)*(s+10);
G=syslin('c',num/den);
evans(G);
axes_handle.grid=[1 1]
mtlb_axis([-5 5 -5 5]);
```