Scilab Textbook Companion for Control Systems by S. Ghosh[1]

Created by
Anuradha Singhania
B. Tech. (Elec. & Commun.)
Electrical Engineering
Sri Mata Vaishno Devi University, Jammu
College Teacher
Mr. Amit Kumar Garg
Cross-Checked by
Prof. Madhu Belur, Dept. Of Electrical Engineering

August 9, 2013

[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 Systems

Author: S. Ghosh

Publisher: New Delhi

Edition: 2

Year: 2009

ISBN: 978-81-317-0828-6

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

Chapter 2

Laplace Transform and Matrix Algebra

Scilab code Exa 2.01.01 laplace

```
//laplace//
 syms t s;
 y=laplace('13',t,s);
 disp(y,"ans=")
```

Scilab code Exa 2.01.02 laplace

```
//laplace//
 syms t s;
 y=laplace('4+5*%e^(-3*t)',t,s);
 disp(y,"ans=")
```

Scilab code Exa 2.01.03 laplace

```
//laplace//
syms t s;
 y=laplace('2*t-3*%e^(-t)',t,s);
 disp(y,"ans=")
```

Scilab code Exa 2.03 value theorem

```
//value theorem//
p=poly([9 1],'s','coeff')
q=poly([3 7 1],'s','coeff')
f=p/q;
disp(f,"F(s)=")
x=s*f;
y=limit(x,s,0); // final value theorem
disp(y,"f(inf)=")
z=limit(x,s,%inf); // initial value theorem
disp(z,"f(0)=")
```

Scilab code Exa 2.04 ilaplace

```
//ilaplace//
s=%s
syms t ;
disp((s+3)/((s+1)*(s+2)*(s+4)),"F(s)=")
[A]=pfss((s+3)/((s+1)*(s+2)*(s+4))) //partial
   fraction of F(s)
F1= ilaplace(A(1),s,t)
F2= ilaplace(A(2),s,t)
F3= ilaplace(A(3),s,t)
F=F1+F2+F3;
disp(F,"f(t)=")
```

Scilab code Exa 2.05.01 laplace

```
//laplace//
syms t s;
y= laplace('%e^(-t)+5*t+6*%e^(-3*t)',t,s);
disp(y,"ans=")
```

Scilab code Exa 2.05.02 laplace

```
//laplace//
syms t s;
y=laplace('5+6*t^2+3*%e^(-2*t)',t,s);
disp(y,"ans=")
```

Scilab code Exa 2.06 laplace

```
//laplace//
syms t s;
y=laplace('3- %e^(-3*t)',t,s);
disp(y,"ans=")
```

Scilab code Exa 2.07 laplace

```
//laplace//
syms t s w;
y=laplace('5*sin(w*t)+7*cos(w*t)',t,s);
disp(y,"ans=")
```

Scilab code Exa 2.11 value theorem

```
//value theorem//
p=poly([0.38],'s','coeff');
q=poly([0 0.543 2.48 1],'s','coeff');
F=p/q;

syms s;
x=s*F;
y=limit(x,s,0);  // final value theorem
y=dbl(y);
disp(y,"f(inf)=")
z=limit(x,s,%inf) // // initial value theorem
z=dbl(z);
disp(z,"f(0)=")
```

Scilab code Exa 2.12 ilaplace

```
//ilaplace//
s=%s;
p=poly([3 1],'s','coeff');
q=poly([0 -1 -2],'s','roots');
F=p/q;
syms t s;
y=ilaplace(F,s,t);
disp(y,"f(t)=")
```

Scilab code Exa 2.13 ilaplace

```
//ilaplace//
s=%s;
p=poly([3 1],'s','coeff');
q=poly([0 -1 -1 -2],'s','roots');
f=p/q;
syms t s;
y=ilaplace(f,s,t);
disp(y,"f(t)=")
```

Chapter 3

Transfer Function

Scilab code Exa 3.02 laplace

```
//laplace//
syms t s;
f=%e^(-3*t);
y=laplace('%e^(-3*t)',t,s);
disp(y,"G(s)=")
```

Scilab code Exa 3.03 laplace

```
//laplace//
syms t s;
disp(%e^(-3*t),"g(t)=");
y1=laplace('%e^(-3*t)',t,s);
disp(y1,"G(s)=")
disp(%e^(-4*t),"r(t)=");
y2=laplace('%e^(-4*t)',t,s);
disp(y2,"R(s)=")
disp(y1*y2,"G(s)R(s)=")
```

Scilab code Exa 3.04 laplace

```
//laplace//
s=%s;
H=syslin('c',(4*(s+2)*((s+2.5)^3))/((s+6)*((s+4)^2))
   );
plzr(H)
```

Scilab code Exa 3.15 laplace

```
//laplace//
syms t s
y=laplace(%e^(-3*t)*sin(2*t),t,s);
disp(y,"ans=")
```

Scilab code Exa 3.16 laplace

```
//laplace//
syms t s;
x=6-4*%e^(-5*t)/5+%e^(-3*t) //Given step Response of
    the system
printf("Derivative of step response gives impulse
   response \n")
y=diff(x,t);      //Derivative of step response
printf("Laplace Transform of Impulse Response gives
   the  Transfer Function \n ")
p=laplace(y,t,s);
disp(p,"Transfer Function=")
```

Scilab code Exa 3.17 laplace

```
//laplace//
printf("Given:Poles are s=-1,(-2+i),(-2-i); zeros s=-3+i,-3-i,Gain factor(k)=5 \n")
num=poly([-3+%i,-3-%i],'s','roots');
den=poly([-1,-2+%i,-2-%i],'s','roots');
G=(5*num)/den;
disp(G,"G(s)=")
```

Scilab code Exa 3.18 laplace

```
//laplace//
s=%s;
G=syslin('c',(5*(s+2))/((s+3)*(s+4)));
disp(G,"G(s)=")
x=denom(G);
disp(x,"Characteristics Polynomial=")
y=roots(x);
disp(y,"Poles of a system=")
```

Scilab code Exa 3.19 laplace

```
//laplace//
printf("Given: Poles are s=-3, Zeros are s=-2, Gain Factor(k)=5 \n ")
num=poly([-2],'s','roots');
den=poly([-3],'s','roots');
G=5*num/den;
```

```
disp(G,"G(s)=")
disp("Input is Step Function ")
syms t s;
R=laplace(1,t,s);
disp(R,"R(s)=")
printf("C(s)=R(s)G(s) \n")
C=R*G;
disp(C,"C(s)=")
c=ilaplace(C,s,t);
disp(c,"c(t)=")
```

Scilab code Exa 3.23 laplace

```
//laplace//
//pole zero plot for g(s)=(s^2+3s+2)/(s^2+7s+12)
s=%s;
p=poly([2 3 1],'s',"coeff")
q=poly([12 7 1],'s',"coeff")
V=syslin('c',p,q)
plzr(V)
syms s t ;
v =ilaplace('(2+(3*s)+s^2)/(s^2+(7*s)+12)',s,t)
disp(v,"V(t)=')
```

Chapter 4

Control system Components

Scilab code Exa 4.01 laplace

```
//laplace//
printf("Given a)Excitation voltage(Ein)=2V \n b)
   Setting Ratio(a)= 0.4 \n")
Ein=2;
disp(Ein,"Ein=")
a=0.4;
disp(a,"a=")
Rt=10^3;
disp(Rt,"Rt=")
Rl=5*10^3;
disp(Rl,"Rl=")
printf("Eo = (a*Ein)/(1+(a*(1-a)*Rt)/Rl) \n")
Eo = (a*Ein)/(1+(a*(1-a)*Rt)/Rl);
disp(Eo,"output voltage(E0)=")
printf("e=((a^2)*(1-a))/((a*(1-a))+(Rl/Rt)) \n")
e=((a^2)*(1-a))/((a*(1-a))+(Rl/Rt));
disp(e,"loading error=")
printf("E=Ein*e \n")
E=Ein*e; //Voltage error=Excitation voltage(Ein)*
   Loading error(e)
disp(E,"Voltage error=")
```

Scilab code Exa 4.02 laplace

```
//laplace//
printf("n=5 ,Helical turn \n")
n=5; //Helical turn
disp(n,"n=")
printf("N=9000 ,Winding Turn \n")
N=9000;//Winding Turn
disp(N,"N=")
printf("R=10000 ,Potentiometer Resistance \n")
R=10000;//Potentiometer Resistance
disp(R,"R=")
printf("Ein=90 ,Input voltage \n")
Ein=90;//Input voltage
disp(Ein,"Ein=")
printf("r=5050 ,Resistance at mid point  \n")
r=5050;//Resistance at mid point
disp(r,"r=")
printf("D=r-5000, Deviation from nominal at mid-
    point \n")
D=r-5000; //Deviation from nominal at mid-point
disp(D,"D=")
printf("L=D/R*100 ,Linearity \n")
L=D/R*100;//Linearity
disp(L,"L=")
printf("R=Ein/N ,Resolution \n")
R=Ein/N; //Resolution
disp(R,"R=")
printf("Kp=Ein/(2*pi*n) ,Potentiometer Constant \n")
Kp=Ein/(2*%pi*n); //Potentiometer Constant
disp(Kp,"Kp=")
```

Scilab code Exa 4.03 laplace

```
//laplace//
printf("since S2 is the referance stator winding ,
   Es2=KVcos0 \n where Es2 & Er are rms voltages \n
   ')
k=1
Theta=60;
disp(Theta,"Theta=")
V=28;
disp(V,"V(applied)=")
printf("Es2=V*cos(Theta) \n")
Es2=k*V*cos(Theta*(%pi/180));
disp(Es2,"Es2=")
printf("Es1=k*V*cos(Theta-120)\n")
Es1=k*V*cos((Theta-120)*(%pi/180)); // Given Theta
   =60 in degrees
disp(Es1,"Es1=')
printf("Es3=k*V*cos(Theta+120) \n")
Es3=k*V*cos((Theta+120)*(%pi/180));
disp(Es3,"Es3=')
printf("Es31=sqrt(3)*k*Er*sin(Theta)")
Es31=sqrt(3)*k*V*sin(Theta*(%pi/180));
disp(Es31,"Es31=')
printf("Es12=sqrt(3)*k*Er*sin((Theta-120)")
Es12=sqrt(3)*k*V*sin((Theta-120)*(%pi/180));
disp(Es12,"Es12=')
printf("Es23=sqrt(3)*k*Er*sin((Theta+120)")
Es23=sqrt(3)*k*V*sin((Theta+120)*(%pi/180));
disp(Es23,"Es23=')
```

Scilab code Exa 4.04 laplace

```
//laplace//
printf("Sensitivity=5v/1000rpm \n")
```

```
Vg=5;
disp(Vg,"Vg=")
printf("w(in radians/sec)=(1000/60)*2*pi \n")
w=(1000/60)*2*%pi;
disp(w,"w=")
printf("Kt=Vg/w \n')
Kt=Vg/w;
disp(Kt,"Gain constant(Kt)=")
```

Scilab code Exa 4.05 laplace

```
//laplace//
printf("Torque=KmVm=2 \n")
t=2;
disp(t,"Torque(t)=")
Fm=0.2;
disp(Fm,"Coefficient of Viscous friction(Fm)=")
N=4
I=0.2
F1=0.05
printf("Wnl=t/Fm")
Wnl=t/Fm;
disp(Wnl,"No Load Speed(Wnl)=")
printf("Fwt=I+(N^2*F1) \n")
Fwt=I+(N^2*F1);
disp(Fwt,"Total Viscous Friction(Fwt)=")
printf("Te=t-(Fwt*w) \n")
Te=0.8 //load
w=(t-Te)/Fwt;
disp(w,"Speed of Motor(w)=")
```

Chapter 6

Control system Components

Scilab code Exa 6.01 syslin

```
//syslin//
exec series.sce;
s=%s;
sys1=syslin('c',(s+3)/(s+1))
sys2=syslin('c',0.2/(s+2))
sys3=syslin('c',50/(s+4))
sys4=syslin('c',10/(s))
a=series(sys1,sys2);
b=series(a,sys3);
y=series(b,sys4);
y=simp(y);
disp(y,"C(s)/R(s)=")
```

Scilab code Exa 6.02 syslin

```
//syslin//
exec parallel.sce;
s=%s;
```

```
sys1=syslin('c',1/s)
sys2=syslin('c',2/(s+1))
sys3=syslin('c',3/(s+3))
a=parallel(sys1,sys2);
y=parallel(a,sys3);
y=simp(y);
disp(y,"C(s)/R(s)=")
```

Scilab code Exa 6.03 syslin

```
//syslin//
exec series.sce;
s=%s;
sys1=syslin('c',3/(s*(s+1)))
sys2=syslin('c',s^2/(3*(s+1)))
sys3=syslin('c',6/(s))
a=(-1)*sys3;
b=series(sys1,sys2);
y=b/.a   // feedback operation
y=simp(y)
disp(y,"C(s)/R(s)=")
```

Scilab code Exa 6.04 syslin

```
//syslin//
exec parallel.sce;
syms G1 G2 G3 H;
a=series(G1,G2);
b=parallel(a,G3);
y=b/.H  //negative feedback operation
y=simple(y)
disp(y,"C(s)/R(s)=")
```

Scilab code Exa 6.05 syslin

```
//syslin//
exec series.sce;
syms G1 G2 H1 H2 s;
a=G1/.H1; // negative feedback operation
b=a/.H2;// negative feedback operation
y=series(b,G2);
y=simple(y);
disp(y,"C(s)/R(s)=")
```

Scilab code Exa 6.06 syslin

```
//syslin//
exec parallel.sce;
exec series.sce;
syms G1 G2 G3 G4 G5 G6 H1 H2;
a=parallel(G3,G5);
b=parallel(a,-G4);
c=series(G1,G2);
d=c/.H1;
e=series(b,d);
f=e/.H2;
y=series(f,G6);
y=simple(y);
disp(y,"C(s)/R(s)=")
```

Scilab code Exa 6.07 syslin

```
//syslin//
exec series.sce;
syms G1 G2 G3 H1 H2 R X;
//putting x=0,then solving the block
a=G3/.H1;
b=series(G1,G2);
c=series(a,b);
x1=c/.H2;
C1=R*x1;
disp(x1,"C1(s)/R(s)=")
//putting r=0 ,then solving the block
d=series(G1,G2);
e=series(d,H2);
f=G3/.H1;
x2=f/.e;
C2=X*x2;
disp(x2,"C2(s)/X(s)=")
//resultant output C=C1+C2
C=C1+C2;
C=simple(C);
disp(C,"Resultant Output=")
```

Scilab code Exa 6.08 syslin

```
//syslin//
exec parallel.sce;
exec series.sce;
syms G1 G2 G3 H1 H2;
//shift the take-off point after the block G2
a=G3/G2;
b=parallel(a,1);
c=series(G1,G2);
d=c/.H1 //negative feedback operation
e=series(d,b);
y=e/.H2;
```

```
y=simple(y);
disp(y,"C(s)/R(s)=")
```

Scilab code Exa 6.09 syslin

```
//syslin//
exec series.sce
syms G1 G2 G3 H1 H2 H3;
//Remove the feedback loop having feedback path
    transfer function H2
a=G3/.H2;
//Interchange the summer .as well as replace the
    cascade block by its equivalent block
b=series(G1,G2);
c=b/.H1; //Negative Feedback Operation
d=series(c,a);
y=d/.H3;  //Negative Feedback Operation
y=simple(y);
disp(y,"C(s)/R(s)=")
```

Scilab code Exa 6.10 syslin

```
//syslin//
exec parallel.sce;
exec series.sce;
syms G1 G2 G3 G4 G5 H1 H2;
a=G2/.H1;  //negative feedback operation
b=series(G1,a);
c=series(b,G3);
d=parallel(c,G4);
e=series(d,G5);
y=e/.H2;   //negative feedback operation
y=simple(y);
```

```
disp(y,"C(s)/R(s)=")
```

Scilab code Exa 6.11 syslin

```
//syslin//
exec parallel.sce;
exec series.sce;
syms G1 G2 G3 G4 G5 G6 G7 H1 H2 H3;
a=parallel(G1,G2);
b=parallel(a,G3);
//shift the take off point to the right of the block
    G4
c=G4/.H1;  //negative feedback operation
d=G5/G4;    //negative feedback operation
e=parallel(d,1);
f=G6/.H2;   //negative feedback operation
g=series(b,c);
h=series(g,e);
i=series(h,f);
j=series(i,G7);
y=j/.H3;
y=simple(y);
disp(y,"C(s)/R(s)=")
```

Scilab code Exa 6.12 syslin

```
//syslin//
exec series.sce;
exec parallel.sce;
syms G1 G2 G3 G4 H1 H2 H3;
//shift the take-off point after the block G1
a=G3/G1;
b=parallel(a,G2);
```

```
c=G1/.H1; // Negative Feedback Operation
d=1/b;    // Negative Feedback Operation
e=parallel(d,H3);
f=series(e,H2);
g=series(c,b);
h=g/.f ;    // Negative Feedback Operation
y=series(h,G4);
y=simple(y);
disp(y,"C(s)/R(s)=")
```

Scilab code Exa 6.13 syslin

```
//syslin//
exec series.sce;
exec parallel.sce;
syms G1 G2 G3 G4 H1 H2 ;
//reduce the internal feedback loop
a=G2/.H2;
//place the summer left to the block G1
b=G3/G1;
//exchange the summer
c=parallel(b,1);
d=series(a,G1);
e=series(d,G4);
f=e/.H1;
y=series(c,f);
y=simple(y);
disp(y,"C(s)/R(s)=")
```

Scilab code Exa 6.14 syslin

```
//syslin//
exec series.sce;
```

```
exec parallel.sce;
syms G1 G2 G3 G4 H1 H2 ;
//shift the take-off point to the right of the block
    G3
a=H1/G3;
b=series(G2,G3);
c=parallel(H2,a);
d=b/.c;
e=series(d,G1);
f=e/.a;
y=series(f,G4);
y=simple(y);
disp(y,"C(s)/R(s)=")
```

Scilab code Exa 6.15 syslin

```
//syslin//
exec series.sce;
exec parallel.sce;
syms G1 G2 G3 G4 H1 H2 H3;
//shift the take-off point to the right of the block
    H1
//shift the other take-off point to the right of the
    block H1 &H2
a=series(H1,H2);
b=1/a;
c=1/H1;
d=G3/.a;
//move the summer to the left of the block G2
e=G4/G2;
f=series(d,G2);
//exchange the summer
g=f/.H1;
h=parallel(G1,e);
i=series(h,g);
```

```
j=series(a,H3);
y=i/.j;
y=simple(y);
disp(y,"C(s)/R(s)=")
```

Chapter 8

Time Domain Analysis of Control Systems

Scilab code Exa 8.02 Velocity

```
//Velocity//
s=%s;
p=poly([3 1],'s','coeff');
q=poly([0 1 0.95 0.21],'s','coeff');
G=p/q;
disp(G,"G(s)=")
H=1;
y=G*H; //Type 1 System
disp(y,"G(s)H(s)=")
//Refering the table 8.2 given in the book ,For type
    1 system  Kp=%inf & Ka=0
printf("For type1  Kp=inf & Ka=0 \n")
syms s;
Kv=limit(s*y,s,0); //Kv= velocity error coefficient
disp(Kv,"Velocity Error Coefficient(Kv)=")
```

Scilab code Exa 8.03.01 coefficient

```
//coefficient//
s=%s;
p=poly([10],'s','coeff');
q=poly([0 0 1],'s','coeff');
G=p/q;
H=0.7;
y=G*H; //type 2
disp(y,"G(s)H(s)=")
//refering the table 8.2 given in the book ,for type
    1 Kp=%inf & Kv=%inf
printf("For type1  Kp=inf & Kv=inf \n")
syms s;
Ka=limit(s^2*y,s,0); //Ka=accelaration error
    coefficient
disp(Ka,"Ka=")
```

Scilab code Exa 8.03.02 coefficient

```
//coefficient//
p=poly([5],'s','coeff');
q=poly([5 3 1],'s','coeff');
G=p/q
H=0.6
y=G*H //type 0
//refering the table 8.2 given in the book ,for type
    1 Ka=0 & Kv=0
syms s
Kp=limit(s*y/s,s,0) // Kp=positional error
    coefficient
```

Scilab code Exa 8.03.03 coefficient

```
//coefficient//
p=poly([1 0.13 0.4],'s','coeff');
q=poly([0 0 5 3 1],'s','coeff');
G=10*p/q //gain FACTOR=10
H=0.8
y=G*H //type 2
//refering the table 8.2 given in the book ,for type
    2 Kp=%inf & Kv=%inf
syms s
Ka=limit(s^2*y,s,0) //Ka=accelaration error
    coefficient
```

Scilab code Exa 8.03.04 coefficient

```
//coefficient//
s= poly ( 0,'s' );
sys = syslin ('c',10/(s+2)); //G(s)H(s)
disp(sys,"G(s)H(s)")
F=1/(1+sys)
syms t s;
Co=limit(s*F/s,s,0) //Ko=Lt s->0 (1/(1+G(s)H(S))
d=diff(s*F/s,s)
C1=limit(diff(s*F/s,s),s,0) //K1=Lt s->0 (dF(s)/ds)
C2=limit(diff(d,s),s,0)    //K2=Lt s->0 (d2F(s)/ds)
//given input is r(t)=1+2*t+(t^2)/2 & R(s)=laplace(r
    (t))
a=(1+2*t+(t^2)/2);
b=diff(a,t);
c=diff(b,t);
e=Co*a+C1*b+C2*c //error by dynamic coefficient
    method
```

Scilab code Exa 8.04 coefficient

```
//coefficient//
p=poly([4 1],'s','coeff');
q=poly([0 0 6 5 1],'s','coeff');
syms K real;
y=K*p/q //gain FACTOR=K
disp(y,"G(s)H(s)=")
//G(s)H(s)=y ,and it is of type 2
//refering the table 8.2 given in the book ,for type
    2 Kp=%inf & Kv=%inf
printf("For type1  Kp=inf & Kv=inf \n")
syms A s t;
Ka=limit(s^2*y,s,0);  //Ka=accelaration error
   coefficient
disp(Ka,"Ka=")
//given input is r(t)=A(t^2)/2 & R(s)=laplace(r(t))
printf("Given r(t)=A(t^2)/2 \n")
R=laplace('A*t^2/2',t,s);
disp(R,"R(s)=")
//steady state error =Lt s->0 sR(S)/1+G(s)H(S)
e=limit(s*R/(1+y),s,0)
disp(e,"Ess=")
```

Scilab code Exa 8.05 coefficient

```
//coefficient//
p=poly([60],'s','coeff');
q=poly([12 7 1],'s','coeff');
G=p/q;
disp(G,"G(s)=")
H=1;
y=G*H
F=1/(1+y);
disp(F,"1/(1+G(s)H(s))=")
syms t s;
Ko=limit(s*F/s,s,0) //Ko=Lt s->0 (1/(1+G(s)H(S))
```

```
d=diff(s*F/s,s);
K1=limit(diff(s*F/s,s),s,0) //K1=Lt s->0 (dF(s)/ds)
K2=limit(diff(d,s),s,0)    //K2=Lt s->0 (d2F(s)/ds)
//given input is r(t)=4+3*t+8(t^2)/2 & R(s)=laplace(
   r(t))
a=(4+3*t+8*(t^2)/2)
b=diff(4+3*t+8*(t^2)/2 ,t)
c=diff(b,t)
e=Ko*a+K1*b+K2*c //error by dynamic coefficient
   method
disp(e,"error")
```

Scilab code Exa 8.06 coefficient

```
//coefficient//
s=poly(0,'s'); //Defines s as polynomial variable
F=syslin('c',[25/((s+1)*s)]); //Creates transfer
   function in forward path
B=syslin('c',(1+0*s)/(1+0*s)); //Creates transfer
   function in backward path
CL=F/.B //Calculates closed-loop transfer function
// compare CL with Wn^2/(s^2+2*zeta*Wn+Wn^2)
y=denom(CL) //extracting the denominator of CL
z=coeff(y)  //extracting the coefficients of the
   denominator polynomial
//Wn^2=z(1,1) ,comparing the coefficients
Wn=sqrt(z(1,1))               // Wn=natural frequency
//2*zeta*Wn=z(1,2)
zeta=z(1,2)/(2*Wn)                // zeta=damping
   factor
Wd=Wn*sqrt(1-zeta^2)
Tp=%pi/Wd
Mp=100*exp((-%pi*zeta)/sqrt(1-zeta^2))
```

Scilab code Exa 8.07 coefficient

```
//coefficient//
s=poly(0,'s'); //Defines s as polynomial variable
F=syslin('c',[20/(s^2+5*s+5)]); //Creates transfer
    function in forward path
B=syslin('c',(1+0*s)/(1+0*s)); //Creates transfer
    function in backward path
CL=F/.B //Calculates closed-loop transfer function
// compare CL with Wn^2/(s^2+2*zeta*Wn+Wn^2)
y=denom(CL) //extracting the denominator of CL
z=coeff(y) //extracting the coefficients of the
    denominator polynomial
//Wn^2=z(1,1) ,comparing the coefficients
Wn=sqrt(z(1,1))                // Wn=natural frequency
//2*zeta*Wn=z(1,2)
zeta=z(1,2)/(2*Wn)                  // zeta=damping
    factor
Wd=Wn*sqrt(1-zeta^2)
Tp=%pi/Wd      //Tp=peak time
Mp=100*exp((-%pi*zeta)/sqrt(1-zeta^2))   //peak
    overshoot
Td=(1+0.7*zeta)/Wn       //Td=delay time
a=atan(sqrt(1-zeta^2)/zeta)
Tr=(%pi-a)/Wd            //Tr=rise time
Ts=4/(zeta*Wn)           //Ts=settling time
```

Scilab code Exa 8.08 coefficient

```
//coefficient//
p=poly([140,35],'s','coeff');
q=poly([0 ,10 ,7 ,1],'s','coeff');
```

```
G=p/q
H=1
y=G*H //type 1
//refering the table 8.2 given in the book ,for type 1 Kp=%inf & Ka=0
syms s
Kv=limit(s*y,s,0) //Kv= velocity error coefficient
//given input is r(t)=5*t & R(s)=laplace(r(t))
R=laplace('5*t',t,s)
//steady state error =Lt s->0 sR(S)/1+G(s)H(S)
e=limit(s*R/(1+y),s,0) //e=error for ramp input
disp(e,"steady state error")
```

Scilab code Exa 8.09 coefficient

```
//coefficient//
p=poly([40,20],'s','coeff');
q=poly([0,0,5,6,1],'s','coeff');
G=p/q;
H=1;
y=G*H; //type 2
disp(y,"G(s)H(s=")
//refering the table 8.2 given in the book ,for type 2 Kp=%inf & Kv=%inf
syms s t;
Ka=limit(s^2*y,s,0) //Ka= accelaration error coefficient
//given input is r(t)=1+3t+t^2/2 & R(s)=laplace(r(t))
R=laplace('(1+3*t+(t^2/2))',t,s)
//steady state error =Lt s->0 sR(S)/1+G(s)H(S)
e=limit(s*R/(1+y),s,0) //e=error for ramp input
disp(e,"steady state error")
```

Scilab code Exa 8.10 coefficient

```
//coefficient//
s = poly ( 0,'s' )
sys = syslin ('c',(20)/(s^2+7*s+10))
a=sys/.(2*(s+1))  //simplifying the internal
   feedback loop
b=20*2*a;
disp(b,"G(s)")
c=1;
disp(c,"H(s)")
OL=b*c;
disp(OL,"G(s)H(s)")
,"G(s)*H(s)")
syms t s;
Kp=limit(s*OL/s,s,0) //Kp= position error
   coefficient
Kv=limit(s*OL,s,0) //Kv= velocity error coefficient
Ka=limit(s^2*OL,s,0) //Ka= accelaration error
   coefficient
//given input r(t)=6
R=laplace('6',t,s)
//steady state error =Lt s->0 sR(S)/1+G(s)H(S)
e1=limit(s*R/(1+OL),s,0); //e=error for given input
disp(e1,"error")
//given input r(t)=8t
M=laplace('8*t',t,s)
//steady state error =Lt s->0 sR(S)/1+G(s)H(S)
e2=limit(s*M/(1+OL),s,0); //e=error for given input
disp(e2,"error")
//given input r(t)=10+4t+3t^2/2
N=laplace('10+4*t+(3*t^2)/2',t,s)
//steady state error =Lt s->0 sR(S)/1+G(s)H(S)
e3=limit(s*N/(1+OL),s,0); //e=error for given input
```

```
disp(e3,"error")
```

Scilab code Exa 8.11 coefficient

```
//coefficient//
s = poly ( 0,'s' )
sys1 = syslin ('c',(s)/(s+6));
sys2 = syslin ('c',(s+2)/(s+3));
sys3 = syslin ('c',(5)/((s+3)*s^3));
a=sys2+sys3;
disp(a,"H(s)")
b=sys1;
disp(b,"G(S)")
y=a*b;
disp(y,"G(S)H(S)")
syms s
Kp=limit(s*y/s,s,0) //Kp= position error coefficient
Kv=limit(s*y,s,0) //Kv= velocity error coefficient
Ka=limit(s^2*y,s,0) //Ka= accelaration error
    coefficient
```

Scilab code Exa 8.12 coefficient

```
//coefficient//
s=%s;
syms k t;
y=k/((s+1)*s^2*(s+4));
disp(y,"G(s)H(s)=")
r=1+(8*t)+(18*t^2/2);
disp(r,"r(t)=")
R=laplace(r,t,s);
disp(R,"R(s)=")
e=limit((s*R)/(1+y),s,0)
```

```
disp(e,"Ess=")
printf('Given Ess = 0.8 \n")
e=0.8;
k=72/e;
disp(k,"k=")
```

Scilab code Exa 8.13 coefficient

```
//coefficient//
syms s t k;
s = poly ( 0,'s' );
y=k/(s*(s+2)); //G(s)H(s)
disp(y,"G(s)H(s)")
//R=laplace('0.2*t',t,s)
R=laplace('0.2*t',t,s)
e=limit(s*R/(1+y),s,0)
//given e<=0.02
a=[0.02];
b=[-0.4];
m=linsolve(a,b);   //Solves The Linear Equation
disp(m,"k")
```

Scilab code Exa 8.14 coefficient

```
//coefficient//
syms s,t,k;
s=%s;
y=k/(s*(s+2)*(1+0.5*s)) //G(s)H(s)
disp(y,"G(s)H(s)")
//R=laplace('3*t',t,s)
R=laplace('3*t',t,s)
e=limit(s*R/(1+y),s,0);
disp(e," steady state error")
```

```
k=4;//given
y=e;
disp(y,"state state error when k=4")
```

Scilab code Exa 8.15 coefficient

```
//coefficient//
s = poly ( 0,'s' );
sys = syslin ('c',180/(s*(s+6))) //G(s)H(s)
disp(sys,"G(s)H(s)")
syms t s;
//R=laplace('4*t',t,s)
R=laplace('4*t',t,s);
e=limit(s*R/(1+sys),s,0);
y=dbl(e);
disp(y,"steady state error")
syms  k real;
//value of k if error reduced by 6%;
e1=limit(s*R/(1+k/(s*(s+6))),s,0)     //---------1
e1=0.94*e     // ----------2
//now solving these two equations
a=[47];
b=[-9000];
m=linsolve(a,b);
disp(m,"k")
```

Scilab code Exa 8.16 coefficient

```
//coefficient//
s=%s;
F=syslin('c',(81)/(s^2+6*s)); //Creates transfer
   function in forward path
```

```
B=syslin('c',(1+0*s)/(1+0*s)); //Creates transfer function in backward path
CL=F/.B //Calculates closed-loop transfer function
// compare CL with Wn^2/(s^2+2*zeta*Wn+Wn^2)
y=denom(CL) //extracting the denominator of CL
z=coeff(y) //extracting the coefficients of the denominator polynomial
//Wn^2=z(1,1) ,comparing the coefficients
Wn=sqrt(z(1,1))                // Wn=natural frequency
//2*zeta*Wn=z(1,2)
zeta=z(1,2)/(2*Wn)                // zeta=damping factor
Wd=Wn*sqrt(1-zeta^2)
Tp=%pi/Wd     //Tp=peak time
Mp=100*exp((-%pi*zeta)/sqrt(1-zeta^2))  //peak overshoot
```

Scilab code Exa 8.17 coefficient

```
//coefficient//
s=%s;
F=syslin('c',(25)/(s^2+7*s)); //Creates transfer function in forward path
B=syslin('c',(1+0*s)/(1+0*s)); //Creates transfer function in backward path
k=20/25;   //k=gain factor
CL=k*(F/.B) //Calculates closed-loop transfer function
// compare CL with Wn^2/(s^2+2*zeta*Wn+Wn^2)
y=denom(CL) //extracting the denominator of CL
z=coeff(y) //extracting the coefficients of the denominator polynomial
//Wn^2=z(1,1) ,comparing the coefficients
Wn=sqrt(z(1,1))                // Wn=natural frequency
//2*zeta*Wn=z(1,2)
```

```
zeta=z(1,2)/(2*Wn)                    // zeta=damping factor
Wd=Wn*sqrt(1-zeta^2)
Tp=%pi/Wd      //Tp=peak time
Mp=100*exp((-%pi*zeta)/sqrt(1-zeta^2))   //peak overshoot
Td=(1+0.7*zeta)/Wn       //Td=delay time
a=atan(sqrt(1-zeta^2)/zeta)
Tr=(%pi-a)/Wd            //Tr=rise time
Ts=4/(zeta*Wn)           //Ts=settling time
//y(t)=expression for output
y=(20/25)*(1-(exp(-1*zeta*Wn*t)/sqrt(1-zeta^2))*sin(Wd*t+atan(zeta/sqrt(1-zeta^2))));
disp(y,"Y(t)")
```

Scilab code Exa 8.19 coefficient

```
//coefficient//
s=%s;
F=syslin('c',(144)/(s^2+12*s)); //Creates transfer function in forward path
B=syslin('c',(1+0*s)/(1+0*s)); //Creates transfer function in backward path
k=20/25;   //k=gain factor
CL=k*(F/.B) //Calculates closed-loop transfer function
// compare CL with Wn^2/(s^2+2*zeta*Wn+Wn^2)
y=denom(CL) //extracting the denominator of CL
z=coeff(y)  //extracting the coefficients of the denominator polynomial
//Wn^2=z(1,1) ,comparing the coefficients
Wn=sqrt(z(1,1))                 // Wn=natural frequency
//2*zeta*Wn=z(1,2)
zeta=z(1,2)/(2*Wn)                    // zeta=damping factor
```

```
Wd=Wn*sqrt(1-zeta^2)
Tp=%pi/Wd      //Tp=peak time
Mp=100*exp((-%pi*zeta)/sqrt(1-zeta^2))  //peak overshoot
Td=(1+0.7*zeta)/Wn      //Td=delay time
a=atan(sqrt(1-zeta^2)/zeta)
Tr=(%pi-a)/Wd           //Tr=rise time
Ts=4/(zeta*Wn)          //Ts=settling time
```

Scilab code Exa 8.20 coefficient

```
//coefficient//
ieee(2);
syms k T;
num=k;
den=s*(1+s*T);
G=num/den;
disp(G,"G(s)=")
H=1;
CL=G/.H;
CL=simple(CL);
disp(CL,"C(s)/R(s)=")  //Calculates closed-loop transfer function
// compare CL with Wn^2/(s^2+2*zeta*Wn+Wn^2)
[num,den]=numden(CL) //extracts num & den of symbolic function (CL)
den=den/T;
cof_a_0 = coeffs(den,'s',0) // coeff of den of symbolic function (CL)
cof_a_1 = coeffs(den,'s',1)
//Wn^2= cof_a_0 ,comparing the coefficients
Wn=sqrt(cof_a_0)
disp(Wn,"natural frequency Wn")          // Wn=natural frequency
//cof_a_1=2*zeta*Wn
```

```
zeta=cof_a_1/(2*Wn)
```

Scilab code Exa 8.23.01 coefficient

```
//coefficient//
s= poly ( 0,'s' );
sys = syslin ('c',10/(s+2)); //G(s)H(s)
disp(sys,"G(s)H(s)")
F=1/(1+sys)
syms t s;
Co=limit(s*F/s,s,0) //Ko=Lt s->0 (1/(1+G(s)H(S))
a=(3);
e=Co*a;
disp(e,"steady state error")
```

Scilab code Exa 8.23.02 coefficient

```
//coefficient//
s= poly ( 0,'s' );
sys = syslin ('c',10/(s+2)); //G(s)H(s)
disp(sys,"G(s)H(s)")
F=1/(1+sys)
syms t s;
Co=limit(s*F/s,s,0) //Ko=Lt s->0 (1/(1+G(s)H(S))
d=diff(s*F/s,s)
C1=limit(diff(s*F/s,s),s,0) //K1=Lt s->0 (dF(s)/ds)
a=(2*t);
b=diff((2*t) ,t);
e=Co*a+C1*b;
disp(e," steadt state error")
```

Scilab code Exa 8.23.03 coefficient

```
//coefficient//
s= poly ( 0,'s' );
sys = syslin ('c',10/(s+2)); //G(s)H(s)
disp(sys,"G(s)H(s)")
F=1/(1+sys)
syms t s;
Co=limit(s*F/s,s,0) //Ko=Lt s->0 (1/(1+G(s)H(S))
d=diff(s*F/s,s)
C1=limit(diff(s*F/s,s),s,0) //K1=Lt s->0 (dF(s)/ds)
C2=limit(diff(d,s),s,0)    //K2=Lt s->0 (d2F(s)/ds)
a=((t^2)/2);
b=diff((t^2)/2 ,t);
c=diff(b,t);
e=Co*a+C1*b+C2*c;
disp(e,"steady state error")
```

Scilab code Exa 8.24 coefficient

```
//coefficient//
s= poly ( 0,'s' );
sys1 = syslin ('c',(s+3)/(s+5));
sys2= syslin ('c',(100)/(s+2));
sys3= syslin ('c',(0.15)/(s+3));
G=sys1*sys2*sys3*2*5
H=1;
y=G*H; //G(s)H(s)
disp(y,"G(s)H(s)")
F=1/(1+y)
syms t s;
Co=limit(s*F/s,s,0) //Ko=Lt s->0 (1/(1+G(s)H(S))
d=diff(s*F/s,s)
C1=limit(diff(s*F/s,s),s,0) //K1=Lt s->0 (dF(s)/ds)
C2=limit(diff(d,s),s,0)    //K2=Lt s->0 (d2F(s)/ds)
```

```
a=(1+(2*t)+(5*(t^2/2)));
b=diff(a ,t);
c=diff(b,t);
e=Co*a+C1*b+C2*c;
disp(e," steadt state error")
```

Scilab code Exa 8.32 coefficient

```
//coefficient//
s=%s;
sys=syslin('c',(9*(1+2*s))/(s^2+0.6*s+9));
disp(sys,"C(s)/R(s)=")
//given r(t)=u(t)
syms t s;
R=laplace('1',t,s);
disp(R,"R(s)=")
C=R*sys;
disp(C,"C(s)=")
c=ilaplace(C,s,t)
disp(c,"c(t)=")
G=9/(s^3+0.6*s^2);
disp(G,"G(s)=")
H=1;
y=1+G*H;
syms t s;
Kp=limit(s*G/s,s,0)
Kv=limit(s*G,s,0)
Ka=limit(s^2*G,s,0)
R=laplace('(1+t+(t^2/2))',t,s)
//steady state error =Lt s->0 sR(S)/1+G(s)H(S)
e=limit(s*R/(1+y),s,0); //e=error for ramp input
disp(e,"steady state error(Ess)")
```

Chapter 9

Feedback Characteristics of control Systems

Scilab code Exa 9.01 calculates

```
//calculates//
s=%s;
G=syslin('c',20/(s*(s+4)))
H=0.35;
y=G*H;

S=1/(1+y);
disp(S,"1/(1+G(s)*H(s))")

//given w=1.2
w=1.2
s=%i*w
S=horner(S,s) //calculates value of S at s
a=abs(S)
disp(a,"sensitivity of open loop")

F=-y/(1+y)
disp(F,"(-G(s)*H(s))/(1+G(s)*H(s))")
S=horner(F,s)  //calculates value of F at s
```

```
b=abs(S)
disp(b,"sensitivity of closed loop")
```

Scilab code Exa 9.02 calculates

```
//calculates//
s=%s;
sys1=syslin('c',9/(s*(s+1.8)));
syms Td ;
sys2=1+(s*Td);
sys3=sys1*sys2;
H=1;
CL=sys3/.H; //Calculates closed-loop transfer function
disp(CL,"C(s)/R(s)")
// compare CL with Wn^2/(s^2+2*zeta*Wn+Wn^2)
[num,den]=numden(CL) //extracts num & den of symbolic function CL
den=den/5;
cof_a_0 = coeffs(den,'s',0) // coeff of den of symbolic function CL
cof_a_1 = coeffs(den,'s',1)
//Wn^2= cof_a_0 ,comparing the coefficients
Wn=sqrt(cof_a_0)
disp(Wn,"natural frequency Wn")            // Wn= natural frequency
//cof_a_1=2*zeta*Wn
zeta=cof_a_1/(2*Wn)
zeta=1;disp(zeta,"for criticaly damped function zeta")
Td=((2*Wn)-1.8)/9
Ts=4/(zeta*Wn);
Ts=dbl(Ts);
disp(Ts,"settling time Ts")
```

Scilab code Exa 9.03 calculates

```
//calculates//
s=%s;
G=syslin('c',40/(s*(s+4)))
H=0.50;
y=G*H;
S=1/(1+y);
disp(S,"1/(1+G(s)*H(s))")
//given w=1.3
w=1.3
s=%i*w
S=horner(S,s)
a=abs(S)
disp(a,"sensitivity of open loop")
F=-y/(1+y)
disp(F,"(-G(s)*H(s))/(1+G(s)*H(s))")
S=horner(F,s)
b=abs(S)
disp(b,"sensitivity of closed loop")
```

Scilab code Exa 9.04 calculates

```
//calculates//
s=%s;
syms s;
syms Wn zeta A H real;
T=6.28;
Wn=(8*%pi)/T;
zeta=0.3
n=Wn^2;
d=s^2+2*zeta*Wn*s+Wn^2;
```

```
G1=n/d;
disp(G1,"G1(s)")
G=A*G1;
disp(G,"G(s)")
S1=(diff(G,A))*(A/G);
a=simple(S1);
disp(a,"open loop sensitivity for changes in A")
M=G/.H;
p=simple(M)
S2=(diff(p,A))*(A/p);
b=simple(S2);
disp(b,"closed loop sensitivity for changes in A")
S3=(diff(p,H))*(H/p);
c=simple(S3);
disp(c,"closed loop sensitivity for changes in H")
```

Scilab code Exa 9.05 calculates

```
//calculates//
s=%s;
sys1=syslin('c',(s+3)/s);
syms u rp k H RL;
num2=u*RL*s*(s+2);
den2=(rp+RL)*(s+3);
sys2=num2/den2;
num3=k;
den3=s+2;
sys3=num3/den3;
sys=sys1*sys2*sys3;
disp(sys,"G(s)=");
RL=10*10^3;
rp=4*10^3;
sys=eval(sys)
sys=float(sys)
disp(sys,"sys=");
```

```
disp(H,"H(s)");
M=sys/.H  //G(s)/1+G(s)H(S)
M=simple(M)
S=(diff(M,u))*(u/M);
S=simple(S);
disp(S,"system sensitivity due to variation of u=");
H=0.3;
u=12;
S=eval(S)  //----------eq 1
S=0.04;
k=((7/S)-7)/18 ; //from eq 1
disp(k,"K=")
```

Scilab code Exa 9.06 calculates

```
//calculates//
G=210;
H=0.12;
syms Eo Er
printf("for closed loop system")
sys=G/.H;  //Eo/Er=G/(1+GH)
disp(sys,"Eo/Er=")
Eo=240 //given
Er=Eo/8.0152;
disp(Er,"Er=")
printf("for open loop system")
disp(G,"Eo/Er=")
Er=Eo/G;
G=210;
disp(Er,"Er=")
//printf("since G is reduced by 12%, the new value
    of gain is 784.8V");
S=1/(1+G*H)
disp(S,"(%change in M)/(%change in G)=")
disp(12,"%CHANGE IN G=")
```

```
y=12*0.03869;
disp(y,"%CHANGE IN M=")
printf("for open loop system")
disp(28.8*100/240,"%change in Eo")
```

Scilab code Exa 9.07 calculates

```
//calculates//
s=%s;
sys1=syslin('c',20/(s*(s+2)));
syms Kt;
sys2=Kt*s;
sys3=sys1/.sys2;
p=simple(sys3);
disp(p,"G(s)=")
H=1;
sys=sys3/.H;
sys=simple(sys);
disp(sys,"C(s)/R(s)=")
[num,den]=numden(sys)
cof_a_0 = coeffs(den,'s',0) // coeff of den of
   symbolic function CL
cof_a_1 = coeffs(den,'s',1)
//Wn^2= cof_a_0,comparing the coefficients
Wn=sqrt(cof_a_0)
Wn=dbl(Wn);
disp(Wn,"natural frequency Wn=")              // Wn=
   natural frequency
//cof_a_1=2*zeta*Wn
zeta=cof_a_1/(2*Wn)
zeta=0.6;
Kt=((2*zeta*Wn)-2)/20;
disp(Kt,"Kt=")
Wd=Wn*sqrt(1-zeta^2);
disp(Wd,"Wd=")
```

```
Tp=%pi/Wd;
disp(Tp,"Tp=")
Mp=100*exp((-%pi*zeta)/sqrt(1-zeta^2));
disp(Mp,"Mp=")
Ts=4/(zeta*Wn);
disp(Ts,"Ts=")
```

Scilab code Exa 9.08 calculates

```
//calculates//
s=%s;
printf( "1)zeta & Wn without Kd")
G=60*syslin('c',1/(s*(s+4)));
disp(G,"G(S)=")
CL=G/.H;
disp(CL,"C(s)/R(s)=")
y=denom(CL) //extracting the denominator of CL
z=coeff(y)  //extracting the coefficients of the
    denominator polynomial
//Wn^2=z(1,1) ,comparing the coefficients
Wn=sqrt(z(1,1))                // Wn=natural frequency
//2*zeta*Wn=z(1,2)
zeta=z(1,2)/(2*Wn)
sys1=syslin('c',1/(s*(s+4)));
syms Kd;
printf("2)Kd for zeta=0.60 with controller")
sys2=s*Kd;
sys3=sys1/.sys2;
G=sys3*60;
disp(G,"G(s)=")
H=1;
sys=G/.H;
disp(sys,"C(s)/R(s)=")
[num,den]=numden(sys)
cof_a_0 = coeffs(den,'s',0)
```

```
cof_a_1 = coeffs(den,'s',1)
//Wn^2= cof_a_0 ,comparing the coefficients
Wn=sqrt(cof_a_0)
Wn=dbl(Wn);
disp(Wn,"natural frequency Wn=")
//cof_a_1=2*zeta*Wn
zeta=0.60
Kd=(2*zeta*Wn)-4
```

Scilab code Exa 9.09 calculates

```
//calculates//
s=%s;
printf("1)without controller")
G=syslin('c',120/(s*(s+12.63)));
disp(G,"G(s)=")
H=1;
CL=G/.H;
disp(CL,"C(s)/R(s)=")
y=denom(CL) //extracting the denominator of CL
z=coeff(y) //extracting the coefficients of the denominator polynomial
//Wn^2=z(1,1) ,comparing the coefficients
Wn=sqrt(z(1,1));
disp(Wn,"Wn=")              // Wn=natural frequency
//2*zeta*Wn=z(1,2)
zeta=z(1,2)/(2*Wn);
disp(zeta,"zeta=")
printf("2) with controller')
G=syslin('c',(120*(30+s))/(s*(s+12.63)*30));
disp(G,"G(s)=")
CL=G/.H;
disp(CL,"C(s)/R(s)=")
den=denom(CL)
den=den/30 //extracting the denominator of CL
```

```
z=coeff(den)  //extracting the coefficients of the
    denominator polynomial
//Wn^2=z(1,1) ,comparing the coefficients
Wn=sqrt(z(1,1));
disp(Wn,"Wn=")               // Wn=natural frequency
//2*zeta*Wn=z(1,2)
zeta=z(1,2)/(2*Wn);
Mp=100*exp((-%pi*zeta)/sqrt(1-zeta^2));
disp(Mp,"Mp=")
Ts=4/(zeta*Wn);
disp(Ts,"Ts=")
```

Scilab code Exa 9.10 calculates

```
//calculates//
s=%s;
printf("1)without controller")
G=64*syslin('c',1/(s*(s+4)));
disp(G,"G(s)=")
H=1;
CL=G/.H;
disp(CL,"C(s)/R(s)=")
//Extracting the denominator of CL
y=denom(CL)
//Extracting the coefficients of the denominator
    polynomial
z=coeff(y)
//Wn^2=z(1,1) ,comparing the coefficients
Wn=sqrt(z(1,1));
//Wn=natural frequency
disp(Wn,"Wn=")
printf("2) with controller')
syms K;
sys1=syslin('c',1/(s*(s+4)));
sys2=sys1 /.(s*K);
```

```
G=64*sys2
disp(G,"G(s)=")

sys=G/.H;
sys=simple(sys);
disp(sys,"C(s)/R(s)=")
[num,den]=numden(sys)
//Coeff of den of symbolic function CL
cof_a_0 = coeffs(den,'s',0)
cof_a_1 = coeffs(den,'s',1)
//Wn^2= cof_a_0 ,comparing the coefficients
Wn=sqrt(cof_a_0)
Wn=dbl(Wn);
//Wn=natural frequency
disp(Wn,"natural frequency Wn=")
//cof_a_1=2*zeta*Wn
zeta=cof_a_1/(2*Wn)
zeta=0.6;
k=(16*zeta)-4
disp(k,"K=")
```

Scilab code Exa 9.11 calculates

```
//calculates//
printf("2) with controller')
syms K;
sys1=syslin('c',1/(s*(s+1.2)));
sys2=sys1/.(s*K);
G=16*sys2;
G=simple(G)
disp(G,"G(s)=")
sys=G/.H;
sys=simple(sys);
disp(sys,"C(s)/R(s)=")
[num,den]=numden(sys)
```

```
den=den/5; //so that coeff of s^2=1
cof_a_0 = coeffs(den,'s',0) // coeff of den of symbolic function CL
cof_a_1 = coeffs(den,'s',1)
//Wn^2= cof_a_0,comparing the coefficients
Wn=sqrt(cof_a_0)
Wn=dbl(Wn);
disp(Wn,"natural frequency Wn=")            // Wn=natural frequency
//cof_a_1=2*zeta*Wn
//zeta=cof_a_1/(2*Wn)
zeta=0.56;
k=(8*zeta)-1.2
disp(k,"K=")
Wd=Wn*sqrt(1-zeta^2);
disp(Wd,"Wd=")
Tp=%pi/Wd;
disp(Tp,"Tp=")
Mp=100*exp((-%pi*zeta)/sqrt(1-zeta^2));
disp(Mp,"Mp=")
Ts=4/(zeta*Wn);
disp(Ts,"Ts=")
```

Chapter 10

Stability

Scilab code Exa 6.0 coefficient

```
//coefficient//
ieee(2);
s=%s;
m=s^5+2*s^4+4*s^3+8*s^2+3*s+1
r=coeff(m); //Extracts the coefficient of the polynomial
n=length(r);
routh=[r([6,4,2]);r([5,3,1])]
syms eps;
routh=[routh;eps,-det(routh(1:2,2:3))/routh(2,2),0];
routh=[routh;-det(routh(2:3,1:2))/routh(3,1),-det(routh(2:3,2:3))/routh(3,2),0];
routh=[routh;-det(routh(3:4,1:2))/routh(4,1),-det(routh(3:4,2:3))/routh(4,2),0];
routh=[routh;-det(routh(4:5,1:2))/routh(5,1),0,0];
disp(routh,"routh=")
//To check the stability
routh(4,1)=8-limit(5/eps,eps,0); //Putting the value of eps=0 in routh(4,1)
disp(routh(4,1),"routh(4,1)=")
routh(5,1)= limit(routh(5,1),eps,0); //Putting the
```

```
   value of eps=0 in routh(5,1)
disp(routh(5,1),"routh(5,1)=')
routh
printf("There are two sign changes of first column,
   hence the system is unstable \n")
```

Scilab code Exa 10.02.01 equation

```
//equation//
s = poly(0, "s");
p=poly([12],'s','coeff');
q=poly([0 2 4 1],'s','coeff');
G=p/q;
H=poly([0.5],'s','coeff');
//characteristic equation is 1+G(s)H(s)=0
y=1+G*H
r=numer(y)
disp('=0',r,"characteristics equation is")
```

Scilab code Exa 10.02.02 equation

```
//equation//
s = poly(0, "s");
p=poly([7],'s','coeff');
q=poly([ 2 3 1],'s','coeff');
G=p/q;
H=poly([0 1],'s','coeff');
//characteristic equation is 1+G(s)H(s)=0
y=1+G*H
r=numer(y)
disp('=0',r,"characteristics equation is")
```

Scilab code Exa 10.02.03 equation

```
//equation//
s = poly(0, "s");
G=syslin('c',2/(s^2+2*s))
H=syslin('c',1/s);
//characteristic equation is 1+G(s)H(s)=0
y=1+G*H
r=numer(y)
disp('=0',r,"characteristics equation is")
```

Scilab code Exa 10.03 equation

```
//equation//
s=%s;
m=s^3+5*s^2+10*s+3;
r=coeff(m)
n=length(r);
routh=[r([4,2]);r([3,1])];
routh=[routh;-det(routh)/routh(2,1),0];
t=routh(2:3,1:2); //extracting the square sub block
   of routh matrix
routh=[routh;-det(t)/t(2,1),0]
c=0;
for i=1:n
if (routh(i,1)< 0)
c=c+1;
end
end
if(c>=1)
printf("system is unstable")
else ("system is stable")
```

```
end
```

Scilab code Exa 10.04 equation

```
//equation//
s=%s;
m=s^3+2*s^2+3*s+10;
r=coeff(m)
n=length(r);
routh=[r([4,2]);r([3,1])];
routh=[routh;-det(routh)/routh(2,1),0];
t=routh(2:3,1:2); //extracting the square sub block
    of routh matrix
routh=[routh;-det(t)/t(2,1),0]
c=0;
for i=1:n
if (routh(i,1)<0)
c=c+1;
end
end
if(c>=1)
printf("system is unstable")
else ("system is stable")
end
```

Scilab code Exa 10.05.01 equation

```
//equation//
ieee(2)
s=%s;
m=s^4+6*s^3+21*s^2+36*s+20
r=coeff(m)
n=length(r);
```

```
routh=[r([5,3,1]);r([4,2]),0]
routh=[routh;-det(routh(1:2,1:2))/routh(2,1),-det(
   routh(1:2,2:3))/routh(2,2),0];
routh=[routh;-det(routh(2:3,1:2))/routh(3,1),-det(
   routh(2:3,2:3))/routh(3,2),0];
routh=[routh;-det(routh(3:4,1:2))/routh(4,1),0,0];
disp(routh,"routh=")
c=0;
for i=1:n
if (routh(i,1)<0)
c=c+1;
end
end
if(c>=1)
printf("system is unstable")
else ("system is stable")
end
```

Scilab code Exa 10.05.02 equation

```
//equation//
s=%s;
m=s^5+6*s^4+3*s^3+2*s^2+s+1
r=coeff(m)
n=length(r)
routh=[r([6,4,2]);r([5,3,1])]
routh=[routh;-det(routh(1:2,1:2))/routh(2,1),-det(
   routh(1:2,2:3))/routh(2,2),0]
routh=[routh;-det(routh(2:3,1:2))/routh(3,1),-det(
   routh(2:3,2:3))/routh(3,2),0]
routh=[routh;-det(routh(3:4,1:2))/routh(4,1),-det(
   routh(3:4,2:3))/routh(4,2),0]
routh=[routh;-det(routh(4:5,1:2))/routh(5,1),0,0]
c=0;
for i=1:n
```

```
if (routh(i,1)<0)
c=c+1;
end
end
if(c>=1)
printf("system is unstable")
else ("system is stable")
end
```

Scilab code Exa 10.06 equation

```
//equation//
ieee(2)
s=%s;
m=s^5+2*s^4+4*s^3+8*s^2+3*s+1
r=coeff(m);  //Extracts the coefficient of the
    polynomial
n=length(r);
routh=[r([6,4,2]);r([5,3,1])]
syms eps;
routh=[routh;eps,-det(routh(1:2,2:3))/routh(2,2),0];
routh=[routh;-det(routh(2:3,1:2))/routh(3,1),-det(
    routh(2:3,2:3))/routh(3,2),0];
routh=[routh;-det(routh(3:4,1:2))/routh(4,1),-det(
    routh(3:4,2:3))/routh(4,2),0];
routh=[routh;-det(routh(4:5,1:2))/routh(5,1),0,0];
disp(routh,"routh=")
//To check the stability
routh(4,1)=8-limit(5/eps,eps,0); //Putting the value
     of eps=0 in routh(4,1)
disp(routh(4,1),"routh(4,1)=")
routh(5,1)= limit(routh(5,1),eps,0); //Putting the
    value of eps=0 in routh(5,1)
disp(routh(5,1),"routh(5,1)=')
routh
```

```
printf("There are two sign changes of first column,
   hence the system is unstable \n")
```

Scilab code Exa 10.07 equation

```
//equation//
s=%s;
m=s^5+2*s^4+2*s^3+4*s^2+4*s+8
routh=routh_t(m) //This Function generates the Routh
    table
c=0;
for i=1:n
if (routh(i,1)<0)
c=c+1;
 end
  end
  if(c>=1)
    printf("system is unstable")
  else ("system is stable")
  end
```

Scilab code Exa 10.08 equation

```
//equation//
s=%s;
m=s^4+5*s^3+2*s^2+3*s+2
r=coeff(m)
n=length(r);
routh=[r([5,3,1]);r([4,2]),0]
routh=[routh;-det(routh(1:2,1:2))/routh(2,1),-det(
   routh(1:2,2:3))/routh(2,2),0];
routh=[routh;-det(routh(2:3,1:2))/routh(3,1),-det(
   routh(2:3,2:3))/routh(3,2),0];
```

```
routh=[routh;-det(routh(3:4,1:2))/routh(4,1),0,0]
  c=0;
 for i=1:n
    if (routh(i,1)<0)
      c=c+1;
    end
  end
  if(c>=1)
    printf("system is unstable")
  else ("system is stable")
  end
```

Scilab code Exa 11.08 value

```
//value//
s=%s;
H=syslin('c',(s+2)/((s+1)*s*(s+4)));
evans(H,100)
printf("From the graph we observed that, \n a)The no of loci ending at inf is 2 \n b)Three loci will start from s= 0,-1 & -4,\n c)One loci will end at -2 & remaining two will end at inf")
```

Scilab code Exa 10.09 equation

```
//equation//
ieee(2);
s=%s;
m=s^5+s^4+3*s^3+3*s^2+4*s+8
r=coeff(m);  //Extracts the coefficient of the polynomial
n=length(r);
routh=[r([6,4,2]);r([5,3,1])]
```

```
syms eps;
routh=[routh;eps,-det(routh(1:2,2:3))/routh(2,2),0];
routh=[routh;-det(routh(2:3,1:2))/routh(3,1),-det(
   routh(2:3,2:3))/routh(3,2),0];
routh=[routh;-det(routh(3:4,1:2))/routh(4,1),-det(
   routh(3:4,2:3))/routh(4,2),0];
routh=[routh;-det(routh(4:5,1:2))/routh(5,1),0,0];
disp(routh,"routh=")
//To check the stability
routh(4,1)=limit(routh(4,1),eps,0); //Putting the
   value of eps=0 in routh(4,1)
disp(routh(4,1),"routh(4,1)=")
routh(5,1)= limit(routh(5,1),eps,0); //Putting the
   value of eps=0 in routh(5,1)
disp(routh(5,1),"routh(5,1)=')
routh
printf("There are two sign changes of first column,
   hence the system is unstable \n")
```

Scilab code Exa 10.10 equation

```
//equation//
ieee(2);
syms  s k;
m=s^4+4*s^3+7*s^2+6*s+k;
cof_a_0 = coeffs(m,'s',0);
cof_a_1 = coeffs(m,'s',1);
cof_a_2 = coeffs(m,'s',2);
cof_a_3 = coeffs(m,'s',3);
cof_a_4 = coeffs(m,'s',4);

r=[cof_a_0 cof_a_1 cof_a_2 cof_a_3  cof_a_4]

n=length(r);
routh=[r([5,3,1]);r([4,2]),0]
```

```
routh=[routh;-det(routh(1:2,1:2))/routh(2,1),-det(
    routh(1:2,2:3))/routh(2,2),0];
routh=[routh;-det(routh(2:3,1:2))/routh(3,1),-det(
    routh(2:3,2:3))/routh(3,2),0];
routh=[routh;-det(routh(3:4,1:2))/routh(4,1),0,0];
disp(routh,"routh=")
```

Scilab code Exa 10.11 equation

```
//equation//
ieee(2);
syms  s k;
m=(24/100)*s^3+s^2+s+k;
cof_a_0 = coeffs(m,'s',0);
cof_a_1 = coeffs(m,'s',1);
cof_a_2 = coeffs(m,'s',2);
cof_a_3 = coeffs(m,'s',3);
r=[cof_a_0 cof_a_1 cof_a_2 cof_a_3]
n=length(r);
routh=[r([4,2]);r([3,1])]
routh=[routh;-det(routh)/routh(2,1),0]
t=routh(2:3,1:2); //extracting the square sub block
    of routh matrix
routh=[routh;-det(t)/routh(3,1),0]
disp(routh,"routh=");
routh(3,1)=0  //For marginaly stable system
k=1/0.24;
disp(k,"K(marginal)=")
disp('=0',(s^2)+k,"auxillary equation")
s=sqrt(-k);
disp(s,"Frequency of oscillation(in rad/sec)=")
```

Scilab code Exa 10.12 equation

```
//equation//
ieee(2);
syms p K s;
m=s^3+(p*s^2)+(K+3)*s+(2*(K+1))
cof_a_0 = coeffs(m,'s',0);
cof_a_1 = coeffs(m,'s',1);
cof_a_2 = coeffs(m,'s',2);
cof_a_3 = coeffs(m,'s',3);
r=[cof_a_0 cof_a_1 cof_a_2 cof_a_3]
n=length(r);
routh=[r([4,2]);r([3,1])];
routh=[routh;-det(routh)/routh(2,1),0];
t=routh(2:3,1:2); //extracting the square sub block
    of routh matrix
routh=[routh;-det(t)/routh(3,1),0];
disp(routh,"routh=")
```

Scilab code Exa 10.13 equation

```
//equation//
ieee(2);
s=%s;
m=2*s^4+(4*s^2)+1
routh=routh_t(m)     //Generates the Routh Table
roots(m)             //Gives the Roots of the
    Polynomial(m)
disp(0,"the number of roots lying in the left half
    plane of s-plane=")
disp(0,"the number of roots lying in the right half
    plane of s-plane=")
disp(4,"the number of roots lying on the imaginary
    axis=")
```

Scilab code Exa 10.14 equation

```
//equation//
ieee(2);
syms  s k;
m=s^4+6*s^3+10*s^2+8*s+k;
cof_a_0 = coeffs(m,'s',0);
cof_a_1 = coeffs(m,'s',1);
cof_a_2 = coeffs(m,'s',2);
cof_a_3 = coeffs(m,'s',3);
cof_a_4 = coeffs(m,'s',4);
r=[cof_a_0 cof_a_1 cof_a_2 cof_a_3  cof_a_4]
n=length(r);
routh=[r([5,3,1]);r([4,2]),0]
routh=[routh;-det(routh(1:2,1:2))/routh(2,1),-det(
    routh(1:2,2:3))/routh(2,2),0];
routh=[routh;-det(routh(2:3,1:2))/routh(3,1),-det(
    routh(2:3,2:3))/routh(3,2),0];
routh=[routh;-det(routh(3:4,1:2))/routh(4,1),0,0];
disp(routh,"routh=")
```

Scilab code Exa 10.15 equation

```
//equation//
ieee(2);
syms s T;
m=s^2+(2-T)*s+1
cof_a_0 = coeffs(m,'s',0);
cof_a_1 = coeffs(m,'s',1);
cof_a_2 = coeffs(m,'s',2);
r=[cof_a_0 cof_a_1 cof_a_2]
n=length(r);
routh=[r([3,1]);r(2),0];
routh=[routh;-det(routh)/routh(2,1),0];
disp(routh,"routh=")
```

Scilab code Exa 10.16 equation

```
//equation//
ieee(2)
s=%s;
m=s^6+2*s^5+7*s^4+10*s^3+14*s^2+8*s+8
routh=routh_t(m);
disp(routh,"routh=")
 c=0;
 for i=1:n
    if (routh(i,1)<0)
      c=c+1;
    end
  end
  if(c>=1)
    printf("system is unstable")
  else ("system is stable")
  end
```

Chapter 11

Root Locus Method

Scilab code Exa 11.01 value

```
//value//
s=%s;
sys1=syslin('c',1/(s+1));
evans(sys1,200)
printf("If k is varied from 0 to any value ,root
    locus varies from -k to 0 \n ")
```

Scilab code Exa 11.02 value

```
//value//
s=%s;
sys1=syslin('c',(s+1)/(s+4));
evans(sys1,100)
printf("rootlocus begins at s=-4 & ends at s=-1")
```

Scilab code Exa 11.03 value

```
//value//
s=%s;
sys1=syslin('c',(s+3-%i)*(s+3+%i)/((s+2-%i)*(s+2+%i)
   ));
evans(sys1,100)
printf("Rootlocus starts from s=-2+i & -2-i ends at
   s=-3+i &-3- i \n")
```

Scilab code Exa 11.04 value

```
//value//
s=%s;
H=syslin('c',(s+1)/(s+2));
evans(H,100)
printf(" Clearly from the graph it  observed that
   given point -1+i & -3+i does not lie on the root
   locus \n")
// there is another process to check whether the
   points lie on the locus of the system
P=-1+%i;    //P=selected point
k1=-1/real(horner(H,P))
Ns=H('num');Ds=H('den');
roots(Ds+k1*Ns)      //does not contains P as
   particular root
P=-3+%i;    //P=selected point
k2=-1/real(horner(H,P));
Ns=H('num');Ds=H('den')
roots(Ds+k2*Ns)      //does not contains P as
   particular root
```

Scilab code Exa 11.05 value

```
//value//
```

```
s=%s;
H=syslin('c',1/(s*(s+1)*(s+3)));
evans(H,100)
printf("Clearly from the graph it  observed that
   given point -0.85  lies on the root locus \n")
// there is another process to check whether the
   points lie on the locus of the system
P=-0.85;     //P=selected point
k=-1/real(horner(H,P));
disp(k,"k=')
Ns=H('num');Ds=H('den');
roots(Ds+k*Ns)      //contains P as particular root
```

Scilab code Exa 11.06 value

```
//value//
s=%s;
H=syslin('c',1/((s+1)*(s+5)));
evans(H,100)
printf("Clearly from the graph it  observed that
   given point -0.85  lies on the root locus \n")
// there is another process to check whether the
   points lie on the locus of the system
P=-3+5*%i;     //P=selected point
k=-1/real(horner(H,P));
disp(k,"k=')
Ns=H('num');Ds=H('den');
roots(Ds+k*Ns)      //contains P as particular root
```

Scilab code Exa 11.08 value

```
//value//
s=%s;
```

```
H=syslin('c',(s+2)/((s+1)*s*(s+4)));
evans(H,100)
printf("From the graph we observed that, \n a)The no
    of loci ending at inf is 2 \n b)Three loci will
   start from s= 0,-1 & -4,\n c)One loci will end at
    -2 & remaining two will end at inf")
```

Scilab code Exa 11.09 value

```
//value//
s=%s;
H=syslin('c',(s+2)/((s+1)*s*(s+4)));
evans(H,100)
```

Scilab code Exa 11.10 value

```
//value//
n=3;
disp(n,"no of poles=")
m=1;
disp(m,"no of poles=")
q=0;
O=((2*q)+1)/(n-m)*180;
disp(O,"q=")
q=1;
O=((2*q)+1)/(n-m)*180;
disp(O,"q=")

printf("Centroid=((sum of all real part of poles of
   G(s)H(s))-(sum of all real part of zeros of G(s)H
   (s))/(n-m) \n")
C=((0-1-4)-(-2))/2;
disp(C,"centroid=")
```

Scilab code Exa 11.11 value

```
//value//
n=3;
disp(n,"no of poles=")
m=0;
disp(m,"no of poles=")
q=0;
O=((2*q)+1)/(n-m)*180;
disp(O,"q=")
q=1;
O=((2*q)+1)/(n-m)*180;
disp(O,"q=")
q=2;
O=((2*q)+1)/(n-m)*180;
disp(O,"q=")

printf("Centroid=((sum of all real part of poles of
   G(s)H(s))-(sum of all real part of zeros of G(s)H
   (s))/(n-m) \n")
C=((0-1-1)-(-0))/3;
disp(C,"centroid=")
```

Scilab code Exa 11.12 value

```
//value//
n=4;
disp(n,"no of poles=")
m=1;
disp(m,"no of poles=")
q=0;
```

```
O=((2*q)+1)/(n-m)*180;
disp(O,"q=")
q=1;
O=((2*q)+1)/(n-m)*180;
disp(O,"q=")
q=2;
O=((2*q)+1)/(n-m)*180;
disp(O,"q=")

printf("Centroid=((sum of all real part of poles of
    G(s)H(s))-(sum of all real part of zeros of G(s)H
    (s))/(n-m) \n")
C=((0-2-4-5)-(-3))/3;
disp(C,"centroid=")
```

Scilab code Exa 11.13 value

```
//value//
s=%s;
H=syslin('c',(s+2)/((s+1)*s*(s+3)));
plzr(H)
printf("There are two adjacent placed poles at s=0 &
    s=-1 \n")
printf("One breakaway point exists between s=0 & s
    =-1 \n")
```

Scilab code Exa 11.14 value

```
//value//
s=%s;
H=syslin('c',((s+2)*(s+4))/((s^2)*(s+5)));
plzr(H)
```

```
printf("There are two adjacent placed zeros at s=-2
    &s=-4 \n")
printf("One breakin point exists between s=-2 & s=-4
     \n")
```

Scilab code Exa 11.15 value

```
//value//
s=%s;
H=syslin('c',(s+6)/((s+1)*(s+3)));
plzr(H)
printf("There are two adjacent placed poles at s=-3
    &s=-1 \n")
printf("One breakaway point exists between s=-3 & s
    =-1 \n")
printf("One breakin point exists to the left of
    zeros at s=-6 \n")
```

Scilab code Exa 11.16 value

```
//value//
s=%s;
H=syslin('c',1/((s+1)*s*(s+3)));
plzr(H)
printf("There are two adjacent placed poles at s=0 &
    s=-1 \n")
printf("One breakaway point exists between s=0 & s
    =-1 \n")
```

Scilab code Exa 11.17 value

```
//value//
s=%s;
H=syslin('c',1/((s+1)*s*(s+3)));
evans(H,100)
syms k;
m=s^3+6*s^2+8*s+k;
 cof_a_0 = coeffs(m,'s',0);
    cof_a_1 = coeffs(m,'s',1);
    cof_a_2 = coeffs(m,'s',2);
    cof_a_3 = coeffs(m,'s',3);
    r=[cof_a_0 cof_a_1 cof_a_2 cof_a_3]

n=length(r);
routh=[r([4,2]);r([3,1])];
routh=[routh;-det(routh)/routh(2,1),0];
t=routh(2:3,1:2); //extracting the square sub block of routh matrix
routh=[routh;-det(t)/t(2,1),0]
disp(48,"K(marginal)=")
disp('=0',(6*s^2)+k,"auxillary equation")
k=48;
s=sqrt(-k/6);
disp(s,"s=")
```

Scilab code Exa 11.19 value

```
//value//
s=%s;
H=syslin('c',1/((s+1+%i)*s*(s+1-%i)));
evans(H,100)
```

Scilab code Exa 11.20 value

```
//value//
s=%s;
H=syslin('c',1/((s+3)*s*(s+5)));
evans(H,100)
```

Scilab code Exa 11.21 value

```
//value//
s=%s;
H=syslin('c',1/((s+2+%i)*(s+1)*(s+2-%i)));
evans(H,100)
```

Scilab code Exa 11.22 value

```
//value//
s=%s;
num=real(poly([1],'s',"coeff"))
den=real(poly([-1,-2+%i,-2-%i],'s'))
H=num/den
evans(H,100)
k=1.5;
disp(k,"K(design)=")
//Kpure calculates the value of k at imaginary
    crossover
[K,Y]=kpure(H)
GM=K/k;
disp(GM,"value of k at imaginary crossover/k(design)
    =")
disp(GM,"gain margin=")
```

Scilab code Exa 11.23 value

```
//value//
s=%s;
H=syslin('c',1/(s*((s+3)^2)));
evans(H,100)
K=25;
y=K*H; //------eq 1)
disp(K*H,"G(s)H(s)=");
disp('=1',K*H,"mod(G(s)H(s))");
//on solving eq 1 for s=%i*w this we get an equation
    m
w=poly(0,'w');
m=w^3+9*w-25
n=roots(m)
s=%i*n(1)
p=horner(y,s)
[R,Theta]=polar(p)
PM=180+Theta
```

Scilab code Exa 11.25 value

```
//value//
s=%s;
H=syslin('c',1/((s+1)*s*(s+2)*(s+4)));
evans(H,100)
```

Scilab code Exa 11.26 value

```
//value//
s=%s;
H=syslin('c',(s+4)*(s+5)/((s+1)*(s+3)));
evans(H,100)
```

Scilab code Exa 11.27 value

```
//value//
s=%s;
syms k Wn;
H=syslin('c',1/((s+3)^2*s));
evans(H,100) //root locus
printf("To determine the value of Wn \n")
disp(k*H,"G(s)H(s)=")
y=1+(k*H);
disp('=0',y,"1+G(s)H(s)")
evans(H,100)
[num,den]=numden(y)

 cof_a_0 = coeffs(num,'s',0);
    cof_a_1 = coeffs(num,'s',1);
    cof_a_2 = coeffs(num,'s',2);
    cof_a_3 = coeffs(num,'s',3);
    r=[cof_a_0 cof_a_1 cof_a_2 cof_a_3]

n=length(r);
routh=[r([4,2]);r([3,1])];
routh=[routh;-det(routh)/routh(2,1),0];
t=routh(2:3,1:2); //extracting the square sub block of routh matrix
routh=[routh;-det(t)/t(2,1),0]
//to obtain Wn
disp('=0',((6*s^2)+54),"auxillary eq")
p=(6*(s^2))+k;
s=%i*Wn
k=54;
p=eval(p)
Wn=sqrt(k/6)
printf("With gvn values of zeta adding a grid on
```

```
    root locus \n")

zeta=0.5; //given
sgrid(zeta,Wn,7) //add a grid over an existing
    continuous s-plane root with given values for
    zeta and wn.
printf("NOTE:-click on the point where locus
    intersects z=0.5 for desired value of k \n")
k=-1/real(horner(H,[1,%i]*locate(1)))  //To obtain
    the gain at a given point of the locus


p=locate(1) //desired point on the root locus
```

Scilab code Exa 11.28 value

```
//value//
s=%s;
H=syslin('c',1/((s+4)*s*(s+6)));
evans(H,100)
```

Chapter 12

Frequency Domain Analysis

Scilab code Exa 12.01 denominator polynomial

```
//denominator polynomial//
s=poly(0,'s'); //Defines s as polynomial variable
F=syslin('c',[225/((s+6)*s)]); //Creates transfer
    function in forward path
B=syslin('c',(1+0*s)/(1+0*s)); //Creates transfer
    function in backward path
CL=F/.B //Calculates closed-loop transfer function
// compare CL with Wn^2/(s^2+2*zeta*Wn+Wn^2)
y=denom(CL) //extracting the denominator of CL
z=coeff(y)  //extracting the coefficients of the
    denominator polynomial
//Wn^2=z(1,1) ,comparing the coefficients
Wn=sqrt(z(1,1))              // Wn=natural frequency
//2*zeta*Wn=z(1,2)
zeta=z(1,2)/(2*Wn)                // zeta=damping
    factor
Mr=1/(2*zeta*sqrt(1-zeta^2))
Wr=Wn*sqrt(1-zeta^2)
Wc=Wn*sqrt((1-2*zeta^2)+sqrt(4*zeta^4-4*zeta^2+2))
BW=Wc //BANDWIDTH
```

Scilab code Exa 12.02 denominator polynomial

```
//denominator polynomial//
s=poly(0,'s'); //Defines s as polynomial variable
F=syslin('c',[36/((s+8)*s)]); //Creates transfer
    function in forward path
B=syslin('c',(1+0*s)/(1+0*s)); //Creates transfer
    function in backward path
CL=F/.B //Calculates closed-loop transfer function
// compare CL with Wn^2/(s^2+2*zeta*Wn+Wn^2)
y=denom(CL) //extracting the denominator of CL
z=coeff(y) //extracting the coefficients of the
    denominator polynomial
//Wn^2=z(1,1) ,comparing the coefficients
Wn=sqrt(z(1,1))                // Wn=natural frequency
//2*zeta*Wn=z(1,2)
zeta=z(1,2)/(2*Wn)                // zeta=damping
    factor
Mr=1/(2*zeta*sqrt(1-zeta^2))
Wr=Wn*sqrt(1-zeta^2)
Wc=Wn*sqrt((1-2*zeta^2)+sqrt(4*zeta^4-4*zeta^2+2))
BW=Wc //BANDWIDTH
```

Scilab code Exa 12.03 denominator polynomial

```
//denominator polynomial//
s=poly(0,'s'); //Defines s as polynomial variable
F=syslin('c',[81/(s^2+7*s)]); //Creates transfer
    function in forward path
B=syslin('c',(1+0*s)/(1+0*s)); //Creates transfer
    function in backward path
CL=F/.B //Calculates closed-loop transfer function
```

```
// compare CL with Wn^2/(s^2+2*zeta*Wn+Wn^2)
y=denom(CL) //extracting the denominator of CL
z=coeff(y)  //extracting the coefficients of the denominator polynomial
//Wn^2=z(1,1) ,comparing the coefficients
Wn=sqrt(z(1,1))                 // Wn=natural frequency
//2*zeta*Wn=z(1,2)
zeta=z(1,2)/(2*Wn)                  // zeta=damping factor
Mr=1/(2*zeta*sqrt(1-zeta^2))
Wr=Wn*sqrt(1-zeta^2)
Wc=Wn*sqrt((1-2*zeta^2)+sqrt(4*zeta^4-4*zeta^2+2))
BW=Wc //BANDWIDTH
```

Scilab code Exa 12.04 denominator polynomial

```
//denominator polynomial//
//Defines s as polynomial variable
s=poly(0,'s');
//Creates transfer function in forward path
F=syslin('c',[81/((s+18)*s)]);
//Creates transfer function in backward path
B=syslin('c',(1+0*s)/(1+0*s));
//Calculates closed-loop transfer function
CL=F/.B
// compare CL with Wn^2/(s^2+2*zeta*Wn+Wn^2)
y=denom(CL)//extracting the denominator of CL
//Extracting the coefficients of the denominator polynomial
z=coeff(y)
//Wn^2=z(1,1) ,comparing the coefficients
Wn=sqrt(z(1,1))
//2*zeta*Wn=z(1,2)
zeta=z(1,2)/(2*Wn)
// zeta=damping factor
```

```
//NOTE= here sqrt(1-2zeta^2) becomes complex so the other solution is Wr=0 & Mr=1
Mr=1
Wr=0
Wc=Wn*((1-2*zeta^2)+sqrt(4*zeta^4-4*zeta^2+2))
BW=Wc //BANDWIDTH
```

Scilab code Exa 12.05 denominator polynomial

```
//denominator polynomial//
ieee(2);
syms k a;
num=k ;
den=s*(a+s);
G=num/den;
disp(G,"G(s)=")
H=1;
CL=G/.H;
CL=simple(CL);
disp(CL,"C(s)/R(s)=") //Calculates closed-loop transfer function
// compare CL with Wn^2/(s^2+2*zeta*Wn+Wn^2)
[num,den]=numden(CL); //extracts num & den of symbolic function (CL)
cof_a_0 = coeffs(den,'s',0); // coeff of den of symbolic function (CL)
cof_a_1 = coeffs(den,'s',1);
//Wn^2= cof_a_0 ,comparing the coefficients
Wn=sqrt(cof_a_0)
// cof_a_1=2*zeta*Wn
zeta=cof_a_1/(2*Wn)
Mr=1/(2*zeta*sqrt(1-zeta^2)) //-----------1)
printf("Given ,Mr=1.25 \n");
//On solving eq (1) we get k=1.25a^2--------2
printf("k=1.25*a^2 \n")
```

```
Wr=Wn*sqrt(1-2*zeta^2)    //----------------3)
printf("Given, Wr=12.65 \n") ;
//on soving eq (3),we get 2k-a^2=320------------4)
printf("2k-a^2=320 \n")
//now eq 2 &4 can be simultaneously soved to take
   out values of k &a
// Let k=x & a^2=y
A=[1,-1.25;2,-1]; // coefficient matrix
b=[0;320];
m=A\b;
x=m(1,1);
k=x
y=m(2,1);
a=sqrt(y)
Wn=dbl(eval(Wn));
disp(Wn,"Wn=")
zeta=dbl(eval(zeta));
disp(zeta,"zeta=')
Ts=4/(zeta*Wn) ;
disp(Ts,"Settling Time (Ts)=")
Wc=Wn((1-(2*zeta^2))+sqrt(4*zeta^4-4*zeta^2+2))
disp(Wc,"BW=")
```

Chapter 13

Bode Plot

Scilab code Exa 13.01 polynomial

```
//polynomial//
s=poly(0,'s'); //Defines s as polynomial variable
F=syslin('c',[20/(s+2)]) //Creates transfer function
    in forward path
B=syslin('c',(1+0*s)/(1+0*s)) //Creates transfer
   function in backward path
OL=F*B //Calculates open-loop transfer function
fmin=0.1; //Min freq in Hz
fmax=100; //Max freq in Hz
scf(1);clf;
bode(OL,fmin,fmax); //Plots frequency response of
   open-loop system in Bode diagram
show_margins(OL) //display gain and phase margin and
    associated crossover frequencies
```

Scilab code Exa 13.02 polynomial

```
//polynomial//
```

```
s=poly(0,'s'); //Defines s as polynomial variable
F=syslin('c',[20/((2+s)*s)]); //Creates transfer
   function in forward path
B=syslin('c',(1+0*s)/(1+0*s)); //Creates transfer
   function in backward path
OL=F*B; //Calculates open-loop transfer function
fmin=0.01; //Min freq in Hz
fmax=20; //Max freq in Hz
scf(1);clf;
bode(OL,fmin,fmax); //Plots frequency response of
   open-loop system in Bode diagram
show_margins(OL) //display gain and phase margin and
    associated crossover frequencies
```

Scilab code Exa 13.03 polynomial

```
//polynomial//
s=poly(0,'s'); //Defines s as polynomial variable
F=syslin('c',[40/((2+s)*s*(s+5))]) //Creates
   transfer function in forward path
B=syslin('c',(1+0*s)/(1+0*s)) //Creates transfer
   function in backward path
OL=F*B; //Calculates open-loop transfer function
fmin=0.1; //Min freq in Hz
fmax=20; //Max freq in Hz
scf(1);clf;
bode(OL,fmin,fmax); //Plots frequency response of
   open-loop system in Bode diagram
[GainMargin,freqGM]=g_margin(OL) //Calculates gain
   margin [dB] and corresponding frequency [Hz]
[Phase,freqPM]=p_margin(OL) //Calculates phase [deg]
    and corresponding freq [Hz] of phase margin
PhaseMargin=180+Phase //Calculates actual phase
   margin [deg]
show_margins(OL) //display gain and phase margin and
```

associated crossover frequencies

Scilab code Exa 13.04 polynomial

```
//polynomial//
s=poly(0,'s'); //Defines s as polynomial variable
F=syslin('c',[40/((s+5)*(2+s)*s^2)]) //Creates
   transfer function in forward path
B=syslin('c',(1+0*s)/(1+0*s)); //Creates transfer
   function in backward path
OL=F*B; //Calculates open-loop transfer function
fmin=0.1; //Min freq in Hz
fmax=20; //Max freq in Hz
scf(1);clf;
bode(OL,fmin,fmax); //Plots frequency response of
   open-loop system in Bode diagram
show_margins(OL) //display gain and phase margin and
    associated crossover frequencies
```

Scilab code Exa 13.05 polynomial

```
//polynomial//
s=poly(0,'s'); //Defines s as polynomial variable
F=syslin('c',[(400*(s+2))/((s+5)*(10+s)*s^2)]) //
   Creates transfer function in forward path
B=syslin('c',(1+0*s)/(1+0*s)) //Creates transfer
   function in backward path
OL=F*B //Calculates open-loop transfer function
fmin=0.1; //Min freq in Hz
fmax=20; //Max freq in Hz
scf(1);clf;
bode(OL,fmin,fmax); //Plots frequency response of
   open-loop system in Bode diagram
```

```
show_margins(OL) //display gain and phase margin and
    associated crossover frequencies
```

Scilab code Exa 13.06 polynomial

```
//polynomial//
s=poly(0,'s'); //Defines s as polynomial variable
F=syslin('c',[(288*(s+4))/((s+2)*(144+4.8*s+s^2)*s)
  ]) //Creates transfer function in forward path
B=syslin('c',(1+0*s)/(1+0*s)) //Creates transfer
  function in backward path
OL=F*B //Calculates open-loop transfer function
fmin=0.1; //Min freq in Hz
fmax=100; //Max freq in Hz
scf(1);clf;
bode(OL,fmin,fmax); //Plots frequency response of
  open-loop system in Bode diagram
show_margins(OL) //display gain and phase margin and
    associated crossover frequencies
```

Chapter 15

Nyquist Plot

Scilab code Exa 15.01 system

```
//system//
s=%s;
sys=syslin('c',1/(s+2))
nyquist(sys)
show_margins(sys,'nyquist')
printf("Since P=0(no of poles in RHP)=Poles of G(s)H
   (s) \n here the number of zeros of 1+G(s)H(s) in
   the RHP is zero \n hence the system is stable")
```

Scilab code Exa 15.02 system

```
//system//
s=%s;
sys=syslin('c',1/(s*(s+2)))
nyquist(sys)
show_margins(sys,'nyquist')
printf("Since P=0(no of poles in RHP)=Poles of G(s)H
   (s) \n here the number of zeros of 1+G(s)H(s) in
   the RHP is zero \n hence the system is stable")
```

Scilab code Exa 15.03 system

```
//system//
s=%s;
sys=syslin('c',1/(s^2*(s+2)))
nyquist(sys)
show_margins(sys,'nyquist')
printf("Since P=0(no of poles in RHP)=Poles of G(s)H
   (s) \n here the number of zeros of 1+G(s)H(s) in
   the RHP is not equal to zero \n hence the system
   is unstable")
```

Scilab code Exa 15.04 system

```
//system//
s=%s;
sys=syslin('c',1/(s^3*(s+2)))
nyquist(sys)
show_margins(sys,'nyquist')
printf("Since P=0(no of poles in RHP)=Poles of G(s)H
   (s) \n here the number of zeros of 1+G(s)H(s) in
   the RHP is N>0 \n hence the system is unstable")
```

Scilab code Exa 15.05 system

```
//system//
s=%s;
sys=syslin('c',1/(s^2*(s+2)))
nyquist(sys)
```

```
show_margins(sys,'nyquist')
printf("Since P=0(no of poles in RHP)=Poles of G(s)H
    (s) \n here the number of zeros of 1+G(s)H(s) in
    the RHP is N>0 \n hence the system is unstable")
```

Scilab code Exa 15.06 system

```
//system//
s=%s;
P1=1;
P2=2;
sys=syslin('c',1/((s+1)*(s+2)))
nyquist(sys)
show_margins(sys,'nyquist')
printf("Since P=0(no of poles in RHP)=Poles of G(s)H
    (s) \n\n Here the number of zeros of 1+G(s)H(s)
    in the RHP is zero \n\n Hence the system is
    stable")
```

Scilab code Exa 15.07 system

```
//system//
s=%s;
sys=syslin('c',12/(s*(s+1)*(s+2)))
nyquist(sys)
show_margins(sys,'nyquist')
gm=g_margin(sys)
if (gm<=0)
  printf("system is unstable")
else
printf("system is stable");end;
```

Scilab code Exa 15.08 system

```
//system//
s=%s;
sys=syslin('c',(30)/((s^2+2*s+2)*(s+3)))
nyquist(sys)
gm=g_margin(sys)
show_margins(sys,'nyquist')
printf("Since P=0(no of poles in RHP)=Poles of G(s)H
    (s) \n Here the number of zeros of 1+G(s)H(s) in
    the RHP is zero \n Hence the system is stable")
if (gm<=0)
  printf("system is unstable")
else
  printf("system is stable")
  end
```

Chapter 17

State Variable Approach

Scilab code Exa 17.03 funtion

```
//function//
s=%s;
//Creating cont-time transfer function
TFcont=syslin('c',3/(s^4+(2*s^3)+(3*s)+2))
SScont=tf2ss(TFcont)
//CCF form
[Ac,Bc,U,ind]=canon(SScont(2),SScon(3))
```

Scilab code Exa 17.04 funtion

```
//function//
s=%s;
TFcont=syslin('c',[(7 + 2*s + 3*(s^2))/(5 + 12*s +
   5*(s^2) + s^3 )])
SScont=tf2ss(TFcont)
[Ac,Bc,U,ind]=canon(SScont(2),SScont(3))
```

Scilab code Exa 17.06 funtion

```
//function//
s=%s;
//Creating cont-time transfer function
TFcont=syslin('c',[(5*(s+1)*(s+2))/((s+4)*(s+5))])
SScont=tf2ss(TFcont)
 //CCF form
[Ac,Bc,U,ind]=canon(SScont(2),SScont(3))
```

Scilab code Exa 17.07 funtion

```
//function//
s=%s;
//Creating cont-time transfer function
TFcont=syslin('c',(s+1)/((s+2)*(s+5)*(s+3)))
SScont=tf2ss(TFcont)
//CCF form
[Ac,Bc,U,ind]=canon(SScont(2),SScont(3))
```

Scilab code Exa 17.08 funtion

```
//function//
s=%s;
//Creating cont-time transfer function
TFcont=syslin('c',(6)/((s+2)^2*(s+1)))
SScont=tf2ss(TFcont)
//CCF form
[Ac,Bc,U,ind]=canon(SScont(2),SScont(3))
```

Scilab code Exa 17.09 funtion

```
//function//
A=[0 1;-6 -5]
[Row Col]=size(A)//Size of a matrix
l=poly(0,'l');
m=l*eye(Row,Col)-A   //lI-A
n=det(m) //To Find The Determinant of li-A
roots(n) //To Find The Value Of l
```

Scilab code Exa 17.10 funtion

```
//function//
A=[-2 1;0 -3]
B=[0;1]
C=[1 1]
s=poly(0,'s');
[Row Col]=size(A)     //Size of a matrix
m=s*eye(Row,Col)-A   //sI-A
n=det(m)     //To Find The Determinant of si-A
p=inv(m)       // To Find The Inverse Of sI-A
y=C*p*B  ;      //To Find C*(sI-A)^-1*B
disp(y,"Transfer Function=")
```

Scilab code Exa 17.11 funtion

```
//function//
A=[0 1;-6 -5]
x=[1;0];
disp(x,"x(t)=')
s=poly(0,'s');
[Row Col]=size(A)      //Size of a matrix
m=s*eye(Row,Col)-A     //sI-A
```

```
n=det(m)                    //To Find The Determinant of si
    -A
p=inv(m) ;                  // To Find The Inverse Of sI-A
syms t s;
disp(p,"phi(s)=")     //Resolvent Matrix
for i=1:Row
for j=1:Col
//Taking Inverse Laplace of each element of Matrix
    phi(s)
q(i,j)=ilaplace(p(i,j),s,t);
end;
end;
disp(q,"phi(t)=")//State Transition Matrix
r=inv(q);
r=simple(r); //To Find phi(-t)
disp(r,"phi(-t)=")
y=q*x; //x(t)=phi(t)*x(0)
disp(y,"Solution To The given eq.=")
```

Scilab code Exa 17.12 funtion

```
//function//
A=[0 1;-6 -5]
B=[0;1]
x=[1;0]
disp(x,"x(t)=')
s=poly(0,'s');
[Row Col]=size(A)     //Size of a matrix A
m=s*eye(Row,Col)-A //sI-A
n=det(m)              //To Find The Determinant of si-A
p=inv(m) ;            // To Find The Inverse Of sI-A
syms t s m;
disp(p,"phi(s)=") //Resolvent Matrix
for i=1:Row
for j=1:Col
```

```
//Inverse Laplace of each element of Matrix(phi(s))
q(i,j)=ilaplace(p(i,j),s,t);
end;
end;
disp(q,"phi(t)=") //State Transition Matrix
t=(t-m);
q=eval(q)          //At t=t-m ,evaluating q i.e phi(t-m)
//Integrate q w.r.t m(Indefinite Integration)
r=integ(q*B,m)
m=0                //Upper limit is t
g=eval(r)          //Putting the value of upper limit in q
m=t                //Lower Limit is 0
h=eval(r)          //Putting the value of lower limit in q
y=(h-g);
disp(y,"y=")
printf("x(t)= phi(t)*x(0) + integ(phi(t-m)*B) w.r.t m from 0 t0 t \n")
//x(t)=phi(t)*x(0)+integ(phi(t-m)*B) w.r.t m from 0 t0 t
y1=(q*x)+y;
disp(y1,"x(t)=")
```

Scilab code Exa 17.13 funtion

```
//function//
A=[3 0;2 4]
B=[0;1]
Cc=cont_mat(A,B);
disp(Cc,"Controlability Matrix=")
//To Check Whether the matrix(Cc) is singular i.e determint of Cc=0
if determ(Cc)==0;
```

```
  printf("Since the matrix is Singular, the system
     is not controllable \n");
else;
 printf("The system is  controllable \n")
 end;
```

Scilab code Exa 17.14 funtion

```
//function//
A=[-2 1;0 -3]
B=[4;1]
C=[1 0]
[O]=obsv_mat(A,C);
disp(O,"Observability Matrix=")
//To Check Whether the matrix(Cc)is singular i.e
   determint of Cc=0
if determ(O)==0;
printf("Since the matrix is Singular, the system is
   not Observable \n");
else;
printf("The system is  Observable \n")
 end;
```

Scilab code Exa 17.16 funtion

```
//function//
s=%s;
//Creating cont-time transfer function
TFcont=syslin('c',((5*s^2)+(2*s)+6)/(s^3+(7*s^2)
   +(11*s)+8))
SScont=tf2ss(TFcont)
//CCF form
[Ac,Bc,U,ind]=canon(SScont(2),SScont(3))
```

Scilab code Exa 17.17 funtion

```
//function//
s=%s;
//Creating cont-time transfer function
TFcont=syslin('c',(8)/(s*(s+2)*(s+3)))
SScont=tf2ss(TFcont)
//CCF form
[Ac,Bc,U,ind]=canon(SScont(2),SScont(3))
```

Scilab code Exa 17.18 funtion

```
//function//
s=%s;
//Creating cont-time transfer function
TFcont=syslin('c',(8)/(s*(s+2)*(s+3)))
SScont=tf2ss(TFcont)
//CCF form
[Ac,Bc,U,ind]=canon(SScont(2),SScont(3))
```

Scilab code Exa 17.19 funtion

```
//function//
A=[0 1;-3 -4]
B=[0;1]
C=[1 0]
x=[0;0]
disp(x,"x(t)=')
s=poly(0,'s');
```

```
[Row Col]=size(A)    //Size of a matrix A
m=s*eye(Row,Col)-A   //sI-A
n=det(m)             //To Find The Determinant of si-A
p=inv(m) ;           // To Find The Inverse Of sI-A
syms t s m;
disp(p,"phi(s)=")  //Resolvent Matrix
for i=1:Row
for j=1:Col
//Taking Inverse Laplace of each element of Matrix(phi(s))
q(i,j)=ilaplace(p(i,j),s,t);
end;
end;
disp(q,"phi(t)=")        //State Transition Matrix
t=(t-m)
q=eval(q)    //At t=t-m ,evaluating q i.e phi(t-m)
r=integ(q*B,m)//Integrate q w.r.t m  (Indefinite Integration)
m=0              //Upper limit is t
g=eval(r)     //Putting the value of upper limit in q
m=t           //Lower Limit is 0
h=eval(r)  //Putting the value of lower limit in q
y=(h-g);
printf("x(t)= phi(t)*x(0) + integ(phi(t-m)*B) w.r.t m from 0 t0 t \n")
//x(t)=phi(t)*x(0) + integ(phi(t-m)*B) w.r.t m from 0 t0 t
y1=(q*x)+y;
disp(y1,"x(t)=")
y2=C*y1;
disp(y2,"Output Response=")
```

Scilab code Exa 17.20 funtion

```
//function//
A=[0 1;-2 0]
B=[0;3]
Cc=cont_mat(A,B);
disp(Cc,"Controlability Matrix=")
//To Check Whether the matrix(Cc) is singular i.e
   determint of Cc=0
if determ(Cc)==0;
printf("Since the matrix is Singular, the system is
   not controllable \n");
else;
 printf("The system is  controllable \n")
 end;
```

Scilab code Exa 17.21 funtion

```
//function//
A=[-3 0;0 -2]
B=[4;1]
C=[2 0]
[O]=obsv_mat(A,C);
disp(O,"Observability Matrix=")
//To Check Whether the matrix(Cc) is singular i.e
   determint of Cc=0
if determ(O)==0;
printf("Since the matrix is Singular, the system is
   not Observable \n");
else;
printf("The system is  Observable \n")
 end;
```

Scilab code Exa 17.22 funtion

```
//function//
ieee(2)
A=[-3 0 0;0 -1 1 ; 0 0 -1]
B=[0;1;0]
s=poly(0,'s');
[Row Col]=size(A)     //Size of a matrix
m=s*eye(Row,Col)-A    //sI-A
n=det(m)              //To Find The Determinant of si-
    A
p=inv(m);             // To Find The Inverse Of sI-A
syms t s;
disp(p,"phi(s)=") //Resolvent Matrix
```

Scilab code Exa 17.23 funtion

```
//function//
A=[-2 0;1 -1]
B=[0;1]
x=[0;0]
disp(x,"x(t)=')
s=poly(0,'s');
[Row Col]=size(A)     //Size of a matrix A
m=s*eye(Row,Col)-A    //sI-A
n=det(m)              //To Find The Determinant of si-
    A
p=inv(m) ;            // To Find The Inverse Of sI-A
syms t s m;
disp(p,"phi(s)=")   //Resolvent Matrix
t=(t-m)
q=eval(q)           //At t=t-m ,evaluating q i.e phi(t
    -m)
//Integrate q w.r.t m (Indefinite Integration)
r=integ(q*B,m)
m=0           //Upper limit is t
g=eval(r)   //Putting the value of upper limit in q
```

```
m=t         //Lower Limit is 0
h=eval(r)  //Putting the value of lower limit in q
y=(h-g);
disp(y,"y=")
printf("x(t)= phi(t)*x(0) + integ(phi(t-m)*B) w.r.t
   m from 0 t0 t \n")
//x(t)=phi(t)*x(0)+integ(phi(t-m)*B)w.r.t m from 0
   t0 t
y1=(q*x)+y;
disp(y1,"x(t)=")
// CONTROLABILITY OF THE SYSTEM
Cc=cont_mat(A,B);
disp(Cc,"Controlability Matrix=")
//To Check Whether the matrix(Cc)is singular i.e
   determint of Cc=0
if determ(Cc)==0;
printf("Since the matrix is Singular, the system is
   not controllable \n");
else;
printf("The system is  controllable \n")
end;
```

Chapter 18

Digital Control Systems

Scilab code Exa 18.01.01 symsum

```
//symsum//
syms n z;
x=(-0.5)^n
y=(4*((0.2)^n))
f1=symsum(x*(z^(-n)),n,0,%inf)
f2=symsum(y*(z^(-n)),n,0,%inf)
y=(f1+f2);
disp(y,"ans=")
```

Scilab code Exa 18.08.01 symsum

```
//symsum//
exec ztransfer.sce;
sequence=[0 2 0 0 -3 0 0 8]
y=ztransfer(sequence);
disp(y,"ans=")
```