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

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 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

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

Scilab code Exa 2.01.02 laplace

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

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

Scilab code Exa 2.05.02 laplace

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

Scilab code Exa 2.06 laplace

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

Scilab code Exa 2.07 laplace

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

Scilab code Exa 2.11 value theorem

```
1 //value theorem//
2 p=poly([0.38],'s','coeff');
3 q=poly([0 0.543 2.48 1],'s','coeff');
4 F=p/q;
5
6 syms s;
7 x=s*F;
8 y=limit(x,s,0);  // final value theorem
9 y=dbl(y);
10 disp(y,"f(inf)=")
11 z=limit(x,s,%inf) // // initial value theorem
12 z=dbl(z);
13 disp(z,"f(0)=")
```

Scilab code Exa 2.12 ilaplace

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

Scilab code Exa 2.13 ilaplace

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

Chapter 3

Transfer Function

Scilab code Exa 3.02 laplace

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

Scilab code Exa 3.03 laplace

```
1 //laplace//
2 syms t s;
3 disp(%e^(-3*t),"g(t)=");
4 y1=laplace('%e^(-3*t)',t,s);
5 disp(y1,"G(s)=")
6 disp(%e^(-4*t),"r(t)=");
7 y2=laplace('%e^(-4*t)',t,s);
8 disp(y2,"R(s)=")
9 disp(y1*y2,"G(s)R(s)=")
```

Scilab code Exa 3.04 laplace

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

Scilab code Exa 3.15 laplace

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

Scilab code Exa 3.16 laplace

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

Scilab code Exa 3.17 laplace

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

Scilab code Exa 3.18 laplace

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

Scilab code Exa 3.19 laplace

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

```
6  disp(G,"G(s)=")
7  disp("Input is Step Function ")
8  syms t s;
9  R=laplace(1,t,s);
10 disp(R,"R(s)=")
11 printf("C(s)=R(s)G(s) \n")
12 C=R*G;
13 disp(C,"C(s)=")
14 c=ilaplace(C,s,t);
15 disp(c,"c(t)=")
```

Scilab code Exa 3.23 laplace

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

Chapter 4

Control system Components

Scilab code Exa 4.01 laplace

```
1  //laplace//
2  printf("Given a)Excitation voltage(Ein)=2V \n b)
       Setting Ratio(a)= 0.4 \n")
3  Ein=2;
4  disp(Ein,"Ein=")
5  a=0.4;
6  disp(a,"a=")
7  Rt=10^3;
8  disp(Rt,"Rt=")
9  Rl=5*10^3;
10 disp(Rl,"Rl=")
11 printf("Eo = (a*Ein)/(1+(a*(1-a)*Rt)/Rl) \n")
12 Eo = (a*Ein)/(1+(a*(1-a)*Rt)/Rl);
13 disp(Eo,"output voltage(E0)=")
14 printf("e=((a^2)*(1-a))/((a*(1-a))+(Rl/Rt)) \n")
15 e=((a^2)*(1-a))/((a*(1-a))+(Rl/Rt));
16 disp(e,"loading error=")
17 printf("E=Ein*e \n")
18 E=Ein*e; //Voltage error=Excitation voltage(Ein)*
       Loading error(e)
19 disp(E,"Voltage error=")
```

Scilab code Exa 4.02 `laplace`

```
1  //laplace//
2  printf("n=5 ,Helical turn \n")
3  n=5; //Helical turn
4  disp(n,"n=")
5  printf("N=9000 ,Winding Turn \n")
6  N=9000;//Winding Turn
7  disp(N,"N=")
8  printf("R=10000 ,Potentiometer Resistance \n")
9  R=10000;//Potentiometer Resistance
10 disp(R,"R=")
11 printf("Ein=90 ,Input voltage \n")
12 Ein=90;//Input voltage
13 disp(Ein,"Ein=")
14 printf("r=5050 ,Resistance at mid point \n")
15 r=5050;//Resistance at mid point
16 disp(r,"r=")
17 printf("D=r-5000, Deviation from nominal at mid-
     point \n")
18 D=r-5000; //Deviation from nominal at mid-point
19 disp(D,"D=")
20 printf("L=D/R*100 ,Linearity \n")
21 L=D/R*100;//Linearity
22 disp(L,"L=")
23 printf("R=Ein/N ,Resolution \n")
24 R=Ein/N; //Resolution
25 disp(R,"R=")
26 printf("Kp=Ein/(2*pi*n) ,Potentiometer Constant \n")
27 Kp=Ein/(2*%pi*n); //Potentiometer Constant
28 disp(Kp,"Kp=")
```

Scilab code Exa 4.03 laplace

```
1 //laplace//
2 printf("since S2 is the referance stator winding ,
    Es2=KVcos0 \n where Es2 & Er are rms voltages \n
    ')
3 k=1
4 Theta=60;
5 disp(Theta,"Theta=")
6 V=28;
7 disp(V,"V(applied)=")
8 printf("Es2=V*cos(Theta) \n")
9 Es2=k*V*cos(Theta*(%pi/180));
10 disp(Es2,"Es2=")
11 printf("Es1=k*V*cos(Theta-120)\n")
12 Es1=k*V*cos((Theta-120)*(%pi/180)); // Given Theta
    =60 in degrees
13 disp(Es1,"Es1=')
14 printf("Es3=k*V*cos(Theta+120) \n")
15 Es3=k*V*cos((Theta+120)*(%pi/180));
16 disp(Es3,"Es3=')
17 printf("Es31=sqrt(3)*k*Er*sin(Theta)")
18 Es31=sqrt(3)*k*V*sin(Theta*(%pi/180));
19 disp(Es31,"Es31=')
20 printf("Es12=sqrt(3)*k*Er*sin((Theta-120)")
21 Es12=sqrt(3)*k*V*sin((Theta-120)*(%pi/180));
22 disp(Es12,"Es12=')
23 printf("Es23=sqrt(3)*k*Er*sin((Theta+120)")
24 Es23=sqrt(3)*k*V*sin((Theta+120)*(%pi/180));
25 disp(Es23,"Es23=')
```

Scilab code Exa 4.04 laplace

```
1 //laplace//
2 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`

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

Scilab code Exa 6.02 `syslin`

```
1  //syslin//
2  exec parallel.sce;
3  s=%s;
```

```
4  sys1=syslin('c',1/s)
5  sys2=syslin('c',2/(s+1))
6  sys3=syslin('c',3/(s+3))
7  a=parallel(sys1,sys2);
8  y=parallel(a,sys3);
9  y=simp(y);
10 disp(y,"C(s)/R(s)=")
```

Scilab code Exa 6.03 `syslin`

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

Scilab code Exa 6.04 `syslin`

```
1  //syslin//
2  exec parallel.sce;
3  syms G1 G2 G3 H;
4  a=series(G1,G2);
5  b=parallel(a,G3);
6  y=b/.H  //negative feedback operation
7  y=simple(y)
8  disp(y,"C(s)/R(s)=")
```

Scilab code Exa 6.05 syslin

```
1 //syslin//
2 exec series.sce;
3 syms G1 G2 H1 H2 s;
4 a=G1/.H1; // negative feedback operation
5 b=a/.H2;// negative feedback operation
6 y=series(b,G2);
7 y=simple(y);
8 disp(y,"C(s)/R(s)=")
```

Scilab code Exa 6.06 syslin

```
1 //syslin//
2 exec parallel.sce;
3 exec series.sce;
4 syms G1 G2 G3 G4 G5 G6 H1 H2;
5 a=parallel(G3,G5);
6 b=parallel(a,-G4);
7 c=series(G1,G2);
8 d=c/.H1;
9 e=series(b,d);
10 f=e/.H2;
11 y=series(f,G6);
12 y=simple(y);
13 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;
```

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

Scilab code Exa 6.09 syslin

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

Scilab code Exa 6.10 syslin

```
1 //syslin//
2 exec parallel.sce;
3 exec series.sce;
4 syms G1 G2 G3 G4 G5 H1 H2;
5 a=G2/.H1;  //negative feedback operation
6 b=series(G1,a);
7 c=series(b,G3);
8 d=parallel(c,G4);
9 e=series(d,G5);
10 y=e/.H2;   //negative feedback operation
11 y=simple(y);
```

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

Scilab code Exa 6.11 syslin

```
1  //syslin//
2  exec parallel.sce;
3  exec series.sce;
4  syms G1 G2 G3 G4 G5 G6 G7 H1 H2 H3;
5  a=parallel(G1,G2);
6  b=parallel(a,G3);
7  //shift the take off point to the right of the block
       G4
8  c=G4/.H1;  //negative feedback operation
9  d=G5/G4;    //negative feedback operation
10 e=parallel(d,1);
11 f=G6/.H2;   //negative feedback operation
12 g=series(b,c);
13 h=series(g,e);
14 i=series(h,f);
15 j=series(i,G7);
16 y=j/.H3;
17 y=simple(y);
18 disp(y,"C(s)/R(s)=")
```

Scilab code Exa 6.12 syslin

```
1 //syslin//
2 exec series.sce;
3 exec parallel.sce;
4 syms G1 G2 G3 G4 H1 H2 H3;
5 //shift the take-off point after the block G1
6 a=G3/G1;
7 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;
```

```
3 exec parallel.sce;
4 syms G1 G2 G3 G4 H1 H2 ;
5 //shift the take-off point to the right of the block
      G3
6 a=H1/G3;
7 b=series(G2,G3);
8 c=parallel(H2,a);
9 d=b/.c;
10 e=series(d,G1);
11 f=e/.a;
12 y=series(f,G4);
13 y=simple(y);
14 disp(y,"C(s)/R(s)=")
```

Scilab code Exa 6.15 syslin

```
1 //syslin//
2 exec series.sce;
3 exec parallel.sce;
4 syms G1 G2 G3 G4 H1 H2 H3;
5 //shift the take-off point to the right of the block
      H1
6 //shift the other take-off point to the right of the
      block H1 &H2
7 a=series(H1,H2);
8 b=1/a;
9 c=1/H1;
10 d=G3/.a;
11 //move the summer to the left of the block G2
12 e=G4/G2;
13 f=series(d,G2);
14 //exchange the summer
15 g=f/.H1;
16 h=parallel(G1,e);
17 i=series(h,g);
```

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

Chapter 8

Time Domain Analysis of Control Systems

Scilab code Exa 8.02 `Velocity`

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

Scilab code Exa 8.03.01 coefficient

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

Scilab code Exa 8.03.02 coefficient

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

Scilab code Exa 8.03.03 coefficient

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

Scilab code Exa 8.03.04 coefficient

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

Scilab code Exa 8.04 coefficient

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

Scilab code Exa 8.05 coefficient

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

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

Scilab code Exa 8.06 coefficient

```
1 //coefficient//
2 s=poly(0,'s'); //Defines s as polynomial variable
3 F=syslin('c',[25/((s+1)*s)]); //Creates transfer
   function in forward path
4 B=syslin('c',(1+0*s)/(1+0*s)); //Creates transfer
   function in backward path
5 CL=F/.B //Calculates closed-loop transfer function
6 // compare CL with Wn^2/(s^2+2*zeta*Wn+Wn^2)
7 y=denom(CL) //extracting the denominator of CL
8 z=coeff(y)  //extracting the coefficients of the
   denominator polynomial
9 //Wn^2=z(1,1) ,comparing the coefficients
10 Wn=sqrt(z(1,1))              // Wn=natural frequency
11 //2*zeta*Wn=z(1,2)
12 zeta=z(1,2)/(2*Wn)              // zeta=damping
   factor
13 Wd=Wn*sqrt(1-zeta^2)
14 Tp=%pi/Wd
15 Mp=100*exp((-%pi*zeta)/sqrt(1-zeta^2))
```

Scilab code Exa 8.07 coefficient

```
1 //coefficient//
2 s=poly(0,'s'); //Defines s as polynomial variable
3 F=syslin('c',[20/(s^2+5*s+5)]); //Creates transfer
    function in forward path
4 B=syslin('c',(1+0*s)/(1+0*s)); //Creates transfer
    function in backward path
5 CL=F/.B //Calculates closed-loop transfer function
6 // compare CL with Wn^2/(s^2+2*zeta*Wn+Wn^2)
7 y=denom(CL) //extracting the denominator of CL
8 z=coeff(y)  //extracting the coefficients of the
    denominator polynomial
9 //Wn^2=z(1,1) ,comparing the coefficients
10 Wn=sqrt(z(1,1))              // Wn=natural frequency
11 //2*zeta*Wn=z(1,2)
12 zeta=z(1,2)/(2*Wn)              // zeta=damping
    factor
13 Wd=Wn*sqrt(1-zeta^2)
14 Tp=%pi/Wd      //Tp=peak time
15 Mp=100*exp((-%pi*zeta)/sqrt(1-zeta^2))  //peak
    overshoot
16 Td=(1+0.7*zeta)/Wn       //Td=delay time
17 a=atan(sqrt(1-zeta^2)/zeta)
18 Tr=(%pi-a)/Wd            //Tr=rise time
19 Ts=4/(zeta*Wn)           //Ts=settling time
```

Scilab code Exa 8.08 coefficient

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

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

Scilab code Exa 8.09 coefficient

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

Scilab code Exa 8.10 coefficient

```
1  //coefficient//
2  s = poly ( 0,'s' )
3  sys = syslin ('c',(20)/(s^2+7*s+10))
4  a=sys/.(2*(s+1))  //simplifying the internal
       feedback loop
5  b=20*2*a;
6  disp(b,"G(s)")
7  c=1;
8  disp(c,"H(s)")
9  OL=b*c;
10 disp(OL,"G(s)H(s)")
11 ,"G(s)*H(s)")
12 syms t s;
13 Kp=limit(s*OL/s,s,0) //Kp= position error
       coefficient
14 Kv=limit(s*OL,s,0) //Kv= velocity error coefficient
15 Ka=limit(s^2*OL,s,0) //Ka= accelaration error
       coefficient
16 //given input r(t)=6
17 R=laplace('6',t,s)
18 //steady state error =Lt s->0 sR(S)/1+G(s)H(S)
19 e1=limit(s*R/(1+OL),s,0); //e=error for given input
20 disp(e1,"error")
21 //given input r(t)=8t
22 M=laplace('8*t',t,s)
23 //steady state error =Lt s->0 sR(S)/1+G(s)H(S)
24 e2=limit(s*M/(1+OL),s,0); //e=error for given input
25 disp(e2,"error")
26 //given input r(t)=10+4t+3t^2/2
27 N=laplace('10+4*t+(3*t^2)/2',t,s)
28 //steady state error =Lt s->0 sR(S)/1+G(s)H(S)
29 e3=limit(s*N/(1+OL),s,0); //e=error for given input
```

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

Scilab code Exa 8.11 coefficient

```
1  //coefficient//
2  s = poly ( 0,'s' )
3  sys1 = syslin ('c',(s)/(s+6));
4  sys2 = syslin ('c',(s+2)/(s+3));
5  sys3 = syslin ('c',(5)/((s+3)*s^3));
6  a=sys2+sys3;
7  disp(a,"H(s)")
8  b=sys1;
9  disp(b,"G(S)")
10 y=a*b;
11 disp(y,"G(S)H(S)")
12 syms s
13 Kp=limit(s*y/s,s,0) //Kp= position error coefficient
14 Kv=limit(s*y,s,0) //Kv= velocity error coefficient
15 Ka=limit(s^2*y,s,0) //Ka= accelaration error
      coefficient
```

Scilab code Exa 8.12 coefficient

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

```
11 disp(e,"Ess=")
12 printf('Given Ess = 0.8 \n")
13 e=0.8;
14 k=72/e;
15 disp(k,"k=")
```

Scilab code Exa 8.13 coefficient

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

Scilab code Exa 8.14 coefficient

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

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

Scilab code Exa 8.15 coefficient

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

Scilab code Exa 8.16 coefficient

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

```
4 B=syslin('c',(1+0*s)/(1+0*s)); //Creates transfer
    function in backward path
5 CL=F/.B //Calculates closed-loop transfer function
6 // compare CL with Wn^2/(s^2+2*zeta*Wn+Wn^2)
7 y=denom(CL) //extracting the denominator of CL
8 z=coeff(y)  //extracting the coefficients of the
    denominator polynomial
9 //Wn^2=z(1,1) ,comparing the coefficients
10 Wn=sqrt(z(1,1))              // Wn=natural frequency
11 //2*zeta*Wn=z(1,2)
12 zeta=z(1,2)/(2*Wn)              // zeta=damping
    factor
13 Wd=Wn*sqrt(1-zeta^2)
14 Tp=%pi/Wd      //Tp=peak time
15 Mp=100*exp((-%pi*zeta)/sqrt(1-zeta^2))  //peak
    overshoot
```

Scilab code Exa 8.17 coefficient

```
1 //coefficient//
2 s=%s;
3 t = %t;
4 F=syslin('c',(25)/(s^2+7*s)); //Creates transfer
    function in forward path
5 B=syslin('c',(1+0*s)/(1+0*s)); //Creates transfer
    function in backward path
6 k=20/25;    //k=gain factor
7 CL=k*(F/.B) //Calculates closed-loop transfer
    function
8 // compare CL with Wn^2/(s^2+2*zeta*Wn+Wn^2)
9 y=denom(CL) //extracting the denominator of CL
10 z=coeff(y)  //extracting the coefficients of the
    denominator polynomial
11 //Wn^2=z(1,1) ,comparing the coefficients
12 Wn=sqrt(z(1,1))              // Wn=natural frequency
```

```
13 //2*zeta*Wn=z(1,2)
14 zeta=z(1,2)/(2*Wn)                    // zeta=damping
      factor
15 Wd=Wn*sqrt(1-zeta^2)
16 Tp=%pi/Wd      //Tp=peak time
17 Mp=100*exp((-%pi*zeta)/sqrt(1-zeta^2))   //peak
      overshoot
18 Td=(1+0.7*zeta)/Wn       //Td=delay time
19 a=atan(sqrt(1-zeta^2)/zeta)
20 Tr=(%pi-a)/Wd            //Tr=rise time
21 Ts=4/(zeta*Wn)           //Ts=settling time
22 //y(t)=expression for output
23 y=(20/25)*(1-(exp(-1*zeta*Wn*t)/sqrt(1-zeta^2))*sin(
      Wd*t+atan(zeta/sqrt(1-zeta^2))));
24 disp(y,"Y(t)")
```

Scilab code Exa 8.19 coefficient

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

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

Scilab code Exa 8.20 coefficient

```
1  //coefficient//
2  ieee(2);
3  syms k T;
4  num=k;
5  den=s*(1+s*T);
6  G=num/den;
7  disp(G,"G(s)=")
8  H=1;
9  CL=G/.H;
10 CL=simple(CL);
11 disp(CL,"C(s)/R(s)=")  //Calculates closed-loop
    transfer function
12 // compare CL with Wn^2/(s^2+2*zeta*Wn+Wn^2)
13 [num,den]=numden(CL) //extracts num & den of
    symbolic function (CL)
14 den=den/T;
15 cof_a_0 = coeffs(den,'s',0) // coeff of den of
    symbolic function (CL)
16 cof_a_1 = coeffs(den,'s',1)
17 //Wn^2= cof_a_0 ,comparing the coefficients
18 Wn=sqrt(cof_a_0)
19 disp(Wn,"natural frequency Wn")          // Wn=
    natural frequency
```

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

Scilab code Exa 8.23.01 coefficient

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

Scilab code Exa 8.23.02 coefficient

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

Scilab code Exa 8.23.03 coefficient

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

Scilab code Exa 8.24 coefficient

```
1  //coefficient//
2  s= poly ( 0,'s' );
3  sys1 = syslin ('c',(s+3)/(s+5));
4  sys2= syslin ('c',(100)/(s+2));
5  sys3= syslin ('c',(0.15)/(s+3));
6  G=sys1*sys2*sys3*2*5
7  H=1;
8  y=G*H; //G(s)H(s)
9  disp(y,"G(s)H(s)")
10 F=1/(1+y)
11 syms t s;
12 Co=limit(s*F/s,s,0) //Ko=Lt s->0 (1/(1+G(s)H(S))
```

```
13 d=diff(s*F/s,s)
14 C1=limit(diff(s*F/s,s),s,0) //K1=Lt s->0 (dF(s)/ds)
15 C2=limit(diff(d,s),s,0)    //K2=Lt s->0 (d2F(s)/ds)
16 a=(1+(2*t)+(5*(t^2/2)));
17 b=diff(a ,t);
18 c=diff(b,t);
19 e=Co*a+C1*b+C2*c;
20 disp(e," steadt state error")
```

Scilab code Exa 8.32 coefficient

```
1 //coefficient//
2 s=%s;
3 sys=syslin('c',(9*(1+2*s))/(s^2+0.6*s+9));
4 disp(sys,"C(s)/R(s)=")
5 //given r(t)=u(t)
6 syms t s;
7 R=laplace('1',t,s);
8 disp(R,"R(s)=")
9 C=R*sys;
10 disp(C,"C(s)=")
11 c=ilaplace(C,s,t)
12 disp(c,"c(t)=")
13 G=9/(s^3+0.6*s^2);
14 disp(G,"G(s)=")
15 H=1;
16 y=1+G*H;
17 syms t s;
18 Kp=limit(s*G/s,s,0)
19 Kv=limit(s*G,s,0)
20 Ka=limit(s^2*G,s,0)
21 R=laplace('(1+t+(t^2/2))',t,s)
22 //steady state error =Lt s->0 sR(S)/1+G(s)H(S)
23 e=limit(s*R/(1+y),s,0); //e=error for ramp input
24 disp(e,"steady state error(Ess)")
```

Chapter 9

Feedback Characteristics of control Systems

Scilab code Exa 9.01 sensitivity of closed loop

```
1  //calculates//
2  s=%s;
3  G=syslin('c',20/(s*(s+4)))
4  H=0.35;
5  y=G*H;
6
7  S=1/(1+y);
8  disp(S,"1/(1+G(s)*H(s))")
9
10 //given w=1.2
11 w=1.2
12 s=%i*w
13 S=horner(S,s) //calculates value of S at s
14 a=abs(S)
15 disp(a,"sensitivity of open loop")
16
17 F=-y/(1+y)
18 disp(F,"(-G(s)*H(s))/(1+G(s)*H(s))")
19 S=horner(F,s)  //calculates value of F at s
```

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

Scilab code Exa 9.02 settling time Ts

```
1 //calculates//
2 s=%s;
3 sys1=syslin('c',9/(s*(s+1.8)));
4 syms Td ;
5 sys2=1+(s*Td);
6 sys3=sys1*sys2;
7 H=1;
8 CL=sys3/.H; //Calculates closed-loop transfer
    function
9 disp(CL,"C(s)/R(s)")
10 // compare CL with Wn^2/(s^2+2*zeta*Wn+Wn^2)
11 [num,den]=numden(CL) //extracts num & den of
    symbolic function CL
12 den=den/5;
13 cof_a_0 = coeffs(den,'s',0) // coeff of den of
    symbolic function CL
14 cof_a_1 = coeffs(den,'s',1)
15 //Wn^2= cof_a_0 ,comparing the coefficients
16 Wn=sqrt(cof_a_0)
17 disp(Wn,"natural frequency Wn")          // Wn=
    natural frequency
18 //cof_a_1=2*zeta*Wn
19 zeta=cof_a_1/(2*Wn)
20 zeta=1;disp(zeta,"for criticaly damped function zeta
    ")
21 Td=((2*Wn)-1.8)/9
22 Ts=4/(zeta*Wn);
23 Ts=dbl(Ts);
24 disp(Ts,"settling time Ts")
```

Scilab code Exa 9.03 sensitivity of closed loop

```
1 //calculates//
2 s=%s;
3 G=syslin('c',40/(s*(s+4)))
4 H=0.50;
5 y=G*H;
6 S=1/(1+y);
7 disp(S,"1/(1+G(s)*H(s))")
8 //given w=1.3
9 w=1.3
10 s=%i*w
11 S=horner(S,s)
12 a=abs(S)
13 disp(a,"sensitivity of open loop")
14 F=-y/(1+y)
15 disp(F,"(-G(s)*H(s))/(1+G(s)*H(s))")
16 S=horner(F,s)
17 b=abs(S)
18 disp(b,"sensitivity of closed loop")
```

Scilab code Exa 9.04 closed loop sensitivity for changes in H

```
1 //calculates//
2 s=%s;
3 syms s;
4 syms Wn zeta A H real;
5 T=6.28;
6 Wn=(8*%pi)/T;
7 zeta=0.3
8 n=Wn^2;
9 d=s^2+2*zeta*Wn*s+Wn^2;
```

```
10 G1=n/d;
11 disp(G1,"G1(s)")
12 G=A*G1;
13 disp(G,"G(s)")
14 S1=(diff(G,A))*(A/G);
15 a=simple(S1);
16 disp(a,"open loop sensitivity for changes in A")
17 M=G/.H;
18 p=simple(M)
19 S2=(diff(p,A))*(A/p);
20 b=simple(S2);
21 disp(b,"closed loop sensitivity for changes in A")
22 S3=(diff(p,H))*(H/p);
23 c=simple(S3);
24 disp(c,"closed loop sensitivity for changes in H")
```

Scilab code Exa 9.05 system sensitivity due to variation

```
1 //calculates//
2 s=%s;
3 sys1=syslin('c',(s+3)/s);
4 syms u rp k H RL;
5 num2=u*RL*s*(s+2);
6 den2=(rp+RL)*(s+3);
7 sys2=num2/den2;
8 num3=k;
9 den3=s+2;
10 sys3=num3/den3;
11 sys=sys1*sys2*sys3;
12 disp(sys,"G(s)=");
13 RL=10*10^3;
14 rp=4*10^3;
15 sys=eval(sys)
16 sys=float(sys)
17 disp(sys,"sys=");
```

```
18 disp(H,"H(s)");
19 M=sys/.H  //G(s)/1+G(s)H(S)
20 M=simple(M)
21 S=(diff(M,u))*(u/M);
22 S=simple(S);
23 disp(S,"system sensitivity due to variation of u=");
24 H=0.3;
25 u=12;
26 S=eval(S)  //----------eq 1
27 S=0.04;
28 k=((7/S)-7)/18 ; //from eq 1
29 disp(k,"K=")
```

Scilab code Exa 9.06 change in Eo

```
1 //calculates//
2 G=210;
3 H=0.12;
4 syms Eo Er
5 printf("for closed loop system")
6 sys=G/.H;  //Eo/Er=G/(1+GH)
7 disp(sys,"Eo/Er=")
8 Eo=240 //given
9 Er=Eo/8.0152;
10 disp(Er,"Er=")
11 printf("for open loop system")
12 disp(G,"Eo/Er=")
13 Er=Eo/G;
14 G=210;
15 disp(Er,"Er=")
16 //printf("since G is reduced by 12%, the new value
     of gain is 784.8V");
17 S=1/(1+G*H)
18 disp(S,"(%change in M)/(%change in G)=")
19 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 natural frequency Wn

```
//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)
```

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

Scilab code Exa 9.09 controller

```
1 //calculates//
2 s=%s;
3 printf("1)without controller")
4 G=syslin('c',120/(s*(s+12.63)));
5 disp(G,"G(s)=")
6 H=1;
7 CL=G/.H;
8 disp(CL,"C(s)/R(s)=")
9 y=denom(CL) //extracting the denominator of CL
10 z=coeff(y)  //extracting the coefficients of the
     denominator polynomial
11 //Wn^2=z(1,1) ,comparing the coefficients
12 Wn=sqrt(z(1,1));
13 disp(Wn,"Wn=")              // Wn=natural frequency
14 //2*zeta*Wn=z(1,2)
15 zeta=z(1,2)/(2*Wn);
16 disp(zeta,"zeta=")
17 printf("2) with controller')
18 G=syslin('c',(120*(30+s))/(s*(s+12.63)*30));
19 disp(G,"G(s)=")
20 CL=G/.H;
21 disp(CL,"C(s)/R(s)=")
22 den=denom(CL)
23 den=den/30 //extracting the denominator of CL
```

```
24 z=coeff(den)  //extracting the coefficients of the
     denominator polynomial
25 //Wn^2=z(1,1) ,comparing the coefficients
26 Wn=sqrt(z(1,1));
27 disp(Wn,"Wn=")              // Wn=natural frequency
28 //2*zeta*Wn=z(1,2)
29 zeta=z(1,2)/(2*Wn);
30 Mp=100*exp((-%pi*zeta)/sqrt(1-zeta^2));
31 disp(Mp,"Mp=")
32 Ts=4/(zeta*Wn);
33 disp(Ts,"Ts=")
```

Scilab code Exa 9.10 natural frequency

```
1 //calculates//
2 s=%s;
3 printf("1)without controller")
4 G=64*syslin('c',1/(s*(s+4)));
5 disp(G,"G(s)=")
6 H=1;
7 CL=G/.H;
8 disp(CL,"C(s)/R(s)=")
9 //Extracting the denominator of CL
10 y=denom(CL)
11 //Extracting the coefficients of the denominator
     polynomial
12 z=coeff(y)
13 //Wn^2=z(1,1) ,comparing the coefficients
14 Wn=sqrt(z(1,1));
15 //Wn=natural frequency
16 disp(Wn,"Wn=")
17 printf("2) with controller')
18 syms K;
19 sys1=syslin('c',1/(s*(s+4)));
20 sys2=sys1 /.(s*K);
```

```
21 G=64*sys2
22 disp(G,"G(s)=")
23
24 sys=G/.H;
25 sys=simple(sys);
26 disp(sys,"C(s)/R(s)=")
27 [num,den]=numden(sys)
28 //Coeff of den of symbolic function CL
29 cof_a_0 = coeffs(den,'s',0)
30 cof_a_1 = coeffs(den,'s',1)
31 //Wn^2= cof_a_0,comparing the coefficients
32 Wn=sqrt(cof_a_0)
33 Wn=dbl(Wn);
34 //Wn=natural frequency
35 disp(Wn,"natural frequency Wn=")
36 //cof_a_1=2*zeta*Wn
37 zeta=cof_a_1/(2*Wn)
38 zeta=0.6;
39 k=(16*zeta)-4
40 disp(k,"K=")
```

Scilab code Exa 9.11 `natural frequency Wn`

```
1 //calculates//
2 printf("2) with controller')
3 syms K;
4 sys1=syslin('c',1/(s*(s+1.2)));
5 sys2=sys1/.(s*K);
6 G=16*sys2;
7 G=simple(G)
8 disp(G,"G(s)=")
9 sys=G/.H;
10 sys=simple(sys);
11 disp(sys,"C(s)/R(s)=")
12 [num,den]=numden(sys)
```

```
13 den=den/5; //so that coeff of s^2=1
14 cof_a_0 = coeffs(den,'s',0) // coeff of den of
      symbolic function CL
15 cof_a_1 = coeffs(den,'s',1)
16 //Wn^2= cof_a_0 ,comparing the coefficients
17 Wn=sqrt(cof_a_0)
18 Wn=dbl(Wn);
19 disp(Wn,"natural frequency Wn=")            // Wn=
      natural frequency
20 //cof_a_1=2*zeta*Wn
21 //zeta=cof_a_1/(2*Wn)
22 zeta=0.56;
23 k=(8*zeta)-1.2
24 disp(k,"K=")
25 Wd=Wn*sqrt(1-zeta^2);
26 disp(Wd,"Wd=")
27 Tp=%pi/Wd;
28 disp(Tp,"Tp=")
29 Mp=100*exp((-%pi*zeta)/sqrt(1-zeta^2));
30 disp(Mp,"Mp=")
31 Ts=4/(zeta*Wn);
32 disp(Ts,"Ts=")
```

Chapter 10

Stability

Scilab code Exa 6.0 coefficient

```
1  //coefficient//
2  ieee(2);
3  s=%s;
4  m=s^5+2*s^4+4*s^3+8*s^2+3*s+1
5  r=coeff(m);  //Extracts the coefficient of the
      polynomial
6  n=length(r);
7  routh=[r([6,4,2]);r([5,3,1])]
8  syms eps;
9  routh=[routh;eps,-det(routh(1:2,2:3))/routh(2,2),0];
10 routh=[routh;-det(routh(2:3,1:2))/routh(3,1),-det(
      routh(2:3,2:3))/routh(3,2),0];
11 routh=[routh;-det(routh(3:4,1:2))/routh(4,1),-det(
      routh(3:4,2:3))/routh(4,2),0];
12 routh=[routh;-det(routh(4:5,1:2))/routh(5,1),0,0];
13 disp(routh,"routh=")
14 //To check the stability
15 routh(4,1)=8-limit(5/eps,eps,0); //Putting the value
       of eps=0 in routh(4,1)
16 disp(routh(4,1),"routh(4,1)=")
17 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

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

Scilab code Exa 10.03 equation

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

```
19 end
```

Scilab code Exa 10.04 equation

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

Scilab code Exa 10.05.01 equation

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

```
13 if (routh(i,1)<0)
14 c=c+1;
15 end
16 end
17 if(c>=1)
18 printf("system is unstable")
19 else ("system is stable")
20 end
```

Scilab code Exa 10.06 equation

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

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

Scilab code Exa 10.07 equation

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

Scilab code Exa 10.08 equation

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

```
9  routh=[routh;-det(routh(3:4,1:2))/routh(4,1),0,0]
10   c=0;
11  for i=1:n
12     if (routh(i,1)<0)
13       c=c+1;
14     end
15   end
16   if(c>=1)
17     printf("system is unstable")
18   else ("system is stable")
19   end
```

Scilab code Exa 11.08 value

```
1 //value//
2 s=%s;
3 H=syslin('c',(s+2)/((s+1)*s*(s+4)));
4 evans(H,100)
5 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

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

```
8  syms eps;
9  routh=[routh;eps,-det(routh(1:2,2:3))/routh(2,2),0];
10 routh=[routh;-det(routh(2:3,1:2))/routh(3,1),-det(
       routh(2:3,2:3))/routh(3,2),0];
11 routh=[routh;-det(routh(3:4,1:2))/routh(4,1),-det(
       routh(3:4,2:3))/routh(4,2),0];
12 routh=[routh;-det(routh(4:5,1:2))/routh(5,1),0,0];
13 disp(routh,"routh=")
14 //To check the stability
15 routh(4,1)=limit(routh(4,1),eps,0); //Putting the
       value of eps=0 in routh(4,1)
16 disp(routh(4,1),"routh(4,1)=")
17 routh(5,1)= limit(routh(5,1),eps,0); //Putting the
       value of eps=0 in routh(5,1)
18 disp(routh(5,1),"routh(5,1)=')
19 routh
20 printf("There are two sign changes of first column,
       hence the system is unstable \n")
```

Scilab code Exa 10.10 equation

```
1  //equation//
2  ieee(2);
3  syms  s k;
4  m=s^4+4*s^3+7*s^2+6*s+k;
5  cof_a_0 = coeffs(m,'s',0);
6  cof_a_1 = coeffs(m,'s',1);
7  cof_a_2 = coeffs(m,'s',2);
8  cof_a_3 = coeffs(m,'s',3);
9  cof_a_4 = coeffs(m,'s',4);
10
11 r=[cof_a_0 cof_a_1 cof_a_2 cof_a_3  cof_a_4]
12
13 n=length(r);
14 routh=[r([5,3,1]);r([4,2]),0]
```

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

Scilab code Exa 10.11 equation

```
1 //equation//
2 ieee(2);
3 syms  s k;
4 m=(24/100)*s^3+s^2+s+k;
5 cof_a_0 = coeffs(m,'s',0);
6 cof_a_1 = coeffs(m,'s',1);
7 cof_a_2 = coeffs(m,'s',2);
8 cof_a_3 = coeffs(m,'s',3);
9 r=[cof_a_0 cof_a_1 cof_a_2 cof_a_3]
10 n=length(r);
11 routh=[r([4,2]);r([3,1])]
12 routh=[routh;-det(routh)/routh(2,1),0]
13 t=routh(2:3,1:2); //extracting the square sub block
      of routh matrix
14 routh=[routh;-det(t)/routh(3,1),0]
15 disp(routh,"routh=");
16 routh(3,1)=0  //For marginaly stable system
17 k=1/0.24;
18 disp(k,"K(marginal)=")
19 disp('=0',(s^2)+k,"auxillary equation")
20 s=sqrt(-k);
21 disp(s,"Frequency of oscillation(in rad/sec)=")
```

Scilab code Exa 10.12 equation

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

Scilab code Exa 10.13 equation

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

Scilab code Exa 10.14 equation

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

Scilab code Exa 10.15 equation

```
1 //equation//
2 ieee(2);
3 syms s T;
4 m=s^2+(2-T)*s+1
5 cof_a_0 = coeffs(m,'s',0);
6 cof_a_1 = coeffs(m,'s',1);
7 cof_a_2 = coeffs(m,'s',2);
8 r=[cof_a_0 cof_a_1 cof_a_2]
9 n=length(r);
10 routh=[r([3,1]);r(2),0];
11 routh=[routh;-det(routh)/routh(2,1),0];
12 disp(routh,"routh=")
```

Scilab code Exa 10.16 equation

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

Chapter 11

Root Locus Method

Scilab code Exa 11.01 root locus

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

Scilab code Exa 11.02 rootlocus

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

Scilab code Exa 11.03 Rootlocus

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

Scilab code Exa 11.04 Root locus calculation

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

Scilab code Exa 11.05 root locus value

```
1 //value//
```

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

Scilab code Exa 11.06 Root locus value

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

Scilab code Exa 11.08 Root locus

```
1 //value//
2 s=%s;
```

```
3 H=syslin('c',(s+2)/((s+1)*s*(s+4)));
4 evans(H,100)
5 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 Root locus

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

Scilab code Exa 11.10 Centroid

```
1 //value//
2 n=3;
3 disp(n,"no of poles=")
4 m=1;
5 disp(m,"no of poles=")
6 q=0;
7 O=((2*q)+1)/(n-m)*180;
8 disp(O,"q=")
9 q=1;
10 O=((2*q)+1)/(n-m)*180;
11 disp(O,"q=")
12
13 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")
14 C=((0-1-4)-(-2))/2;
15 disp(C,"centroid=")
```

Scilab code Exa 11.11 Centroid

```
1 //value//
2 n=3;
3 disp(n,"no of poles=")
4 m=0;
5 disp(m,"no of poles=")
6 q=0;
7 O=((2*q)+1)/(n-m)*180;
8 disp(O,"q=")
9 q=1;
10 O=((2*q)+1)/(n-m)*180;
11 disp(O,"q=")
12 q=2;
13 O=((2*q)+1)/(n-m)*180;
14 disp(O,"q=")
15
16 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")
17 C=((0-1-1)-(-0))/3;
18 disp(C,"centroid=")
```

Scilab code Exa 11.12 Centroid

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

```
7  O=((2*q)+1)/(n-m)*180;
8  disp(O,"q=")
9  q=1;
10 O=((2*q)+1)/(n-m)*180;
11 disp(O,"q=")
12 q=2;
13 O=((2*q)+1)/(n-m)*180;
14 disp(O,"q=")
15
16 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")
17 C=((0-2-4-5)-(-3))/3;
18 disp(C,"centroid=")
```

Scilab code Exa 11.13 breakaway point

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

Scilab code Exa 11.14 breakin point

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

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

Scilab code Exa 11.15 breakaway point

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

Scilab code Exa 11.16 breakaway point

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

Scilab code Exa 11.17 auxillary equation

```
//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 Equation

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

Scilab code Exa 11.20 equation

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

Scilab code Exa 11.21 equation

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

Scilab code Exa 11.22 gain margin

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

Scilab code Exa 11.23 value

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

Scilab code Exa 11.25 equation

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

Scilab code Exa 11.26 equation

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

Scilab code Exa 11.27 root locus

```
1  //value//
2  s=%s;
3  syms k Wn;
4  H=syslin('c',1/((s+3)^2*s));
5  evans(H,100) //root locus
6  printf("To determine the value of Wn \n")
7  disp(k*H,"G(s)H(s)=")
8  y=1+(k*H);
9  disp('=0',y,"1+G(s)H(s)")
10 evans(H,100)
11 [num,den]=numden(y)
12
13  cof_a_0 = coeffs(num,'s',0);
14     cof_a_1 = coeffs(num,'s',1);
15     cof_a_2 = coeffs(num,'s',2);
16     cof_a_3 = coeffs(num,'s',3);
17     r=[cof_a_0 cof_a_1 cof_a_2 cof_a_3]
18
19 n=length(r);
20 routh=[r([4,2]);r([3,1])];
21 routh=[routh;-det(routh)/routh(2,1),0];
22 t=routh(2:3,1:2); //extracting the square sub block
      of routh matrix
23 routh=[routh;-det(t)/t(2,1),0]
24 //to obtain Wn
25 disp('=0',((6*s^2)+54),"auxillary eq")
26 p=(6*(s^2))+k;
27 s=%i*Wn
28 k=54;
29 p=eval(p)
30 Wn=sqrt(k/6)
31 printf("With gvn values of zeta adding a grid on
```

```
    root locus \n")
32
33 zeta=0.5; //given
34 sgrid(zeta,Wn,7) //add a grid over an existing
    continuous s-plane root with given values for
    zeta and wn.
35 printf("NOTE:-click on the point where locus
    intersects z=0.5 for desired value of k \n")
36 k=-1/real(horner(H,[1,%i]*locate(1))) //To obtain
    the gain at a given point of the locus
37
38
39 p=locate(1) //desired point on the root locus
```

Scilab code Exa 11.28 Equation

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

Chapter 12

Frequency Domain Analysis

Scilab code Exa 12.01 denominator polynomial

```
1 //denominator polynomial//
2 s=poly(0,'s'); //Defines s as polynomial variable
3 F=syslin('c',[225/((s+6)*s)]); //Creates transfer
    function in forward path
4 B=syslin('c',(1+0*s)/(1+0*s)); //Creates transfer
    function in backward path
5 CL=F/.B //Calculates closed-loop transfer function
6 // compare CL with Wn^2/(s^2+2*zeta*Wn+Wn^2)
7 y=denom(CL) //extracting the denominator of CL
8 z=coeff(y)  //extracting the coefficients of the
    denominator polynomial
9 //Wn^2=z(1,1) ,comparing the coefficients
10 Wn=sqrt(z(1,1))              // Wn=natural frequency
11 //2*zeta*Wn=z(1,2)
12 zeta=z(1,2)/(2*Wn)              // zeta=damping
    factor
13 Mr=1/(2*zeta*sqrt(1-zeta^2))
14 Wr=Wn*sqrt(1-zeta^2)
15 Wc=Wn*sqrt((1-2*zeta^2)+sqrt(4*zeta^4-4*zeta^2+2))
16 BW=Wc //BANDWIDTH
```

Scilab code Exa 12.02 denominator polynomial

```
1 //denominator polynomial//
2 s=poly(0,'s'); //Defines s as polynomial variable
3 F=syslin('c',[36/((s+8)*s)]); //Creates transfer
    function in forward path
4 B=syslin('c',(1+0*s)/(1+0*s)); //Creates transfer
    function in backward path
5 CL=F/.B //Calculates closed-loop transfer function
6 // compare CL with Wn^2/(s^2+2*zeta*Wn+Wn^2)
7 y=denom(CL) //extracting the denominator of CL
8 z=coeff(y)  //extracting the coefficients of the
    denominator polynomial
9 //Wn^2=z(1,1) ,comparing the coefficients
10 Wn=sqrt(z(1,1))              // Wn=natural frequency
11 //2*zeta*Wn=z(1,2)
12 zeta=z(1,2)/(2*Wn)              // zeta=damping
    factor
13 Mr=1/(2*zeta*sqrt(1-zeta^2))
14 Wr=Wn*sqrt(1-zeta^2)
15 Wc=Wn*sqrt((1-2*zeta^2)+sqrt(4*zeta^4-4*zeta^2+2))
16 BW=Wc //BANDWIDTH
```

Scilab code Exa 12.03 denominator polynomial

```
1 //denominator polynomial//
2 s=poly(0,'s'); //Defines s as polynomial variable
3 F=syslin('c',[81/(s^2+7*s)]); //Creates transfer
    function in forward path
4 B=syslin('c',(1+0*s)/(1+0*s)); //Creates transfer
    function in backward path
5 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
```

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

Scilab code Exa 12.05 denominator polynomial

```
1  //denominator polynomial//
2  ieee(2);
3  syms k a;
4  num=k ;
5  den=s*(a+s);
6  G=num/den;
7  disp(G,"G(s)=")
8  H=1;
9  CL=G/.H;
10 CL=simple(CL);
11 disp(CL,"C(s)/R(s)=")  //Calculates closed-loop
     transfer function
12 // compare CL with Wn^2/(s^2+2*zeta*Wn+Wn^2)
13 [num,den]=numden(CL); //extracts num & den of
     symbolic function (CL)
14 cof_a_0 = coeffs(den,'s',0); // coeff of den of
     symbolic function (CL)
15 cof_a_1 = coeffs(den,'s',1);
16 //Wn^2= cof_a_0 ,comparing the coefficients
17 Wn=sqrt(cof_a_0)
18 //cof_a_1=2*zeta*Wn
19 zeta=cof_a_1/(2*Wn)
20 Mr=1/(2*zeta*sqrt(1-zeta^2)) //----------1)
21 printf("Given ,Mr=1.25 \n");
22 //On solving eq (1) we get k=1.25a^2-------2
23 printf("k=1.25*a^2  \n")
```

```
24 Wr=Wn*sqrt(1-2*zeta^2)    //-----------------3)
25 printf("Given, Wr=12.65  \n") ;
26 //on soving eq (3),we get 2k-a^2=320----------4)
27 printf("2k-a^2=320 \n")
28 //now eq 2 &4 can be simultaneously soved to take
      out values of k &a
29 // Let k=x & a^2=y
30 A=[1,-1.25;2,-1]; // coefficient matrix
31 b=[0;320];
32 m=A\b;
33 x=m(1,1);
34 k=x
35 y=m(2,1);
36 a=sqrt(y)
37 Wn=dbl(eval(Wn));
38 disp(Wn,"Wn=")
39 zeta=dbl(eval(zeta));
40 disp(zeta,"zeta=')
41 Ts=4/(zeta*Wn) ;
42 disp(Ts,"Settling Time (Ts)=")
43 Wc=Wn((1-(2*zeta^2))+sqrt(4*zeta^4-4*zeta^2+2))
44 disp(Wc,"BW=")
```

Chapter 13

Bode Plot

Scilab code Exa 13.01 gain and phase margin

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

Scilab code Exa 13.02 gain and phase margin and associated crossover frequencies

```
1 //polynomial//
```

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

Scilab code Exa 13.03 Gain and phase margin and associated crossover frequencies

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

```
    associated crossover frequencies
```

Scilab code Exa 13.04 gain and phase margin

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

Scilab code Exa 13.05 gain and phase margin

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

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

Scilab code Exa 13.06 gain and phase margin

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

Chapter 15

Nyquist Plot

Scilab code Exa 15.01 no of poles

```
1 //system//
2 s=%s;
3 sys=syslin('c',1/(s+2))
4 nyquist(sys)
5 show_margins(sys,'nyquist')
6 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 no of poles

```
1 //system//
2 s=%s;
3 sys=syslin('c',1/(s*(s+2)))
4 nyquist(sys)
5 show_margins(sys,'nyquist')
6 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 no of poles

```
1 //system//
2 s=%s;
3 sys=syslin('c',1/(s^2*(s+2)))
4 nyquist(sys)
5 show_margins(sys,'nyquist')
6 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 no of poles

```
1 //system//
2 s=%s;
3 sys=syslin('c',1/(s^3*(s+2)))
4 nyquist(sys)
5 show_margins(sys,'nyquist')
6 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 no of poles

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

```
5 show_margins(sys,'nyquist')
6 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 nyquist

```
1 //system//
2 s=%s;
3 P1=1;
4 P2=2;
5 sys=syslin('c',1/((s+1)*(s+2)))
6 nyquist(sys)
7 show_margins(sys,'nyquist')
8 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 unstable and stable system

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

Scilab code Exa 15.08 stable and unstable

```
1 //system//
2 s=%s;
3 sys=syslin('c',(30)/((s^2+2*s+2)*(s+3)))
4 nyquist(sys)
5 gm=g_margin(sys)
6 show_margins(sys,'nyquist')
7 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")
8 if (gm<=0)
9   printf("system is unstable")
10 else
11   printf("system is stable")
12   end
```

Chapter 17

State Variable Approach

Scilab code Exa 17.03 Creating cont-time transfer function

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

Scilab code Exa 17.04 Transfer funtion

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

Scilab code Exa 17.06 Creating cont-time transfer function

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

Scilab code Exa 17.07 Creating cont-time transfer function

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

Scilab code Exa 17.08 Creating cont-time transfer function

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

Scilab code Exa 17.09 Find The Determinant

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

Scilab code Exa 17.10 transfer Function

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

Scilab code Exa 17.11 Transfer funtion

```
1 //function//
2 A=[0 1;-6 -5]
3 x=[1;0];
4 disp(x,"x(t)=')
5 s=poly(0,'s');
6 [Row Col]=size(A)    //Size of a matrix
7 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 `Transfer 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
```

```
15 //Inverse Laplace of each element of Matrix(phi(s))
16 q(i,j)=ilaplace(p(i,j),s,t);
17 end;
18 end;
19 disp(q,"phi(t)=") //State Transition Matrix
20 t=(t-m);
21 q=eval(q)        //At t=t-m ,evaluating q i.e phi(t-m
    )
22 //Integrate q w.r.t m(Indefinite Integration)
23 r=integ(q*B,m)
24 m=0              //Upper limit is t
25 g=eval(r)       //Putting the value of upper limit in
     q
26 m=t              //Lower Limit is 0
27 h=eval(r)       //Putting the value of lower limit in
     q
28 y=(h-g);
29 disp(y,"y=")
30 printf("x(t)= phi(t)*x(0) + integ(phi(t-m)*B) w.r.t
    m from 0 t0 t \n")
31 //x(t)=phi(t)*x(0)+integ(phi(t-m)*B) w.r.t m from 0
    t0 t
32 y1=(q*x)+y;
33 disp(y1,"x(t)=")
```

Scilab code Exa 17.13 Singular matrix

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

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

Scilab code Exa 17.14 Observable system

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

Scilab code Exa 17.16 Creating cont-time transfer function

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

Scilab code Exa 17.17 Creating cont-time transfer function

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

Scilab code Exa 17.18 time transfer function

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

Scilab code Exa 17.19 Output Response

```
1 //function//
2 A=[0 1;-3 -4]
3 B=[0;1]
4 C=[1 0]
5 x=[0;0]
6 disp(x,"x(t)=')
7 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

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

Scilab code Exa 17.21 `Transfer funtion`

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

Scilab code Exa 17.22 `Resolvent Matrix`

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

Scilab code Exa 17.23 `Transfer funtion`

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

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

Chapter 18

Digital Control Systems

Scilab code Exa 18.01.01 symsum

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

Scilab code Exa 18.08.01 symsum

```
1 //symsum//
2  function [Ztransfer]= ztransfer_new (sequence,n)
3   z = poly (0, 'z' , 'r' );  Ztransfer = sequence
      *(1/z )^n'
4 endfunction
5 sequence=[0 2 0 0 -3 0 0 8]
6 y=ztransfer(sequence);
7 disp(y,"ans=")
```