Scilab Textbook Companion for
Linear Control Systems
by B. S. Manke [1]

Created by
Ashish Kumar
B.Tech (Pursuing)
Electronics Engineering
M. N. N. I. T., Allahabad
College Teacher
Dr. Rajesh Gupta, MNNIT, Allahabad
Cross-Checked by
K Sryanarayan and Sonanya Tatikola. IIT Bombay

August 9, 2013

Book Description

Title: Linear Control Systems

Author: B. S. Manke

Publisher: Khanna Publishers

Edition: 9

Year: 2009

ISBN: 81-7409-107-6

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

5

9

List of Figures

Chapter 1

INTRODUCTION

Scilab code Exa 1.6.1.i inverse laplace transform

```
1  //Caption:inverse_laplace_transform
2  // example 1.6.1.(i)
3  //page 7
4  // F(s)=1/(s*(s+1))
5  s =%s ;
6  syms t ;
7  [A]=pfss(1/((s)*(s+1))) // partial fraction of F(s)
8  F1 = ilaplace (A(1),s,t)
9  F2 = ilaplace (A(2),s,t)
10 F=F1+F2;
11 disp (F," f(t)=")//result
```

Scilab code Exa 1.6.1.ii inverse laplace transform

```
1  //Caption:inverse_laplace_transform
2  // example 1.6.1.(ii)
3  //page 7
4  // F(s)=s+6/(s(s^2+4s+3))
```

```
5  s =%s ;
6  syms t ;
7  [A]= pfss((s+6)/(s*(s^2+4*s+3))) // partial fraction
        of F(s)
8  A(1)=2/s;
9  F1 = ilaplace (A(1),s,t)
10 F2 = ilaplace (A(2),s,t)
11 F3 = ilaplace (A(3),s,t)
12 F=F1+F2+F3;
13 disp (F," f (t)=")// result
```

Scilab code Exa 1.6.1.iii inverse laplace transform

```
1  //Caption:inverse_laplace_transform
2  // example 1.6.1.(iii)
3  //page 8
4  // F(s)=1/(s^2+4s+8)
5  s =%s ;
6  syms t ;
7  disp(1/(s^2+4*s+8),"F(s)=")
8  f=ilaplace(1/(s^2+4*s+8),s,t)
9  disp (f," f(t)=")//result
```

Scilab code Exa 1.6.1.iv inverse laplace transform

```
1  //Caption:inverse_laplace_transform
2  // example 1.6.1.(iv)
3  //page 8
4  // F(s)=s+2/(s^2+4s+6)
5  s =%s ;
6  syms t ;
7  disp((s+2)/(s^2+4*s+6),"F(s)=")
8  F=ilaplace((s+2)/(s^2+4*s+6),s,t)
```

```
9  disp (F," f(t)=")//result
```

Scilab code Exa 1.6.1.v inverse laplace transform

```
1  //Caption:inverse_laplace_transform
2  // example 1.6.1.(v)
3  //page 8
4  // F(s)=5/(s(s^2+4s+5))
5  s =%s ;
6  syms t ;
7  [A]= pfss (5/(s*(s^2+4*s+5))) // partial fraction of
       F(s)
8  F1= ilaplace (A(1),s,t)
9  F2= ilaplace (A(2),s,t)
10 F=F1+F2;
11 disp (F," f(t)=")//result
```

Scilab code Exa 1.6.1.vi program laplace transform

```
1  //Caption:program_laplace_transform
2  //example 1.6.1.(v)
3  //page 9
4  //this problem is solved in two parts because in
       this problem pfss function donot work.So, First
       we find partial fraction using method as we do in
        maths and then secondly we find inverse laplace
       transform as usual.
5  // partial fraction
6  s=%s
7  syms t;
8  num=(s^2+2*s+3);
9  den=(s+2)^3;
10 g=syslin('c',num/den);
```

```
11  rd=roots(den);
12  [n d k]=factors(g)
13  a(3)=horner(g*d(1)^3,rd(1))
14  a(2)=horner(derivat(g*d(1)^3),rd(1))
15  a(1)=horner(derivat(derivat(g*d(1)^3)),rd(1))
16  //inverse laplace
17  // partial fraction will be: a(1)/(s+1)+a(2)/((s+2)
       ^2)+a(3)/((s+2)^3)
18  F1 = ilaplace (1/d(1),s,t)
19  F2 = ilaplace (-2/(d(1)^2),s,t)
20  F3 = ilaplace (2*1.5/(d(1)^3),s,t)
21  F=F1+F2+F3
22  disp (F," f(t)=")//result
```

Scilab code Exa 1.6.2 solution of differential equation

```
1  //Caption:solution_of_differential equation
2  // example 1.6.2
3  //page 9
4  //after taking laplace transform and applying given
      condition , we get :
5  //X(s)=2s+5/(s(s+4))
6  s=%s;
7  syms t
8  [A]=pfss((2*s+5)/(s*(s+4)))
9  A(1)=1.25/s
10  F1 =ilaplace(A(1),s,t)
11  F2 = ilaplace(A(2),s,t)
12  f=F1+F2;
13  disp (f,"f(t)=")//result
```

Scilab code Exa 1.6.3 solution of differential equation

16

```
1 // Caption : solution_of_differential_equation
2 // example 1.6.3
3 // page 10
4 // after taking laplace transform and applying given
      condition , we get :
5 //X( s )=1/( s ^2+2 s +2)
6 s =%s ;
7 syms t
8 f = ilaplace (1/( s ^2+2* s +2) ,s ,t ) ;
9 disp (f ," f ( t )=") // result
```

Scilab code Exa 1.6.4 solution of differential equation

```
1 // Caption : solution_of_differential_equation
2 // example 1.6.4
3 // page 10
4 // after taking laplace transform and applying given
      condition , we get :
5 //Y( s )=(6* s +6) /(( s −1) *( s +2) *( s +3) )
6 s =%s ;
7 syms t
8 [A]= pfss ((6* s +6) /(( s -1) *( s +2) *( s +3) ) )
9 F1 = ilaplace (A (1) ,s ,t )
10 F2 = ilaplace (A (2) ,s ,t )
11 F3 = ilaplace (A (3) ,s ,t )
12 F=F1+F2+F3 ;
13 disp (F ," f ( t )=") // result
```

Scilab code Exa 1.6.5 initial value

```
1 // Caption : initial_value
2 // example 1.6.5
3 // page 11
```

```
4  // I ( s )=(C∗s /(RCs+1) )∗E( s )
5  // given :  E( s )=100/s ,R=2  megaohm   ,C=1  uF
6  //  so ,   I ( s )=(((1∗10^−6)∗s )/(2∗s+1) )∗(100/ s )
7  syms t
8  p=poly ( [0  10^-6] , 's ' , ' coeff ' ) ;
9  q=poly ( [1  2] , 's ' , ' coeff ' ) ;
10 r=poly ( [0  1] , 's ' , ' coeff ' ) ;
11 F1=p/q ;
12 F2=1/r ;
13 F=F1∗F2
14 f=ilaplace (F, s , t ) ;
15 z=limit ( f , t , 0 ) ;// initial  value  theorem
16 z=dbl ( z ) ;
17 disp ( z ,” i (0+)=”)
```

Scilab code Exa 1.6.7 final value

```
1  // Caption : final_value
2  //  example  1 . 6 . 7
3  // page  12
4  //X( s )=100/(s ∗( s ^2+2∗s+50) )
5  p=poly ( [100] , 's ' , ' coeff ' ) ;
6  q=poly ( [0  50  2  1] , 's ' , ' coeff ' ) ;
7  F=p/q ;
8  syms s
9  x=s∗F ;
10 y=limit ( x , s , 0 ) ;// final  value  theorem
11 y=dbl ( y )
12 disp ( y ,” x ( inf )=”)// result
```

Scilab code Exa 1.6.8 steady state value

```
1  // Caption : steady_state_value
```

18

```
2 // example 1.6.7
3 //page 12
4 //X(s)=s/(s^2*(s^2+6*s+25))
5 p=poly([0 1],'s','coeff');
6 q=poly([0 0 25 6 1],'s','coeff');
7 F=p/q;
8 syms s
9 x=s*F;
10 y=limit(x,s,0);//final value theorem
11 y=dbl(y)
12 disp(y,"x(inf)=")//result
```

Scilab code Exa 1.6.9 initial values

```
1 //Caption: initial_values
2 // example 1.6.7
3 //page 13
4 //F(s)=(4*s+1)/(s^3+2*s)
5 s=%s;
6 syms t;
7 F=(4*s+1)/(s^3+2*s)
8 f = ilaplace (F,s,t);
9 y=limit(f,t,0);//initial value theorem
10 y=dbl(y);
11 disp(y,"f(0+)=")
12 // since F'(s)=sF(s)-f(0+) where L(f'(t))=F'(s)=F1
13 F1=(4*s+1)/(s^2+2)
14 f1= ilaplace(F1,s,t);
15 y1=limit(f1,t,0);//initial value theorem
16 y1=dbl(y1);
17 disp(y1,"f_prime(0+)=")
18 // since F''(s)=(s^2)*F(s)-s*f(0+)-f'(0+) where L(f
        ''(t))=F''(s)=F2
19 F2=(s-8)/(s^2+2)
20 f2= ilaplace(F2,s,t);
```

```
21  y2=limit(f2,t,0);//initial value theorem
22  y2=dbl(y2);
23  disp(y2,"f_doubleprime(0+)=")
```

Scilab code Exa 1.6.10 final value

```
1  //Caption:final_value
2  // example 1.6.10
3  //page 13
4  syms t s;
5  F=4/(s^2+2*s)
6  x=s*F
7  x=simple(x)
8  z=limit(x,s,0);//final value theorem
9  z=dbl(z);
10  disp(z,"f(0+)=")
```

Chapter 2

TRANSFER FUNCTIONS

Scilab code Exa 2.4.1 pole zero plot

```
1  //Caption: pole_zero_Plot
2  // example 2.4.1
3  //page 19
4  //transfer function:G(s)=(1/2((s^2+4)*(1+2.5*s))/((s
      ^2+2)*(1+0.5*s)))
5  s=%s;
6  G=syslin('c',(1/2*((s^2+4)*(1+2.5*s))/((s^2+2)
      *(1+0.5*s))));
7  disp(G,"G(s)=");
8  x=plzr(G)
9  xtitle('pole-zero_configuration','Real_part','
      Img_part');
10 // value at s=2
11 a=2;
12 g=(1/2*((a^2+4)*(1+2.5*a))/((a^2+2)*(1+0.5*a)));
13 disp(g,"G(2)=");
```
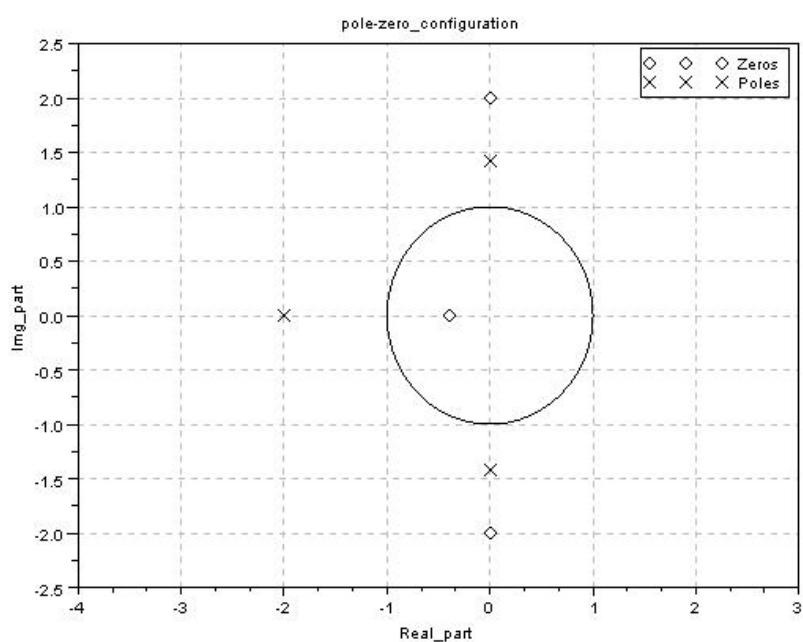
Figure 2.1: pole zero plot

Scilab code Exa 2.4.2 final value

```scilab
1  //Caption:final_value
2  // example  2.4.2
3  //page  20
4  //refer  to  fig.2.4.2  given  on  page  20
5  //poles  are  located  at  s=0,-2  and  -4
6  //zero  at  s=-3
7  s=%s;
8  syms  K;
9  g=syslin('c',((s+3))/(s*(s+2)*(s+4)));//transfer
       function
10 G=K*g;//transfer  function
11 disp(G,"G(s)=");
12 //G(s)=3.2  at  s=1;
13 //on  solving  we  find  K=12
14 K=12;
15 G=K*g;
16 disp(G,"G(s)=")
```

Chapter 3

BLOCK DIAGRAMS

Scilab code Exa 3.2.1 Transfer Function

```
1  //Caption: transfer_function
2  // example 3.2.1
3  //page 32
4  // we have defined parallel and series function
      which we are going to use here
5  //exec parallel.sce;
6  //exec series.sce;
7  syms G1 G2 G3 H1;
8  // shifting take off point after block G2 to a
      position before block G2
9  a= G2*H1;
10 b=parallel(G2,G3);
11 //shifting take off point before (G2+G3) to After (
      G2+G3)
12 c=a/b;
13 m=1;
14 d=b/(1+m*b);
15 e=series(G1,d);
16 y=(e/(1+c*e));
17 y=simple (y);
18 disp (y,"C(s)/R(s)=");
```

check Appendix AP 2 for dependency:

parallel.sce

check Appendix AP 1 for dependency:

series.sce

Scilab code Exa 3.2.2 Transfer Function

```
1  //Caption: transfer_function
2  // example 3.2.2
3  //page 34
4  // we have defined parallel and series function
       which we are going to use here
5  //exec parallel.sce;
6  //exec series.sce;
7  syms G1 G2 G3 H1;
8  // shifting take off point before block G1 to a
       position after block G1
9  a=G1*H1;
10 b=parallel(G1,G2);
11 c=G3/.a
12 y=series(b,c);
13 y=simple (y);
14 disp(y,"C(s)/R(s)=")
```

check Appendix AP 2 for dependency:

parallel.sce

check Appendix AP 1 for dependency:

series.sce

Scilab code Exa 3.2.3 Transfer Function

```
1  //Caption:transfer_function
2  // example 3.2.3
3  //page 35
4  // we have defined parallel and series function
       which we are going to use here
5  //exec parallel.sce;
6  //exec series.sce;
7  syms G1 G2 G3 G4 G5 G6 H1 H2;
8  //shift the takeof point placed before G2 towards
       right side of block G2
9  a= G5/G2;
10 b=parallel(G3,G4);
11 c=series(b,G6);
12 d=parallel(a,c);
13 e=series(G1,G2);
14 l=series(H1,d);
15 g=e/.H2
16 y=g/(1+g*l)
17 y=simple (y);
18 disp(y,"C(s)/R(s)=")
```

check Appendix AP 2 for dependency:

parallel.sce

check Appendix AP 1 for dependency:

series.sce

Scilab code Exa 3.2.4 Transfer Function

```
1  //Caption:transfer_function
2  // example 3.2.4
3  //page 36
```

```
4 // we have defined parallel and series function
      which we are going to use here
5 // exec parallel.sce;
6 // exec series.sce;
7 syms G1 G2 G3 H1 H2;
8 // shift the take off point placed before G1 to after
      block G1
9 a=G3/G1;
10 b=parallel(a,G2);
11 // shift the take off point before block b towards
      right side of same block
12 c=1/b;
13 d=series(H1,c);
14 e=series(G1,b);
15 g=parallel(H2,d);
16 y=e/(1+e*g);
17 y=simple (y);
18 disp(y,"C(s)/R(s)=")
```

check Appendix AP 2 for dependency:

parallel.sce

check Appendix AP 1 for dependency:

series.sce

Scilab code Exa 3.2.5 Transfer Function

```
1 // Caption: transfer_function
2 // example 3.2.5
3 // page 37
4 // we have defined parallel and series function
      which we are going to use here
5 // exec parallel.sce;
6 // exec series.sce;
7 syms G1 G2 G3 H1 H2 H3;
```

```
8  //shift the take off point placed before G3 to after
      block G3
9  a=H3/G3;
10 b=G3/.H2;
11 c=series(G2,b);
12 //shift the summing point after block G1 to before
      block G1
13 d=a/G1;
14 e=G1/.H1;
15 f=series(e,c);
16 y=f/(1+f*d);
17 y=simple (y);
18 disp(y,"C(s)/R(s)=")
```

check Appendix AP 2 for dependency:

parallel.sce

check Appendix AP 1 for dependency:

series.sce

Scilab code Exa 3.2.6 Transfer Function

```
1  //Caption:transfer_function
2  // example 3.2.6
3  //page 38
4  // we have defined parallel and series function
      which we are going to use here
5  //exec parallel.sce;
6  //exec series.sce;
7  syms G1 G2 G3 H1;
8  a=parallel(G1,G3);
9  b=G2/.H1;
10 y=series(a,b);
11 y=simple (y);
12 disp(y,"C(s)/R(s)=");
```

check Appendix for dependency:

```
parallel.sce
```

check Appendix for dependency:

```
series.sce
```

Scilab code Exa 3.2.7 Transfer Function

```
1  //Caption:transfer_function
2  // example 3.2.7
3  //page 39
4  // we have defined parallel and series function
       which we are going to use here
5  //exec parallel.sce;
6  //exec series.sce;
7  syms G1 G2 G3 H1 H2;
8  //shift the take off point placed before G2 to after
        block G2
9  a=G2*H1;
10 //shift the take off point placed before G2 to after
        block G1 and shift the summing point before
       block G2 to before block G2
11 //interchange consecutive summing point and shift
       the take off point placed before H2 to after
       block H2
12 b=G2*H2;
13 c=b*H1;
14 d=parallel(G3,G2);
15 e=1/.b
16 f=series(d,e);
17 g=G1/.a
18 h=series(g,f);
19 y=h/(1-h*c);
```

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

check Appendix for dependency:

`parallel.sce`

check Appendix for dependency:

`series.sce`

Scilab code Exa 3.2.8 Transfer Function

```
 1  // Caption : transfer_function
 2  //  example  3.2.8
 3  // page 41
 4  //  we  have  defined  parallel  and  series  function
         which  we  are  going  to  use  here
 5  // exec  parallel.sce;
 6  // exec  series.sce;
 7  syms  G1  G2  G3  G4  G5  H1  H2;
 8  // shift  the  summing  point  before  block  G5  towards
         left  of  block  G5
 9  a=G2*G5;
10  b=G4/.H1;
11  c=series(G5,H2);
12  d=series(b,G3);
13  e=d/(1+d*c);
14  e=simple(e)
15  f=parallel(G1,a);
16  y=series(f,e);
17  y=simple (y);
18  disp(y,"C(s)/R(s)=");
```

check Appendix for dependency:

`parallel.sce`

check Appendix AP 1 for dependency:

```
series.sce
```

Scilab code Exa 3.2.9 Transfer Function

```
1  //Caption:transfer_function
2  // example 3.2.9
3  //page 42
4  // we have defined parallel and series function
       which we are going to use here
5  //exec parallel.sce;
6  //exec series.sce;
7  syms G1 G2 G3 G4 G5 H1 H2 H3;
8  //shift the take off point placed before G3 to after
        block G3
9  a=G5/G3;
10 b=G1/.H1;
11 c=series(b,G2);
12 d=G3/.H2;
13 e=parallel(G4,a);
14 e=simple(e);
15 f=series(c,d);
16 g=series(f,e);
17 g=simple(g);
18 y=g/(1+g*H3);
19 y=simple(y);
20 disp (y,"C(s)/R(s)=");
```

check Appendix AP 2 for dependency:

```
parallel.sce
```

check Appendix AP 1 for dependency:

```
series.sce
```

Scilab code Exa 3.2.10 Transfer Function

```
1  //Caption: transfer_function
2  // example 3.2.10
3  //page 43
4  // we have defined parallel and series function
        which we are going to use here
5  //exec parallel.sce;
6  //exec series.sce;
7  syms G1 G2 G3 G4 G5 H1 H2 H3;
8  a=parallel(G3,G4);
9  //shift off the take off point before block 'a' to
        after block 'a'
10 b=1/a;
11 d=1;
12 c=G2/(1+G2*d);
13 e=parallel(H1,b);
14 f=series(c,a);
15 g=series(H2,e);
16 h=f/(1+f*g);
17 h=simple(h);
18 i=series(h,G1);
19 y=i/(1+i*H3);
20 y=simple(y);
21 disp(y,"C(s)/R(s)=")
```

check Appendix AP 2 for dependency:

parallel.sce

check Appendix AP 1 for dependency:

series.sce

Scilab code Exa 3.2.11 Transfer Function

```
1  //Caption: transfer_function
2  // example 3.2.11
3  //page 45
4  // we have defined parallel and series function
        which we are going to use here
5  //exec parallel.sce;
6  //exec series.sce;
7  syms G1 G2 G3 G4 H1 H2 H3;
8  a=G4/.H1;
9  //shift the summing point after block G3 towards
        left of block G3
10 b=G2/G3;
11 c=series(a,G3);
12 d=c/(1+c*H2);
13 d=simple(d)
14 //shift the summing point before block G1 towards
        right of block G1
15 e=G1*H3;
16 f=d/(1+d*e);
17 f=simple(f)
18 g=parallel(G1,b);
19 g=simple(g);
20 y=series(g,f)
21 y=simple (y);
22 disp(y,"C(s)/R(s)=");
```

check Appendix AP 2 for dependency:

parallel.sce

check Appendix AP 1 for dependency:

series.sce

Scilab code Exa 3.2.12 Transfer Function

```
1  //Caption:transfer_function
2  // example 3.2.12
3  //page 47
4  // we have defined parallel and series function
       which we are going to use here
5  //exec parallel.sce;
6  //exec series.sce;
7  syms G1 G2 G3 G4 H1 H2 H3;
8  //shift the summing point before block G1 towards
       right of block G1
9  //shift the summing point after block G3 towards
       left of block G3
10 a=G2/G1
11 b=H3/G3;
12 c=G1/.H1
13 d=G3/.H2
14 e=series(G4,d)
15 //shift the summing point after block e towards left
       of block e
16 g=1/e
17 f=series(a,g);
18 f=simple(f)
19 h=parallel(f,1);
20 h=simple(h)
21 i=e/(1+e*b);
22 i=simple(i)
23 j=series(c,h);
24 y=series(j,i);
25 y=simple (y);
26 disp(y,"C(s)/R(s)=");
```

check Appendix AP 2 for dependency:

parallel.sce

check Appendix AP 1 for dependency:

series.sce

Scilab code Exa 3.2.13 Transfer Function

```
1  //Caption: transfer_function
2  // example 3.2.13
3  //page 49
4  // we have defined parallel and series function
       which we are going to use here
5  //exec parallel.sce;
6  //exec series.sce;
7  syms G1 G2 G3 G4 H1 H2;
8  //shift the summing point after block H2 towards
       right of block H2
9  //shift the summing point after block H1 towards
       right of block H2
10 a=H1*H2
11 b=parallel(G1,G2)
12 c=G4/.a
13 d=series(G3,c)
14 e=d/(1+d*H2);
15 e=simple(e)
16 f=series(b,e)
17 y=f/(1+f*a);
18 y=simple (y);
19 disp(y,"C(s)/R(s)=");
```

check Appendix AP 2 for dependency:

parallel.sce

check Appendix AP 1 for dependency:

series.sce

Scilab code Exa 3.2.14 Transfer Function

```
 1  //Caption: transfer_function
 2  // example 3.2.14
 3  //page 50
 4  // we have defined parallel and series function
        which we are going to use here
 5  //exec parallel.sce;
 6  //exec series.sce;
 7  syms G1 G2 G3 G4 H1 H2 R D;
 8  a=parallel(G1,G2)
 9  b=G3/.H2
10  c=series(a,b)
11  //inorder to determine C(s)/R(s) consider D=0
12  d=series(c,G4);
13  y=d/(1+d*H1);
14  y=simple (y);
15  disp(y,"C(s)/R(s)=")
16  // now consider R=0 for calculating C(s)/D(s)
17  e=series(c,H1)
18  z=G4/(1+G4*e);
19  z=simple(z);
20  disp(z,"C(s)/D(s)=");
21  x=(y*R)+(z*D);
22  x=simple(x);
23  printf("total output");
24  disp(x,"C(s)")
```

check Appendix AP 2 for dependency:

parallel.sce

check Appendix AP 1 for dependency:

series.sce

Scilab code Exa 3.2.15 Transfer Function

36

```
 1  //Caption:transfer_function
 2  // example 3.2.15
 3  //page 52
 4  // we have defined parallel and series function
        which we are going to use here
 5  //exec parallel.sce;
 6  //exec series.sce;
 7  syms G1 G2 H1 H2 H3 D1 D2 R;
 8  a=G2/.H2
 9  b=a/(1+a*H3);
10  b=simple(b)
11  //inorder to determine C(s)/R(s) consider D1=0,D2=0
12  c=series(b,G1)
13  y=c/(1+c*H1);
14  y=simple(y)
15  disp(y,"C(s)/R(s)");
16  // now consider R=0,D2=0 for calculating C(s)/D1(s)
17  d=series(G1,H1);
18  z=b/(1+b*d);
19  z=simple(z)
20  disp(z,"D1(s)/R(s)");
21  // now consider R=0,D1=0 for calculating C(s)/D2(s)
22  e=G1*(-H1);
23  f=series(b,e);
24  x=f/(1+f);
25  x=simple(x);
26  disp(x,"D2(s)/R(s)");
27  out=(y*R)+(z*D1)+(x*D2);
28  out=simple(out);
29  disp(out,"C(s)");
```

check Appendix AP 2 for dependency:

```
parallel.sce
```

check Appendix AP 1 for dependency:

```
series.sce
```

Chapter 4

SIGNAL FLOW GRAPHS

Scilab code Exa 4.3.1 Overall Transmittance

```
 1  //Caption: overall_transmittance
 2  //  example  4.3.1
 3  //page  63
 4  syms  G1  G2  H1;
 5  //  forward  path  denoted  by  P1,P2  and  so  on  and  loop
        by  L1,L2  and  so  on
 6  //path  factor  by  D1,D2  and  so  on  and  graph
        determinant  by  D
 7  L1=-G1*H1;
 8  L2=-G2*H1;
 9  P1=G1;
10  P2=G2;
11  D1=1;
12  D2=1;
13  D=1-(L1+L2);
14  Y=(P1*D1+P2*D2)/D;
15  Y=simple(Y)
16  disp(Y,"C(s)/R(s)=")
```

Scilab code Exa 4.3.2 Overall Transmittance

```
1  // Caption : overall_transmittance
2  // example 4.3.2
3  // page 64
4  syms G1 G2 H1;
5  // forward path denoted by P1,P2 and so on and loop
       by L1,L2 and so on
6  // path factor by D1,D2 and so on and graph
       determinant by D
7  P1=G1;
8  P2=G2;
9  L1=-G1*H1;
10  D1=1;
11  D2=1;
12  D=1-(L1);
13  Y=(P1*D1+P2*D2)/D;
14  Y=simple(Y);
15  disp(Y,"C(s)/R(s)=")
```

Scilab code Exa 4.4.1 Closed Loop Transfer Function

```
1  // Caption : closed_loop_transfer_function
2  // example 4.4.1
3  // page 64
4  syms G1 G2 G3 H1;
5  // forward path denoted by P1,P2 and so on and loop
       by L1,L2 and so on
6  // path factor by D1,D2 and so on and graph
       determinant by D
7  P1=G1*G3;
8  P2=G2*G3;
9  L1=-G3*H1;
10  D1=1;
11  D2=1;
```

```
12  D=1-(L1);
13  Y=(P1*D1+P2*D2)/D;
14  Y=simple(Y)
15  disp(Y,"C(s)/R(s)=")
```

Scilab code Exa 4.4.2 Overall Gain

```
 1  //Caption:overall_gain
 2  // example  4.4.2
 3  //page 65
 4  syms a b c d e f g h
 5  // forward path denoted by P1,P2 and so on and loop
       by L1,L2 and so on
 6
 7  //path factor by D1,D2 and so on and graph
       determinant by D
 8  P1=a*b*c*d;
 9  P2=a*g;
10  L1=f;
11  L2=c*e;
12  L3=d*h;
13  //nontouching loops are  L1L2,  L1L3
14  L1L2=L1*L2;
15  L1L3=L1*L3;
16  D1=1;
17  D2=1-L2;
18  D=1-(L1+L2+L3)+(L1L2+L1L3);
19  D=simple(D);
20  Y=(P1*D1+P2*D2)/D;
21  Y=simple(Y);
22  disp(Y,"x5/x1");
```

Scilab code Exa 4.4.3 to find various signal flow graph parameter

```
1  // Caption :
       to_find_various_signal_flow_graph_parameter
2  // example 4.4.3
3  // page 66
4  syms a b c d e f g h ij
5  // forward path denoted by P1,P2 and so on and loop
       by L1,L2 and so on
6  // path factor by D1,D2 and so on and graph
       determinant by D
7  // six independent path
8  P1=a*b*d
9  P2=e*f*h
10 P3=a*j*h
11 P4=e*i*d
12 P5=-e*i*c*j*h
13 P6=-a*j*g*i*d
14 //3 INDIVIDUAL LOOPS
15 L1=-b*c
16 L2=-f*g
17 L3=-i*c*j*g
18 //NON TOUCHING LOOPS
19 L1L2=L1*L2
20 //PATH FACTORS
21 D3=1;D4=1;D5=1;D6=1
22 D1=1-L2
23 D2=1-L1
24 //GRAPH DETERMINANT
25 D=1-(L1+L2+L3)+(L1L2);
26 D=simple(D)
27 disp(D,"graph_determinant=")
```

Scilab code Exa 4.4.4 Transfer function using mason gain formula

```
1  // Caption : transfer_function_using_mason '
       s_gain_formula
```

```
2  // example 4.4.4
3  //page 67
4  syms G1 G2 G3 H1;
5  // forward path denoted by P1,P2 and so on and loop
        by L1,L2 and so on
6  //path factor by D1,D2 and so on and graph
        determinant by D
7  P1=G1*G2*1;
8  P2=G1*G3;
9  L1=G2*(-1);
10 L2=G3*(-1);
11 L3=-G1*G2*H1
12 D1=1;
13 D2=1;
14 D=1-(L1+L2+L3);
15 Y=(P1*D1+P2*D2)/D;
16 Y=simple(Y);
17 disp(Y,"C(s)/R(s)=")
```

Scilab code Exa 4.4.5 Transfer function using mason gain formula

```
1  //Caption:transfer_function_using_mason '
        s_gain_formula
2  // example 4.4.5
3  //page 68
4  syms G1 G2 G3 H1 H2;
5  // forward path denoted by P1,P2 and so on and loop
        by L1,L2 and so on
6  //path factor by D1,D2 and so on and graph
        determinant by D
7  P1=G1*G2;
8  P2=G1*G3;
9  L1=-G2*H2;
10 L2=-G1*G2*H1;
11 L3=G1*G3*(-H2)*G2*(-H1);
```

```
12  L3=simple(L3)
13  D1=1;
14  D2=1;
15  D=1-(L1+L2+L3);
16  D=simple(D)
17  Y=(P1*D1+P2*D2)/D;
18  Y=simple(Y);
19  disp(Y,"C(s)/R(s)=")
```

Scilab code Exa 4.4.6 Transfer function using mason gain formula

```
1  //Caption:transfer_function_using_mason'
        s_gain_formula
2  // example 4.4.6
3  //page 69
4  syms G1 G2 G3 G4 H1 H2;
5  // forward path denoted by P1,P2 and so on and loop
        by L1,L2 and so on
6  //path factor by D1,D2 and so on and graph
        determinant by D
7  // to find C/R consider D=0
8  P1=G1*G3*G4;
9  P2=G2*G3*G4;
10  L1=-G3*H1;
11  L2=-G1*G3*G4*H2;
12  L3=-G2*G3*G4*H2;
13  D1=1;
14  D2=1;
15  D=1-(L1+L2+L3);
16  D=simple(D)
17  Y=(P1*D1+P2*D2)/D;
18  Y=simple(Y);
19  disp(Y,"C(s)/R(s)=")
```

Scilab code Exa 4.4.7 Transfer function using mason gain formula

```
1  // Caption : transfer_function_using_mason '
       s_gain_formula
2  // example 4.4.7
3  // page 70
4  syms G1 G2 G3 H1;
5  // forward path denoted by P1,P2 and so on and loop
       by L1,L2 and so on
6  // path factor by D1,D2 and so on and graph
       determinant by D
7  P1=G1*G3*G4;
8  P2=G1*(-G2);
9  L1=G3*G4*(-1);
10 L2=G1*G3*H1*(-1);
11 L3=G1*G3*H1*(-1);
12 L4=G1*(-G2)*(-1)*G3*H1*(-1);
13 L5=G1*(-G2)*(-1)*G3*H1*(-1);
14 L4=simple(L4);
15 L5=simple(L5);
16 D1=1;
17 D2=1;
18 D=1-(L1+L2+L3+L4+L5);
19 D=simple(D)
20 Y=(P1*D1+P2*D2)/D;
21 Y=simple(Y);
22 disp(Y,"C(s)/R(s)=")
```

Scilab code Exa 4.4.8 Transfer function using mason gain formula

```
1  // Caption : transfer_function_using_mason '
       s_gain_formula
```

```
2  // example 4.4.8
3  //page 70
4  syms G1 G2 G3 G4 G5 H1 H2;
5  // forward path denoted by P1,P2 and so on and loop
       by L1,L2 and so on
6  //path factor by D1,D2 and so on and graph
       determinant by D
7  P1=G1*G2*G4;
8  P2=G2*G3*G5;
9  P3=G3;
10 L1=-G4*H1;
11 L2=-G2*G4*H2;
12 L3=-G2*G5*H2;
13 D1=1;
14 D2=1;
15 D3=1-L1;
16 D=1-(L1+L2+L3);
17 D=simple(D)
18 Y=(P1*D1+P2*D2+P3*D3)/D;
19 Y=simple(Y);
20 disp(Y,"C(s)/R(s)=")
```

Scilab code Exa 4.4.9 Overall Transfer Function

```
1  //Caption:overall_transfer_function
2  // example 4.4.9
3  //page 71
4  // we have defined parallel and series function
       which we are going to use here
5  exec parallel.sce;
6  exec series.sce;
7  syms G1 G2 G3 G4 H5 H1 H2;
8  //shift the SUMMING point locsted after G3 towards
       left of block G3
9  a=G2/.H1;
```

```
10  b=G5/G3;
11  c=parallel(a,b);
12  c=simple(c);
13  d=G3/.H2;
14  e=series(G1,c);
15  f=series(e,d);
16  y=series(G4,f);
17  y=simple (y);
18  disp(y,"C(s)/R(s)=")
```

Scilab code Exa 4.4.10 Transfer function using mason gain formula

```
1  //Caption:transfer_function_using_mason'
       s_gain_formula
2  // example 4.4.10
3  //page 72
4  syms G1 G2 G3 G4 G5 H1 H2;
5  // forward path denoted by P1,P2 and so on and loop
       by L1,L2 and so on
6  //path factor by D1,D2 and so on and graph
       determinant by D
7  P1=G1*G2*G3*G4;
8  P2=G1*G5*G4;
9  L1=-G2*H1;
10  L2=-G3*H2;
11  D1=1;
12  D2=1;
13  D=1-(L1+L2);
14  D=simple(D)
15  Y=(P1*D1+P2*D2)/D;
16  Y=simple(Y);
17  disp(Y,"C(s)/R(s)=")
```

Chapter 5

MODELLING A CONTROL SYSTEM

Scilab code Exa 5.9.4 Calculate Reference Voltage Vr

```
1  // Caption : calculate_reference_voltage_Vr
2  // example 5.9.4
3  // page 102
4  exec series.sce;
5  A=2// amplifier
6  K=10// motor_field
7  N=100// speed
8  tg=0.1// tacho_generator
9  a=series(A,K);
10 b=a/.tg;
11 disp(b,"N/Vr=");
12 // since N=100
13 Vr=N*(1/b);
14 disp(Vr,"reference_voltage=");
```

check Appendix AP 1 for dependency:

series.sce

Scilab code Exa 5.9.5 Calculate Reference Voltage Vr

```
1 //Caption:calculate_reference_voltage_Vr
2 //example 5.9.5
3 //page 102
4 exec series.sce;
5 Vt=250//output_voltage
6 Rf=100//field_winding_resistance
7 Kg=500//generator_constant
8 A=2//amplifier
9 Vf=0.4//
      fraction_of_output_voltage_compared_with_reference_voltage

10 a=1/Rf;
11 b=series(A,a);
12 c=series(b,Kg);
13 d=c/.Vf;
14 disp(d,"Vt/Vr=");
15 //since Vt=250
16 Vr=Vt*(1/d);
17 disp(Vr,"reference_voltage=");
```

check Appendix AP 1 for dependency:

```
series.sce
```

Scilab code Exa 5.9.6 Calculate Gain of Amplifier Ka

```
1 //Caption:calculate_gain_of_amplifier_Ka
2 //example 5.9.6
3 //page 103
4 //steady state equations: Ei=Vt+Il*Ra and Vt=Il*Rl
5 //where Vt=output_voltage, Rl=load_resistance, Ra=
      armature_resistance, Il=load_current,
6 syms Rl Ra Ka Er e;
7 Ei=450;
```

```
8  If =1.5;// field current
9  Kg=Ei/If;// generator_emf_constant
10 Vt =400;
11 //from stedy state eq. we get : Vt/Ei=Rl/(Rl+Ra)
12 a=Rl/(Rl+Ra);
13 a=Vt/Ei;
14 c=(Kg*a);
15 G=(Ka*c);
16 H=0.1;
17 // transfer function relating erroe 'e' and the
       reference voltage 'Er' is e/Er=1/(1+GH)
18 b=e/Er;
19 b=1/(1+G*H);
20 e=Vt*H*.02 ;// since allowable error is 2%
21 Er=(Vt*H)+e;
22 // since e/Er=1/(1+G*H), on putting value of e,Er,G
       and H and solving we get
23 Ka=1.89;
24 disp(Ka,"gain_of_amplifier_Ka=");
```

Scilab code Exa 5.9.7 Transfer Function of Generator

```
1  // Caption: transfer_function_of_generator
2  // example 5.9.7
3  // page 105
4  syms E Vf Kg R L
5  s=%s;
6  // generator_field_constant_Kg=delta(e)/delta(If)
7  Kg=50/2;
8  L=2;// field_inductance
9  R=200;// field_resistance
10 // transfer function is given by : E/Vf=(Kg/R+s*L)
11 a=Kg/(R+s*L);
12 disp(a,"E(s)/Vf(s)=");
```

Scilab code Exa 5.9.8 Overall Transfer Function of given System

```
1  //Caption:overall_transfer_function_of_given_system
2  //example 5.9.8
3  //page 105
4  syms Rf Ra Kb Jm Lf La Kg Kt Jl s
5  Rf=1000;//field_resistance
6  Lf=100;//field_inductanc
7  Kg=1000;//generator_field_constant
8  Ra=20;//armature_resistance
9  La=0.1;//armature_inductance
10 Kt=1.2;//motor_torque_const
11 Kb=1.2;//motor_back_emf_const
12 Jl=0.00003;//moment_of_inertia
13 Jm=0.00002;//coeff_of_viscous_friction
14 a=Kt/(Ra+s*La);
15 b=1/((Jm+Jl)*s);
16 c=(a*b);
17 d=c/(1+c*Kb);
18 e=Kg/(Rf+s*Lf);
19 f=(d*e);
20 f=simple(f)
21 disp(f,"wss(s)/Vf(s)=");
22
23 //steady state value
24 disp("under steady state condition , on putting s=0
       in expression f,we get:")
25 disp("Vf=1.2*wss")
```

Scilab code Exa 5.9.9 Overall Transfer Function of given System

```
1  //Caption:overall_transfer_function_of_given_system
```

```
2  //example  5.9.9
3  //page  107
4  syms  Ka  Ke  Kt  J  f  N1  N2
5  s=%s;
6  Ke=10;//error_detector_gain
7  Ka=100;//amplifier_transconductance
8  Kt=.0005;//motor_torque_const
9  J=.0000125;//moment_of_inertia
10 f=.0005;//coeff_of_viscous_friction
11 g=N1/N2;
12 g=1/20;
13 a=(Ka*Ke);
14 b=(a*Kt);
15 c=1/(J*s^2+f*s);
16 d=(c*b);
17 e=(g*d);
18 h=e/(1+e);
19 disp(h,"C(s)/R(s)=");//result
```

Scilab code Exa 5.9.10 Overall Transfer Function of given System

```
1  //Caption:overall_transfer_function_of_given_system
2  //example  5.9.10
3  //page  108
4  syms  Ra  Kt  Jm  fm  Kb
5  //where  Ra=armature_resistance;  Kt=
       motor_torque_const;  Jm=moment_of_inertia;  fm=
       coeff_of_viscous_friction;  Kb=
       motor_back_emf_const
6  s=%s;
7  a=Kt/(s*Ra*(s*Jm+fm));
8  a=simple(a);
9  b=s*Kb;
10 c=a/(1+a*b);
11 c=simple(c);
```

```
12  disp(c,"Q(s)/V(s)=");//overall_transfer_function
13  Kt=0.183;Kb=0.183;Ra=4.8;Jm=.004;fm=0.0015;
14  a=Kt/(s*Ra*(s*Jm+fm));
15  b=s*Kb;
16  c=a/(1+a*b);
17  disp(c,"Q(s)/V(s)=");//
        overall_transfer_function_after substituting
        value in above equation
```

Scilab code Exa 5.9.11 Overall Transfer Function of given System

```
 1  //Caption:
        overall_transfer_function_of_positional_control_system
 2  //example 5.9.11
 3  //page 109
 4  syms Ka Ke Kb Kt Jeq feq Ra N1 N2
 5  //where Ka=amplifier_gain; Ke=error_detector_gain;
        Kb=motor_back_emf_const; Kt=motor_torque_const;
        Ra=armature_resistance;  Jeq=moment_of_inertia;
        feq=coeff_of_viscous_friction;
 6  s=%s;
 7  Kt=.0001;Ra=0.2;Jeq=.0001;feq=.0005;Ka=10;Ke=2;Kb
        =0.0001;f=0.1;
 8  a=(Ka*Ke); // in series
 9  b=Kt/(s*Ra*(Jeq*s+feq));
10  c=b/(1+b*s*Kb);
11  d=(a*c);   // in series
12  f=0.1;
13  g=(d*f); // in series
14  h=g/(1+g);
15  disp(h,"C(s)/R(s)=");//
        overall_transfer_function_after substituting
        value in above equation
```

Scilab code Exa 5.9.12 Overall Transfer Function of given System

```
1  //Caption:overall_transfer_function_of_given_system
2  //example 5.9.12
3  //page 111
4  syms Ka Ke Kf Rf Lf Jeq feq  N1 N2
5  //where Ka=amplifier_gain; Ke=error_detector_gain;
       Kf=motor_torque_const; Rf=field_resistance; Lf=
       field_inductance Jeq=moment_of_inertia; feq=
       coeff_of_viscous_friction;
6  s=%s;
7  d=N1/N2;
8  Ka=10; Ke=8; Kf=0.05; Rf=5; Lf=0.25; Jeq=0.05; feq
       =0.075;   d=0.1;
9  a=(Ka*Ke);
10 b=Kf/(Rf+s*Lf);
11 c=1/(s*(Jeq*s+feq));
12 g=(b*c)    //in  series
13 h=(g*a)    //in  series
14 i=(h*d)    //in  series
15 j=i/(1+i);
16 disp(j,"C(s)/R(s)=");
```

Scilab code Exa 5.9.13 Overall Transfer Function of Two Phase ac Motor

```
1  //Caption:
       overall_transfer_function_of_two_phase_ac_motor
2  //example 5.9.13
3  //page 113
4  syms Ka K Ktg Jeq feq  N1 N2 m
5  //where Ka=amplifier_gain; Ktg=tachometer_gain_const
       Jeq=moment_of_inertia; feq=
```

```
        coeff_of_viscous_friction;
 6 s=%s;
 7 //from torque characterstics m and K are determined
 8 Ka=20; K=0.0012; Ktg=0.2; Jeq=0.00015; feq=0.0001;m
        =-0.0003;
 9 a=K/(Jeq*s+(feq-m));
10 b=N1/N2;
11 b=0.1;
12 c=(Ka*a)   // in series
13 d=(c*b)    // in series
14 e=d/(1+Ktg*d);
15 disp(e,"C(s)/R(s)=");//overall_transfer_function
```

Scilab code Exa 5.9.14 Transfer Function Of Motor

```
 1 //Caption:transfer_function_of_motor
 2 //example 5.9.14
 3 //page 114
 4 syms Kt Kb Ra La J
 5 //where Ka=amplifier_gain; Kt=motor_torque_const; Ra
        =armature_resistance; La=armature_inductor; J=
        moment_of_inertia
 6 s=%s;
 7 //since there are two inputs Va(s)and Tl(s).If Va(s)
        is held at fixed value then only effect of Tl(s)
        is considered and Va(s)is taken as Zero.
 8 a=(Kt*Kb)/(Ra+s*La)
 9 b=1/J*s
10 c=b/(1+b*a);
11 disp(-c,"Wm(s)/Tl(s)=");//negative sign indicates
        that an increase in Tl decreases Wm
12 Kb=9.55;Kt=9.55;Ra=0.75;La=0.005;J=50;
13 a=(Kt*Kb)/(Ra+s*La)
14 b=1/(J)*(1/(s))
15 c=b/(1+b*a);
```

```
16  disp(-c,"Wm( s )/ Tl ( s )=") ;// after  putting  values
```

Chapter 6

TIME RESPONSE ANALYSIS OF CONTROL SYSTEMS

Scilab code Exa 6.10.1 time response for step function

```
1  //Caption:time_response_for_step_function
2  //example 6.10.1
3  //page 170
4  // we have defined parallel and series function
       which we are going to use here
5  exec parallel.sce;
6  exec series.sce;
7  s=%s;
8  syms t;
9  a=4/(s*(s+4))
10 b=s+1.2
11 c=s+0.8
12 d=a/(1+a)
13 e=parallel(b,c)
14 f=d/.e;
15 disp(f,"C(s)/R(s)=");//transfer function
16 //since input:r(t)=2, so R(s)=2/s;//
       step_function_of_magnitude_2
17 g=f*(2/s);
```

```
18  disp(g,"C( s )=");
19  [A]=pfss(8/(s*(s+2)*(s+6)))
20  F=ilaplace((8/(s*(s+2)*(s+6))),s,t);
21  disp(F,"
        time_response_for_step_function_of_magnitude_2 , f (
        t )=")
```

Scilab code Exa 6.10.2 Time Response for unit Impulse and Step function

```
 1  //Caption:
        time_response_for_unit_impulse_and_step_function
 2  //example  6.10.2
 3  //page  171
 4  //G( s )=(4*s+1)/4*( s ^2) ;H( s )=1;
 5  clc;
 6  s=%s;
 7  syms t;
 8  G=(4*s+1)/(4*(s^2))//G( s )
 9  b=1;
10  a=G/.(b);
11  disp(a,"C( s )/R( s )=");
12  //for  unit  impulse  response  R( s )=1 ;  so  C( s )=a;
13  disp("for  unit  impulse  response  R( s )=1 ;  so  C( s )=a;"
        )
14  disp(a,"C( s )=");
15  c=ilaplace(a,s,t);
16  disp(c,"c ( t )=");
17  //for  unit  step  response  R( s )=1/s
18  disp('for  unit  step  response  R( s )=1/s ,  so ');
19  d=a*(1/s);
20  disp(d,"C( s )=");
21  e=ilaplace(d,s,t);
22  disp(e,"c ( t )=");
```

Scilab code Exa 6.10.3 Time Response for Unit Step Function

```
1  //Caption: time_response_for_unit_step_function
2  //example 6.10.3
3  //page 172
4  //G(s)=2/(s*(s+3)
5  clc;
6  s=%s;
7  syms t;
8  G=2/(s*(s+3))//G(s)
9  b=1;
10 a=G/.(b);
11 disp(a,"C(s)/R(s)=");
12 disp('for unit step response R(s)=1/s');
13 d=a*(1/s);
14 disp(d,"C(s)=");
15 e=ilaplace(d,s,t);
16 disp(e,"c(t)=");
```

Scilab code Exa 6.10.4 Time Response for Unit Step Function

```
1  //Caption: time_response_for_unit_step_function
2  //example 6.10.4
3  //page 172
4  s=%s;
5  syms t;
6  a=(s+4);
7  b=1/(s*(s+2));
8  c=(a*b); //in series
9  d=0.5;
10 e=c/.d
11 f=1;
```

```
12  g=e/.f;
13  disp(g,"C(s)/R(s)=");
14  disp('for unit step response R(s)=1/s, so');
15  h=g*(1/s);
16  disp(h,"C(s)=");
17  [A]=pfss(h);
18  A(1)=(1/s)
19  F1=ilaplace(A(1),s,t)
20  //A(2)can be written as: A(2)=a1+a2
21  a1=-1/(4*(6+3.5*s+s^2));
22  a2=(-(s+1.75)/(6+3.5*s+s^2));
23  F2=ilaplace(a1,s,t);
24  F3=ilaplace(a2,s,t);
25  //now multiplying by their coefficient
26  F1=(2/3)*F1;
27  F2=(1/6)*F2;
28  F3=(2/3)*F3;
29  //after adding F1,F2 and F3 and simplyfying we get
        time response which is denoted by 'c(t)'
30  disp('c(t)=((2-(%e^(-1.75*t)*(2*cos(1.71*t)-0.29*sin
        (1.71*t))))/3');//time_response
```

Scilab code Exa 6.10.5 Calculate Wn zeta Wd tp Mp

```
1  //Caption:calculate_Wn ,zeta ,Wd,tp ,Mp
2  //example 6.10.5
3  //page 174
4  //given G(s)=20/(s+1)*(s+2)
5  clc;
6  s=%s;
7  G=syslin('c',[20/((s+1)*(s+5))])//G(s):transfer
        function in forward path
8  H=1;//backward path transfer function
9  a=G/.H //closed loop transfer function
10 b=denom(a)
```

```
11  c=coeff(b)
12  //Wn^2=c(1,1)
13  Wn=sqrt(c(1,1))//natural frequency
14  disp(Wn,"natural frequency ,Wn=")
15  //2*zeta*Wn=c(1,2)
16  zeta=c(1,2)/(2*Wn)//damping ratio
17  disp(zeta,"damping ratio ,zeta=")
18  Wd=Wn*sqrt(1-zeta^2)//damped frequency
19  disp(Wd,"damping ratio ,Wd=")
20  Tp=%pi/Wd//peak time
21  disp(Tp,"peak time ,Tp=")
22  Mp=(exp(-(zeta*%pi)/sqrt(1-zeta^2)))*100//max.
        overshoot
23  disp(Mp,"max overshoot ,Mp=")
24  t=(2*%pi)/(Wn*sqrt(1-zeta^2))//period of oscillation
25  disp(t,'time at which first overshoot occurs=')
26  disp(t,"period of oscillation ,t=")
27  ts=4/(zeta*Wn)//settling time
28  disp(ts,"settling time ,ts=")
29  N=Wd/(2*%pi)*ts//no. of oscillations completed
        before reaching steady state
30  disp(N,"no. of oscillations completed before
        reaching steady state ,N=")
```

Scilab code Exa 6.10.6 Time Response

```
1  //Caption:time_response
2  //example 6.10.6
3  //page 174
4  //Kt=torque constant ,J=moment of inertia ,f=coeff. of
        viscous friction
5  clc;
6  syms Kt J f t
7  s=%s;
8  Kt=360,J=10,f=60
```

```
 9  b=1/( J * s ^2+ f * s ) ;
10  G =( Kt * b )    // in   series
11  H =1;// backward  path  transfer  function
12  cl = G /. H ;// closed_loop_transfer_function
13  d = denom ( cl ) /10;// taking  10  common  from  numerator  and
         denominator   for   simplyfying
      closed_loop_transfer_function
14  f = numer ( cl ) /10;
15  CL = f / d ;//
      closed_loop_transfer_function_after_simplifying
16  printf (" overall   transfer   function=  \n ") ;
17  disp ( CL ," C ( s )/ R ( s )=") ;
18  // given  R ( s )=(50*(% pi /180) ) *(1/ s ) ;
19  R =(50*(% pi /180) ) *(1/ s ) ;
20  C = R * CL ;
21  e = coeff ( d )
22  // Wn ^2= e (1 ,1)
23  Wn = sqrt ( e (1 ,1) )// natural   frequency
24  //2* zeta * Wn = c (1 ,2)
25  zeta = e (1 ,2) /(2* Wn )// damping   ratio
26  // c ( t ) : time_response_expression
27  c =(5*% pi /18) *(1 -( exp ( - zeta * Wn * t ) * sin ( Wn * sqrt (1 - zeta
      ^2) * t + atan ( sqrt (1 - zeta ^2) / zeta ) ) )/ sqrt (1 - zeta ^2) )
       ;
28  c = float ( c )
29  disp ( c ," c ( t )=")
```

Scilab code Exa 6.10.7 determine factor by which K should be reduced

```
1  // caption :
      determine_factor_by_which_K_should_be_reduced
2  // example  6.10.7
3  // page  175
4  syms  T  K  // K = forward_path_gain , T = time_constant
5  s =% s ;
```

```
6  G=K/(s*(s*T+1));
7  G=simple(G);
8  printf("the_forward_path_transfer_function: \n" );
9  disp(G,"G(s)=");
10 H=1;//backward path transfer function
11 CL=G/.H;
12 CL=simple(CL);
13 printf("the_overlall_transfer_function: \n" );
14 disp(CL,"C(s)/R(s)=");
15 printf("the_characterstic_equation_is:\n ");
16 disp("s^2+s/T+K/T=0");
17 //from char. eq. we get Wn^2=K/T and 2*zeta*Wn=1/T,
       so
18 Wn=sqrt(K/T);//natural frequency
19 zeta=(1/2)*sqrt(1/K*T)
20 //let K1 be forward path gain when Mp1=60% and zeta1
       be corresponding damping factor
21 syms K1 K2 zeta1 zeta2
22 Mp1=(exp(-(zeta1*%pi)/sqrt(1-zeta1^2)))*100//max.
       overshoot
23 //on solving we get :
24 zeta1=0.158
25 //similarly let K2 be forward path gain when Mp2=20%
       and zeta2 be corresponding damping factor, which
       gives
26 zeta2=0.447
27 //assuming time const. T to be const. we get
28 k=(K1/K2);
29 k=(zeta1/zeta2)^2;
30 disp(k,"K1/K2=");
```

Scilab code Exa 6.10.8 Determine Steady State Speed and Error

```
1  //caption:determine_steady_state_speed_and_error
2  //example 6.10.8
```

```
3  // page 176
4  //Tm=torque  constant ,J=moment  of  inertia , f=coeff .  of
        viscous  friction ,  E=error  detector  gain
5  syms  Tm  J  f  t  E  s
6  Tm=75;J=10;f=5;E=1;
7  a=(Tm*E);
8  b=1/(J*s+f);
9  c=(a*b);
10  H=1;//backward  path  transfer  function
11  CL=c/.H;
12  printf ("the_overlall_transfer_function :  \n" );
13  disp (CL,"Wo(s)/Wr(s)=");
14  //since  Wr(s)=(2*%pi)*(1/s)
15  q=bfloat ((2*%pi),2)
16  Wr=q*(1/s)
17  Wo=CL*Wr;
18  wo=ilaplace (Wo,s,t)
19  printf ("expression_relating_load_speed_and_time :\n")
        ;
20  disp (wo,"wo(t)=");
21  a=Wo*s
22  Woss=limit (a,s,0);//steady  state  speed
23  Woss=dbl (Woss);
24  disp (Woss,"steady  state  speed=");
25  We=Wr*(10*s+5)/(10*s+80);
26  Wess=limit (s*We,s,0);//steady  state  error
27  Wess=dbl (Wess);
28  disp (Wess,"steady  state  error=");
```

Scilab code Exa 6.10.9 determine J f K

```
1  //caption :determine_J ,f ,K
2  //example  6.10.9
3  //page  178
4  //J=moment  of  inertia , f=C,K=forward  path  gain ,Wn=
```

```
        natural frequency, zeta=damping ratio
 5  syms J f K s zeta Wn
 6  CL=1/(J*s^2+f*s+K);
 7  printf("given:transfer function is:\n");
 8  disp(CL,"theta(s)/T(s)=");
 9  T=10*(1/s);
10  theta=T*CL;
11  theta_ss=limit(s*theta,s,0)//steady_state_value
12  printf("given:theta_ss=0.5\n so K=10/0.5");
13  theta_ss=0.5;
14  K=10/theta_ss;
15  disp(K,"forward path gain,K=");
16  Mp=0.06;//max.peak overshoot (given)---------(1)
17  Mp=exp((-zeta*%pi)/sqrt(1-zeta^2))//----------(2)
18  //from eq. (1) and (2), we get
19  zeta=0.66;
20  tp=%pi/(Wn*(sqrt(1-zeta^2)));//--------(3)
21  tp=1//(given)
22  Wn=%pi/(tp*sqrt(1-zeta^2));
23  //also Wn=sqrt(K/J);
24  J=K/Wn^2;
25  //also 2*zeta*Wn=f/J
26  f=J*2*zeta*Wn;
27  disp(J,"moment of inertia,J=");
28  disp(f,"moment of inertia,f=");
```

Scilab code Exa 6.10.10 Determine Transfer Function Wn zeta

```
 1  //caption:determine_transfer_function,Wn,zeta
 2  //example 6.10.10
 3  //page 179
 4  //J=moment of inertia,f=C,Ke=error detector gain,Wn=
        natural frequency, zeta=damping ratio,Km=torque
        constant
 5  syms J f s
```

```
 6  Ke=5.73;
 7  Km=0.045;
 8  n=sym('N1/N2')
 9  n=subs(n,'N1/N2',1/10);
10  J=0.25*float(n^2);//referred to motor side
11  f=1*float(n^2);//referred to motor side
12  //from the block diagram given in fig 6.10.6 on page
       179,
13  a=(Ke*Km)
14  b=1/(J*s^2+f*s)
15  c=(b*float(a))
16  G=(n*c);
17  G=simple(G)
18  H=1;
19  d=(1+G*H);
20  d=simple(d);
21  CL=G/d;
22  CL=simple(CL)
23  disp(CL,"C(s)/R(s)=");
24  e=poly([328800 127516 31879],'s','coeff')
25  printf("the characterstics eq. is:\n");
26  disp(e);
27  f=coeff(e)
28  Wn=sqrt((f(1,1)/f(1,3)))//natural_frequency
29  zeta=((f(1,2)/f(1,3))/(2*Wn))//damping ratio
30  //part(b)
31  syms Td
32  g=1+s*Td
33  h=(g*G)
34  i=1+h*H
35  i=simple(i);
36  CL2=(h/i)
37  CL2=simple(CL2);
38  disp(CL2,"C(s)/R(s)=");
39  poly(0,'s');
40  l=s^2+(10.32*Td+4)*s+10.32;
41  printf("on simplyfing the characterstics eq. is:");
42  disp(float(l));
```

```
43  Wn1=sqrt(10.32)//natural_frequency
44  //2*zeta1*Wn1=10.32*Td+4
45  zeta1=1;//as damping is critical
46  Td=(2*zeta1*Wn1-4)/10.32;
47  disp(Td,"time const. of advanced phase circuit ,Td=")
        ;
```

Scilab code Exa 6.10.11 Determine Characterstics eq and Steady State Error

```
 1  //caption:
        determine_characterstics_eq_and_steady_state_error

 2  //example 6.10.11
 3  //page 181
 4  //J=moment of inertia ,f=C,K=controller gain ,Wn=
        natural frequency , zeta=damping ratio
 5  syms f J K Kt
 6  s=%s;
 7  A=sym((1/(J*s^2+f*s)));
 8  J=250;
 9  K=8*10^4;
10  B=eval(A)
11  a=(K*B);
12  H1=s*Kt;
13  b=(1+a*H1);
14  b=simple(b);
15  CL1=a/b;
16  CL1=simple(CL1);
17  H=1;
18  c=1+CL1*H;
19  c=simple(c);
20  CL=CL1/c
21  CL=simple(CL);
22  disp(CL,"C(s)/R(s)=");
```

```scilab
23  Wn=sqrt(80000/250)//natural frequency
24  //2*zeta*Wn=(80000*Kt+f)/250
25  zeta=1;//for critical damping
26  d=2*zeta*Wn;
27  v=[320 d 1];
28  CH=poly(v,'s','coeff');
29  r=float(5*2*%pi/60);
30  //steady state error for unit ramp input is:Ess= (2*
        zeta/Wn)
31  Ess=(2*zeta/Wn)*r;
32  disp(Ess,"steady_state_error=");
```

Scilab code Exa 6.10.12 determine WnWd zeta and steady state error

```scilab
 1  //caption:determine_Wn ,Wd,
        zeta_and_steady_state_error
 2  //example 6.10.12
 3  //page 182
 4  clc;
 5  s=%s;
 6  G=sym('25/(s*(s+5))');
 7  G=simple(G);
 8  H=1;
 9  CL=G/.H;
10  CL=simple(CL);
11  disp(CL,"C(s)/R(s)=");
12  printf("the char. eq is:")
13  disp("s^2+5*s+25")
14  Wn=sqrt(25)//natural_frequency
15  //2*zeta*Wn=5
16  zeta=5/(2*Wn);//damping ratio
17  d=zeta*Wn;//damping_factor
18  z=sqrt(1-zeta^2);
19  Wd=Wn*z;//damped_frequency_of_oscillation
20  Mp=exp((-zeta*%pi)/z)*100;//%_max.peak_overshoot
```

```
21 //steady state error for unit ramp input is:Ess= (2*
       zeta/Wn)
22 Ess=(2*zeta/Wn);//steady state error
23 disp(" part(a):")
24 disp(Wn,"natural_frequency,Wn=");
25 disp(zeta,"damping ratio,zeta=");
26 disp(Wd,"damped_frequency_of_oscillation,Wd=");
27 disp(Mp,"%_max.peak_overshoot,Mp=");
28 disp(Ess,"steady state error,Ess=");
29 //if damping ratio is increased from 0.5 to 0.75 by
       incorporating tachometer feedback
30 zeta=0.75;
31 H1=sym('s*Kt')//tachometer_feedback
32 CL1=G/(1+G*H1);
33 CL1=simple(CL1);
34 CL2=CL1/(1+H*CL1);
35 CL2=simple(CL2);
36 disp(CL2,"C(s)/R(s)=");
37 Wn=sqrt(25);
38 //2*zeta*Wn=25*Kt+5;
39 Kt=(2*zeta*Wn-5)/25;//tachometer_gain
40 Mp1=exp((-zeta*%pi)/sqrt(1-zeta^2))*100;//
       %_peak_overshoot
41 disp("After applying tachometer feedback:")
42 disp(Kt,"tachometer_gain,Kt=");
43 disp(Mp1,"%_peak_overshoot,Mp1=");
```

Scilab code Exa 6.10.13 determine K ts tp and Mp

```
1 //caption:determine_K,ts,tp,Mp
2 //example 6.10.13
3 //page 184
4 clc;
5 syms K;
6 s=%s;
```

```
 7  G=sym('K/(s*(s+6))');
 8  H=0.2;
 9  CL=G/(1+G*H);
10  CL=simple(CL);
11  Wn=sqrt(K/5)
12  zeta=0.7//(given) damping ratio
13  //2*zeta*Wn=6
14  Wn=6/(2*zeta);
15  K=Wn^2*5;
16  ts=4/(zeta*Wn);//settling_time
17  Mp=exp((-zeta*%pi)/sqrt(1-zeta^2))*100;//
        %_peak_overshoot
18  tp=%pi/(Wn*sqrt(1-zeta^2));//peak_time
19  disp(Wn,"natural_frequency,Wn=");
20  disp(Mp,"%_max.peak_overshoot,Mp=");
21  disp(ts,"settling_time,ts=");
22  disp(tp,"peak_time,tp=");
```

Scilab code Exa 6.10.14 determine Mp Ess and steady state value

```
 1  //caption:determine_Mp,Ess_and_steady_state_value
 2  //example 6.10.14
 3  //page 185
 4  //there are two inputs: R(s) is reference input and
        Tl(s)is load torque
 5  clc;
 6  s=%s;
 7  A=sym('1/(0.15*s^2+0.9*s)');
 8  K=6;
 9  //while considering R(s),we take Tl(s)=0
10  G=(A*K);  //in series
11  H=1;
12  CL=G/(1+G*H);
13  CL=simple(CL);
14  disp(CL,"C(s)/R(s)=");
```

```
15  Wn=sqrt(40);//natural_frequency
16  //2*zeta*Wn=6
17  zeta=6/(2*Wn);//damping ratio
18  Mp=exp((-zeta*%pi)/sqrt(1-zeta^2))*100;//
        %_peak_overshoot
19  //steady state error for unit ramp input is:Ess= (2*
        zeta/Wn)
20  Ess=(2*zeta/Wn);//steady state error
21  disp(Mp,"%_peak_overshoot=");
22  disp(Ess,"steady state error=");
23  printf("Now considering effect of Tl(s),put R(s)=0\n
        ");
24  H=6;
25  CL1=A/(1+A*H);
26  CL1=simple(CL1);
27  disp(CL1,"C(s)/-Tl(s)=");//-ve sign indicates output
        position lags behind the input
28  Tl=1/s;//given
29  C=-Tl*CL1;
30  Css=limit(s*C,s,0);
31  disp(Css,"steady state value of output=");
```

Scilab code Exa 6.10.16 determine Wn zeta and M

```
1   //caption:determine_Wn,zeta_and_Mp
2   //example 6.10.16
3   //page 187
4   s=%s;
5   num=1;
6   den=sym('s*(1+0.5*s)*(1+0.2*s)');
7   c=num/den;
8   c=simple(c);
9   disp(c,"C(s)/E(s)=");
10  G=c;
11  H=1;
```

```
12  CL=G/(1+G*H);
13  CL=simple(CL);
14  disp(CL,"C(s)/R(s)=");
15  A=pfss((10/(s^3+7*s^2+10*s+10)));
16  d=denom(A(1));
17  b=coeff(denom(A(1)))
18  printf("for oscillatory roots:")
19  Wn=sqrt(b(1,1));//natural_frequency
20  //2*zeta*Wn=1.5;
21  zeta=1.5/(2*Wn);//damping ratio
22  Mp=exp((-zeta*%pi)/sqrt(1-zeta^2))*100;//
        %_peak_overshoot
23  disp(Wn,"natural_frequency=");
24  disp(zeta,"damping ratio=");
25  disp(Mp,"%_peak_overshoot=");
```

Scilab code Exa 6.10.17 determine Kp Kv and Ka

```
1  //caption:determine_Kp_Kv_Ka
2  //example 6.10.17
3  //page 188
4  s=%s;
5  syms t;
6  num=sym('2*(s^2+3*s+20)');
7  den=sym('s*(s+2)*(s^2+4*s+10)');
8  GH=num/den;
9  GH=simple(GH);
10  disp(GH,"G(s)H(s)=");
11  input1=5;
12  Kp=limit(GH,s,0);//static positional error
        coefficient
13  Ess=5*(1/(1+Kp));//steady state error
14  e=(1/(%inf+1));
15  e=0;
16  Ess=e;
```

```
17  disp(Kp,"static positional error coefficient=");
18  disp(Ess,"steady state error=");
19  input2=4*t;
20  Kv=limit(s*GH,s,0);//static velocity error
       coefficient
21  Ess=(1/Kv)*4;//steady state error
22  disp(Kv,"static velocity error coefficient=");
23  disp(Ess,"steady state error=");
24  input3=(4*t^2)/2;
25  Ka=limit(s^2*GH,s,0);//static acceleration error
       coefficient
26  Ess=(1/Ka)*4;//steady state error
27  disp(Ka,"static acceleration error coefficient=");
28  disp("steady state error=");
29  disp("infinity")
```

Scilab code Exa 6.10.18 determine Kp Kv and Ka

```
1  //caption:determine_Kp_Kv_Ka
2  //example 6.10.17
3  //page 188
4  s=%s;
5  syms t K zeta Wn;
6  num=K;
7  den=sym('s*(s^2+2*zeta*Wn*s+Wn^2)');
8  G=num/den;
9  G=simple(G);
10 disp(G,"G(s)=");
11 Kp=limit(G,s,0);//static positional error
       coefficient
12 disp(Kp,"static positional error coefficient ,Kp=");
13 Kv=limit(s*G,s,0);//static velocity error
       coefficient
14 disp(Kv,"static velocity error coefficient ,Kv=");
15 Ka=limit(s^2*G,s,0);//static acceleration error
```

```
        coefficient
16  disp(Ka,"static  acceleration  error  coefficient ,Ka=")
        ;
17  printf("for(ii):");
18  num=sym('100*(s+2)*(s+40)');
19  den=sym('s^3*(s^2+4*s+200)');
20  GH=num/den;
21  GH=simple(GH);
22  disp(GH,"G(s)H(s)=");
23  Kp=limit(GH,s,0);//static  positional  error
        coefficient
24  disp(Kp,"static  positional  error  coefficient ,Kp=");
25  Kv=limit(s*GH,s,0);//static  velocity  error
        coefficient
26  disp(Kv,"static  velocity  error  coefficient ,Kv=");
27  Ka=limit(s^2*GH,s,0);//static  acceleration  error
        coefficient
28  disp(Ka,"static  acceleration  error  coefficient ,Ka=")
        ;
```

Scilab code Exa 6.10.19 determine steady state error

```
1  //caption:determine_steady_state_error
2  //example  6.10.19
3  //page  189
4  s=%s;
5  syms K bta alpha G
6  num=sym('K*s+bta');
7  den=sym('s^2+alpha*s+bta');
8  CL=num/den;
9  disp(CL,"C(s)/R(s)=");//----(1)
10 H=1;
11 //also
12 cl=G/(1+G*H);
13 disp(cl,"also ,C(s)/R(s)=");//------(2)
```

```
14  //from eq. (1) and (2), we get
15  G=num/(s^2+s*(alpha-K));
16  disp(G,"G(s)=");
17  B=1/(1+G);
18  B=simple(B);
19  disp(B,"E(s)/R(s)=");
20  R=1/s^2;
21  E=B*R;
22  E=simple(E);
23  Ess=limit(s*E,s,0);
24  disp(Ess,"steady state error=");
```

Scilab code Exa 6.10.20 determine steady state error and error coefficient

```
1  //caption:
       determine_steady_state_error_and_error_coefficient

2  //example 6.10.20
3  //page 189
4  s=%s;
5  syms t a0 a1 a2;
6  r=a0+a1*t+(a2/2)*t^2;
7  //since for 'r' only upto 2nd order derivative is
       non zero, so only coeff. C0 C1 C2 exist
8  num=20;
9  den=sym('(s*(s+2))');
10 G=num/den;
11 disp(G,"G(s)=");
12 A=1/(1+G);
13 A=simple(A);
14 disp(A,"E(s)/R(s)=");
15 C0=limit(A,s,0);
16 B=sym('((20)/(s^2+2*s+20))');//on simplyfying A=1-B
17 d=diff(-B,s);
18 C1=limit(d,s,0);
```

74

```
19  d1=diff(-B,s,2);
20  C2=limit(d1,s,0);
21  r1=diff(r,t);
22  r2=diff(r,t,2);
23  e=(C0*r)+(C1*r1)+(C2*r2)/2;
24  disp(C0,"C0=");
25  disp(float(C1),"C1=");
26  disp(float(C2),"C2=");
27  disp(float(e),"steady_state_error ,e=");
```

Scilab code Exa 6.10.21 determine steady state error and error coefficient

```
1  //caption:
      determine_steady_state_error_and_error_coefficient
2  //example  6.10.21
3  //page  191
4  s=%s;
5  syms t;
6  r=2+3*t+2*t^3;
7  //since for 'r' only upto 2nd order derivative is
      non zero, so only coeff. C0 C1 C2 exist
8  G=sym('1/(s*(s+2))');
9  A=1/(1+G);
10  A=simple(A);
11  disp(A,"E(s)/R(s)=");
12  C0=limit(A,s,0);
13  B=sym('((1)/(s^2+2*s+1))');//on simplyfying A=1-B
14  d=diff(-B,s);
15  C1=limit(d,s,0);
16  d1=diff(-B,s,2);
17  C2=limit(d1,s,0);
18  r1=diff(r,t);
19  r2=diff(r,t,2);
20  d2=diff(-B,s,3)
```

```
21  C3=limit(d2,s,0)
22  e=(C0*r)+(C1*r1)+(C2*r2)/2;
23  disp(float(e),"steady_state_error ,e=");
24  disp(C0,"C0=");
25  disp(float(C1),"C1=");
26  disp(float(C2),"C2=");
27  disp(float(C3),"C3=")
```

Scilab code Exa 6.10.22 determine voltage Er and change in terminal-voltage

```
 1  //caption : determine_reference_voltage ,
        Er_and_%change_in_terminal_voltage
 2  //example 6.10.22
 3  //page 191
 4  clc ;
 5  syms G H
 6  s=%s ;
 7  A=G/.H ;
 8  disp(A,"E0/Er=");
 9  G=200;   //gain
10  H=0.1;
11  B=eval(A);
12  disp(B,"E0/Er=");
13  E0=250;
14  Er=(1/B)*E0;
15  disp("for_closed_loop_system :");
16  disp(float(Er),"reference_voltage ,Er=");
17  disp("for_open_loop_system :");
18  disp(float(E0/G),"Er=E0/G=");
19
20  disp("part (b)")
21  disp("for closed loop system :")
22  disp("as the forward path gain is reduced by 10%,
        the new value of gain ,G is 180 ")
```

76

```
23  G=180    // gain
24  pcG=10   // percentage  change  in  G
25  S=1/(1+G*H)
26  disp(S,"sensitivity ,S=");
27  disp("since  S=1/19,       (given)")
28  pcM=S*pcG   // percentage  change  in  overall  gain ,M
29  pcEo=pcM
30  disp(pcM,"percentage  change  in  overall  gain ,M=")
31  disp(pcEo,"percentage  change  in  terminal  voltage ,Eo=
        ")
32  disp("for  open  loop  system :")
33  pcEo=(25/250)*100
34  disp(pcEo,"percentage  change  in  terminal  voltage ,Eo=
        ")
```

Scilab code Exa 6.10.23 determine sensitivity wrt K and H

```
 1  // caption : determine_sensitivity_w . r . t_K_and_H
 2  // example  6.10.23
 3  // page  192
 4  syms  Wn  zeta  K  H;
 5  s=%s ;
 6  A=sym ( 'Wn^2/( s^2+2* zeta *Wn* s ) ');
 7  B=(A*K);
 8  CL=B/(1+B*H);
 9  CL=simple(CL);
10  disp(CL," overall_transfer_function :M( s )=");
11  disp("sensitivity_w . r . t_K :");
12  a=diff(CL,K);
13  b=K/CL;
14  b=simple(b);
15  Sk=a*b;
16  Sk=simple(Sk);
17  disp(Sk," sensitivity_w . r . t_K , Sk=");
18  disp("sensitivity_w . r . t_H :");
```

```
19  c=diff(CL,H);
20  d=H/CL;
21  d=simple(d);
22  Sh=c*d;
23  Sh=simple(Sh);
24  disp(Sh,"sensitivity_w.r.t_H,Sh=");
```

Chapter 7

STABILITY ANALYSIS OF CONTROL SYSTEMS

Scilab code Exa 1.0 root locus

```
1 // caption : root_locus
2 // example 1
3 // page 291
4 s=%s ;
5 g=s /( s +1) ;
6 G=syslin ( 'c ' ,g)
7 evans (g ,200)
```

Scilab code Exa 2.0 root locus

```
1 // caption : root_locus
2 // example 2
```
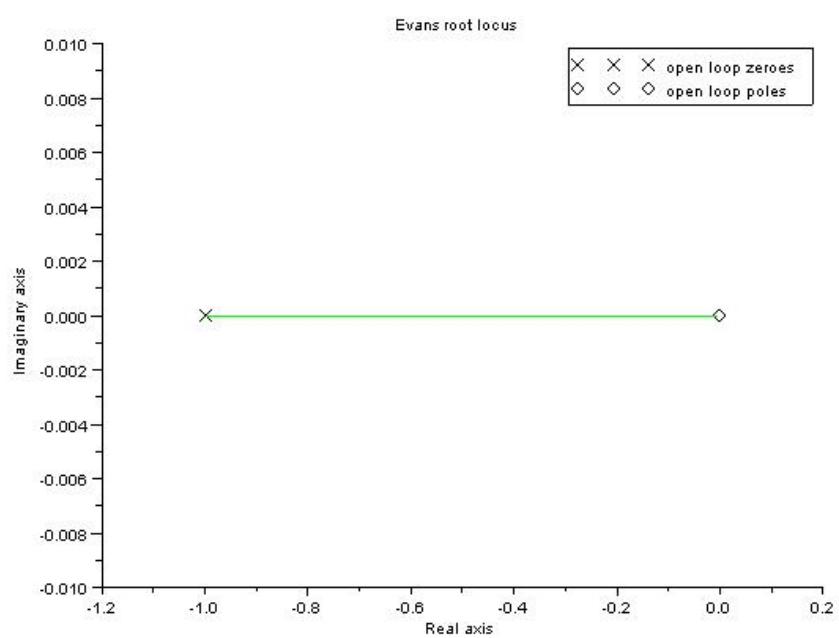
Figure 7.1: root locus
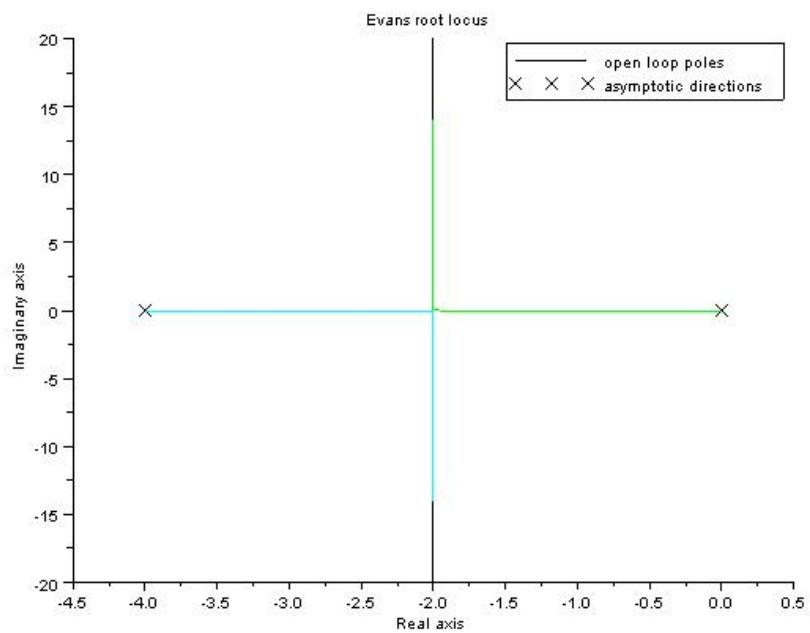
Figure 7.2: root locus

81
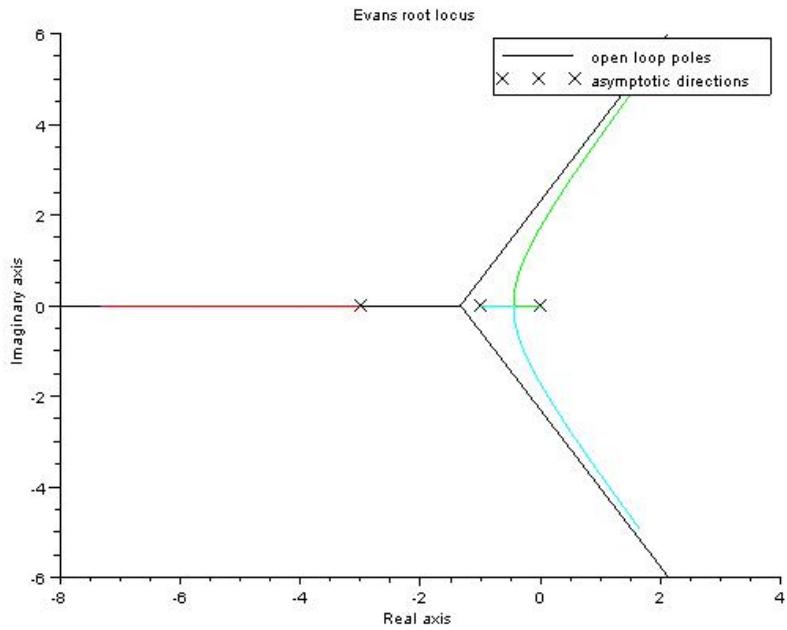
Figure 7.3: root locus

```
3 // page 291
4 s=%s;
5 g=1/(s*(s+4));
6 G=syslin('c',g)
7 evans(g,200)
```

Scilab code Exa 3.0 root locus

```
1 // caption : root_locus
2 // example 3
3 // page 292
```

```
4  s=%s;
5  g=1/(s*(s+1)*(s+3));
6  G=syslin('c',g)
7  evans(g,200)
```

Scilab code Exa 7.5.1 stability using Routh hurwitz criterion

```
1  //caption:stability_using_Routh−hurwitz_criterion
2  //example 7.5.1
3  //page 202
4  clc;
5  s=%s;
6  A=s^3+4.5*s^2+3.5*s+1.5;
7  b=coeff(A)
8  n=length(b)
9  B=routh_t(A)
10 disp(B,"routh table:");
11 c=0;
12 for(i=1:n)
13    if(B(i,1)<0)
14       c=c+1;
15    end
16 end
17 if(c>=1)
18    disp("system is unstable")
19 else("system is stable")
20 end
```

Scilab code Exa 7.5.2 stability using Routh hurwitz criterion

```
1  //caption:stability_using_Routh−hurwitz_criterion
2  //example 7.5.2
3  //page 202
```

```
4  s=%s;
5  A=s^3+4*10^2*s^2+5*10^4*s+2*10^6;
6  b=coeff(A)
7  n=length(b)
8  B=routh_t(A)
9  disp(B,"routh table:");
10 c=0;
11 for(i=1:n)
12    if(B(i,1)<0)
13       c=c+1;
14    end
15 end
16 if(c>=1)
17    printf("\n system is unstable")
18 else("system is stable")
19 end
```

Scilab code Exa 7.5.3 stability using Routh hurwitz criterion

```
1  //caption:stability_using_Routh−hurwitz_criterion
2  //example 7.5.3
3  //page 203
4  s=%s;
5  A=s^5+4*1.5*s^4+4*s^3+4*s^2+5*s+10;
6  C=2*s+5;
7  CL=A/C;
8  disp(CL,"C(s)/R(s)=");
9  disp('=0',A,"characteristics eq is:")
10 b=coeff(A)
11 n=length(b)
12 B=routh_t(A)
13 disp(B,"routh table:");
14 c=0;
15 r=1;
16 for(i=1:n)
```

```
17    if(B(i,1)<0)
18       c=c+1;
19       if(i==n & B(n,1)<0)
20          r=r;
21       else
22          r=r+1;
23       end
24    end
25 end
26 if(c>=1)
27    printf("system is unstable\n");
28 else("system is stable " );
29 end
30 mprintf('no. of roots with positive real parts=%d',r
      );
```

Scilab code Exa 7.5.4 stability using Routh hurwitz criterion

```
1 //caption:stability_using_Routh-hurwitz_criterion
2 //example 7.5.4
3 //page 203
4 ieee (2)
5 s=%s;
6 A=s^5+s^4+2*s^3+2*s^2+11*s+10;
7 b=coeff(A);
8 n=length(b);
9 B=routh_t(A);
10 K=B;
11 c=0;
12 syms eps;
13 x=limit(eps,eps,0);
14 y=limit(((-1+2*eps)/eps),eps,0);
15 z=limit(((-1+2*eps-10*eps^2)/(-1+2*eps)),eps,0);
16 //after putting the limit we get:
17 K(3,1)=0;
```

```
18  K(4,1)=-%inf;
19  K(5,1)=1;
20  disp(K,"routh_table:")
21  printf("There are two sign changes of first column ,
        hence the system is unstable \n" )
```

Scilab code Exa 7.5.5 stability using Routh hurwitz criterion

```
1  //caption:stability_using_Routh-hurwitz_criterion
2  //example 7.5.5
3  //page 204
4  s=%s;
5  A=s^6+s^5+5*s^4+3*s^3+2*s^2-4*s-8;
6  b=coeff(A)
7  n=length(b)
8  routh=[b([7,5,3,1]);b([6,4,2]),0];
9
10 c=[routh(1,1),routh(1,3);routh(2,1),routh(2,3)]
11 d=[routh(1,1),routh(1,4);routh(2,1),routh(2,4)]
12 routh=[routh;-det(routh(1:2,1:2))/routh(2,1),-det(c)
       /routh(2,1),-det(d)/routh(2,1),0];
13 routh1=routh;
14 e=[routh(2,1),routh(2,2);routh(3,1),routh(3,2)]
15 f=[routh(2,1),routh(2,3);routh(3,1),routh(3,3)]
16 routh=[routh;-det(e)/routh(3,1),-det(f)/routh(3,1)
       ,0,0];
17 disp("since all elements of fourth row are zero , so
       we make auxiliary equation")
18 A=sym('2*s^4+6*s^2-8')
19 B=diff(A,s)
20 routh=[routh1;8,12,0,0]
21 g=[routh(3,1),routh(3,3);routh(4,1),routh(4,3)]
22 routh=[routh;-det(routh(3:4,1:2))/routh(4,1),-det(g)
       /routh(4,1),0,0];
23 routh=[routh;-det(routh(4:5,1:2))/routh(5,1),0,0,0];
```

```
24  routh =[ routh ;-det ( routh (5:6 ,1:2))/ routh (6 ,1) ,0 ,0 ,0]
25  disp ( routh ," routh  table :")
26  c =0;
27  r =1;
28  for (i =1: n)
29     if ( routh (i ,1) <0)
30        c = c +1;
31        if (i == n &  routh (n ,1) <0)
32           r = r ;
33        else
34           r = r +1;
35        end
36     end
37  end
38  if (c >=1)
39     printf (" system  is  unstable \n")
40  else (" system  is  stable  " )
41  end
42  mprintf ( 'no.  of  roots  with  positive  real  parts=%d ',r
        );
```

Scilab code Exa 7.5.6 stability using Routh hurwitz criterion

```
1  // caption : stability_using_Routh −hurwitz_criterion
2  // example  7.5.6
3  // page  206
4  s =%s ;
5  syms  T
6  num = exp (-s *T);
7  den = s *( s +2);
8  G = num / den ;
9  H =1;
10  a =(1+ G *H);
11  CL = G /a;
12  CL = simple ( CL );
```

```
13  CH=s^2+2*s+exp(-s*T);
14  //exp(-s*T)=1-sT+(sT)^2/2+....
15  CH=s^2+(2-T)*s+1;
16  disp('=0',CH,"characterstics_eq ,CH=")
17  c0=coeffs(CH,'s',0);
18  c1=coeffs(CH,'s',1);
19  c2=coeffs(CH,'s',2);
20  b=[c0 c1 c2]
21  n=3;
22  routh=[b([3,1]);b([2]),0];
23  routh=[routh;simple(-det(routh)/routh(2,1)),0]
24  disp(routh,"routh=");
25  disp("for given system to be stable:");
26  disp("2-T>0");
27  disp("which gives:");
28  disp("T<2");
```

Scilab code Exa 7.5.7.a stability using Routh hurwitz criterion

```
1  //caption:stability_using_Routh-hurwitz_criterion
2  //example 7.5.7.(a)
3  //page 207
4  s=%s;
5  syms T K
6  P=s*(s*(s+10)+T);
7  CH=sym('s^3+10*s^2+T*s+K');
8  disp('=0',CH,"characterstics_eq ,CH=")
9  c0=coeffs(CH,'s',0);
10 c1=coeffs(CH,'s',1);
11 c2=coeffs(CH,'s',2);
12 c3=coeffs(CH,'s',3);
13 b=[c0 c1 c2 c3]
14 n=4;
15 routh=[b([4,2]);b([3,1])];
16 routh=[routh;-det(routh)/routh(2,1),0]
```

```
17  t = routh (2:3 ,1:2)
18  routh =[ routh ; - det ( t )/ t (2 ,1) ,0]
19  disp ( routh ," routh =")
20  disp (" for given system to be stable :");
21  disp (" ((10* T–K )/10) >0 and K >0") ;
22  disp (" which gives :") ;
23  disp ("0< K <10T ") ;
```

Scilab code Exa 7.5.7.b value of K of characterstics equation

```
1  // caption : value_of_K_of_characterstics_eq
2  // example 7.5.7.( b)
3  // page 207
4  s =% s ;
5  syms K s1 ;
6  CH = s ^3+10* s ^2+18* s + K
7  s = s1 -1;
8  CH = eval ( CH );
9  CH = simple ( CH );
10  disp ( '=0 ', CH ," characterstics_eq ,CH =") ;
11  c0 = coeffs ( CH , 's1 ' ,0) ;
12  c1 = coeffs ( CH , 's1 ' ,1) ;
13  c2 = coeffs ( CH , 's1 ' ,2) ;
14  c3 = coeffs ( CH , 's1 ' ,3) ;
15  b =[ c0 c1 c2 c3 ]
16  n =4;
17  routh =[ b ([4 ,2]) ; b ([3 ,1]) ];
18  routh =[ routh ; - det ( routh )/ routh (2 ,1) ,0]
19  t = routh (2:3 ,1:2)
20  routh =[ routh ; - det ( t )/ t (2 ,1) ,0]
21  disp ( routh ," routh =")
22  disp (" for given system to be stable :") ;
23  disp (" (( -( K –16) )/7) >0 and K –9 >0") ;
24  disp (" which gives :") ;
25  disp ("9< K <16") ;
```

Scilab code Exa 7.5.8 value of K in terms of T1 and T2

```
1  //caption:value_of_K_in_terms_of_T1_and_T2
2  //example 7.5.8
3  //page 207
4  s=%s;
5  syms K T1 T2;
6  m=s*(s*T1+1)*(s*T2+1)
7  G=K/m;
8  CH=1+G;
9  disp("on simplyfying CH")
10 CH=m+K;
11 CH=simple(CH);
12 disp('=0',CH,"characterstics_eq ,CH=");
13 c0=coeffs(CH,'s',0);
14 c1=coeffs(CH,'s',1);
15 c2=coeffs(CH,'s',2);
16 c3=coeffs(CH,'s',3);
17 b=[c0 c1 c2 c3]
18 n=4;
19 routh=[b([4,2]);b([3,1])];
20 routh=[routh;-det(routh)/routh(2,1),0]
21 t=routh(2:3,1:2)
22 routh=[routh;-det(t)/t(2,1),0]
23 disp(routh,"routh=")
24 disp("for given system to be stable:");
25 disp("K>0 and (-(K*T1*T2-T2-T1)/(T2+T1))>0");
26 disp("which gives:");
27 disp("0<K<((1/T1)+(1/T2))");
```
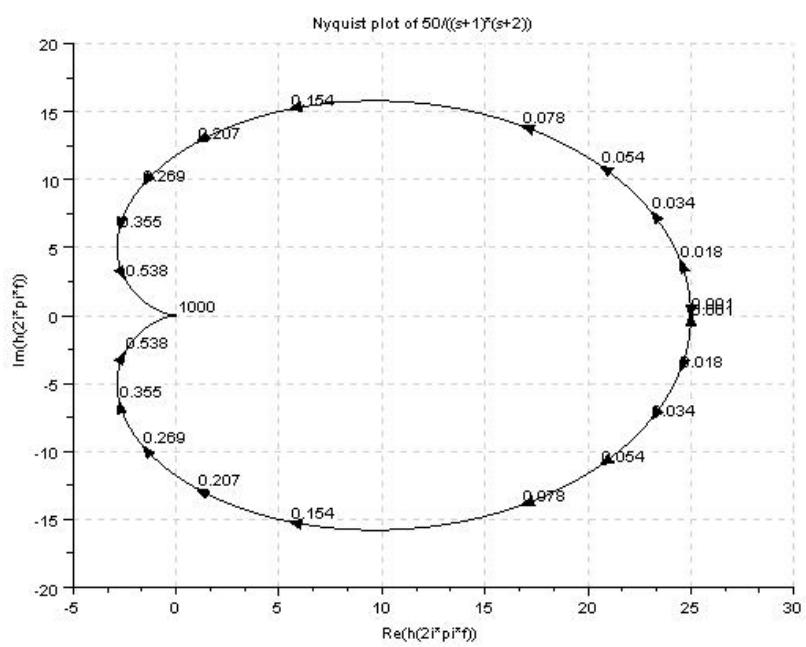
Figure 7.4: stability using Nyquist criterion

Scilab code Exa 7.17.1 stability using Nyquist criterion

```
1  //caption:stability_using_Nyquist_criterion
2  //example 7.17.1
3  //page 236
4  clf;
5  s=%s;
6  s1=-s;
7  g=50/((s+1)*(s+2));
8  g1=50/((s1+1)*(s1+2));
9  GH=syslin('c',g)
10 GH1=syslin('c',g1)
11 nyquist(GH);
12 nyquist(GH1);
13 mtlb_axis([-5 30 -20 20]);
14 xtitle('Nyquist plot of 50/((s+1)*(s+2))')
15 figure;
16 show_margins(GH,'nyquist')
17 disp("since the point(-1+%i0) is not encircled by
        Nyquist plot ,so N=0 and P=0(given)")
18 N=0;//no. of encirclement of -1+%i0 by G(s)H(s) plot
19 P=0;//no. of poles of G(s)H(s) with positive real
        part
20 Z=P-N;//np.of zeros of 1+G(s)H(s)=0 with positive
        real part
21 disp(Z,"Z=")
22 disp("as Z=0,there are no roots of closed loop
        characterstics eq having positive real part,
        hence system is stable.")
```

Scilab code Exa 7.17.2.i stability using Nyquist criterion
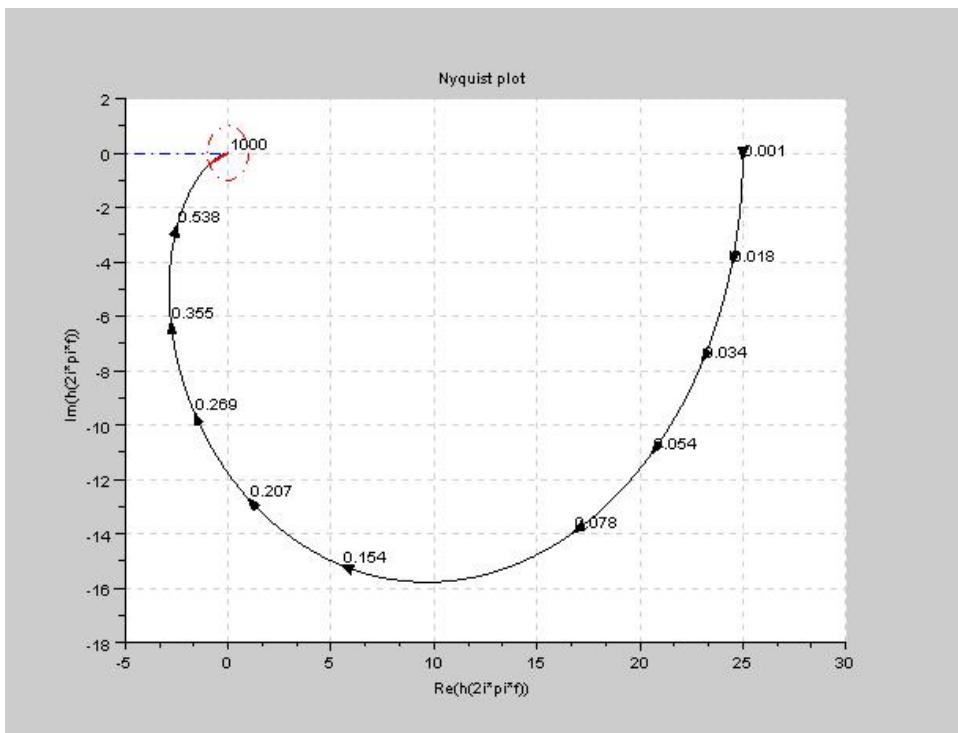
Figure 7.5: stability using Nyquist criterion
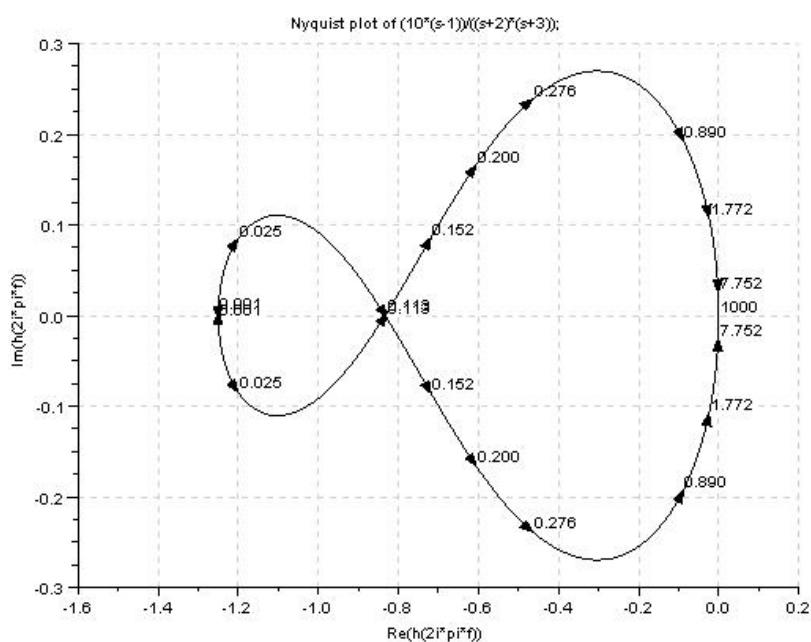
Figure 7.6: stability using Nyquist criterion

```scilab
1  //caption:stability_using_Nyquist_criterion
2  //example 7.17.2
3  //page 237
4  clf();
5  s=%s;
6  s1=-s;
7  disp("for K=1.25")
8  g=(1.25*(s+1))/((s+0.5)*(s-2));
9  g1=(1.25*(s1+1))/((s1+0.5)*(s1-2));
10 GH=syslin('c',g);
11 GH1=syslin('c',g1);
12 nyquist(GH);
13 nyquist(GH1);
14 mtlb_axis([-1.5 0.2 -0.3 0.3]);
15 xtitle('Nyquist plot of (10*(s-1))/((s+2)*(s+3));')
16 figure;
17 show_margins(GH,'nyquist')
18 disp("since the point(-1+%i0) is  encircled
       clockwise by Nyquist plot ,so N=-1 and P=1(given)
       ")
19 N=-1;//no. of encirclement of -1+%i0 by G(s)H(s)
       plot anticlockwise
20 P=1;//no. of poles of G(s)H(s) with positive real
       part
21 Z=P-N;//np.of zeros of 1+G(s)H(s)=0 with positive
       real part
22 disp(Z,"Z=")
23 disp("as Z=2,there are two roots of closed loop
       characterstics eq having positive real part,
       hence system is unstable.")
```
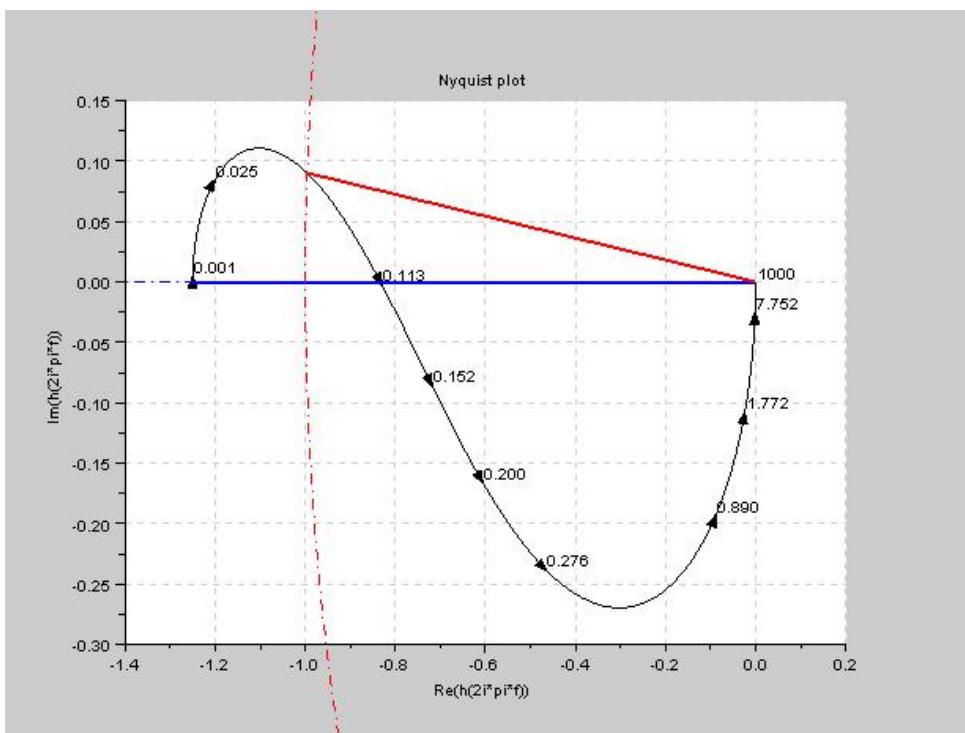
Figure 7.7: stability using Nyquist criterion
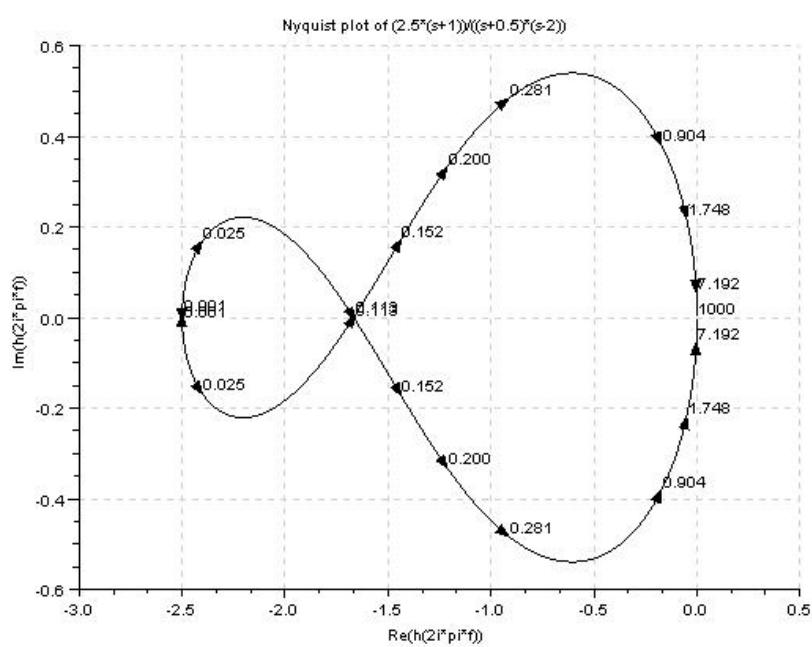
Figure 7.8: stability using Nyquist criterion

Scilab code Exa 7.17.2.ii stability using Nyquist criterion

```
1  //caption:stability_using_Nyquist_criterion
2  //example 7.17.2_ii
3  //page 237
4  disp("for K=2.5")
5  g=(2.5*(s+1))/((s+0.5)*(s-2));
6  g1=(2.5*(s1+1))/((s1+0.5)*(s1-2));
7  GH=syslin('c',g);
8  GH1=syslin('c',g1);
9  nyquist(GH);
10 nyquist(GH1);
11 mtlb_axis([-3 0.5 -0.6 0.6]);
12 xtitle('Nyquist plot of (2.5*(s+1))/((s+0.5)*(s-2))'
       )
13 figure;
14 show_margins(GH,'nyquist')
15 disp("since the point(-1+%i0) is  encircled
       anticlockwise by Nyquist plot ,so N=1 and P=1(
       given)")
16 N=1;//no. of encirclement of -1+%i0 by G(s)H(s) plot
        anticlockwise
17 P=1;//no. of poles of G(s)H(s) with positive real
       part
18 Z=P-N;//np.of zeros of 1+G(s)H(s)=0 with positive
       real part
19 disp(Z,"Z=")
20 disp("as Z=0,there are no roots of closed loop
       characterstics eq having positive real part,
       hence system is stable.")
```
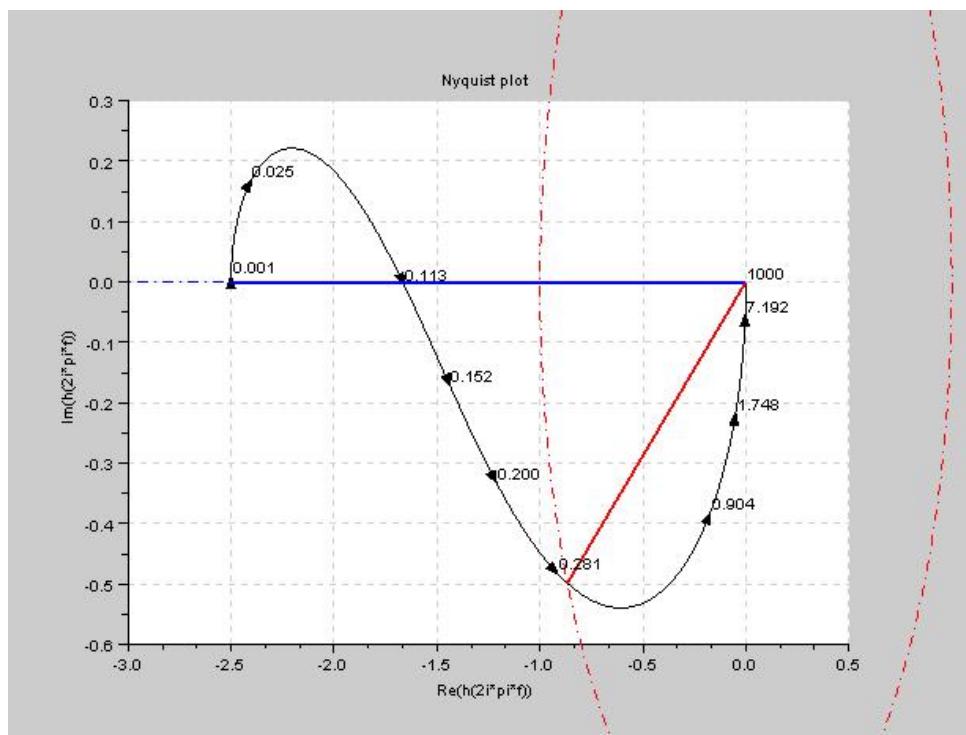
Figure 7.9: stability using Nyquist criterion
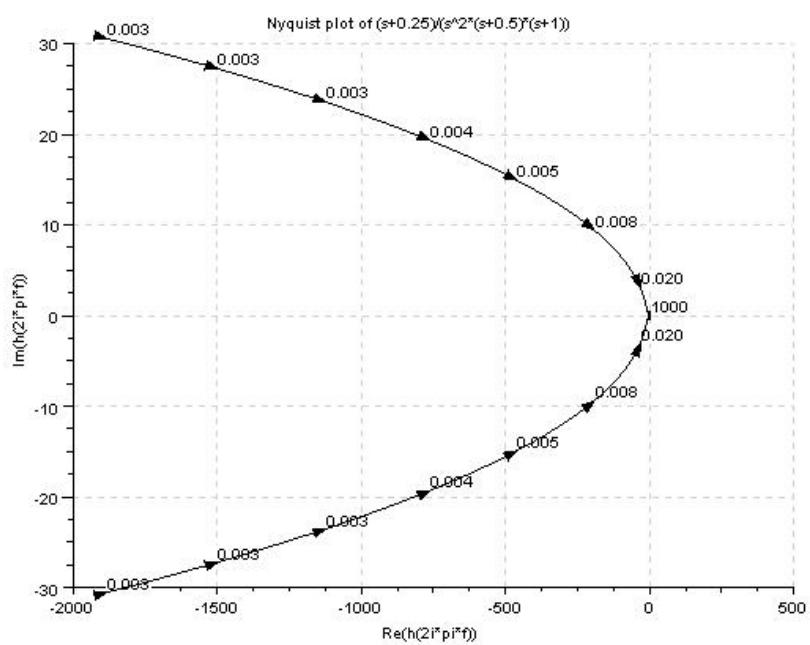
Figure 7.10: stability using Nyquist criterion

Scilab code Exa 7.17.3 stability using Nyquist criterion

```
1  //caption:stability_using_Nyquist_criterion
2  //example 7.17.3
3  //page 238
4  clf();
5  s=%s;
6  s1=-s;
7  g=(s+0.25)/(s^2*(s+0.5)*(s+1));
8  g1=(s1+0.25)/(s1^2*(s1+0.5)*(s1+1));
9  GH=syslin('c',g);
10 GH1=syslin('c',g1);
11 nyquist(GH);
12 nyquist(GH1);
13 mtlb_axis([-2000 500 -30 30]);
14 xtitle('Nyquist plot of (s+0.25)/(s^2*(s+0.5)*(s+1))
      ')
15 figure;
16 show_margins(GH,'nyquist')
17 disp("since the point(-1+%i0) is  encircled
      clockwise by Nyquist plot ,so N=-1 and P=1(given)
      ")
18 N=-1;//no. of encirclement of -1+%i0 by G(s)H(s)
      plot anticlockwise
19 P=1;//no. of poles of G(s)H(s) with positive real
      part
20 Z=P-N;//np.of zeros of 1+G(s)H(s)=0 with positive
      real part
21 disp(Z,"Z=")
22 disp("as Z=2,there are two roots of closed loop
      characterstics eq having positive real part,
      hence system is unstable.")
```
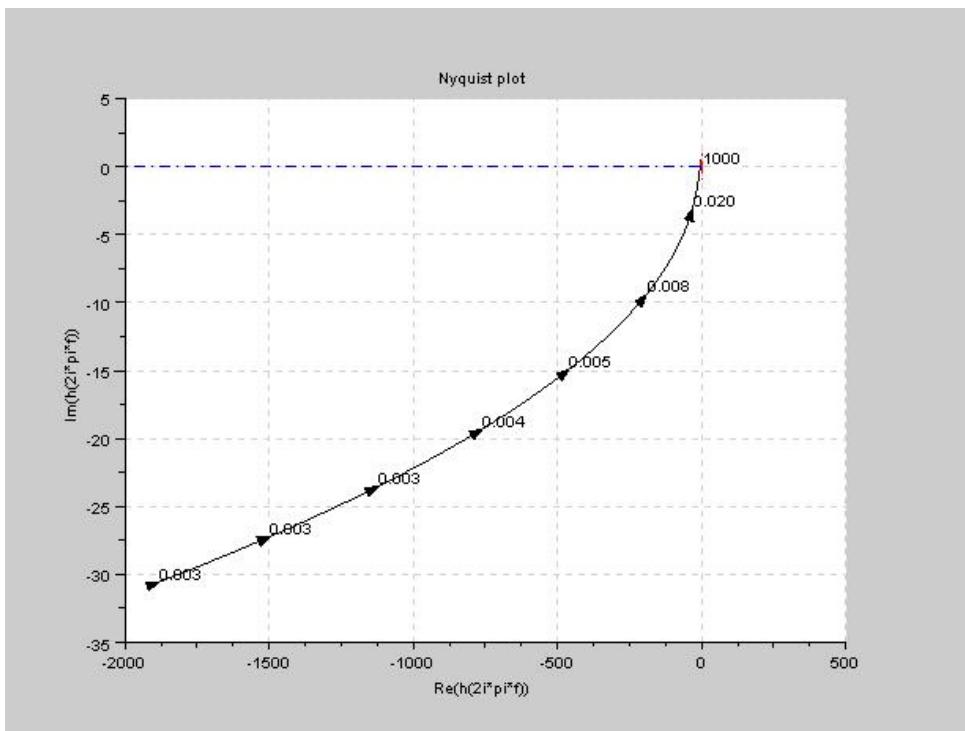
Figure 7.11: stability using Nyquist criterion
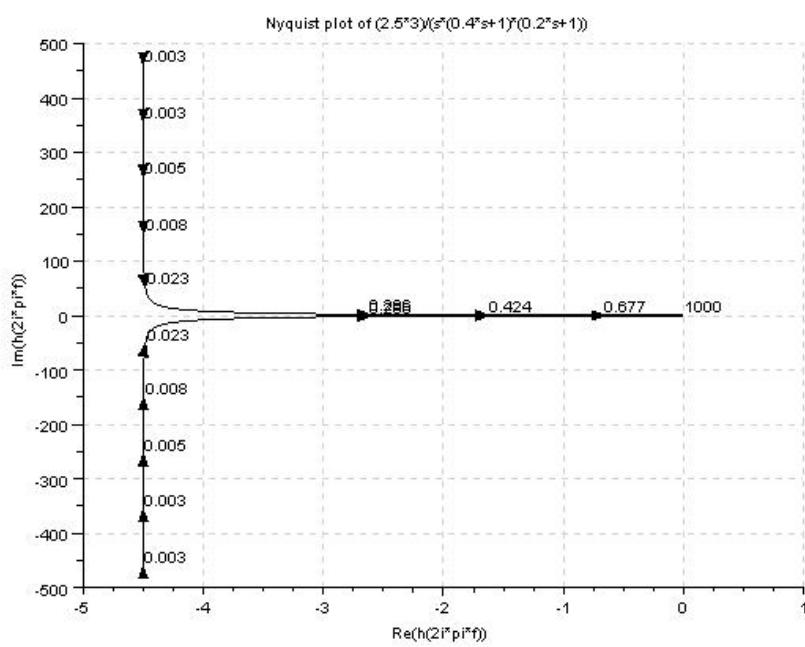
Figure 7.12: Phase Margin

Scilab code Exa 7.17.5 Phase Margin

```
1  // caption : phase_margin
2  // example  7.17.5
3  // page  241
4  clf ();
5  s =% s ;
6  s1 = -s ;
7  disp ("for  K=3")
8  g =(2.5*3)/(s *(0.4* s +1)*(0.2* s +1));
9  g1 =(2.5*3)/(s1 *(0.4* s1 +1)*(0.2* s1 +1));
10 GH = syslin ('c ',g );
11 GH1 = syslin ('c ',g1 );
12 nyquist (GH );
13 nyquist (GH1 );
14 mtlb_axis ([-5 1   -500 500]);
15 xtitle ('Nyquist  plot  of  (2.5*3)/(s *(0.4* s +1)*(0.2* s
       +1))')
16 pm = p_margin (GH )
17 disp (pm ,"phase  margin=")
```

**Scilab code Exa 7.17.7** stability using Nyquist criterion

```
1  // caption : stability_using_Nyquist_criterion
2  // example  7.17.7
3  // page  244
4  clf ();
5  s =% s ;
6  s1 = -s ;
7  g =(2.2/(s *(s +1)*(s^2+2* s +2)))
```
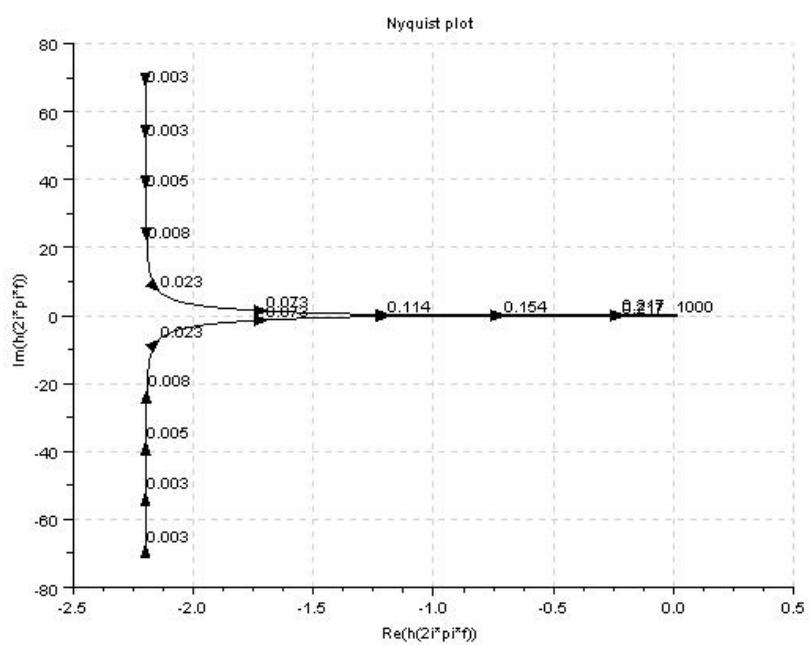
Figure 7.13: stability using Nyquist criterion

```
 8  g1=(2.2/(s1*(s1+1)*(s1^2+2*s1+2)))
 9  GH=syslin('c',g);
10  GH1=syslin('c',g1);
11  nyquist(GH);
12  nyquist(GH1);
13  mtlb_axis([-2.5 0.2 -75 75]);
14  disp("as the nyquist plot passes through the point
        -1+%i*0, so system is marginally stable and
        output represents sustained oscillations.")
```

Scilab code Exa 7.17.9 gain margin and phase margin

```
 1  //caption: gain_margin_and_phase_margin
 2  //example 7.17.9
 3  //page 245
 4  s=%s;
 5  syms w;
 6  s1=-s;
 7  gh=(32/(s*(s+sqrt(6))^3));
 8  g=sym((32/(s*(s+sqrt(6))^3)));
 9  s=%i*w;
10  a=eval(g);
11  w=sqrt(2);
12  b=float(eval(abs(a)));
13  disp(b,"at w=sqrt(2),|G(jw)H(jw)|=");
14  GH=syslin('c',gh);
15  gm=g_margin(GH);
16  pm=p_margin(GH);
17  disp(gm,"Gain margin=");
18  disp(pm,"Phase margin=");
19  disp("since gm=0 and pm=0, so system is marginally
        stable")
```
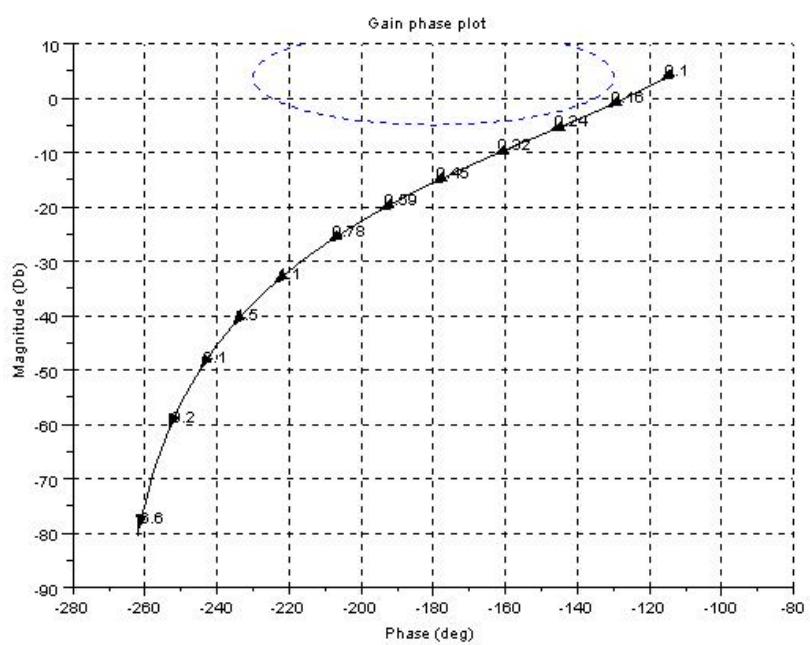
Figure 7.14: gain phase plot

Scilab code Exa 7.17.18 gain phase plot

```
1  //caption:gain_phase_plot
2  //example 7.17.18
3  //page 256
4  k=1;
5  s=%s;
6  G=syslin('c',k/(s*(0.5*s+1)*(0.25*s+1)));
7  // freq range to plot
8  fmin=0.1;
9  fmax=7;
10 black(G, fmin, fmax)
11 xgrid
12 xtitle('Gain phase plot')
13 disp("for GM=8 db, K=2.23")
14 disp("for PM=20 deg. , K=2.69")
```

Scilab code Exa 7.19.1 stability using bode plot

```
1  //caption:stability_using_bode_plot
2  //example 7.19.1
3  //page 280
4  s=%s;
5  g=50/((s+1)*(s+2));
6  G=syslin('c',g)
7  fmin=0.01;
8  fmax=100;
9  bode(G, fmin, fmax)
10 show_margins(G)
11 gm=g_margin(G)
```
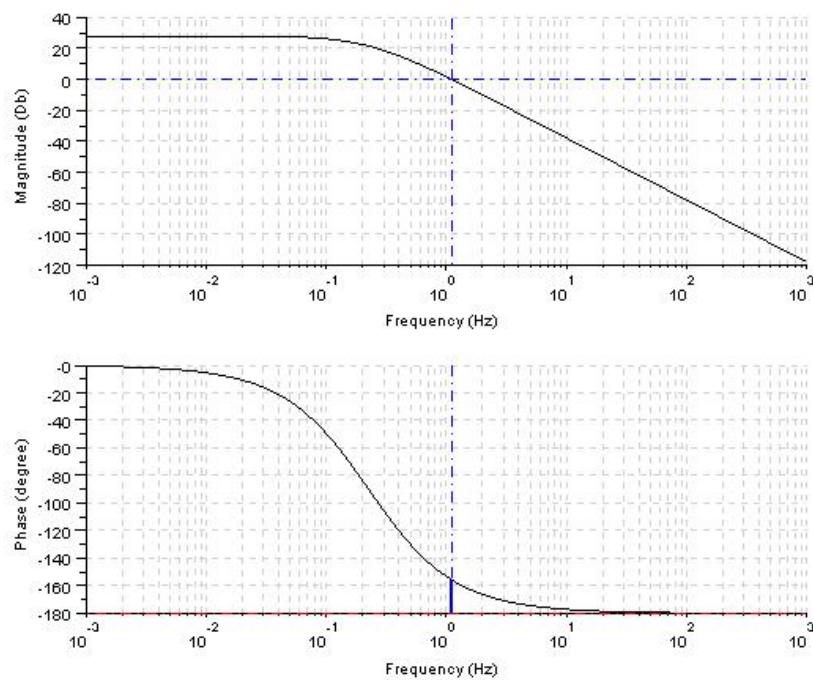
Figure 7.15: stability using bode plot
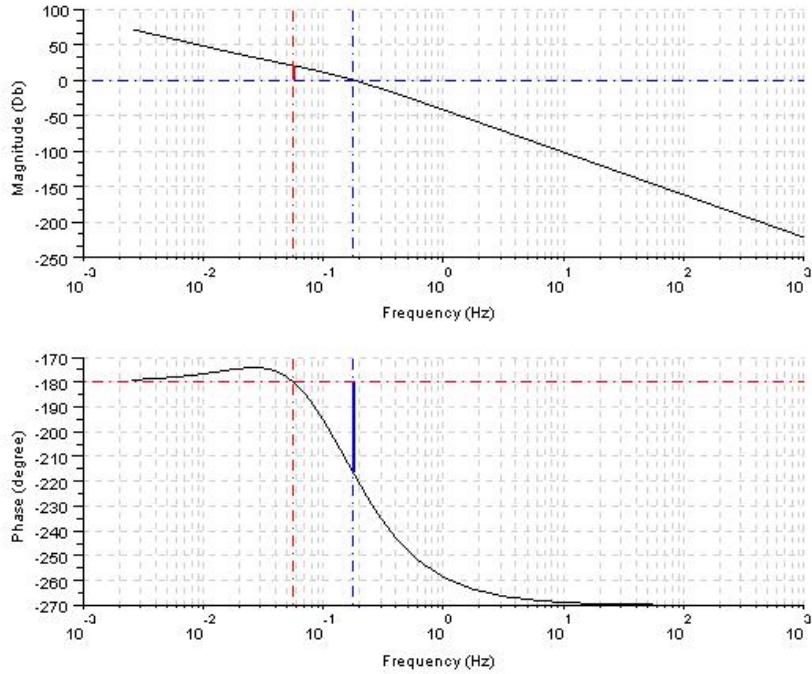
Figure 7.16: gain margin and phase margin

```
12  pm=p_margin(G)
13  disp(gm,"gain_margin=");
14  disp(pm,"phase_margin=");
15  disp("since gain and phase margin are both positive
        so system is stable")
```

Scilab code Exa 7.19.2 gain margin and phase margin

```
1  //caption:gain_margin_and_phase_margin
2  //example 7.19.2
3  //page 282
```

```
4  s=%s;
5  g=((2*(s+0.25))/(s^2*(s+1)*(s+0.5)));
6  G=syslin('c',g)
7  fmin=0.1;
8  fmax=100;
9  bode(G, fmin, fmax)
10 [gm,freqGM]=g_margin(G);
11 [pm,freqPM]=p_margin(G);
12 show_margins(G);
13 disp(gm,"gain_margin=");
14 disp((freqGM*2*%pi),"gain_margin_freq=");
15 disp(pm,"phase_margin=");
16 disp((freqPM*2*%pi),"phase_margin_freq=");
17 show_margins(G);
18 disp("since gain and phase margin are both negative
      so system is unstable")
```

Scilab code Exa 7.19.3 stability using bode plot

```
1  //caption:stability_using_bode_plot
2  //example 7.19.3
3  //page 283
4  s=%s;
5  g=(48*(s+10))/(s*(s+20)*(s^2+2.4*s+16));
6  G=syslin('c',g)
7  fmin=0.01;
8  fmax=100;
9  bode(G, fmin, fmax)
10 show_margins(G)
11 gm=g_margin(G)
12 pm=p_margin(G)
13 disp(gm,"gain_margin=");
14 disp(pm,"phase_margin=");
```
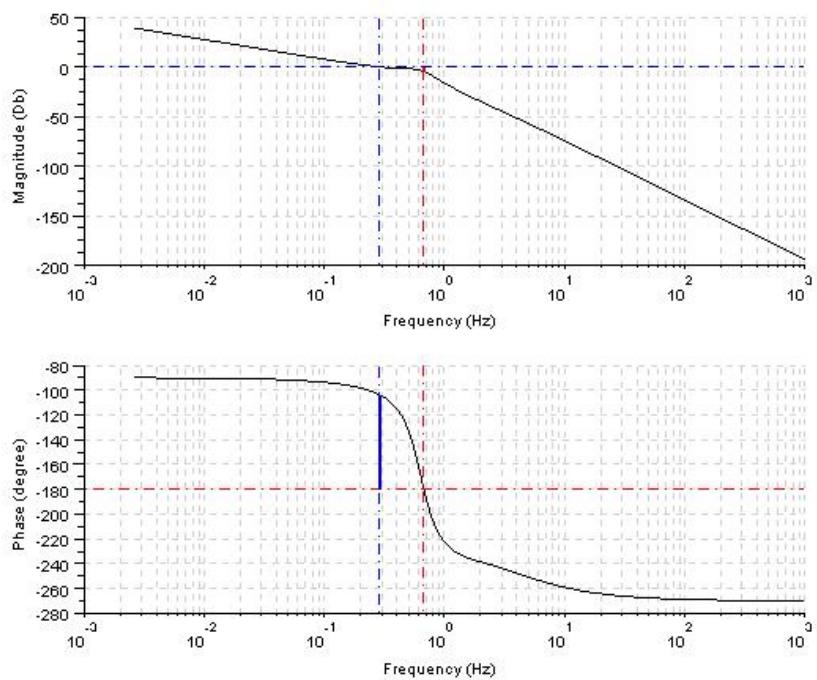
Figure 7.17: stability using bode plot
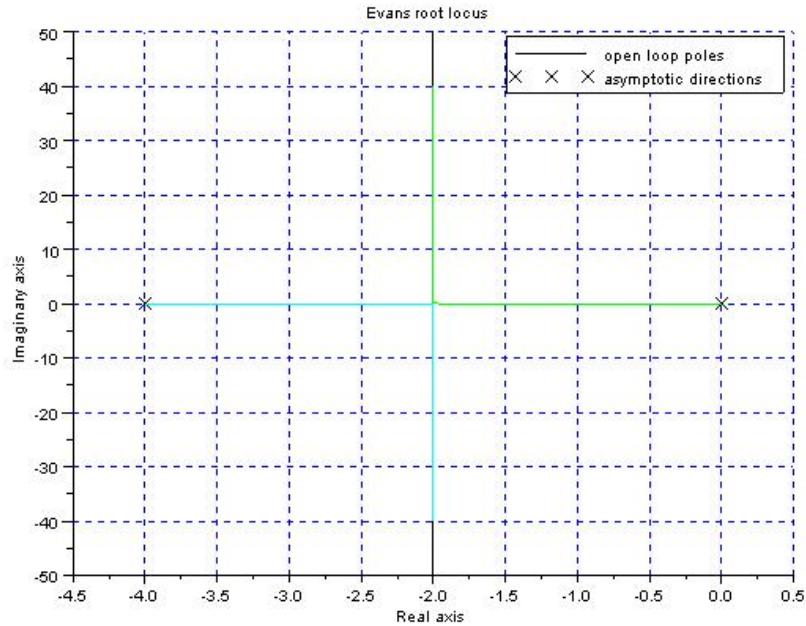
Figure 7.18: root locus description

```
15  disp("since gain and phase margin are both positive
        so system is stable")
```

Scilab code Exa 7.24.1 root locus description

```
1  //caption:root_locus_description
2  //example 7.24.1
3  //page 295
4  clc;
5  s=%s;
6  syms K
```

```
 7  G=K/( s ∗( s +4) ) ;
 8  disp ( " the characterstics eq. is determined as : " )
 9  CH=( s ∗( s +4) )+K
10  CH=sym ( ' ( s ∗( s +4) )+K ' ) ;
11  disp ( '=0 ' ,CH , " characterstics_eq ,CH=" )
12  eq =( s ∗( s +4) )
13  p=roots ( eq )
14  disp (p , " open loop poles are : " )
15  K=sym ( '−( s ∗( s +4) ) ' )
16  d=diff (K , s )
17  e=2∗s+4
18  P=2;
19  Z=0;
20  disp ( " since −2 lies on root locus so breakaway point
         is −2" )
21  for (k=0:1)
22    A=((2∗k+1) ∗180) /(P−Z) ;
23    disp (A , " asymptote are at angle : " )
24  end
25  x=((0−4)−0) /(P−Z)   // ( sum_of_P − sum_of_Z ) /(P−Z)
26  disp (x , " asymptotes intersect at " )
27  disp ( " since |G( s ) ∗H( s ) |=1" )
28  disp ( " which gives K=8" )
29  k=8
30  g=k/( s ∗( s +4) )
31  G=syslin ( ' c ' ,g )
32  evans (g ,200)
33  xgrid (2)
```

Scilab code Exa 7.24.2 root locus description

```
1  // caption : root_locus_description
2  // example 7.24.2
```
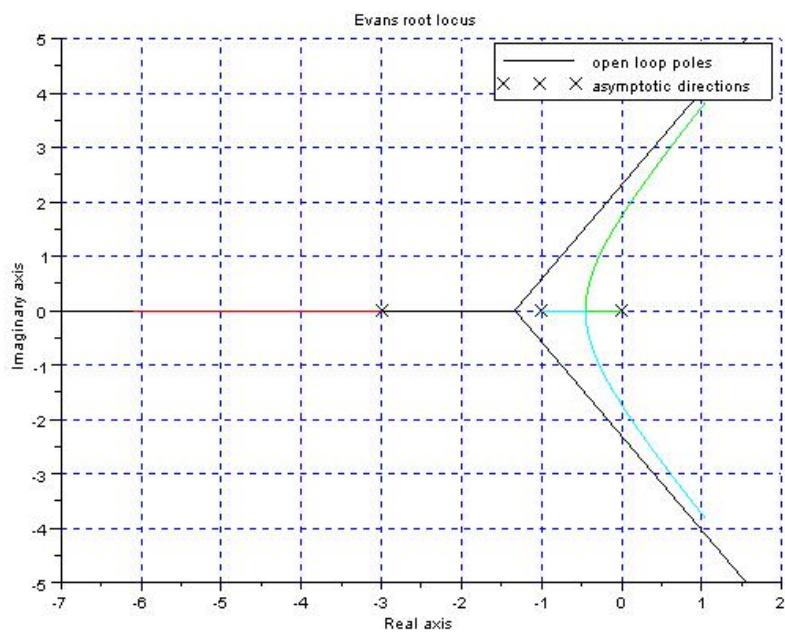
Figure 7.19: root locus description

```scilab
 3  //page 296
 4  s=%s;
 5  syms K;
 6  GH=K/(s*(s+1)*(s+3))
 7  zeta=0.5
 8  //from given data
 9  disp("the characterstics eq. is determined as:")
10  CH=(s*(s+1)*(s+3))+K
11  CH=sym('(s*(s+1)*(s+3))+K');
12  disp('=0',CH,"characterstics_eq ,CH=")
13  c0=coeffs(CH,'s',0);
14  c1=coeffs(CH,'s',1);
15  c2=coeffs(CH,'s',2);
16  c3=coeffs(CH,'s',3);
17  b=[c0 c1 c2 c3]
18  n=4;
19  routh=[b([4,2]);b([3,1])];
20  routh=[routh;-det(routh)/routh(2,1),0]
21  t=routh(2:3,1:2)
22  routh=[routh;-det(t)/t(2,1),0]
23  K=sym('-(s^3+4*s^2+3*s)')
24  d=diff(K,s)
25  e=-3*s^2-8*s-3
26  r1=roots(e)
27  disp(r1,"roots=")
28  disp("-0.45 is break away point since it lies on
        root locus")
29  disp(routh,"routh=")
30  disp("for given system to be marginally stable:");
31  disp("(12-K)=0 ");
32  disp("which gives:");
33  disp("K=12, for margianl stability");
34  K=12;
35  k=12
36  a=4*s^2+k//intersection of root locus with imaginary
          plane
37  r=roots(a)
38  g=k/(s*(s+1)*(s+3))
```

116

```
39  G=syslin('c',g)
40  evans(g,8)
41  xgrid(2)
42  disp("the line theta=acos(zeta)=60 intersects root
        locus at sa=(-0.35+i0.6)")
43  disp("the value of K at s=sa is find to be 1.66 ")
44  disp("the value of K at s=-4")
45  disp("K=12")
46  disp("at K=1.66")
47  k=1.66
48  H=1
49  G=k/(s*(s+1)*(s+3))
50  CL=G/(1+G*H)
51  disp(CL,"C(s)/R(s)=")
```

Scilab code Exa 7.24.3 root locus description

```
1   //caption:root_locus_description
2   //example 7.24.3
3   //page 299
4   clc;
5   s=%s;
6   syms K
7   G=K/(s*(s+6)*(s^2+4*s+13));
8   disp("the characterstics eq. is determined as:")
9   CH=(s*(s+6)*(s^2+4*s+13))+K
10  CH=sym('(s*(s+6)*(s^2+4*s+13))+K');
11  disp('=0',CH,"characterstics_eq ,CH=")
12  eq=(s*(s+6)*(s^2+4*s+13))
13  p=roots(eq)
14  disp(p,"open loop poles are:")
15  phi1=180-(atan(3/2)*180/%pi)
16  phi2=atan(3/4)*180/%pi
```
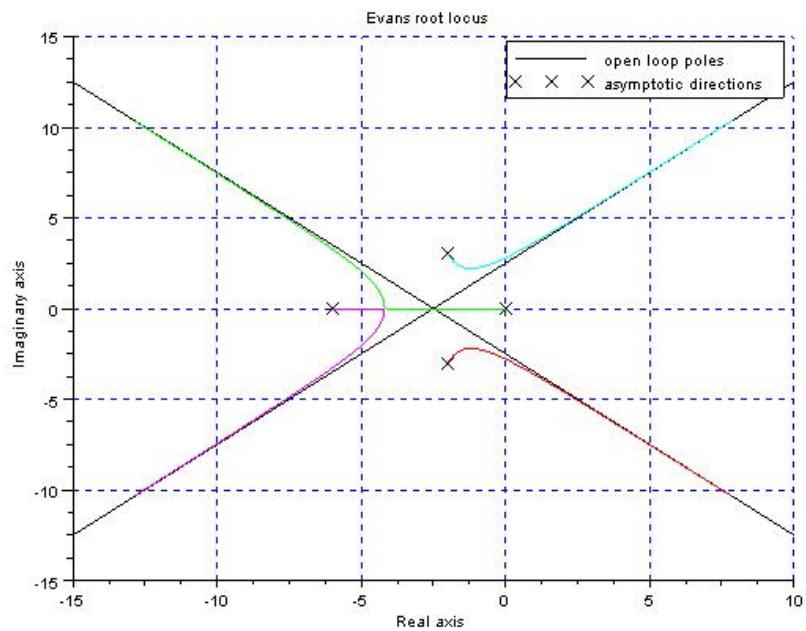
Figure 7.20: root locus description

```
17  phi3=90
18  phi_p2=180-(phi1+phi2+phi3)
19  phi_p3=-phi_p2
20  disp(phi_p2,"angle of departure for -2+3i=")
21  disp(phi_p3,"angle of departure for -2-3i=")
22  c0=coeffs(CH,'s',0);
23  c1=coeffs(CH,'s',1);
24  c2=coeffs(CH,'s',2);
25  c3=coeffs(CH,'s',3);
26  c4=coeffs(CH,'s',4);
27  b=[c0 c1 c2 c3 c4]
28  routh=[b([5,3,1]);b([4,2]),0]
29  routh=[routh;-det(routh(1:2,1:2))/routh(2,1),routh
        (1,3),0]
30  routh(3,1)=simple(routh(3,1))
31  t=routh(2:3,1:2)
32  l=simple(-det(t)/t(2,1))
33  routh=[routh;l,0,0]
34  routh=[routh;K,0,0]
35  K=sym('-(s*(s+6)*(s^2+4*s+13))')
36  d=diff(K,s)
37  e=-(4*s^3+30*s^2+74*s+78)
38  f=-e;
39  r=roots(f);
40  disp(r,"r=")
41  disp("since -4.2 lies on root locus, so the
        breakaway point is -4.2 ")
42  disp(routh,"routh=")
43  disp("for given system to be marginally stable:");
44  disp("(78-0.34*K)=0 ");
45  disp("which gives:");
46  disp("K=229.4");
47  K=229.4;
48  k=229.4;
49  a=29.2*s^2+229.4//intersection of root locus with s
        plane
50  r1=roots(a)
51  g=k/(s*(s+6)*(s^2+4*s+13));
```
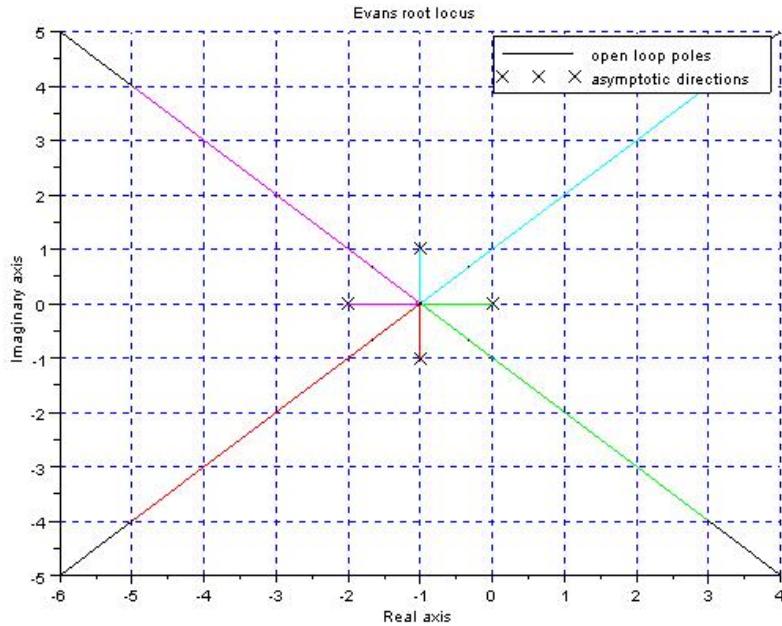
Figure 7.21: root locus

```
52  g=syslin('c',g)
53  evans(g,200)
54  xgrid(2)
55  disp(r1,"the point of intersection of root locus
        with imaginary axis =")
```

Scilab code Exa 7.24.4 root locus

```
1  //caption:root_locus
2  //example 7.24.4
3  //page 301
```

```
 4  clc;
 5  s=%s;
 6  syms K;
 7  GH=K/(s*(s+4)*(s^2+4*s+13))
 8  disp("the characterstics eq. is determined as:")
 9  CH=(s*(s+4)*(s^2+4*s+13))+K
10  CH=sym('(s*(s+4)*(s^2+4*s+13))+K');
11  disp('=0',CH,"characterstics_eq,CH=")
12  c0=coeffs(CH,'s',0);
13  c1=coeffs(CH,'s',1);
14  c2=coeffs(CH,'s',2);
15  c3=coeffs(CH,'s',3);
16  c4=coeffs(CH,'s',4);
17  b=[c0 c1 c2 c3 c4 ]
18  routh=[b([5,3,1]);b([4,2]),0]
19  routh=[routh;-det(routh(1:2,1:2))/routh(2,1),routh
       (1,3),0]
20  routh(3,1)=simple(routh(3,1))
21  t=routh(2:3,1:2)
22  l=simple(-det(t)/t(2,1))
23  routh=[routh;l,0,0]
24  routh=[routh;K,0,0]
25  K=sym('(s*(s+4)*(s^2+4*s+13))')
26  d=diff(K,s)
27  e=(-4*s^3+24*s^2+58*s+52)
28  r=roots(e)
29  disp("since -2 lies on root locus so complex
       breakaway point is -2+i1.58 and -2-i1.58")
30  disp(routh,"routh=")
31  disp("for given system to be marginally stable:");
32  disp("((20-4K)/5)=0 ");
33  disp("which gives:");
34  disp("K=5");
35  K=5;
36  k=5
37  a=5*s^2+5//intersection of root locus with s plane
38  r=roots(a)
39  g=k/(s*(s+2)*(s^2+2*s+2))
```

```
40  G=syslin('c',g)
41  evans(g,200)
42  xgrid(2)
43  eq=(s*(s+4)*(s^2+4*s+13))
44  p=roots(eq)
45  disp(p,"open loop poles are:")
46  phi1=180-(atan(3/2)*180/%pi)
47  phi2=atan(3/2)*180/%pi
48  phi3=90
49  phi_p2=180-(phi1+phi2+phi3)
50  phi_p3=-phi_p2
51  disp(phi_p2,"angle of departure for -2+3i=")
52  disp(phi_p3,"angle of departure for -2-3i=")
```

Scilab code Exa 7.24.6 root locus

```
1   //caption:root_locus
2   //example 7.24.6
3   //page 304
4   clc;
5   s=%s;
6   syms K;
7   clf();
8   g=(K*(s+1)/(s^2+0.4*s+0.4));
9   eq=(s^2+0.4*s+0.4)
10  p=roots(eq)
11  disp(p,"open loop poles are:");
12  P=2;
13  Z=1;
14  k=0
15    A=((2*k+1)*180)/(P-Z);
16    disp(A,"asymptote are at angle:")
17    CH=(s^2+0.4*s+0.4)+K*(s+1)
```
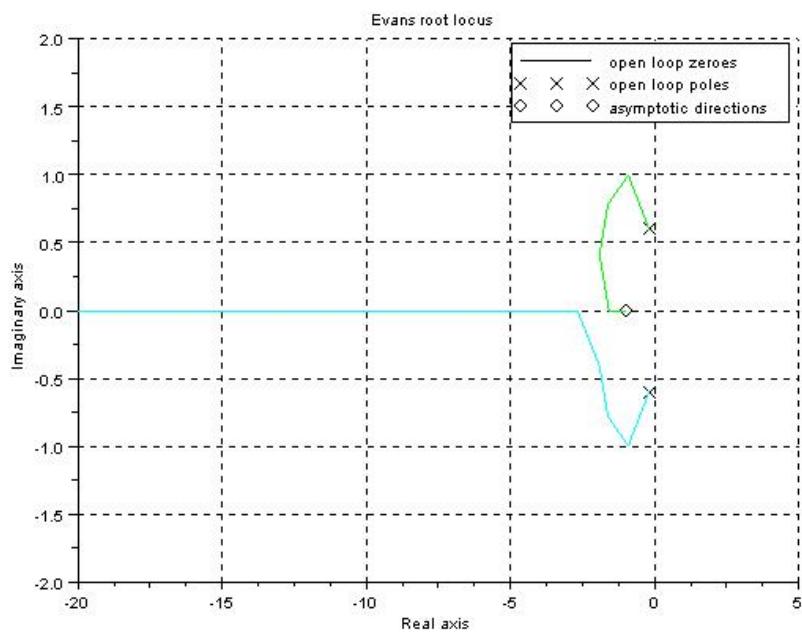
Figure 7.22: root locus

123

```
18  CH=sym('(s^2+0.4*s+0.4)+K*(s+1)');
19  disp('=0',CH,"characterstics_eq ,CH=")
20  K=sym('-(s^2+0.4*s+0.4)/(s+1)')
21  d=diff(K,s)
22  e=s*(s+2)
23  disp("break away point s=-2 as it lies on root locus
        ")
24  disp("since |G(s)*H(s)|=1")
25  disp("which gives K=3.6")
26  k=3.6
27  g=(k*(s+1))/(s^2+0.4*s+0.4);
28  G=syslin('c',g)
29  evans(g,200)
30  xgrid(1)
31  v=[-20 1 -2 2]
32  mtlb_axis(v);
33  disp(" poles s=(-0.2+j0.6 and -0.2-j0.6)are
        equidistant from the zero s=-1, hence rootlocus
        plot is arc of the circle with centre s=-1 and
        radius 1.")
```

Scilab code Exa 7.24.7 root locus

```
1  //caption:root_locus
2  //example 7.24.7
3  //page 306
4  s=%s;
5  syms K;
6  GH=K/(s*(s+2)*(s+4))
7  //since Mp=40%, so .4=exp((-zeta*%pi)/(sqrt(1-zeta
        ^2))
8  zeta=0.3
9  //from given data
```

Figure 7.23: root locus

125

```scilab
10  disp("the characterstics eq. is determined as:")
11  CH=(s*(s+2)*(s+4))+K
12  K=sym('-(s^3+6*s^2+8*s)')
13  d=diff(K,s)
14  e=-3*s^2-12*s-8
15  r1=roots(e)
16  disp(r1,"roots=")
17  disp(" -0.842 is break away point sinc it lies on
        root locus")
18  CH=sym('s^3+6*s^2+8*s+K');
19  disp('=0',CH,"characterstics_eq ,CH=")
20  c0=coeffs(CH,'s',0);
21  c1=coeffs(CH,'s',1);
22  c2=coeffs(CH,'s',2);
23  c3=coeffs(CH,'s',3);
24  b=[c0 c1 c2 c3]
25  n=4;
26  routh=[b([4,2]);b([3,1])];
27  routh=[routh;-det(routh)/routh(2,1),0]
28  t=routh(2:3,1:2)
29  routh=[routh;-det(t)/t(2,1),0]
30  disp(routh,"routh=")
31  disp("for given system to be marginally stable:");
32  disp("(48-K)=0 ");
33  disp("which gives:");
34  disp("K=48");
35  K=48;
36  k=48
37  a=6*s^2+48//intersection of root locus with
        imaginary plane
38  r=roots(a)
39  g=k/(s*(s+2)*(s+4))
40  G=syslin('c',g)
41  evans(g,8)
42  xgrid(2)
43  disp("the line theta=acos(zeta)=72.5 intersects root
        locus at sa=(-0.5+i1.65)")
44  disp("the value of K at s=sa is find to be 14.87 for
```

Figure 7.24: root locus

       Mp=40%")

```
45  K =14.87
46  ts =4/0.5          // ts =4/( zeta ∗wn)
47  Kv = limit ( s ∗GH , s ,0)
48  Kv = eval ( Kv )
49  Kv = float ( Kv )
50  disp ( Kv ,"Kv=" ) ;
```

Scilab code Exa 7.24.8 root locus

```
1  // caption : root_locus
```

```scilab
2  //example 7.24.8
3  //page 308
4  clc;
5  s=%s;
6  syms K;
7  GH=K/(s*(s+1)*(s+3))
8  zeta=0.5
9  //from given data
10 disp("the characterstics eq. is determined as:")
11 CH=(s*(s+1)*(s+3))+K
12 CH=sym('(s*(s+1)*(s+3))+K');
13 disp('=0',CH,"characterstics_eq ,CH=")
14 c0=coeffs(CH,'s',0);
15 c1=coeffs(CH,'s',1);
16 c2=coeffs(CH,'s',2);
17 c3=coeffs(CH,'s',3);
18 b=[c0 c1 c2 c3]
19 n=4;
20 routh=[b([4,2]);b([3,1])];
21 routh=[routh;-det(routh)/routh(2,1),0]
22 t=routh(2:3,1:2)
23 routh=[routh;-det(t)/t(2,1),0]
24 K=sym('-(s^3+4*s^2+3*s)')
25 d=diff(K,s)
26 e=-3*s^2-8*s-3
27 r1=roots(e)
28 disp(r1,"roots=")
29 disp("-0.45 is break away point since it lies on
       root locus")
30 disp(routh,"routh=")
31 disp("for given system to be marginally stable:");
32 disp("(12-K)=0 ");
33 disp("which gives:");
34 disp("K=12, for margianl stability");
35 K=12;
36 k=12
37 a=4*s^2+k//intersection of root locus with imaginary
       plane
```

```
38  r=roots(a)
39  disp(r,"intersection point of root locus with
        imaginary axis=")
40  g=k/(s*(s+1)*(s+3))
41  G=syslin('c',g)
42  evans(g,8)
43  xgrid(2)
44  disp("for K=6")
45  k=6;
46  GH=k/(s*(s+1)*(s+3))
47  gm=K/k   //gm=K(marginal_stability)/K(desired)
48  disp(gm,"gain_margin=")
49  disp("the point where K=6 is s=1.2")
50  pm=180+(-90-(atan(1.2/1)*180/%pi)-(atan(1.2/3)*180/
        %pi))
51  disp(pm,"phase margin=")
```

Scilab code Exa 7.24.9 root locus

```
1   //caption:root_locus
2   //example 7.24.9
3   //page 308
4   clc;
5   s=%s;
6   syms K
7   clf();
8   g=(K*(s^2+4)/(s*(s+1)));
9   Z=2
10  P=2
11  disp("the characterstics eq. is determined as:")
12  CH=(s*(s+1))+K*(s^2+4)
13  CH=sym('(s*(s+1))+K*(s^2+4)');
14  disp('=0',CH,"characterstics_eq ,CH=")
```
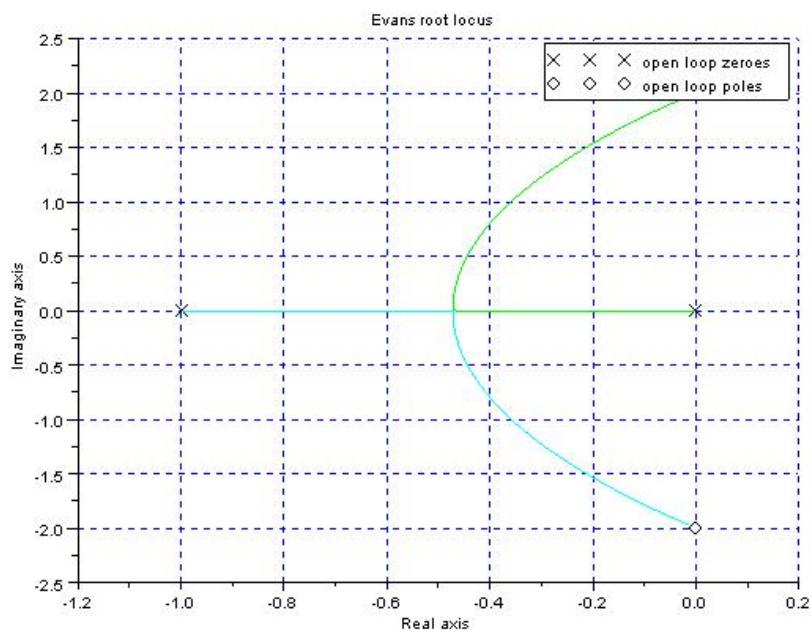
Figure 7.25: root locus

```
15  eq=(s*(s+1))
16  p=roots(eq)
17  disp(p,"open loop poles are:")
18  K=sym('-(s*(s+1))/(s^2+4)')
19  d=diff(K,s)
20  e=2*s^2-8*s-8
21  r=roots(e);
22  disp(r,"r=")
23  disp("since -0.82 lies on root locus, so the
        breakaway point is -0.82 ")
24  disp("by putting s=-0.82 in  |G(s)*H(s)|=1, the
        value of K at s=-0.82 is K=0.2")
25  k=0.2
26  g=(k*(s^2+4)/(s*(s+1)));
27  G=syslin('c',g)
28  evans(g,200)
29  xgrid(2)
30
31  disp("part (b)")
32  disp("by putting s=-0.69+i0.9 in  |G(s)*H(s)|=1, K
        =0.464")
```

Scilab code Exa 7.24.10 Overall Transfer Function and Root Locus

```
1  //Caption:overall_transfer_function_and_root_locus
2  //example 7.24.10
3  //page 314
4  syms Ka Ke Kf Rf Lf eq  N1 N2 N3 N4 N5
5  //where Ka=amplifier_gain; Ke=error_detector_gain;
       Kf=motor_torque_const; Rf=field_resistance; Lf=
       field_inductance Jeq=moment_of_inertia; feq=
       coeff_of_viscous_friction;
6  s=%s;
```
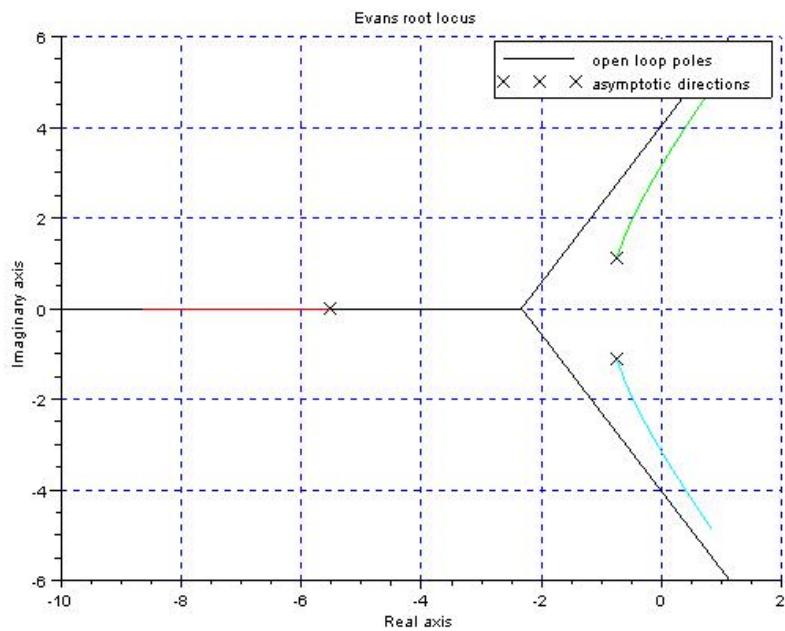
Figure 7.26: Overall Transfer Function and Root Locus

```
7  d=N1/N2;
8  e=N4/N3;
9  f=N4/N5;
10 n=N3/N5
11 Ke=0.05; Kf=2; Rf=10; Lf=2; Jeq=0.5*10^-4; feq
      =10^-4;  d=0.1; e=5; f=0.5;n=0.2;
12 a=(1*Ke); //in series
13 b=Kf/(Rf+s*Lf);
14 c=1/(s*(Jeq*s+feq));
15 g= (b*c)    //in series
16 h= (g*a)    //in series
17 j= (h*0.02)  //in series
18 k=j/(1+j*0.5);
19 a1=1/20;
20 cl=a1*k
21 disp(cl,"C(s)/R(s)=");
22 clf();
23 G=syslin('c',cl)
24 evans(G,200)
25 xgrid(2)
```

Scilab code Exa 7.24.11 root locus

```
1  //caption:root_locus
2  //example 7.24.11
3  //page 308
4  clc;
5  s=%s;
6  syms K
7  clf();
8  g=K*(s+0.1)/(s*(s-0.2)*(s^2+s+0.6));
9  Z=2
10 P=2
```
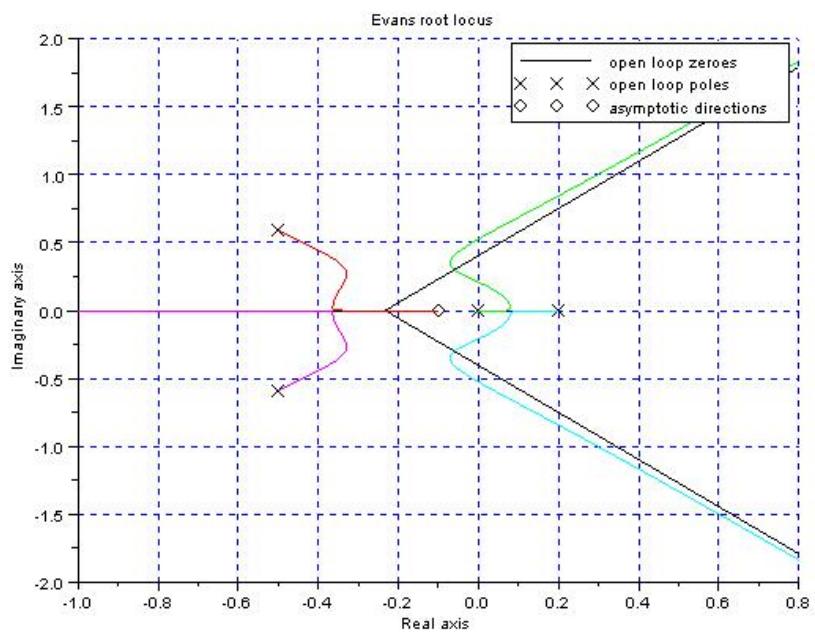
Figure 7.27: root locus

```
11  disp("the characterstics eq. is determined as:")
12  CH=(s*(s-0.2)*(s^2+s+0.6))+K*(s+0.1)
13  CH=sym('(s*(s-0.2)*(s^2+s+0.6))+K*(s+0.1)');
14  disp('=0',CH,"characterstics_eq,CH=")
15  eq=(s*(s-0.2)*(s^2+s+0.6))
16  p=roots(eq)
17  disp(p,"open loop poles are:")
18  c0=coeffs(CH,'s',0);
19  c1=coeffs(CH,'s',1);
20  c2=coeffs(CH,'s',2);
21  c3=coeffs(CH,'s',3);
22  c4=coeffs(CH,'s',4);
23  b=[c0 c1 c2 c3 c4 ]
24  routh=[b([5,3,1]);b([4,2]),0]
25  routh=[routh;-det(routh(1:2,1:2))/routh(2,1),routh
        (1,3),0]
26  routh(3,1)=simple(routh(3,1))
27  t=routh(2:3,1:2)
28  l=simple(-det(t)/t(2,1))
29  routh=[routh;l,0,0]
30  routh=[routh;K,0,0]
31  K=sym('-(s*(s-0.2)*(s^2+s+0.6))/(s+0.1)')
32  d=diff(K,s)
33  e=3*s^4+2*s^3+0.64*s^2+0.08*s-0.12
34  r=roots(e);
35  disp(r,"r=")
36  disp("since -0.37 and 0.08 lies on root locus, so
        the breakaway point is -0.37 and 0.08 ")
37  disp(routh,"routh=")
38  disp("for given system to be marginally stable:");
39  disp("(625*K^2-310*K+33)/(625*K-275)=0 ");
40  disp("which gives:");
41  disp("K=0.148 and 0.352");
42  K1=0.148;
43  K2=0.352
44  a=((11-25*K1)/20)*s^2+K1/10//intersection of root
        locus with jw axis
45  r1=roots(a)
```

```scilab
46  b=((11-25*K2)/20)*s^2+K2/10//intersection of root
       locus with jw axis
47  r2=roots(b)
48  disp(r1,r2,"root locus crosses imaginary axis at:")
49  g=(s+0.1)/(s*(s-0.2)*(s^2+s+0.6));
50  G=syslin('c',g)
51  evans(G,200)
52  xgrid(2)
53  v=[-1 0.7 -2 2];
54  mtlb_axis(v)
```

Chapter 8

COMPENSATION OF CONTROL SYSTEMS

Scilab code Exa 8.6.1 design suitable compensator

```
1  // caption : design_suitable_compensator
2  // example  8.6.1
3  // page  339
4  s=%s;
5  clf();
6  syms K;
7  g=(K/(s*(1+0.2*s)));
8  Kv=limit(s*g,s,0);// static  velocity  error
       coefficient
9  // since  Kv=10
10 K=10;
11 g=(10/(s*(1+0.2*s)));
12 G=syslin('c',g)
13 fmin=0.01;
14 fmax=100;
15 bode(G, fmin, fmax)
16 show_margins(G)
17 xtitle("uncompensated  system")
18 [gm,freqGM]=g_margin(G);
```

```scilab
19  [pm,freqPM]=p_margin(G);
20  disp(gm,"gain_margin=");
21  disp((freqGM*2*%pi),"gain_margin_freq=");
22  disp(pm,"phase_margin=");
23  disp((freqPM*2*%pi),"
        phase_margin_freq_or_gain_cross_over_frequency=")
        ;
24  disp("since P.M is less than desired value so we
        need phase lead network ")
25  disp("selecting zero of lead compensating network at
         w=5.5 rad/sec and pole at w=13.8 rad/sec and
        applying gain to account attenuatin factor .")
26  gc=(1+0.18*s)/(1+0.072*s)
27  Gc=syslin('c',gc)
28  disp(Gc,"transfer function of lead compensator=");
29  G1=G*Gc
30  disp(G1,"overall transfer function=");
31  fmin=0.01;
32  fmax=100;
33  figure();
34  bode(G1, fmin, fmax);
35  show_margins(G1)
36  xtitle("compensated system")
37  [gm,freqGM]=g_margin(G1);
38  [pm,freqPM]=p_margin(G1);
39  disp(pm,"phase_margin_of_compensated_system=");
40  disp((freqPM*2*%pi),"gain_cross_over_frequency=");
```

Scilab code Exa 8.6.2 design phase lead compensator

```scilab
1  //caption:design_phase_lead_compensator
```
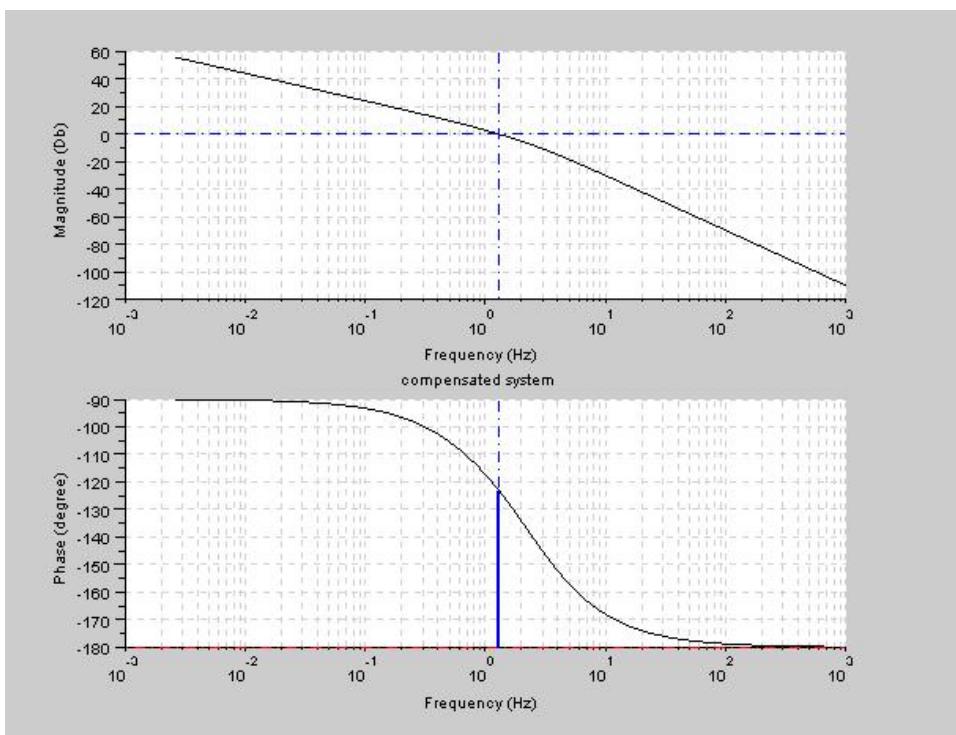
Figure 8.1: design suitable compensator
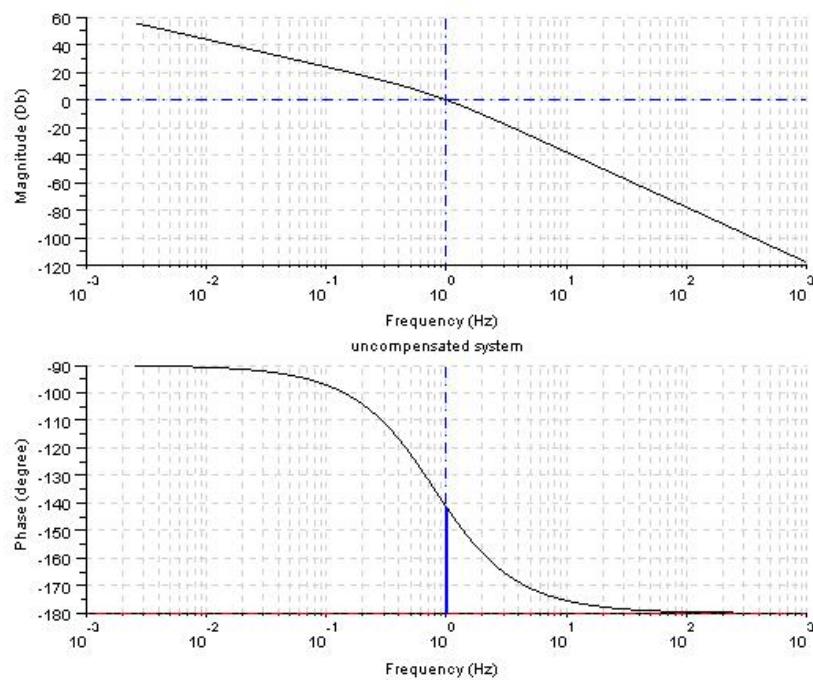
Figure 8.2: design suitable compensator

```scilab
2  //example 8.6.2
3  //page 340
4  clc;
5  s=%s;
6  clf();
7  syms K;
8  g=(K/(s^2*(1+0.05*s)));
9  Ka=limit(s^2*g,s,0);//static acceleration error
      coefficient
10 //since Ka=100
11 K=100;
12 g=(100/(s^2*(1+0.05*s)));
13 G=syslin('c',g)
14 fmin=0.01;
15 fmax=100;
16 bode(G, fmin, fmax)
17 show_margins(G)
18 xtitle("uncompensated system")
19 [gm,freqGM]=g_margin(G);
20 [pm,freqPM]=p_margin(G);
21 disp(gm,"gain_margin=");
22 disp((freqGM*2*%pi),"gain_margin_freq=");
23 disp(pm,"phase_margin=");
24 disp((freqPM*2*%pi),"
      phase_margin_freq_or_gain_cross_over_frequency=")
      ;
25 disp("since P.M is negaative so system is unstable "
      )
26 disp("selecting zero of lead compensating network at
       w=5rad/sec and pole at w=54rad/sec and applying
      gain to account attenuatin factor .")
27 gc=(1+0.2*s)/(1+0.0186*s)
28 Gc=syslin('c',gc)
29 disp(Gc,"transfer function of lead compensator=");
30 G1=G*Gc
31 disp(G1,"overall transfer function=");
32 fmin=0.01;
33 fmax=100;
```
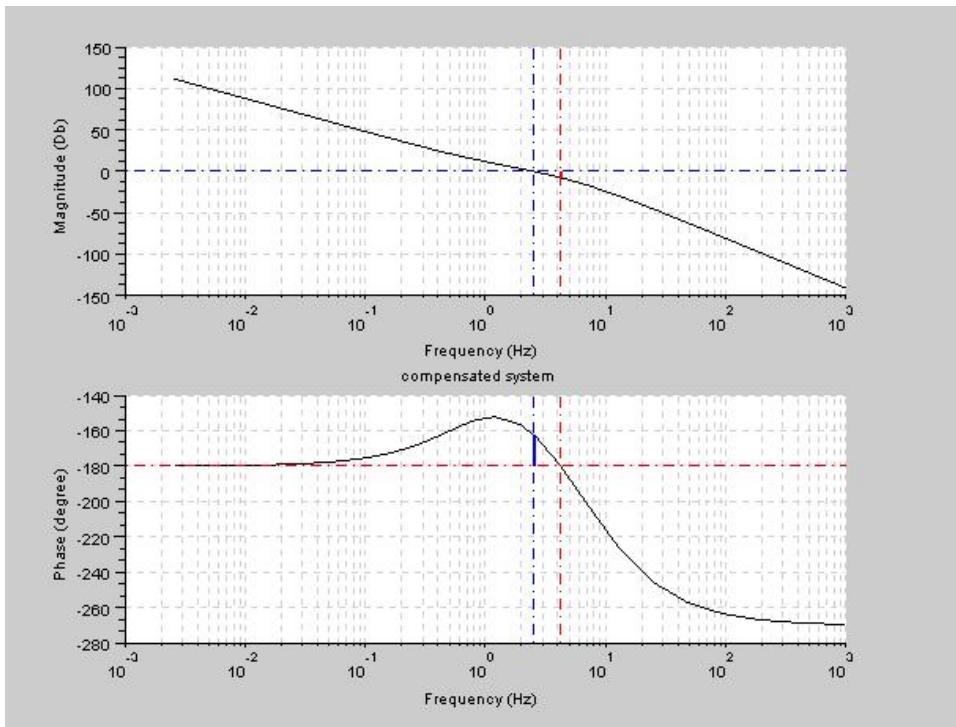
Figure 8.3: design phase lead compensator

```scilab
34  figure();
35  bode(G1, fmin, fmax)
36  show_margins(G1)
37  xtitle("compensated system")
38  [gm,freqGM]=g_margin(G1);
39  [pm,freqPM]=p_margin(G1);
40  disp(pm,"phase_margin_of_compensated_system=");
41  disp((freqPM*2*%pi),"gain_cross_over_frequency=");
```

142
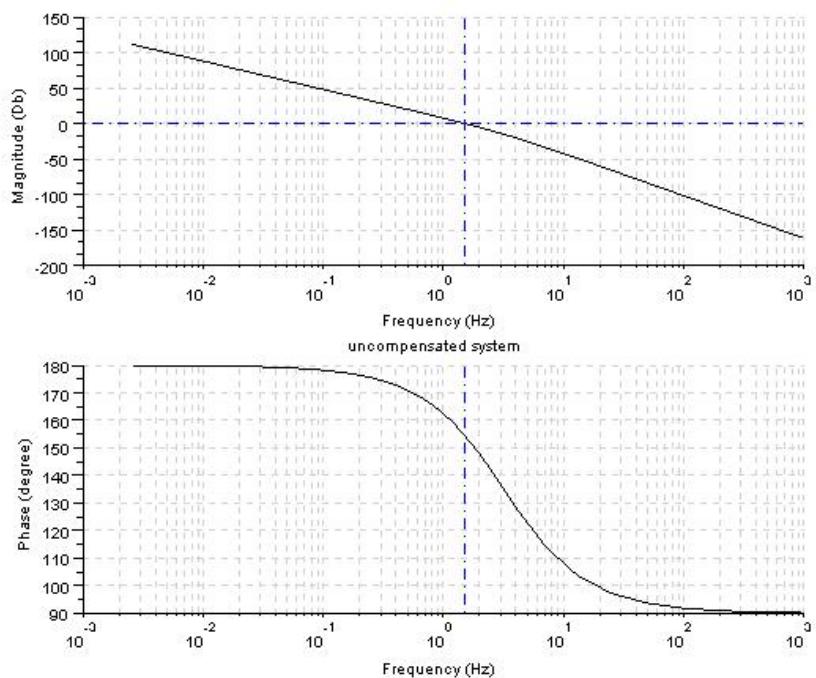
Figure 8.4: design phase lead compensator

Scilab code Exa 8.6.3 design suitable compensator

```
1  //caption:design_suitable_compensator
2  //example 8.6.3
3  //page 344
4  clc;
5  s=%s;
6  clf();
7  syms K;
8  g=(K/(s*(1+0.5*s)*(1+0.2*s)));
9  Kv=1/0.125//static velocity error coefficient(Kv=
       desired output velocity/steady state error)
10 //since Kv=8, as system is type 1 , so K=Kv;
11 K=8;
12 g=(8/(s*(1+0.5*s)*(1+0.2*s)));
13 G=syslin('c',g)
14 fmin=0.01;
15 fmax=100;
16 bode(G, fmin, fmax)
17 show_margins(G)
18 xtitle("uncompensated system")
19 [gm,freqGM]=g_margin(G);
20 [pm,freqPM]=p_margin(G);
21 disp(gm,"gain_margin=");
22 disp((freqGM*2*%pi),"
       gain_margin_freq_or_phase_cross_over_frequency=="
       );
23 disp(pm,"phase_margin=");
24 disp((freqPM*2*%pi),"
       phase_margin_freq_or_gain_cross_over_frequency=")
       ;
25 disp("since gain crossover freq and phase crossover
       freq are very close to each other. So, system is
       marginally stable");
26 disp("so we need phase lag network ")
27 disp("selecting zero of lead compensating network at
        w=0.18rad/sec and pole at w=0.04rad/sec and
       applying gain to account attenuatin factor .")
```

```
28  gc =(1+5.55* s )/(1+24.7* s )
29  Gc = syslin ( ' c ',gc )
30  disp (Gc ," transfer  function  of  lead  compensator=");
31  G1 =G * Gc
32  disp (G1 ," overall  transfer  function=");
33  fmin =0.01;
34  fmax =100;
35  figure ();
36  bode (G1 ,  fmin ,  fmax )
37  show_margins (G1 )
38  xtitle (" compensated  system ")
39  [gm ,freqGM]= g_margin (G1 );
40  [pm ,freqPM]= p_margin (G1 );
41  disp (gm ," gain_margin=");
42  disp ((freqGM*2*%pi) ,"
        gain_margin_freq_or_phase_cross_over_frequency=="
        );
43  disp (pm ," phase_margin_of_compensated_system=");
44  disp ((freqPM*2*%pi) ," gain_cross_over_frequency=");
```
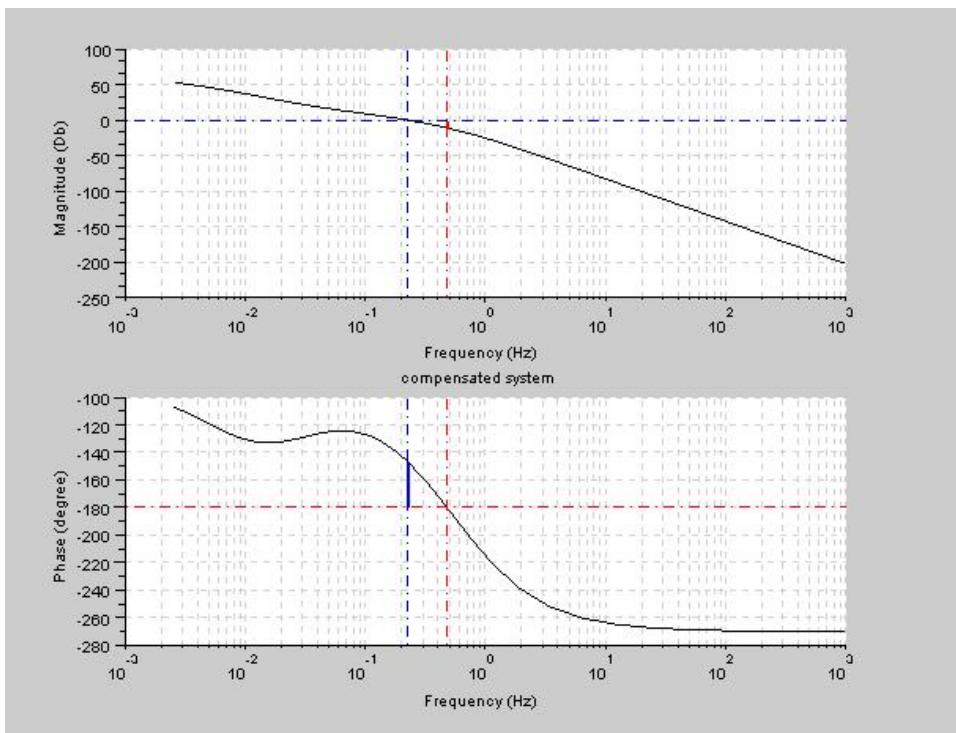
Figure 8.5: design suitable compensator
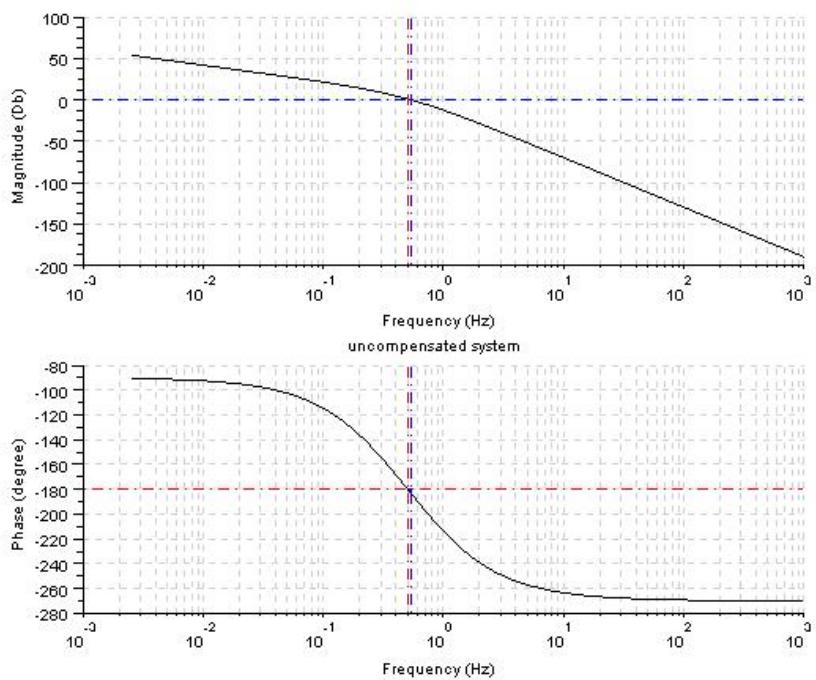
Figure 8.6: design suitable compensator

Chapter 9

INTRODUCTION TO STATE SPACE ANALYSIS OF CONTROL SYSTEMS

Scilab code Exa 9.8.1 Check for Contrallability of System

```
1  //caption:check_for_contrallability_of_system
2  //example 9.8.1
3  //page 381
4  A=[1 1;0 -1]
5  B=[1;0]
6  P=cont_mat(A,B);
7  disp(P,"Controllability Matrix=");
8  d=determ(P)
9  if d==0
10    printf("matrix is singular, so system is
          uncontrollable");
11  else
12    printf("system is controllable");
13  end;
```

Scilab code Exa 9.8.2 Check for Contrallability of System

```scilab
1  // caption : check_for_contrallability_of_system
2  // example  9.8.2
3  // page  381
4  A =[-2  0;0  -1]
5  B =[1;1]
6  P = cont_mat (A ,B);
7  disp(P,"Controllability  Matrix=");
8  d = determ (P)
9  if  d ==0
10    printf("matrix  is  singular ,  so  system  is
           uncontrollable");
11 else
12    printf("system  is  controllable");
13 end;
```

Scilab code Exa 9.9.1.a Check for Observability of System

```scilab
1  // caption : check_for_observability_of_system
2  // example  9.9.1_a
3  // page  383
4  A =[0  1; -2  -3]
5  B =[0;1]
6  C =[1  1]
7  P = obsv_mat (A ,C);
8  disp(P,"Observability  Matrix=");
9  d = determ (P)
10 if  d ==0
11    printf("matrix  is  singular ,  so  system  is
           unobservable");
12 else
13    printf("system  is  observable");
14 end;
```

Scilab code Exa 9.9.1.b Check for Observability of System

```
1  //caption:check_for_observability_of_system
2  //example 9.9.1_b
3  //page 383
4  A=[-2 1;0 1]
5  B=[1;1]
6  C=[1 1]
7  P=obsv_mat(A,C);
8  disp(P,"Observability Matrix=");
9  d=determ(P)
10 if d==0
11   printf("matrix is singular, so system is
          unobservable");
12 else
13   printf("system is observable");
14 end;
```

Scilab code Exa 9.10.4.a Obtain State Matrix

```
1  //caption:obtain_state_matrix
2  //example 9.10.4_a
3  //page 387
4  s=%s;
5  g=1/((s+1)*(s+3));
6  CL=syslin('c',g);
7  disp(CL,"C(s)/R(s)=");
8  SS=tf2ss(CL)
9  [Ac,Bc,U,ind]=canon(SS(2),SS(3))
10 disp(SS,"state space matrix=")
11 disp(Ac,"Ac",Bc,"Bc",U,"U",ind,"ind")
```

150

Scilab code Exa 9.10.4.b Obtain State Matrix

```
1  //caption:obtain_state_matrix
2  //example 9.10.4_b
3  //page 387
4  clc;
5  s=%s;
6  g=(2*s+1)/(s^2+2);
7  CL=syslin('c',g);
8  disp(CL,"C(s)/R(s)=");
9  SS=tf2ss(CL)
10 [Ac,Bc,U,ind]=canon(SS(2),SS(3))
11 disp(SS,"state space matrix=")
12 disp(Ac,"Ac",Bc,"Bc",U,"U",ind,"ind")
```

Scilab code Exa 9.10.5 Obtain State Matrix

```
1  //caption:obtain_state_matrix
2  //example 9.10.5
3  //page 388
4  s=%s;
5  g=(s+3)/(s^2+3*s+4);
6  CL=syslin('c',g);
7  disp(CL,"C(s)/R(s)=");
8  SS=tf2ss(CL)
9  [Ac,Bc,U,ind]=canon(SS(2),SS(3))
10 disp(SS,"state space matrix=")
11 disp(Ac,"Ac",Bc,"Bc",U,"U",ind,"ind")
```

Scilab code Exa 9.10.6 Obtain State Matrix

```
1  // caption : obtain_state_matrix
2  // example  9.10.6
3  // page  389
4  s=%s ;
5  g=(s^2+3*s+2)/(s^3+9*s^2+26*s+24);
6  CL=syslin ( 'c' ,g);
7  disp(CL ,"C( s )/R( s )=");
8  SS=tf2ss(CL)
9  [Ac ,Bc ,U,ind]=canon (SS(2),SS(3))
10 disp(SS ,"state  space  matrix=")
11 disp(Ac ,"Ac" ,Bc ,"Bc" ,U,"U" ,ind ,"ind")
```

Scilab code Exa 9.10.7 Obtain State Matrix

```
1  // caption : obtain_state_matrix
2  // example  9.10.7
3  // page  390
4  s=%s ;
5  g=(s+3)/((s+1)*(s+2));
6  CL=syslin ( 'c' ,g);
7  disp(CL ,"C( s )/R( s )=");
8  SS=tf2ss(CL)
9  [Ac ,Bc ,U,ind]=canon (SS(2),SS(3))
10 disp(SS ,"state  space  matrix=")
11 disp(Ac ,"Ac" ,Bc ,"Bc" ,U,"U" ,ind ,"ind")
```

Scilab code Exa 9.10.9 Obtain State Matrix

```
1  // caption : obtain_state_matrix
2  // example  9.10.9
3  // page  392
```

```
4  s=%s;
5  g=(10*(s+3))/((s+4)*(s+2)^2);
6  CL=syslin('c',g);
7  disp(CL,"C(s)/R(s)=");
8  SS=tf2ss(CL)
9  [Ac,Bc,U,ind]=canon(SS(2),SS(3))
10 disp(SS,"state space matrix=")
11 disp(Ac,"Ac",Bc,"Bc",U,"U",ind,"ind")
```

Scilab code Exa 9.10.10 Obtain State Matrix

```
1  //caption:obtain_state_matrix
2  //example 9.10.10
3  //page 393
4  s=%s;
5  g=(s^2+6*s+8)/((s+3)*(s^2+2*s+5));
6  CL=syslin('c',g);
7  disp(CL,"C(s)/R(s)=");
8  SS=tf2ss(CL)
9  [Ac,Bc,U,ind]=canon(SS(2),SS(3))
10 disp(SS,"state space matrix=")
11 disp(Ac,"Ac",Bc,"Bc",U,"U",ind,"ind")
```

Scilab code Exa 9.10.11 Obtain Time Response

```
1  //caption:obtain_time_response
2  //example 9.10.11
3  //page 394
4  s=%s;
5  syms t
6  A=[0 1;-2 0]
7  B=[1 -1]
8  x0=[1 1]'
```

153

```
9  [r c]=size(A);//size of matrix A
10 p=s*eye(r,c)-A;//s*I-A where I is identity matrix
11 q=det(p)//determinant of sI-A
12 r=inv(p)//inverse of sI-A
13 //for calculating state transistion matrix
14 ip=[0 0;0 0]
15 i=1;j=1;
16 for i=1:2
17   for j=1:2
18     ip(i,j)=ilaplace(r(i,j),s,t);
19     j=j+1;
20   end
21   i=i+1;
22 end
23 disp(ip,"state transistion matrix,ip(t)=");
24 x=ip*x0
25 y=x(1,1)-x(2,1)
26 y=simple(y) //output
27 disp(y,"time response of the system,y(t)=");
```

Scilab code Exa 9.10.12.i Obtain Zero Input Response

```
1  //caption:obtain_zero_input_response
2  //example 9.10.12_(i)
3  //page 395
4  s=%s;
5  syms t
6  A=[0 1;-2 -1]
7  B=[0 1]'
8  x0=[1 0]'
9  [r c]=size(A);//size of matrix A
10 p=s*eye(r,c)-A;//s*I-A where I is identity matrix
11 q=det(p)//determinant of sI-A
12 r=inv(p)//inverse of sI-A
13 //for calculating state transistion matrix
```

```
14  ip =[0  0;0  0]
15  i =1; j =1;
16  for  i =1:2
17    for  j =1:2
18      ip ( i , j )= ilaplace ( r ( i , j ) , s , t ) ;
19      j = j +1;
20    end
21    i = i +1;
22  end
23  disp ( ip , " state  transistion  matrix , ip ( t )= " ) ;
24  x = ip * x0
25  disp ( x , " zero  input  response  of  the  system , x ( t )= " ) ;
```

Scilab code Exa 9.10.12.ii Obtain Zero State Response

```
1  // caption : obtain_zero_state_response
2  // example  9.10.12_( ii )
3  // page  395
4  s =%s ;
5  syms  t
6  A =[0  1; -2  -1]
7  B =[0  1] '
8  x0 =[1  0] '
9  [ r  c ]= size ( A ) ; // size  of  matrix  A
10  p = s * eye ( r , c ) -A ; // s * I -A  where  I  is  identity  matrix
11  q = det ( p ) // determinant  of  sI -A
12  r = inv ( p ) // inverse  of  sI -A
13  m = r * B * (1/ s ) ;
14  // for  calculating  zero  state  response
15  x =[0;0]
16  x ( 1 , 1 )= ilaplace ( m ( 1 , 1 ) , s , t ) ;
17  x ( 2 , 1 )= ilaplace ( m ( 2 , 1 ) , s , t ) ;
18  disp ( x , " zero  input  response  of  the  system , x ( t )= " ) ;
```

Scilab code Exa 9.10.13 Obtain Time Response

```
1  //caption:obtain_time_response
2  //example 9.10.13
3  //page 397
4  syms t;
5  A=[0 1;-2 0]
6  x0=[1 1]'
7  [r c]=size(A);//size of matrix A
8  //since exp(At)=I+A*t+(A*t)^2/2+(A*t)^3/3+...
9  I=eye(r,c)
10 p=I+A*t+(A*t)^2/2+(A*t)^3/3
11 x=p*x0;
12 disp(x(1,1),"time response of the system,x1(t)=");
13 disp(x(2,1),"time response of the system,x2(t)=");
```

Scilab code Exa 9.10.14 Obtain Time Response using Diagonalization Process

```
1  //caption:
       obtain_time_response_using_diagonalization_process
2  //example 9.10.14
3  //page 398
4  syms m11 m22 m21 m12 t;
5  s=%s;
6  poly(0,"l");
7  A=[1 4;-2 -5]
8  B=[0;1]
9  C=[1;0]'
10 x0=[1 0]'
11 [r c]=size(A);//size of matrix A
```

```
12  I=eye(r,c);
13  p=l*I-A;//l*I-A where I is identity matrix
14  q=det(p)//determinant of lI-A
15
16  //roots of q are -1 and -3
17  l1=-1;
18  l2=-3;
19  //for determining modal matrix
20  x1=[m11;m21]
21  q1=(l1*I-A)*x1
22  //on solving we find m11=1, m21=-0.5
23  m11=1;m21=-0.5
24  x2=[m12;m22]
25  q2=(l2*I-A)*x1
26  //on solving we find m12=1, m22=-1
27  m12=1;m22=-1
28  M=[m11 m12;m21 m22]
29  M1=inv(M);
30  k1=M1*A*M;
31  z0=M1*x0;
32  k2=M1*B
33  Z=inv(s*I-k1)*z0+(inv(s*I-k1)*k2)*(1/s);
34  X=M*Z
35  x=[0;0]
36  x(1,1)=ilaplace(X(1,1),s,t);
37  x(2,1)=ilaplace(X(2,1),s,t);
38  y=C*x
39  disp(y,"output_equation_is ,y(t)=")
```

Scilab code Exa 9.10.15 Obtain Time Response using Diagonalization Process

```
1  //caption:
       obtain_time_response_using_diagonalization_process
```

```
2  //example 9.10.15
3  //page 403
4  syms t;
5  s=%s;
6  poly(0,"l");
7  A=[0 1;-6 -5]
8  B=[0;1]
9  C=[6 1]
10 x0=[1 0]'
11 [r c]=size(A);//size of matrix A
12 I=eye(r,c);
13 p=l*I-A;//l*I-A where I is identity matrix
14 q=det(p)//determinant of lI-A
15 //roots of q are -1 and -3
16 l1=-2;
17 l2=-3;
18 //for determining vandermonde's matrix
19 P=[1 1;l1 l2];
20 P1=inv(P);
21 k1=P1*A*P;
22 z0=P1*x0;
23 k2=P1*B
24 Z=inv(s*I-k1)*z0+(inv(s*I-k1)*k2)*(1/s);
25 X=P*Z
26 X(2,1)=-5/(s^2+5*s+6)
27 x=[0;0]
28 x(1,1)=ilaplace(X(1,1),s,t);
29 x(2,1)=ilaplace(X(2,1),s,t);
30 y=C*x
31 y=simple(y)
32 disp(y,"output_equation_is ,y(t)=")
```

Scilab code Exa 9.10.16 Determine Transfer Matrix

```
1  //caption:determine_transfer_matrix
```

```
2  //example 9.10.16
3  //page 406
4  clc;
5  s=%s
6  A=[1 -2;4 -5]
7  B=[2;1]
8  C=[1 1]
9  [r c]=size(A);//size of matrix A
10 p=s*eye(r,c)-A;//s*I-A where I is identity matrix
11 r=inv(p)//inverse of sI-A
12 G=C*r*B//transfer matrix
13 disp(G,"transfer matrix=")
```

Scilab code Exa 9.10.17 Determine Transfer Matrix

```
1  //caption:determine_transfer_matrix
2  //example 9.10.17
3  //page 407
4  s=%s
5  A=[-3 1;0 -1]
6  B=[1;1]
7  C=[1 1]
8  D=0;
9  [r c]=size(A);//size of matrix A
10 p=s*eye(r,c)-A;//s*I-A where I is identity matrix
11 r=inv(p)//inverse of sI-A
12 G=C*r*B//transfer matrix
13 disp(G,"transfer matrix=")
```

Scilab code Exa 9.10.18 Determine Transfer Matrix

```
1  //caption:determine_transfer_matrix
2  //example 9.10.17
```

159

```
3  //page 407
4  s=%s
5  A=[0 3;-2 -5]
6  B=[1 1;1 1]
7  C=[2 1;1 0]
8  D=0;
9  [r c]=size(A);//size of matrix A
10 p=s*eye(r,c)-A;//s*I-A where I is identity matrix
11 r=inv(p)//inverse of sI-A
12 G=C*r*B//transfer matrix
13 disp(G,"transfer matrix=")
```

Scilab code Exa 9.10.20 Check for Contrallability of System

```
1  //caption:check_for_contrallability_of_system
2  //example 9.10.20
3  //page 410
4  A=[0 1 0;0 0 1;0 -2 -3]
5  B=[0 1;0 0;1 1]
6  P=cont_mat(A,B);
7  disp(P,"Controllability Matrix=");
8  S=[P(1) P(4) P(7);P(2) P(5) P(8);P(3) P(6) P(9)];//
      collecting columns from P to form a square matrix
      (3*3)
9  d=det(S);
10 if d==0
11   printf("matrix is singular, so system is
        uncontrollable");
12 else
13   printf("system is controllable");
14 end;
```

Scilab code Exa 9.10.21 Check for Contrallability and Observability

```
1  // caption :
       check_for_contrallability_and_observability_of_system

2  // example  9.10.21
3  // page  411
4  A =[0  1  0;0  0  1; -6  -11  -6]
5  B =[1  0  1] '
6  C =[10  5  1]
7  P = cont_mat (A ,B ) ;
8  disp (P ," Controllability  Matrix=") ;
9  d = det (P )
10 if  d ==0
11    printf (" matrix  is  singular ,  so  system  is
          uncontrollable ") ;
12 else
13    printf (" system  is  controllable ") ;
14 end ;
15 P1 = obsv_mat (A ,C ) ;
16 disp (P1 ," Observability  Matrix=") ;
17 d1 = det (P1 )
18 if  d1 ==0
19    printf (" matrix  is  singular ,  so  system  is
          unobservable ") ;
20 else
21    printf (" system  is  observable ") ;
22 end ;
```

Scilab code Exa 9.10.22 Check for Contrallability and Observability

```
1  // caption :
       check_for_contrallability_and_observability_of_system

2  // example  9.10.21
3  // page  411
4  s =% s ;
```

```
5  g=(10*(s+3))/((s+4)*(s+2)^2);
6  CL=syslin('c',g);
7  disp(CL,"Y(s)/U(s)=");
8  SS=tf2ss(CL)
9  [Ac,Bc,U,ind]=canon(SS(2),SS(3))
10 A=SS(2)
11 B=SS(3)
12 C=SS(4)
13 P=cont_mat(A,B);
14 P=round(P)
15 disp(P,"Controllability Matrix=");
16 d=det(P)
17 if d==0
18   printf("matrix is singular, so system is
         uncontrollable");
19 else
20   printf("system is controllable");
21 end;
22 P1=obsv_mat(A,C);
23 P1=round(P1);
24 disp(P1,"Observability Matrix=");
25 d1=det(P1)
26 if d1==0
27   printf("matrix is singular, so system is
         unobservable");
28 else
29   printf("system is observable");
30 end
```

Chapter 11

SOLUTION OF PROBLEMS USING COMPUTER

Scilab code Exa 11.1 pole zero Plot

```
1 // Caption : pole_zero_plot
2 // example 11_1
3 // page 468
4 // transfer function :G(s)=((8*(s+3)*(s+4))/(s*(s+2)*(
     s^2+2*s+2)))
5 s=%s ;
6 G=syslin ('c ',((8*(s+3)*(s+4))/(s*(s+2)*(s^2+2*s+2)))
     ) ;
7 disp (G ,"G(s)=") ;
8 x=plzr (G)
9 xgrid (2)
```

Scilab code Exa 11.2 transfer function
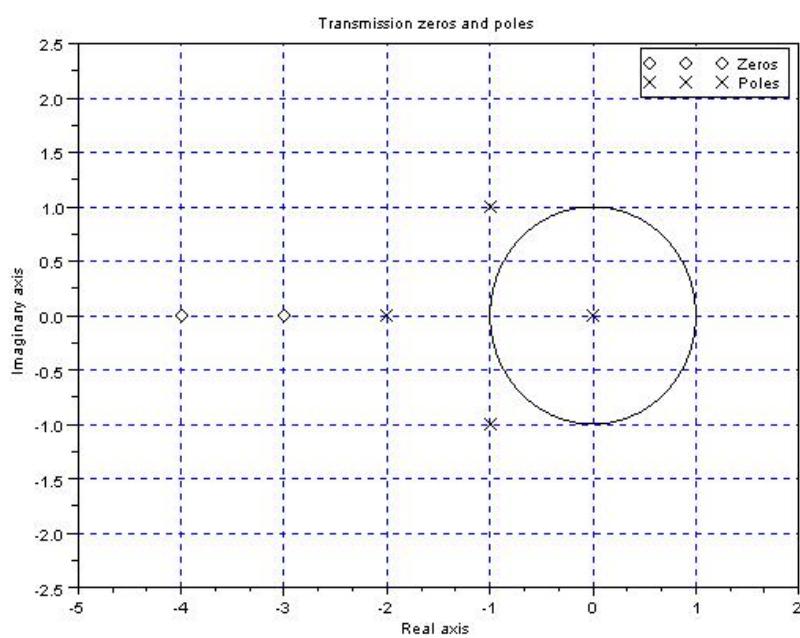
Figure 11.1: pole zero Plot

```
1  //Caption : transfer_function
2  // example 11_2
3  //page 469
4  syms G1 G2 G3 H1;
5  s=%s;
6  G1=4/(s*(s+4));
7  G2=s+1.2;
8  G3=s+0.8;
9  H1=1;
10  H2=(G2+G3);
11  a=G1/.H1;
12  y=a/(1+a*H2)
13  disp(y,"C(s)/R(s)=")
```

Scilab code Exa 11.3 transfer function

```
1  //Caption : transfer_function
2  // example 11_3
3  //page 470
4  syms G1 G2 G3 H1;
5  s=%s;
6  G1=4;
7  G2=s
8  G3=1/(s*(s+2));
9  H1=0.5;
10  H2=1;
11  a=(G1+G2);
12  b=(a*G3);
13  c=b/.H1;
14  y=c/(1+c*H2)
15  disp(y,"C(s)/R(s)=")
```

Scilab code Exa 11.4 determine Wn zeta and Mp

```
1  // caption : determine_Wn , zeta_and_Mp
2  // example 11_4
3  // page 471
4  s=%s;
5  num=1;
6  den=sym('s*(1+0.5*s)*(1+0.2*s)');
7  c=num/den;
8  c=simple(c);
9  disp(c,"C(s)/E(s)=");
10 G=c;
11 H=1;
12 CL=G/(1+G*H);
13 CL=simple(CL);
14 disp(CL,"C(s)/R(s)=");
15 A=pfss((10/(s^3+7*s^2+10*s+10)));
16 d=denom(A(1));
17 b=coeff(denom(A(1)))
18 printf("for oscillatory roots:")
19 Wn=sqrt(b(1,1));// natural_frequency
20 //2*zeta*Wn=1.5;
21 zeta=1.5/(2*Wn);// damping ratio
22 Mp=exp((-zeta*%pi)/sqrt(1-zeta^2))*100;//
       %_peak_overshoot
23 disp(Wn,"natural_frequency ,Wn=");
24 disp(zeta,"damping ratio ,zeta=");
25 disp(Mp," %_peak_overshoot ,Mp=");
```

Scilab code Exa 11.5 time response for unit step function

```
1  // Caption : time_response_for_unit_step_function
2  // example 11_6
3  // page 474
4  s=%s;
5  syms t;
6  G=(s+2)/(s*(s+1))
```

```
7   H=1;
8   CL=G/.H
9   disp(CL,"C(s)/R(s)=")
10  y=ilaplace(CL,s,t);
11  disp(y,"c(t)=");
12  b=denom(CL)
13  c=coeff(b)
14  //Wn^2=c(1,1)
15  Wn=sqrt(c(1,1))//natural frequency
16  //2*zeta*Wn=c(1,2)
17  zeta=c(1,2)/(2*Wn)//damping ratio
18  Wd=Wn*sqrt(1-zeta^2)//damped frequency
19  tr=(%pi-atan(sqrt(1-zeta^2)/zeta))/(Wn*sqrt(1-zeta
        ^2))
20  Mp=(exp(-(zeta*%pi)/sqrt(1-zeta^2)))*100//max.
        overshoot
21  disp(tr,"rise time=");
22  disp(Mp,"max. peak overshoot=");
```

Scilab code Exa 11.7 time response for unit step function

```
1   //Caption:time_response_for_unit_step_function
2   //example 11_7
3   //page 475
4   s=%s;
5   syms t;
6   CL=(s+2)/(s^2+2*s+2)
7   disp(CL,"C(s)/R(s)=")
8   y=ilaplace(CL,s,t);
9   disp(y,"c(t)=");
10  b=denom(CL)
11  c=coeff(b)
12  //Wn^2=c(1,1)
13  Wn=sqrt(c(1,1))//natural frequency
14  //2*zeta*Wn=c(1,2)
```

```
15  zeta=c(1,2)/(2*Wn)//damping ratio
16  Wd=Wn*sqrt(1-zeta^2)//damped frequency
17  Tp=%pi/Wd//peak time
18  Mp=(exp(-(zeta*%pi)/sqrt(1-zeta^2)))*100//max.
        overshoot
19  disp(Tp,"peak time=");
20  disp(Mp,"max. peak overshoot=")
```

Scilab code Exa 11.8 time response for unit step function

```
1  //Caption:time_response_for_unit_step_function
2  //example 11_8
3  //page 476
4  s=%s;
5  syms t;
6  G=(20)/((s+1)*(s+5))
7  H=1;
8  CL=G/.H
9  disp(CL,"C(s)/R(s)=")
10  y=ilaplace(CL,s,t);
11  disp(y,"c(t)=");
12  b=denom(CL)
13  c=coeff(b)
14  //Wn^2=c(1,1)
15  Wn=sqrt(c(1,1))//natural frequency
16  //2*zeta*Wn=c(1,2)
17  zeta=c(1,2)/(2*Wn)//damping ratio
18  Wd=Wn*sqrt(1-zeta^2)//damped frequency
19  Mp=(exp(-(zeta*%pi)/sqrt(1-zeta^2)))*100//max.
        overshoot
20  Tp=%pi/Wd//peak time
21  t=(2*%pi)/(Wn*sqrt(1-zeta^2))//period of oscillation
22  ts=4/(zeta*Wn)//settling time
23  N=Wd/(2*%pi)*ts//no. of oscillations completed
        before reaching steady state
```

```
24  disp(Tp,"peak time=");
25  disp(Mp,"max. peak overshoot=");
26  disp(t,"period of oscillation");
27  disp(N,"no. of oscillations completed before
        reaching steady state=");
28  disp(ts,"settling time=")
```

Scilab code Exa 11.9 time response for unit step function

```
1   //Caption:time_response_for_unit_step_function
2   //example 11_9
3   //page 476
4   s=%s;
5   syms t;
6   CL=(4*s+4)/(s^2+2*s+5)
7   disp(CL,"C(s)/R(s)=")
8   y=ilaplace(CL,s,t);
9   disp(y,"c(t)=");
10  b=denom(CL)
11  c=coeff(b)
12  //Wn^2=c(1,1)
13  Wn=sqrt(c(1,1))//natural frequency
14  //2*zeta*Wn=c(1,2)
15  zeta=c(1,2)/(2*Wn)//damping ratio
16  Wd=Wn*sqrt(1-zeta^2)//damped frequency
17  Mp=(exp(-(zeta*%pi)/sqrt(1-zeta^2)))*100//max.
        overshoot
18  Tp=%pi/Wd//peak time
19  disp(Tp,"peak time=");
20  disp(Mp,"max. peak overshoot=");
```

Scilab code Exa 11.10.a calculate tr Tp Mp

169

```
1  //Caption:calculate_tr ,Tp,Mp
2  //example 11_10_a
3  //page 478
4  s=%s;
5  G=16/(s^2+1.6*s)
6  H=1;
7  CL=G/.H
8  disp(CL,"C(s)/R(s)=");
9  b=denom(CL)
10 c=coeff(b)
11 //Wn^2=c(1,1)
12 Wn=sqrt(c(1,1))//natural frequency
13 //2*zeta*Wn=c(1,2)
14 zeta=c(1,2)/(2*Wn)//damping ratio
15 Wd=Wn*sqrt(1-zeta^2)//damped frequency
16 Mp=(exp(-(zeta*%pi)/sqrt(1-zeta^2)))*100//max.
       overshoot
17 tr=(%pi-atan(sqrt(1-zeta^2)/zeta))/(Wn*sqrt(1-zeta
       ^2))//rise time
18 Tp=%pi/Wd//peak time
19 disp(Tp,"peak time=");
20 disp(Mp,"max. peak overshoot=")
21 disp(tr,"rise time=")
```

Scilab code Exa 11.10.b calculate Td tr Tp Mp

```
1  //Caption:calculate_Td ,tr ,Tp,Mp
2  //example 11_10_b
3  //page 478
4  s=%s;
5  syms Td
6  G=16/(s^2+1.6*s)
7  G1=1+s*Td
8  H=1;
9  a=G*G1
```

```
10  CL=a/.H
11  CL=simple(CL)
12  disp(CL,"C(s)/R(s)=");
13  zeta=0.8//given
14  //since zeta=0.8 so 2*zeta*Wn=1.6+16*Td
15  Wn=sqrt(16)
16  //so on solving
17  Td=0.3
18  //so transfer function takes the form:
19  CL1=(16+4.8*s)/(s^2+6.4*s+6)
20  disp(CL1,"C(s)/R(s)=");
21  Wn1=sqrt(16)
22  //2*zeta1*Wn1=6.4
23  zeta1=6.4/(2*Wn1)//damping ratio
24  Wd=Wn1*sqrt(1-zeta1^2)//damped frequency
25  Mp=(exp(-(zeta1*%pi)/sqrt(1-zeta1^2)))*100//max.
        overshoot
26  tr=(%pi-atan(sqrt(1-zeta1^2)/zeta1))/(Wn1*sqrt(1-
        zeta1^2))//rise time
27  Tp=%pi/Wd//peak time
28  disp(Tp,"peak time=");
29  disp(Mp,"max. peak overshoot=")
30  disp(tr,"rise time=")
```

Scilab code Exa 11.11 expression for unit step response

```
1  //Caption:expression_for_unit_step_response
2  // example 11_11
3  //page 481
4  syms G1 G2 G3 H1 t;
5  s=%s;
6  G1=s+1;
7  G2=1/(s+2);
8  H2=1/(s*(s+1));
9  H1=1/(s+2);
```

```
10  a = G1 /( 1+ G1 * H1 ) ;
11  b = a /( 1+ a * H2 )
12  y = b * G2 ;
13  disp ( y , " C ( s )/ R ( s )= " )
14  // for  unit  step  response  R ( s )=1/ s ;
15  C = y *(1/ s )
16  c = ilaplace ( C , s , t )
17  disp ( c , " expression_for_unit_step_response_is= " )
```

Scilab code Exa 11.12 unit step and impulse response

```
1  // Caption : unit_step_and_impulse_response
2  // example  11_12
3  // page  482
4  s =% s ;
5  syms  t ;
6  G =(4* s +1) /(4* s ^2)
7  H =1;
8  CL = G /. H
9  disp ( CL , " C ( s )/ R ( s )= " )
10  y = ilaplace ( CL , s , t ) ;
11  disp ( y , " unit  impulse  response , c ( t )= " ) ;
12  // for  unit  step  response  R ( s )=1/ s ;
13  C = CL *(1/ s )
14  c = ilaplace ( C , s , t )
15  disp ( c , " expression_for_unit_step_response_is= " )
```

Scilab code Exa 11.13 determine transfer function

```
1  // Caption : determine_transfer_function
2  // example  11_13
3  // page  483
4  syms  t ;
```

```
5  f=exp(-t)*(1-cos(2*t))
6  F=laplace(f,t,s);
7  disp(F,"F(s)=")
```

Scilab code Exa 11.14 determine Wn Wd Tp zeta and steady state error

```
1  //caption:determine_Wn ,Wd,Tp,
       zeta_and_steady_state_error
2  //example 11_14
3  //page 484
4  s=%s;
5  syms t;
6  G=20/((s+5)*(s+1))
7  H=1;
8  CL=G/.H
9  disp(CL,"C(s)/R(s)=")
10 b=denom(CL)
11 disp(0,b,"=","the char. eq is:",)
12 Wn=sqrt(25)//natural_frequency
13 //2*zeta*Wn=6
14 zeta=6/(2*Wn);//damping ratio
15 d=zeta*Wn;//damping_factor
16 z=sqrt(1-zeta^2);
17 Wd=Wn*z;//damped_frequency_of_oscillation
18 Mp=exp((-zeta*%pi)/z)*100;//%_max.peak_overshoot
19 Tp=%pi/Wd//peak time
20 tfirst=(2*%pi)/Wd   //time for first under shoot
21 period=(2*%pi)/Wd    //period of oscillation
22 ts=4/(zeta*Wn)//settling time
23 N=(Wd/(2*%pi))*ts//no. of oscillations completed
       before reaching steady state
24 disp(Wn,"natural frequency=");
25 disp(zeta,"damping ratio=");
26 disp(Wd,"damped frequency of oscillation=");
27 disp(Tp,"peak time=");
```

173

```
28  disp(Mp,"%_max.peak overshoot=");
29  disp(tfirst,"time for first under shoot=");
30  disp(period,"period of oscillation=");
31  disp(N,"no. of oscillations completed before
        reaching steady state=");
```

Scilab code Exa 11.15 determine Wn Wd zeta and steady state error

```
1  //caption:determine_Wn,Wd,
      zeta_and_steady_state_error
2  //example 11_15
3  //page 484
4  s=%s;
5  G=sym('25/(s*(s+5))');
6  G=simple(G);
7  H=1;
8  CL=G/.H;
9  CL=simple(CL);
10 disp(CL,"C(s)/R(s)=");
11 printf("the char. eq is:")
12 disp("s^2+5*s+25")
13 Wn=sqrt(25)//natural_frequency
14 //2*zeta*Wn=5
15 zeta=5/(2*Wn);//damping ratio
16 d=zeta*Wn;//damping_factor
17 z=sqrt(1-zeta^2);
18 Wd=Wn*z;//damped_frequency_of_oscillation
19 Mp=exp((-zeta*%pi)/z)*100;//%_max.peak_overshoot
20 //steady state error for unit ramp input is:Ess=(2*
      zeta/Wn)
21 Ess=(2*zeta/Wn);//steady state error
22 disp("part(a):")
23 disp(Wn,"natural_frequency=");
24 disp(zeta,"damping ratio=");
25 disp(Wd,"damped_frequency_of_oscillation=");
```

```
26  disp(Mp,"%_max.peak_overshoot=");
27  disp(Ess,"steady state error=");
28  //if damping ratio is increased from 0.5 to 0.75 by
        incorporating tachometer feedback
29  zeta=0.75;
30  H1=sym('s*Kt')//tachometer_feedback
31  CL1=G/(1+G*H1);
32  CL1=simple(CL1);
33  CL2=CL1/(1+H*CL1);
34  CL2=simple(CL2);
35  disp(CL2,"C(s)/R(s)=");
36  Wn=sqrt(25);
37  //2*zeta*Wn=25*Kt+5;
38  Kt=(2*zeta*Wn-5)/25;//tachometer_gain
39  Mp1=exp((-zeta*%pi)/sqrt(1-zeta^2))*100;//
        %_peak_overshoot
40  disp("After applying tachometer feedback:")
41  disp(Kt,"tachometer_gain=");
42  disp(Mp1,"%_peak_overshoot=");
```

Scilab code Exa 11.16 determine Kp Kv Ka

```
1   //caption:determine_Kp_Kv_Ka
2   //example 11_16
3   //page 485
4   s=%s;
5   syms t;
6   num=10
7   den=sym('s^2+6*s+10');
8   GH=num/den;
9   GH=simple(GH);
10  disp(GH,"G(s)H(s)=");
11  Kp=limit(GH,s,0);//static positional error
        coefficient
12  disp(Kp,"static positional error coefficient=");
```

```
13  Kv=limit(s*GH,s,0);//static velocity error
        coefficient
14  disp(Kv,"static velocity error coefficient=");
15  Ka=limit(s^2*GH,s,0);//static acceleration error
        coefficient
16  disp(Ka,"static acceleration error coefficient=");
```

Scilab code Exa 11.17 determine Kp Kv Ka

```
1  //caption:determine_Kp_Kv_Ka
2  //example 11_16
3  //page 485
4  s=%s;
5  syms t;
6  num=sym('100*(s+2)*(s+40)');
7  den=sym('(s^3*(s^2+4*s+200))');
8  GH=num/den;
9  GH=simple(GH);
10 disp(GH,"G(s)H(s)=");
11 Kp=limit(GH,s,0);//static positional error
        coefficient
12 disp(Kp,"static positional error coefficient=");
13 Kv=limit(s*GH,s,0);//static velocity error
        coefficient
14 disp(Kv,"static velocity error coefficient=");
15 Ka=limit(s^2*GH,s,0);//static acceleration error
        coefficient
16 disp(Ka,"static acceleration error coefficient=");
```

Scilab code Exa 11.18 determine Kp Kv Ka

```
1  //caption:determine_Kp_Kv_Ka
2  //example 11_18
```

```
3  // page 488
4  s=%s;
5  syms t;
6  num=sym('2*(s^2+3*s+20)');
7  den=sym('s*(s+2)*(s^2+4*s+10)');
8  GH=num/den;
9  GH=simple(GH);
10 disp(GH,"G(s)H(s)=");
11 input1=5;
12 Kp=limit(GH,s,0);//static positional error
       coefficient
13 Ess=5*(1/(1+Kp));//steady state error
14 e=(1/(%inf+1));
15 e=0;
16 Ess=e;
17 disp(Kp,"static positional error coefficient=");
18 disp(Ess,"steady state error=");
19 input2=4*t;
20 Kv=limit(s*GH,s,0);//static velocity error
       coefficient
21 Ess=(1/Kv)*4;//steady state error
22 disp(Kv,"static velocity error coefficient=");
23 disp(Ess,"steady state error=");
24 input3=(4*t^2)/2;
25 Ka=limit(s^2*GH,s,0);//static acceleration error
       coefficient
26 Ess=(1/Ka)*4;//steady state error
27 disp(Ka,"static acceleration error coefficient=");
28 disp("steady state error=");
29 disp("infinity")
```

Scilab code Exa 11.19 determine transfer function

```
1  // Caption: determine_transfer_function
2  // example 11_19
```

```
3  // page 489
4  syms t;
5  s=%s;
6  c=0.5+(1.25*exp(-t))-(1.75*exp(-12*t));
7  C=laplace(c,t,s);
8  disp(C,"C(s)=")
```

Scilab code Exa 11.21.a roots of characterstics equation

```
1  // Caption: roots_of_characterstics_equation
2  // example 11_21_a
3  // page 491
4  s=%s;
5  num=210
6  den=sym('s*(s+2)*(s^2+12*s+6)');
7  G=num/den;
8  G=simple(G);
9  H=1;
10 n1=poly([210],'s',"coeff");
11 d1=poly([210 192 390 44 1],'s',"coeff");
12 CL=syslin('c',n1,d1)
13 disp(CL,"C(s)/R(s)=")
14 b=denom(CL)
15 disp(0,"=",b,"the char. eq is:",)
16 r=roots(b);
17 disp(r,"roots of char. eq. are=");
```

Scilab code Exa 11.21.b bode plot

```
1  // Caption: bode_plot
2  // example 11_21_b
```
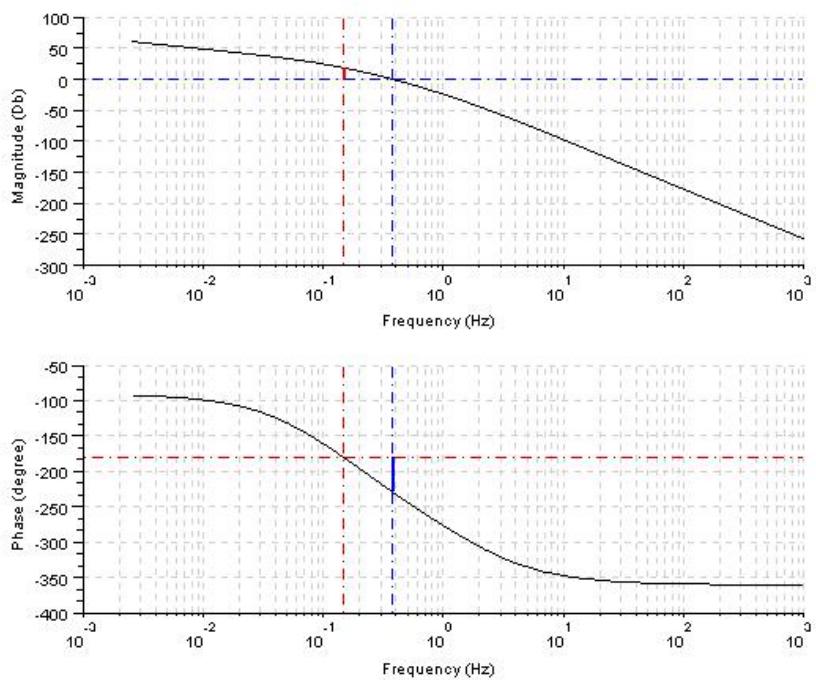
Figure 11.2: bode plot

179

```
3  //page 491
4  s=%s;
5  num=210
6  den=(s*(s+2)*(s^2+12*s+6));
7  g=num/den;
8  G=syslin('c',g)
9  fmin=0.01;
10 fmax=100;
11 bode(G, fmin, fmax)
12 show_margins(G)
13 gm=g_margin(G)
14 pm=p_margin(G)
15 disp(gm,"gain_margin=");
16 disp(pm,"phase_margin=");
```

Scilab code Exa 11.22 gain margin and phase margin

```
1  //caption:gain_margin_and_phase_margin
2  //example 11_22
3  //page 493
4  s=%s;
5  g=((2*(s+0.25))/(s^2*(s+1)*(s+0.5)));
6  G=syslin('c',g)
7  fmin=0.1;
8  fmax=100;
9  bode(G, fmin, fmax)
10 [gm,freqGM]=g_margin(G);
11 [pm,freqPM]=p_margin(G);
12 show_margins(G);
13 disp(gm,"gain_margin=");
14 disp((freqGM*2*%pi),"gain_margin_freq=");
15 disp(pm,"phase_margin=");
16 disp((freqPM*2*%pi),"phase_margin_freq=");
```
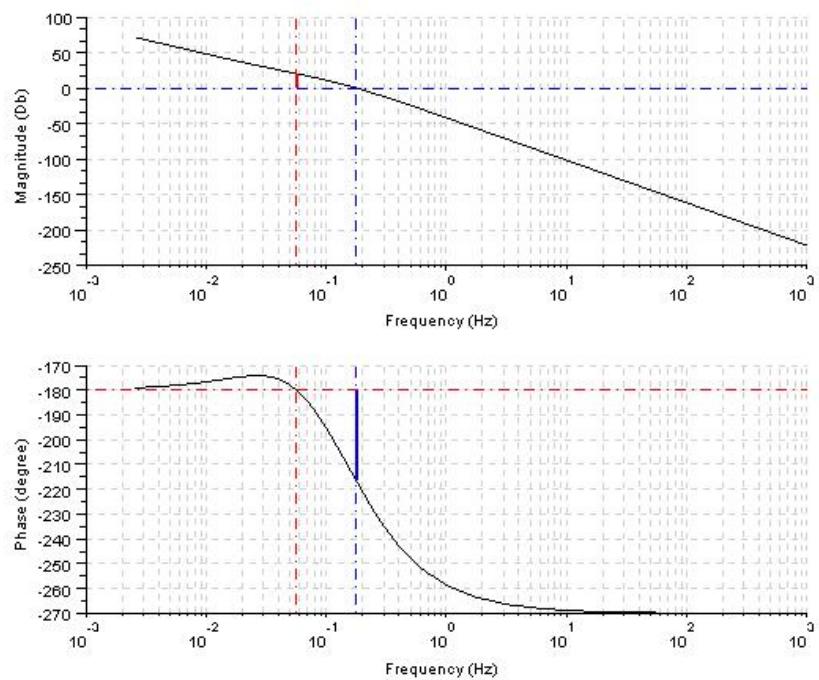
Figure 11.3: gain margin and phase margin
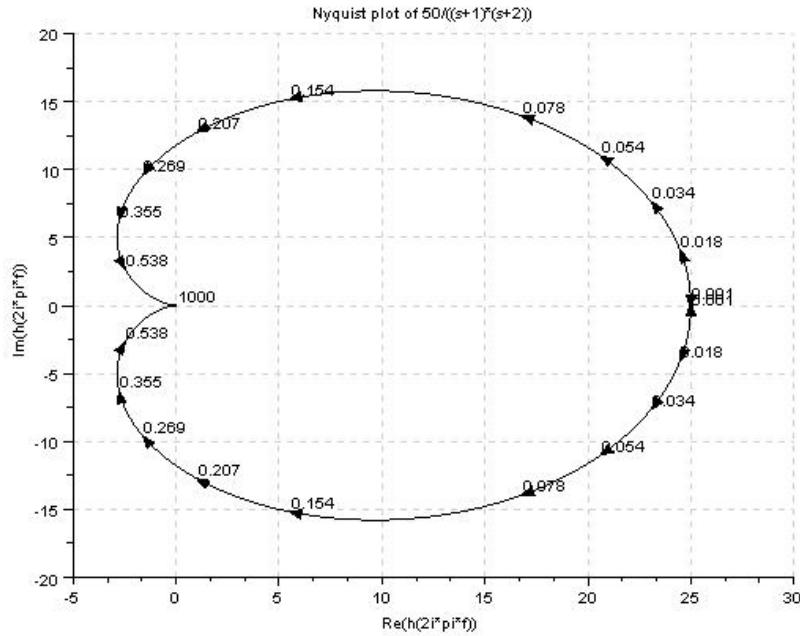
Figure 11.4: stability using Nyquist criterion

```
17  show_margins (G);
18  disp("since  gain  and  phase  margin  are  both  negative
        so  system  is  unstable")
```

Scilab code Exa 11.24 stability using Nyquist criterion

```
1  // caption : stability_using_Nyquist_criterion
2  // example  11_24
3  // page  496
4  clf ;
5  s=%s ;
```

```
 6  s1=-s;
 7  g=50/((s+1)*(s+2));
 8  g1=50/((s1+1)*(s1+2));
 9  GH=syslin('c',g)
10  GH1=syslin('c',g1)
11  nyquist(GH);
12  nyquist(GH1);
13  mtlb_axis([-5 30 -20 20]);
14  xtitle('Nyquist plot of 50/((s+1)*(s+2))')
15  figure;
16  show_margins(GH,'nyquist')
17  disp("since the point(-1+%i0) is not encircled by
        Nyquist plot ,so N=0 and P=0(given)")
18  N=0;//no. of encirclement of -1+%i0 by G(s)H(s) plot
19  P=0;//no. of poles of G(s)H(s) with positive real
        part
20  Z=P-N;//np.of zeros of 1+G(s)H(s)=0 with positive
        real part
21  disp(Z,"Z=")
22  disp("as Z=0,there are no roots of closed loop
        characterstics eq having positive real part,
        hence system is stable.")
```

Scilab code Exa 11.25 stability using Nyquist criterion

```
1  //caption:stability_using_Nyquist_criterion
2  //example 11_25
3  //page 497
4  clf();
5  s=%s;
6  s1=-s;
7  g=(2.2/(s*(s+1)*(s^2+2*s+2)))
8  g1=(2.2/(s1*(s1+1)*(s1^2+2*s1+2)))
```
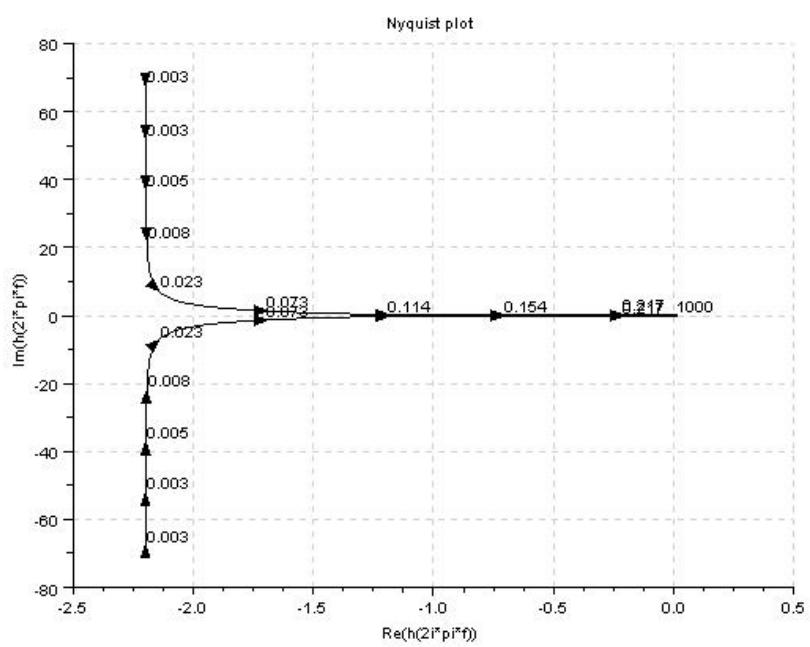
Figure 11.5: stability using Nyquist criterion
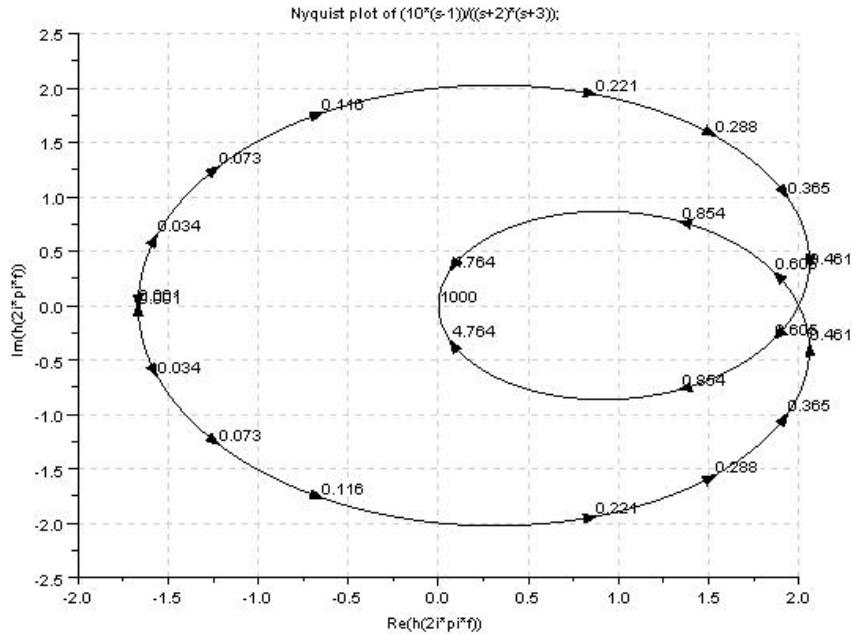
Figure 11.6: stability using Nyquist criterion

```
 9  GH=syslin('c',g);
10  GH1=syslin('c',g1);
11  nyquist(GH);
12  nyquist(GH1);
13  mtlb_axis([-2.5 0.2 -75 75]);
14  disp("as the nyquist plot passes through the point
        -1+%i*0, so system is marginally stable and
        output represents sustained oscillations.")
```

Scilab code Exa 11.26.i stability using Nyquist criterion

```
1  // caption : stability_using_Nyquist_criterion
2  // example 11_26_i
3  // page 497
4  clf();
5  s=%s;
6  s1=-s;
7  disp("for K=10")
8  g=(10*(s-1))/((s+2)*(s+3));
9  g1=(10*(s1-1))/((s1+2)*(s1+3));
10 GH=syslin('c',g);
11 GH1=syslin('c',g1);
12 nyquist(GH);
13 nyquist(GH1);
14 mtlb_axis([-2 2 -2.5 2.5]);
15 xtitle('Nyquist plot of (10*(s-1))/((s+2)*(s+3));')
16 disp("since the point(-1+%i0) is  encircled
       clockwise by Nyquist plot ,so N=-1 and P=1(given)
       ")
17 N=-1;//no. of encirclement of -1+%i0 by G(s)H(s)
       plot anticlockwise
18 P=0;//no. of poles of G(s)H(s) with positive real
       part
19 Z=P-N;//np.of zeros of 1+G(s)H(s)=0 with positive
       real part
20 disp(Z,"Z=")
21 disp("as Z=1,there is one roots of closed loop
       characterstics eq having positive real part,
       hence system is unstable.")
```

Scilab code Exa 11.26.ii stability using Nyquist criterion

```
1  // caption : stability_using_Nyquist_criterion
2  // example 11_26_ii
```
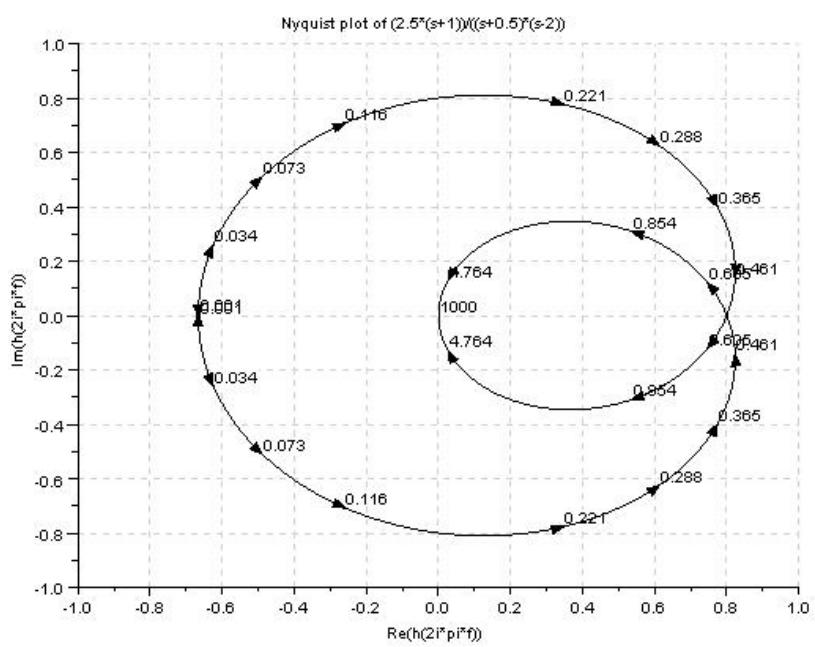
Figure 11.7: stability using Nyquist criterion

```
3  // page 497
4  clf();
5  s=%s;
6  s1=-s;
7  disp("for K=4")
8  g=(4*(s-1))/((s+2)*(s+3));
9  g1=(4*(s1-1))/((s1+2)*(s1+3));
10 GH=syslin('c',g);
11 GH1=syslin('c',g1);
12 nyquist(GH);
13 nyquist(GH1);
14 mtlb_axis([-1 1 -1 1]);
15 xtitle('Nyquist plot of (2.5*(s+1))/((s+0.5)*(s-2))'
      )
16 disp("since the point(-1+%i0) is   encircled
      anticlockwise by Nyquist plot ,so N=1 and P=1(
      given)")
17 N=0;//no. of encirclement of -1+%i0 by G(s)H(s) plot
       anticlockwise
18 P=0;//no. of poles of G(s)H(s) with positive real
      part
19 Z=P-N;//np.of zeros of 1+G(s)H(s)=0 with positive
      real part
20 disp(Z,"Z=")
21 disp("as Z=0,there are no roots of closed loop
      characterstics eq having positive real part,
      hence system is stable.")
```

Scilab code Exa 11.27 root locus

```
1 // caption : root_locus
2 // example 11_27
3 // page 499
```
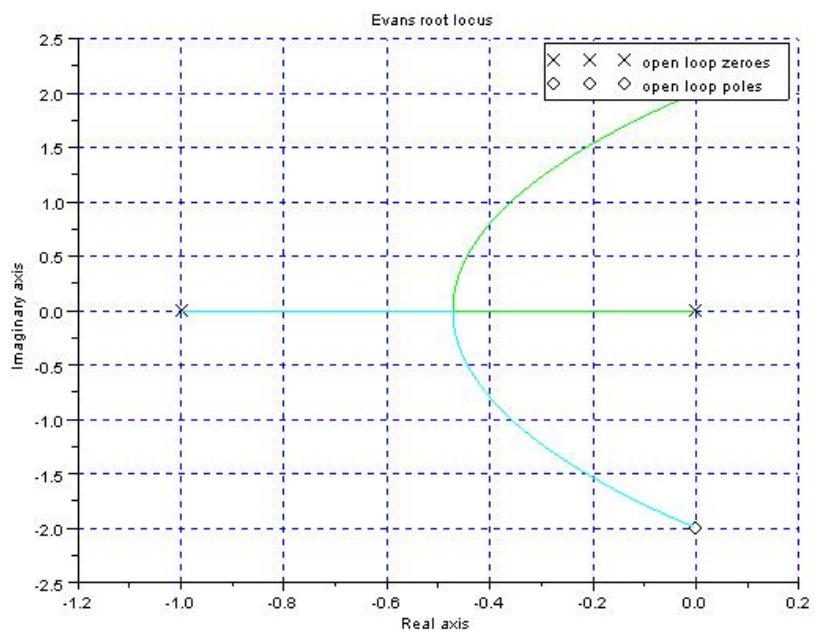
Figure 11.8: root locus
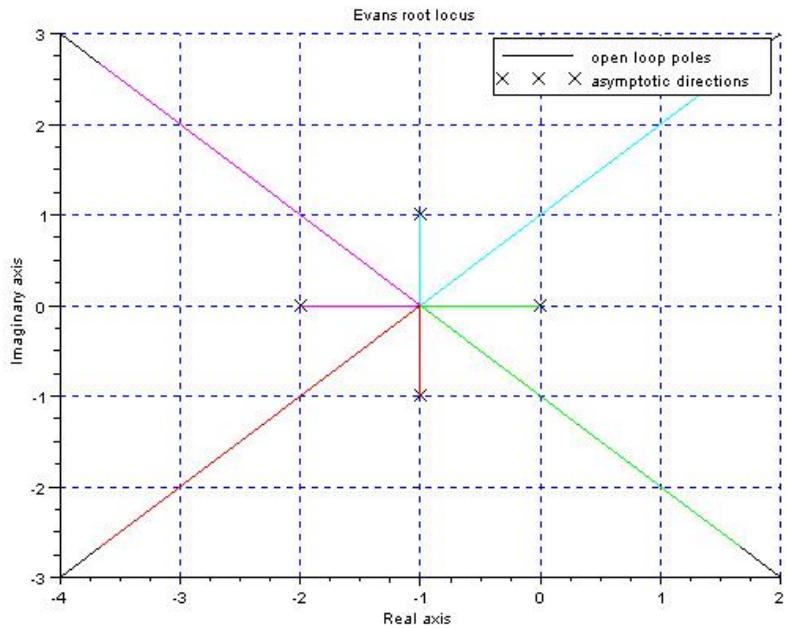
Figure 11.9: root locus

```
4  s=%s ;
5  clf ();
6  g=((s^2+4)/(s*(s+1)));
7  G=syslin('c',g)
8  evans(g,200)
9  xgrid(2)
```

Scilab code Exa 11.28 root locus

```
1  //caption:root_locus
2  //example 11_28
```

190
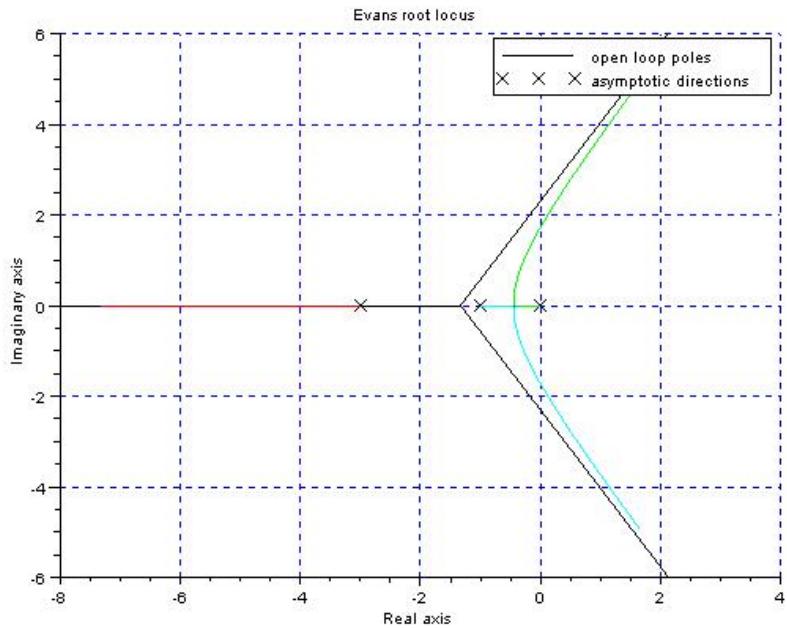
Figure 11.10: root locus

```
3  // page 501
4  s=%s;
5  clf();
6  g=1/(s*(s+2)*(s^2+2*s+2));
7  G=syslin('c',g)
8  evans(g,200)
9  xgrid(2)
```

Scilab code Exa 11.29 root locus

```
1  // caption : root_locus
```
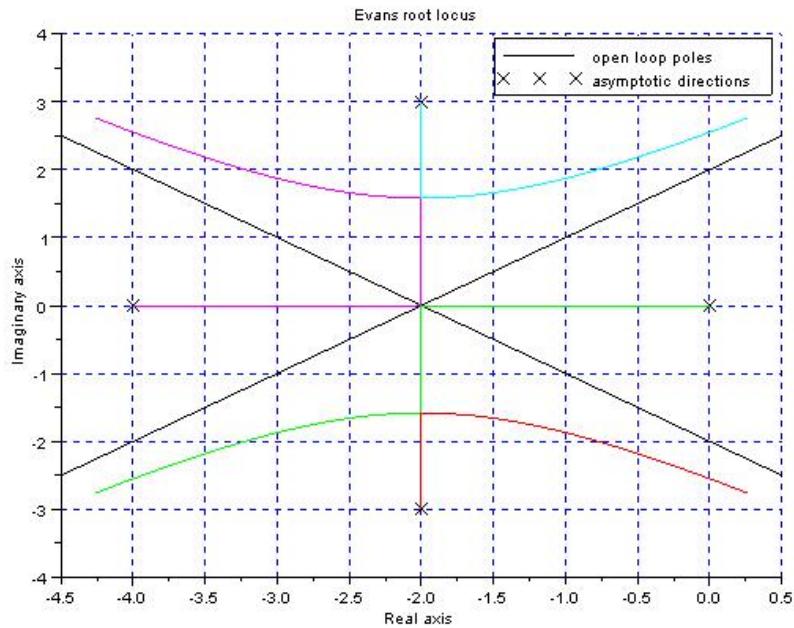
Figure 11.11: root locus

```
2  // example  11_29
3  // page  502
4  s =% s ;
5  g =1/( s *( s +1) *( s +3) ) ;
6  G = syslin ( 'c' , g )
7  evans (g ,200)
8  xgrid (2)
```

Scilab code Exa 11.30 root locus

```
1  // caption : root_locus
```

```
2  //example  11_30
3  //page  503
4  s=%s;
5  g=1/(s*(s+4)*(s^2+4*s+13));
6  G=syslin('c',g)
7  evans(g,200)
8  xgrid(2)
```

Scilab code Exa 11.31 design lead compensator

```
1  //caption:design_lead_compensator
2  //example  11_31
3  //page  339
4  s=%s;
5  clf();
6  syms  K;
7  g=(K/(s*(1+0.2*s)));
8  Kv=limit(s*g,s,0);//static  velocity  error
       coefficient
9  //since  Kv=10
10  K=10;
11  g=(10/(s*(1+0.2*s)));
12  G=syslin('c',g)
13  fmin=0.01;
14  fmax=100;
15  bode(G,  fmin,  fmax)
16  show_margins(G)
17  [gm,freqGM]=g_margin(G);
18  [pm,freqPM]=p_margin(G);
19  disp(gm,"gain_margin=");
20  disp((freqGM*2*%pi),"gain_margin_freq=");
21  disp(pm,"phase_margin=");
22  disp((freqPM*2*%pi),"
       phase_margin_freq_or_gain_cross_over_frequency=")
       ;
```

```
23  disp("since P.M is less than desired value so we
        need phase lead network ")
24  disp("selecting zero of lead compensating network at
         w=5.5rad/sec and pole at w=13.8rad/sec and
        applying gain to account attenuatin factor .")
25  gc=(1+0.18*s)/(1+0.072*s)
26  Gc=syslin('c',gc)
27  disp(Gc,"transfer function of lead compensator=");
28  G1=G*Gc
29  disp(G1,"overall transfer function=");
30  fmin=0.01;
31  fmax=100;
32  figure();
33  bode(G1, fmin, fmax);
34  show_margins(G1)
35  [gm,freqGM]=g_margin(G1);
36  [pm,freqPM]=p_margin(G1);
37  disp(pm,"phase_margin_of_compensated_system=");
38  disp((freqPM*2*%pi),"gain_cross_over_frequency=");
```

Scilab code Exa 11.32 nicholas chart

```
1  //caption: nicholas_chart
2  //example 11_32
3  //page 507
4  s=%s;
5  num=20;
6  den=(s*(s+2)*(s+5))
7  g=num/den
```
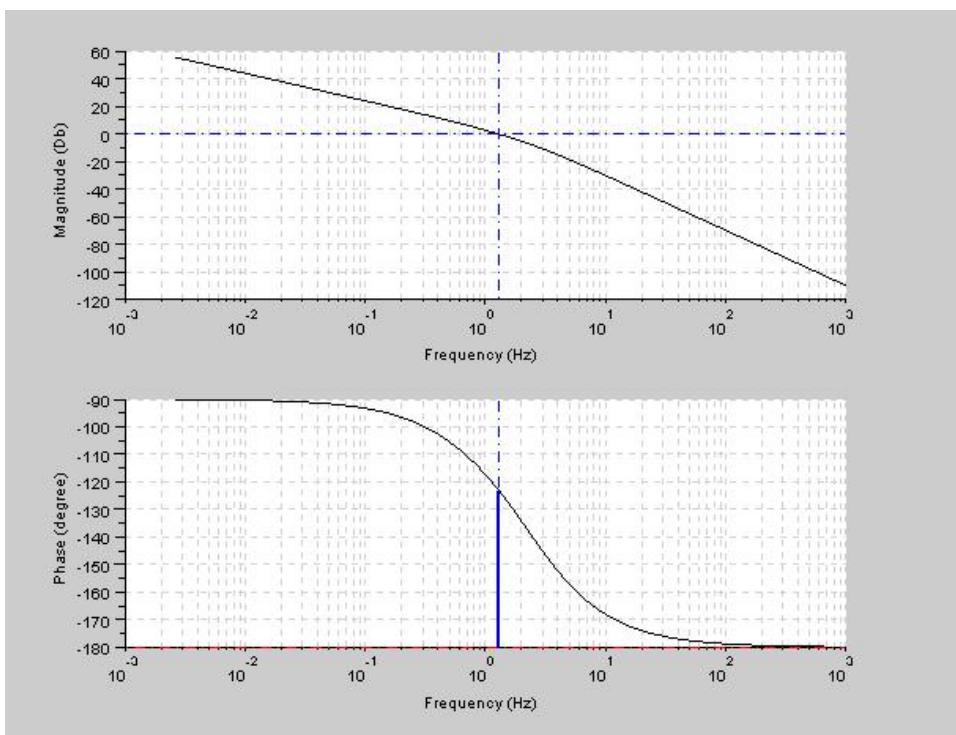
Figure 11.12: design lead compensator
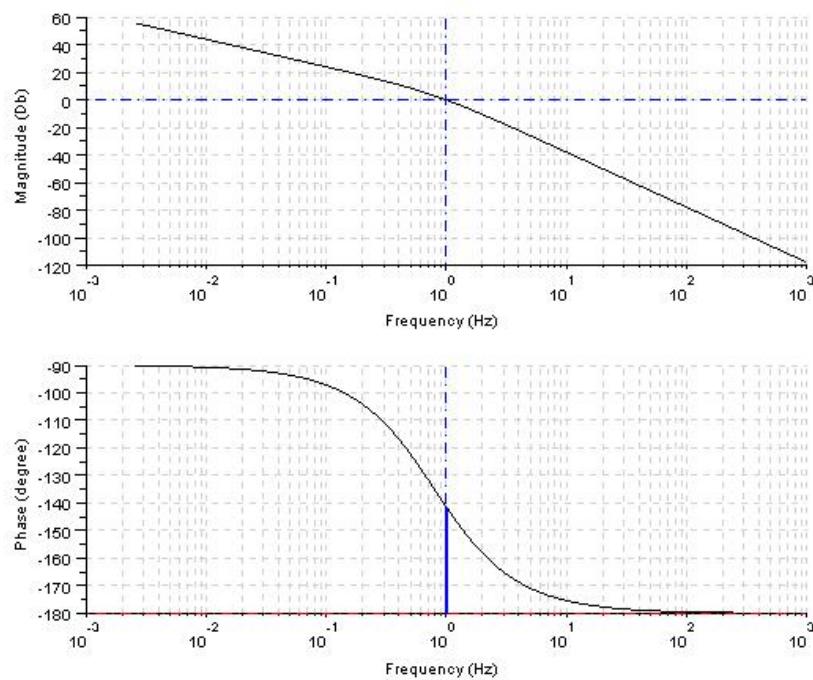
Figure 11.13: design lead compensator
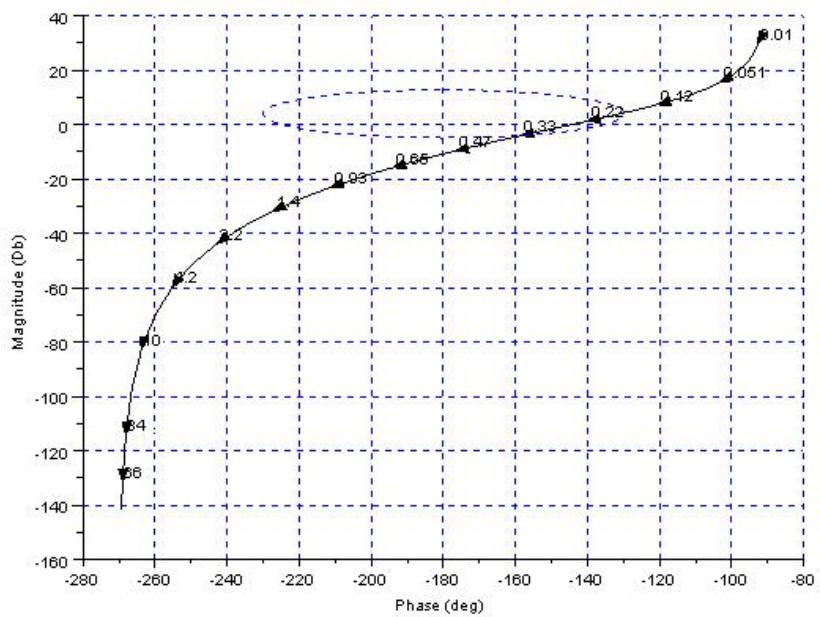
Figure 11.14: nicholas chart

```
 8  G=syslin('c',g)
 9  fmin=0.01
10  fmax=100
11  black(G,fmin,fmax)
12  xgrid(2)
```

Scilab code Exa 11.33 obtain state matrix

```
 1  //caption:obtain_state_matrix
 2  //example 11_33
 3  //page 509
 4  s=%s;
 5  g=(s+2)/((s+1)*(s+3));
 6  CL=syslin('c',g);
 7  disp(CL,"C(s)/R(s)=");
 8  SS=tf2ss(CL)
 9  [Ac,Bc,U,ind]=canon(SS(2),SS(3))
10  disp(SS,"state space matrix=")
11  disp(Ac,"Ac",Bc,"Bc",U,"U",ind,"ind")
```

Scilab code Exa 11.34 obtain state matrix

```
 1  //caption:obtain_state_matrix
 2  //example 11_34
 3  //page 509
 4  s=%s;
 5  g=(s^2+s+2)/(s^3+9*s^2+26*s+24);
 6  CL=syslin('c',g);
 7  disp(CL,"C(s)/R(s)=");
 8  SS=tf2ss(CL)
 9  [Ac,Bc,U,ind]=canon(SS(2),SS(3))
10  disp(SS,"state space matrix=")
11  disp(Ac,"Ac",Bc,"Bc",U,"U",ind,"ind")
```

Scilab code Exa 11.35 obtain state matrix

```
1  //caption:obtain_state_matrix
2  //example 11_35
3  //page 510
4  s=%s;
5  g=1/(s^2+2*s+5);
6  CL=syslin('c',g);
7  disp(CL,"C(s)/R(s)=");
8  SS=tf2ss(CL)
9  [Ac,Bc,U,ind]=canon(SS(2),SS(3))
10 disp(SS,"state space matrix=")
11 disp(Ac,"Ac",Bc,"Bc",U,"U",ind,"ind")
```

Scilab code Exa 11.36 state transition matrix

```
1  //caption:state_transition_matrix
2  //example 11_36
3  //page 511
4  s=%s;
5  syms t
6  A=[1 4;-2 -5]
7  [r c]=size(A);//size of matrix A
8  p=s*eye(r,c)-A;//s*I-A where I is identity matrix
9  q=det(p)//determinant of sI-A
10 r=inv(p)//inverse of sI-A
11 //for calculating state transistion matrix
12 ip=[0 0;0 0]
13 i=1;j=1;
14 for i=1:2
15   for j=1:2
```

199

```
16      ip(i,j)=ilaplace(r(i,j),s,t);
17      j=j+1;
18    end
19    i=i+1;
20 end
21 disp(ip,"state transistion matrix,ip(t)=");
```

Scilab code Exa 11.37 check for contrallability of system

```
1 //caption:check_for_contrallability_of_system
2 //example 11_37
3 //page 512
4 A=[1 1;0 -1]
5 B=[1;0]
6 P=cont_mat(A,B);
7 disp(P,"Controllability Matrix=");
8 d=determ(P)
9 if d==0
10   printf("matrix is singular, so system is
        uncontrollable");
11 else
12   printf("system is controllable");
13 end;
```

Scilab code Exa 11.38 determine transfer function

```
1 //caption:determine_transfer_function
2 //example 11_38
3 //page 513
4 s=%s
5 A=[-5 1;-6 0]
6 B=[1;2]
7 C=[2 1]
```

```
 8  D=0;
 9  [r c]=size(A);//size of matrix A
10  p=s*eye(r,c)-A;//s*I-A where I is identity matrix
11  r=inv(p)//inverse of sI-A
12  G=C*r*B+D//transfer matrix
13  disp(G,"transfer matrix=")
```

Scilab code Exa 11.39 determine transfer matrix

```
 1  //caption:determine_transfer_matrix
 2  //example 11_39
 3  //page 513
 4  s=%s
 5  A=[0 1;-6 -5]
 6  B=[0;1]
 7  C=[2 1]
 8  D=0;
 9  [r c]=size(A);//size of matrix A
10  p=s*eye(r,c)-A;//s*I-A where I is identity matrix
11  r=inv(p)//inverse of sI-A
12  G=C*r*B+D//transfer matrix
13  disp(G,"transfer matrix=")
```

Chapter 12

CLASSIFIED SOLVED EXAMPLES

Scilab code Exa 12.1 Transfer Function

```
1  // Caption : transfer_function
2  //  example  12.1
3  // page  515
4  //  we  are  solving  this  problem  from  signal  flow
        graph  approach
5  syms  G  H;
6  //  forward  path  denoted  by  P1,P2  and  so  on  and  loop
        by  L1,L2  and  so  on
7  // path  factor  by  D1,D2  and  so  on  and  graph
        determinant  by  D
8  P1=1*G*1
9  P2=1;
10  L1=-G;
11  L2=-G*H;
12  L3=G*H;
13  D1=1;
14  D2=1;
15  D=1-(L1+L2+L3);
16  D=simple(D)
```

```
17  Y=(P1*D1+P2*D2)/D;
18  Y=simple(Y);
19  disp(Y,"C(s)/R(s)=");
```

Scilab code Exa 12.2 Transfer Function

```
1  //Caption: transfer_function
2  // example 12.2
3  //page 516
4  // we are solving this problem from signal flow
       graph approach
5  syms G1 G2 G3 G4
6  // forward path denoted by P1,P2 and so on and loop
       by L1,L2 and so on
7  //path factor by D1,D2 and so on and graph
       determinant by D
8  P1=G1;
9  P2=G2;
10  P3=-G1*G2*G3;
11  P4=G1*G2*G4;
12  L1=-G1*G2*G3*G4;
13  D1=1;
14  D2=1;
15  D3=1;
16  D4=1;
17  D=1-(L1);
18  Y=(P1*D1+P2*D2+P3*D3+P4*D4)/D;
19  Y=simple(Y);
20  disp(Y,"C(s)/R(s)=");
```

Scilab code Exa 12.3 Transfer Function

```
1  //Caption: transfer_function
```

```
2 // example 12.3
3 //page 517
4 // we are solving this problem from signal flow
      graph approach
5 syms G1 G2 G3 G4 H2 H1 H3
6 // forward path denoted by P1,P2 and so on and loop
      by L1,L2 and so on
7 //path factor by D1,D2 and so on and graph
      determinant by D
8 P1=G2*G4;
9 P2=G2*G3*G4;
10 P3=G1*G3*G4;
11 L1=-G4*H1;
12 L2=-G3*G4*H1*H2;
13 L3=-G1*G3*G4*H1*H2*H3
14 D1=1;
15 D2=1;
16 D3=1;
17 D=1-(L1+L2+L3);
18 Y=(P1*D1+P2*D2+P3*D3)/D;
19 Y=simple(Y);
20 disp(Y,"C(s)/R(s)=");
```

Scilab code Exa 12.4 Transfer Function

```
1 //Caption:transfer_function
2 // example 12.4
3 //page 518
4 // we are solving this problem from signal flow
      graph approach
5 syms G1 G2
6 // forward path denoted by P1,P2 and so on and loop
      by L1,L2 and so on
7 //path factor by D1,D2 and so on and graph
      determinant by D
```

```
 8  P1=G1;
 9  P2=G2;
10  P3=G1*G2;
11  P4=G1*G2;
12  L1=-G1;
13  L2=-G2;
14  L3=G1*G2;
15  L4=-G1*G2;
16  L5=-G1*G2;
17  D1=1;
18  D2=1;
19  D3=1;
20  D4=1;
21  D=1-(L1+L2+L3+L4+L5);
22  Y=(P1*D1+P2*D2+P3*D3+P4*D4)/D;
23  Y=simple(Y);
24  disp(Y,"C(s)/R(s)=");
```

Scilab code Exa 12.5 Transfer Function

```
 1  //Caption: transfer_function
 2  // example 12.5
 3  //page 518
 4  // we are solving this problem from signal flow
        graph approach
 5  syms G1 G2 G3 G4 G5 H1 H2
 6  // forward path denoted by P1,P2 and so on and loop
        by L1,L2 and so on
 7  //path factor by D1,D2 and so on and graph
        determinant by D
 8  P1=G1*G4*G5;
 9  P2=G1*G2*G3*G5;
10  L1=-G3*G5;
11  L2=-G3*G5*H2;
12  L3=-G1*G4*G5*H1;
```

```
13  L4 = - G1 * G2 * G3 * G5 * H1;
14  D1 =1;
15  D2 =1;
16  D=1 -( L1 + L2 + L3 + L4 );
17  Y=( P1 * D1 + P2 * D2 )/ D;
18  Y = simple ( Y );
19  disp (Y ," C( s )/ R( s )=" );
```

Scilab code Exa 12.7 Determine Peak Time and Peak Overshoot

```
 1  // Caption : determine_peak_time_and_peak_overshoot
 2  // example 12.7
 3  // page 520
 4  s=%s;
 5  syms  t;
 6  G= sym ( '( s +2)/( s *( s +1)) ')// G( s )
 7  H=1;
 8  CL=G /(1+ G*H );
 9  disp (CL ," C( s )/ R( s )=" );
10  // for unit step response R( s )=1/ s;
11  d=CL *(1/ s );
12  a=s*d;
13  disp (d ," C( s )=" );
14  c= ilaplace (d ,s ,t );
15  disp (c ," c ( t )=" );
16  // for peak time we get tp=3*%pi/4
17  tp =3*% pi /4
18  Cmax =1 -( exp (- tp )* cos ( tp ));
19  Css = limit (a ,s ,0);
20  disp ( Css ," Css=" );
21  Mp =(( Cmax - Css )/ Css )*100
22  Mp = float ( Mp )
23  disp (Mp ," peak_overshoot=" )
```

Scilab code Exa 12.8 Time Response and Peak Overshoot

```
1  // Caption : time_response_and_peak_overshoot
2  // example 12.8
3  // page 521
4  s=%s;
5  syms t;
6  num=sym('8*(s+1)');
7  den=sym('(s^2+2*s+2)');
8  CL=num/den;
9  disp(CL,"C(s)/R(s)=");
10 // for unit step response R(s)=1/s;
11 d=CL*(1/s);
12 disp(d,"C(s)=");
13 c=ilaplace(d,s,t);
14 disp(c,"c(t)=");
15 // for peak time we get tp=%pi/2
16 t=%pi/2
17 a=s*d;
18 a=simple(a)
19 Cmax=4*(1+1.414*exp(-t)*sin(t-(%pi/4)))
20 Css=limit(a,s,0);
21 disp(Css,"Css=");
22 Mp=((Cmax-Css)/Css)*100
23 Mp=float(Mp)
24 disp(Mp,"peak_overshoot=")
```

Scilab code Exa 12.9 Determine Peak Overshoot

```
1  // Caption : determine_peak_overshoot
2  // example 12.9
3  // page 523
```

```
4  s=%s;
5  syms t K;
6  CL=sym('(s+1)/(s^2+2*s+5)');
7  CL=K*CL;
8  disp(CL,"C(s)/R(s)=")
9  //for unit step response R(s)=1/s;
10 d=CL*(1/s)
11 Css=limit(s*d,s,0)
12 disp(Css,"Css=");
13 //since Css=0.8 (given)
14 K=0.8*5;
15 CL=eval(CL);
16 disp(CL,"C(s)/R(s)=");
17 //for unit step response R(s)=1/s;
18 d=CL*(1/s)
19 disp(d,"C(s)=");
20 c=ilaplace(d,s,t);
21 disp(c,"c(t)=");
22 //for peak time we get tp=0.785
23 t=0.785
24 a=s*d;
25 a=simple(a)
26 Cmax=(4/5)*(1-exp(-t)*cos(2*%pi/4)+exp(-t)*2*sin(2*
      %pi/4))
27 Css=limit(a,s,0)
28 disp(Css,"Css=");
29 Mp=((Cmax-Css)/Css)*100
30 Mp=float(Mp)
31 disp(Mp,"peak_overshoot=")
```

Scilab code Exa 12.10 Determine Unit Step Response

```
1 //Caption:determine_unit_step_response
2 //example 12.10
3 //page 524
```

```
 4  s=%s ;
 5  syms t ;
 6  CL=sym ( ' 1/( ( s +1)∗( s ^2+1) ) ')
 7  disp (CL , "C( s )/R( s )=") ;
 8  // for unit step response R( s )=1/s ;
 9  d=CL∗(1/ s ) ;
10  a=s ∗d ;
11  c=ilaplace ( d , s , t ) ;
12  disp ( c , " c ( t )=") ;
```

Scilab code Exa 12.11 Determine Unit Step and Unit Impulse Response

```
 1  // Caption :
        determine_unit_step_and_unit_impulse_response
 2  // example 12.11
 3  // page 524
 4  s=%s ;
 5  syms t ;
 6  G=sym ( ' 8/( s +1) ') ;
 7  H=sym ( ' (1/2∗ s ) ') ;
 8  CL=G/(1+G∗H) ;
 9  disp (CL , "C( s )/R( s )=") ;
10  // for unit step response R( s )=1/s ;
11  d=CL∗(1/ s ) ;
12  disp ( d , "C( s )=") ;
13  c=ilaplace ( d , s , t ) ;
14  disp ( c , " unit step response , c ( t )=") ;
15  // for unit impulse response R( s )=1;
16  e=CL∗(1) ;
17  disp ( e , "C( s )=") ;
18  ct=ilaplace ( e , s , t ) ;
19  disp ( ct , " unit impulse response , c ( t )=") ;
```

Scilab code Exa 12.12 Determine Wn Wd zeta and steady state error

```
1  //caption:determine_Wn,Wd,
       zeta_and_steady_state_error
2  //example 12_12
3  //page 526
4  s=%s;
5  G=sym('20/(s*(s^2+6*s+6))');
6  H=0.25;
7  CL=G/(1+G*H);
8  CL=simple(CL);
9  disp(CL,"C(s)/R(s)=");
10 printf("the char. eq is:")
11 disp("s^2+s+1=0")
12 Wn=sqrt(1)//natural_frequency
13 //2*zeta*Wn=1
14 zeta=1/(2*Wn);//damping ratio
15 d=zeta*Wn;//damping_factor
16 z=sqrt(1-zeta^2);
17 Wd=Wn*z;//damped_frequency_of_oscillation
18 Mp=exp((-zeta*%pi)/z)*100;//%_max.peak_overshoot
19 ts=4/(zeta*Wn);//settling_time
20 tp=%pi/(Wn*sqrt(1-zeta^2));//peak_time
21 tu=2*%pi/(Wn*sqrt(1-zeta^2));//first under shoot
22 ti=tu-tp;//time interval between max. and min.
       values
23 disp(Wn,"natural_frequency=");
24 disp(zeta,"damping ratio=");
25 disp(Wd,"damped_frequency_of_oscillation=");
26 disp(Mp,"%_max.peak_overshoot=");
27 disp(ts,"settling_time=");
28 disp(tp,"peak_time=");
29 disp(ti,"time_interval_between_max_and_min_values=")
       ;
```

Scilab code Exa 12.13 Determine Wn Wd zeta and steady state error

```
1  //caption:determine_Wn,Wd,
       zeta_and_steady_state_error
2  //example 12_13
3  //page 527
4  syms Kp K Kd T
5  s=%s;
6  //exec series.sce;
7  //exec parallel.sce;
8  a=(Kp+s*Kd)*K
9  b=1/(s*(s*T+1))
10 G=series(a,b)
11 H=1;
12 er=1/(1+G*H)
13 disp(er,"E(s)/R(s)=");
14 R=1/s^2
15 E=R*er
16 ess=limit(s*E,s,0)
17 disp(ess,"steady state error,ess=")
```

Scilab code Exa 12.15 Stability Using Routh Hurwitz Criterion

```
1  //caption:stability_using_Routh-hurwitz_criterion
2  //example 12.15
3  //page 529
4  s=%s;
5  syms  K
6  G=sym('K/(s*(s^2+s+1)*(s+5)');
7  H=1;
8  CH=(s*(s^2+s+1)*(s+5)+K)
9  disp('=0',CH,"characterstics_eq,CH=")
10 c0=coeffs(CH,'s',0);
11 c1=coeffs(CH,'s',1);
12 c2=coeffs(CH,'s',2);
```

```
13  c3=coeffs(CH,'s',3);
14  c4=coeffs(CH,'s',4);
15  b=[c0 c1 c2 c3 c4 ]
16  routh=[b([5,3,1]);b([4,2]),0]
17  routh=[routh;-det(routh(1:2,1:2)/routh(2,1)),K,0]
18  t=routh(2:3,1:2)
19  routh=[routh;-det(t)/t(2,1),0,0]
20  routh=[routh;K,0,0]
21  disp(routh,"routh=")
22  disp("for given system to be stable:");
23  disp("((5.1*5-6*K)/5.1)>0 and K>0");
24  disp("which gives:");
25  disp("0<K<4.25");
```

Scilab code Exa 12.16 Stability Using Routh Hurwitz Criterion

```
1  //caption:stability_using_Routh-hurwitz_criterion
2  //example 12.16
3  //page 530
4  s=%s;
5  syms  K
6  CH=s^4+2*s^3+10*s^2+(K-10)*s+K
7  disp('=0',CH,"characterstics_eq ,CH=")
8  c0=coeffs(CH,'s',0);
9  c1=coeffs(CH,'s',1);
10 c2=coeffs(CH,'s',2);
11 c3=coeffs(CH,'s',3);
12 c4=coeffs(CH,'s',4);
13 b=[c0 c1 c2 c3 c4 ]
14 routh=[b([5,3,1]);b([4,2]),0]
15 routh=[routh;-det(routh(1:2,1:2)/routh(2,1)),K,0]
16 routh(3,1)=simple(routh(3,1))
17 t=routh(2:3,1:2)
18 l=simple(-det(t)/t(2,1))
19 routh=[routh;l,0,0]
```

```
20  //routh=[routh;K,0 ,0]
21  disp(routh ,"routh=")
22  disp("for given system to be stable ,following
        condition should be satisfied");
23  disp("K<30,K<22.9 and K>13.1,K>0")
24  disp("which gives ,")
25  disp("13.1<K<22.9");
```

Scilab code Exa 12.17 Stability Using Routh Hurwitz Criterion

```
1   //caption : stability_using_Routh −hurwitz_criterion
2   //example 12.17
3   //page 530
4   s=%s;
5   syms  K
6   G=sym('K/(( s^3+3*s+2)*(s^2+6*s+24))');
7   H=1;
8   CH=((s^3+3*s+2)*(s^2+6*s+24)+K)
9   disp('=0',CH,"characterstics_eq ,CH=")
10  c0=coeffs(CH,'s',0);
11  c1=coeffs(CH,'s',1);
12  c2=coeffs(CH,'s',2);
13  c3=coeffs(CH,'s',3);
14  c4=coeffs(CH,'s',4);
15  b=[c0 c1 c2 c3 c4 ]
16  routh=[b([5,3,1]);b([4,2]),0]
17  routh=[routh;-det(routh(1:2,1:2))/routh(2,1),routh
        (1,3),0]
18  routh(3,1)=simple(routh(3,1))
19  t=routh(2:3,1:2)
20  l=simple(-det(t)/t(2,1))
21  routh=[routh;l,0,0]
22  //routh=[routh;K,0 ,0]
23  disp(routh ,"routh=")
24  disp("for given system to be stable ,following
```

```
              condition should be satisfied");
25  disp("78.84−0.259K>0")
26  disp("which gives limiting value of K")
27  disp("K<288.9");
```

Scilab code Exa 12.18 Stability Using Routh Hurwitz Criterion

```
1   //caption:stability_using_Routh−hurwitz_criterion
2   //example 12.18
3   //page 531
4   s=%s;
5   A=s^5+s^4+4*s^3+4*s^2+s+1;
6   b=coeff(A)
7   n=length(b)
8   routh=[b([6 4 2]);b([5 3 1])]
9   routh1=routh;
10  c=[routh(1,1),routh(1,3);routh(2,1),routh(2,3)]
11  routh=[routh;-det(routh(1:2,1:2))/routh(2,1),-det(c)
        /routh(2,1),0]
12  disp("since all elements of third row are zero, so
        we make auxiliary equation")
13  A=sym('s^4+4*s^2+1')//auxiliary equation
14  B=diff(A,s)
15  routh=[routh1;4,8,0]
16  d=[routh(2,1),routh(2,3);routh(3,1),routh(3,3)]
17  routh=[routh;-det(routh(2:3,1:2))/routh(3,1),-det(d)
        /routh(3,1),0]
18  routh2=routh
19  routh=[routh;-det(routh(3:4,1:2))/routh(4,1),0,0]
20  routh=[routh;-det(routh(4:5,1:2))/routh(5,1),0,0];
21  disp(routh,"routh=")
22  disp("stability is examined as below . since roots
        of auxiliary eq are also roots of characterstics
        eq")
23  A=s^4+4*s^2+1
```

```
24  b=roots(A)
25  disp("since the equation has non repeating  roots on
        s plane imaginary axis.hence system  are
        unstable" )
```

Scilab code Exa 12.19 Stability Using Routh Hurwitz Criterion

```
1  // caption : stability_using_Routh−hurwitz_criterion
2  // example 12.19
3  // page 531
4  s=%s ;
5  A=s^5+s^4+4*s^3+4*s^2+4*s+4;
6  b=coeff(A)
7  n=length(b)
8  routh=[b([6,4,2]);b([5 3 1])]
9  routh1=routh ;
10 c=[routh(1,1),routh(1,3);routh(2,1),routh(2,3)]
11 routh=[routh;-det(routh(1:2,1:2))/routh(2,1),-det(c)
       /routh(2,1),0]
12 disp("since all elements of third row are zero , so
       we make auxiliary equation")
13 A=sym('s^4+4*s^2+4')// auxiliary  equation
14 B=diff(A,s)
15 routh=[routh1;4,8,0]
16 d=[routh(2,1),routh(2,3);routh(3,1),routh(3,3)]
17 routh=[routh;-det(routh(2:3,1:2))/routh(3,1),-det(d)
       /routh(3,1),0]
18 routh2=routh
19 routh=[routh;-det(routh(3:4,1:2))/routh(4,1),0,0]
20 disp("since all elements of fifth row are zero , so
       we make auxiliary equation")
21 A=sym('2*s^2+4')// auxiliary  equation
22 B=diff(A,s)
23 routh=[routh2;4,0,0]
24 routh=[routh;-det(routh(4:5,1:2))/routh(5,1),0,0];
```

```
25  disp(routh,"routh=")
26  disp("stability is examined as below . since roots
       of auxiliary eq are also roots of characterstics
       eq")
27  A=s^4+4*s^2+4
28  b=roots(A)
29  disp("since the equation has multiple roots on s
       plane imaginary axis.hence system   are unstable"
       )
```

Scilab code Exa 12.21 Determine Frequency of Oscillations

```
1  //caption:determine_frequency_of_oscillations
2  //example 12.21
3  //page 533
4  s=%s;
5  syms  K
6  G=sym('K*(s*(2*s+1))/(s^3+3*s+20)');
7  H=1;
8  CH=((s^3+3*s+20)+K*(s*(2*s+1)))
9  disp('=0',CH,"characterstics_eq ,CH=")
10  c0=coeffs(CH,'s',0);
11  c1=coeffs(CH,'s',1);
12  c2=coeffs(CH,'s',2);
13  c3=coeffs(CH,'s',3);
14  b=[c0 c1 c2 c3]
15  routh=[b([4,2]);b([3,1])]
16  routh=[routh;-det(routh(1:2,1:2)/routh(2,1)),0]
17  routh(3,1)=simple(routh(3,1))
18  t=routh(2:3,1:2)
19  l=simple(-det(t)/t(2,1))
20  routh=[routh;l,0]
21  disp(routh,"routh=")
22  disp("for sustained oscillations:");
23  disp("2*K^2+6*K-20=0")
```
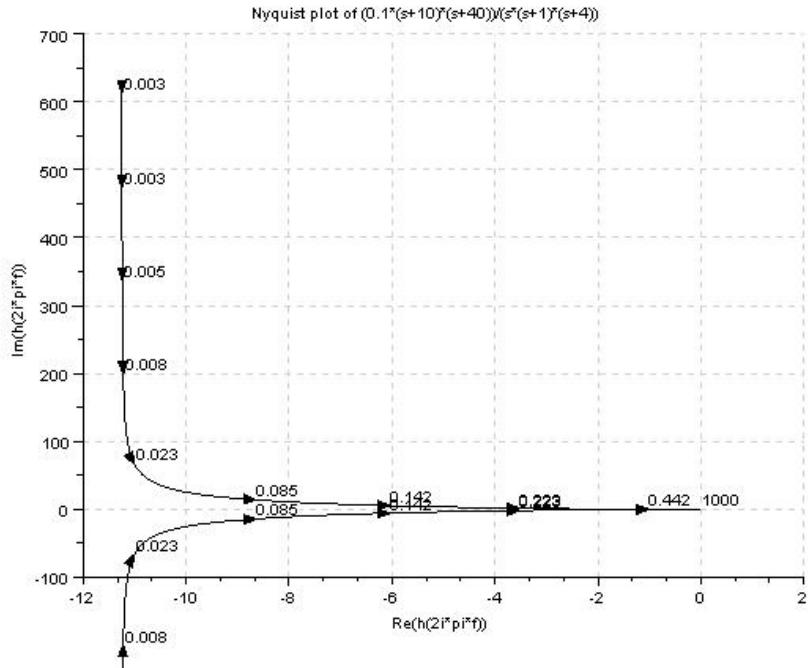
Figure 12.1: Stability Using Nyquist Criterion

```
24  disp("which gives K")
25  disp("K=2");
26  disp("freq of oscillations is calculated by solving
         equation formed by s^2 row of the routh table")
27  K=2
28  k=2
29  A=2*k*s^2+20
30  r=roots(A)
31  disp(%i*r(2),"frequency of oscillations=" )
```

Scilab code Exa 12.23.i Stability Using Nyquist Criterion

```scilab
1  //caption:stability_using_Nyquist_criterion
2  //example 12_23_i
3  //page 535
4  clf();
5  s=%s;
6  s1=-s;
7  disp("for K=0.1")
8  g=(0.1*(s+10)*(s+40))/(s*(s+1)*(s+4));
9  g1=(0.1*(s1+10)*(s1+40))/(s1*(s1+1)*(s1+4));
10 GH=syslin('c',g);
11 GH1=syslin('c',g1);
12 nyquist(GH);
13 nyquist(GH1);
14 //mtlb_axis([-1.5 0.2 -0.3 0.3]);
15 xtitle('Nyquist plot of (0.1*(s+10)*(s+40))/(s*(s+1)
      *(s+4))')
16 figure;
17 show_margins(GH,'nyquist')
18 disp("since the point(-1+%i0) is  not encircled
      clockwise by Nyquist plot ,so N=0 and P=0")
19 N=0;//no. of encirclement of -1+%i0 by G(s)H(s) plot
       anticlockwise
20 P=0;//no. of poles of G(s)H(s) with positive real
      part
21 Z=P-N;//np.of zeros of 1+G(s)H(s)=0 with positive
      real part
22 disp(Z,"Z=")
23 disp("as Z=0,there are no roots of closed loop
      characterstics eq having positive real part,
      hence system is stable.")
```
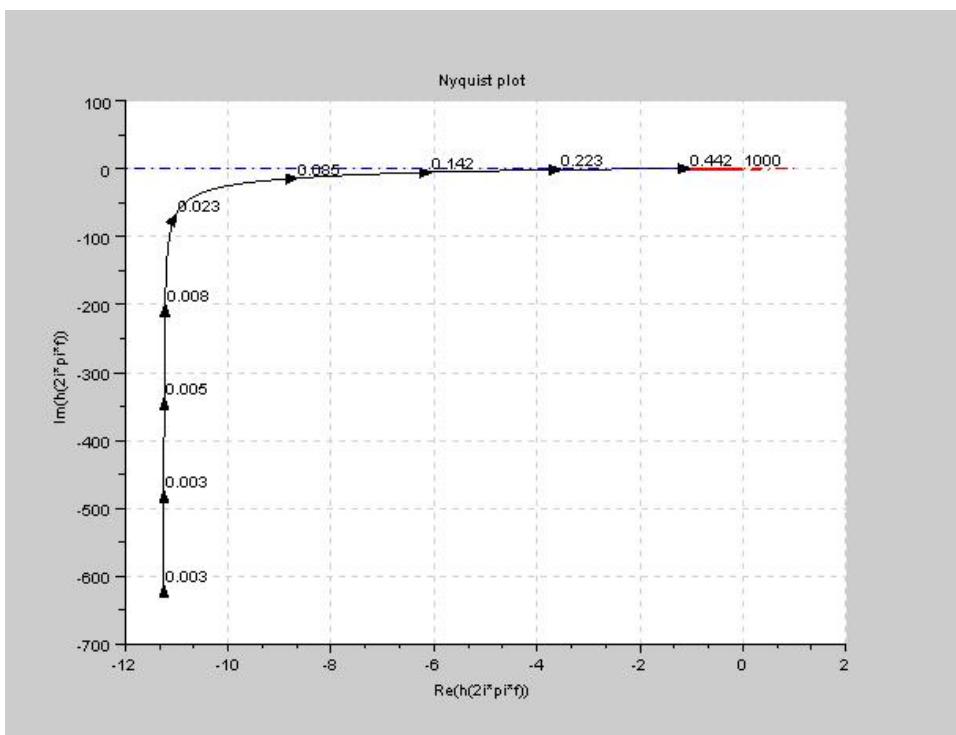
Figure 12.2: Stability Using Nyquist Criterion
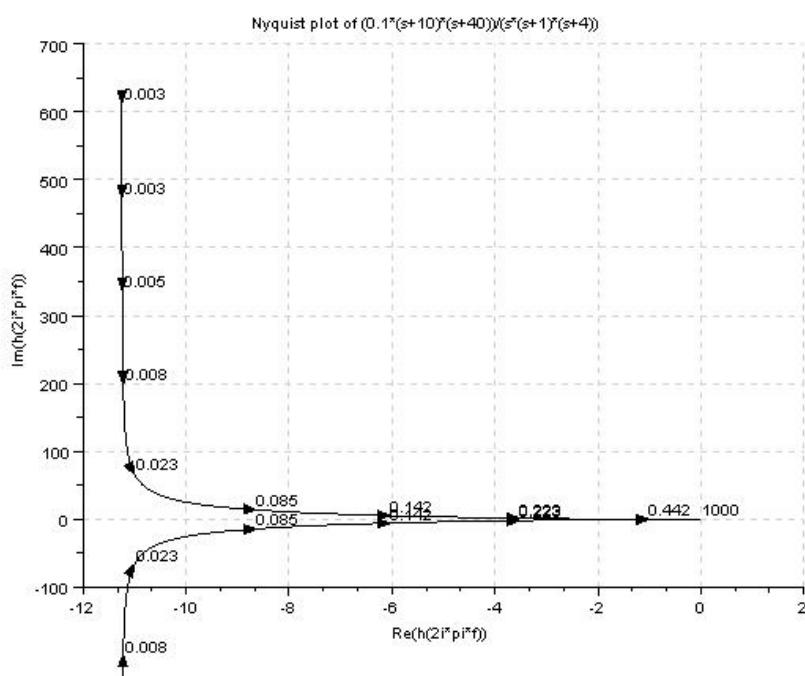
Figure 12.3: Stability Using Nyquist Criterion

Scilab code Exa 12.23.ii Stability Using Nyquist Criterion

```
1  //caption: stability_using_Nyquist_criterion
2  //example 12_23_ii
3  //page 535
4  disp("for K=1")
5  g=(0.1*(s+10)*(s+40))/(s*(s+1)*(s+4));
6  g1=(0.1*(s1+10)*(s1+40))/(s1*(s1+1)*(s1+4));
7  GH=syslin('c',g);
8  GH1=syslin('c',g1);
9  nyquist(GH);
10 nyquist(GH1);
11 //mtlb_axis([-3 0.5 -0.6 0.6]);
12 xtitle('Nyquist plot of (0.1*(s+10)*(s+40))/(s*(s+1)
       *(s+4))')
13 figure;
14 show_margins(GH,'nyquist')
15 disp("since the point(-1+%i0) is  encircled twice
       clockwise by Nyquist plot ,so N=2 and P=0(given)"
       )
16 N=-2;//no. of encirclement of -1+%i0 by G(s)H(s)
       plot anticlockwise
17 P=0;//no. of poles of G(s)H(s) with positive real
       part
18 Z=P-N;//np.of zeros of 1+G(s)H(s)=0 with positive
       real part
19 disp(Z,"Z=")
20 disp("as Z=2,there are two roots of closed loop
       characterstics eq having positive real part,
       hence system is unstable.")
```
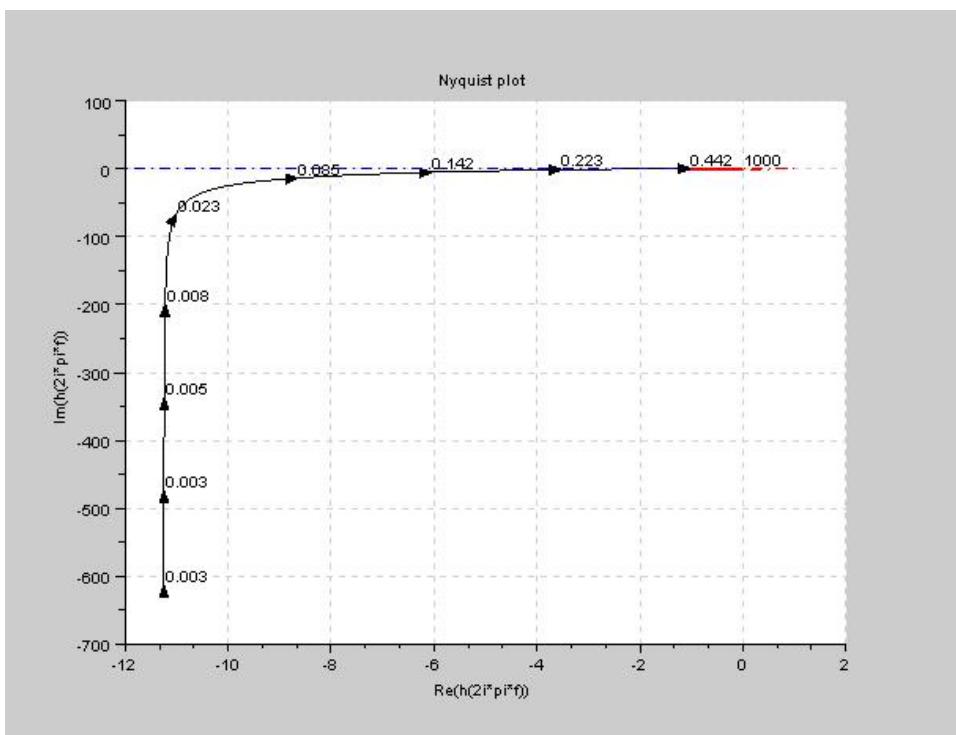
Figure 12.4: Stability Using Nyquist Criterion
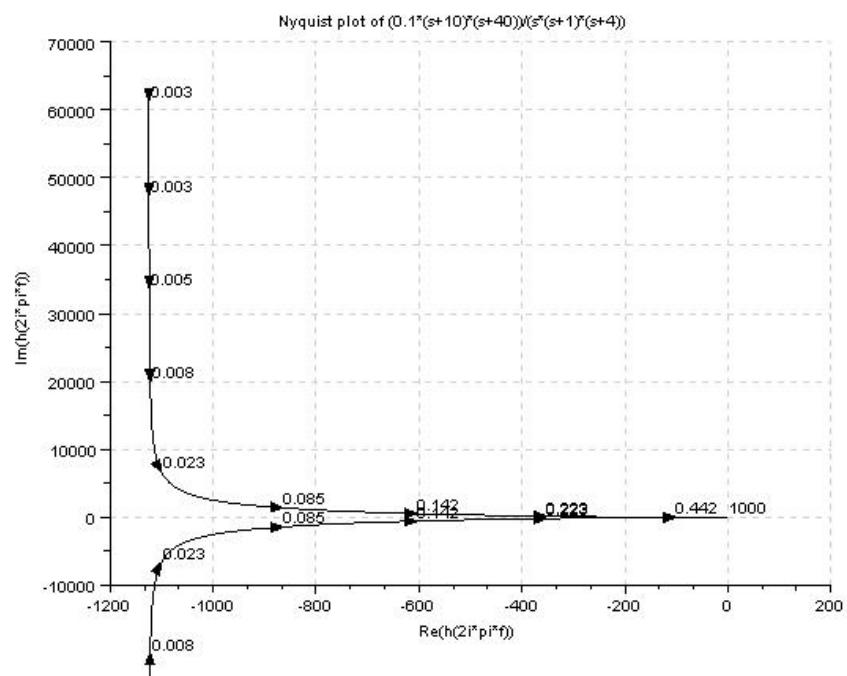
Figure 12.5: Stability Using Nyquist Criterion

Scilab code Exa 12.23.iii Stability Using Nyquist Criterion

```
1  //caption:stability_using_Nyquist_criterion
2  //example 12_23_iii
3  //page 535
4  disp("for K=10")
5  g=(10*(s+10)*(s+40))/(s*(s+1)*(s+4));
6  g1=(10*(s1+10)*(s1+40))/(s1*(s1+1)*(s1+4));
7  GH=syslin('c',g);
8  GH1=syslin('c',g1);
9  nyquist(GH);
10 nyquist(GH1);
11 //mtlb_axis([-1.5 0.2 -0.3 0.3]);
12 xtitle('Nyquist plot of (0.1*(s+10)*(s+40))/(s*(s+1)
      *(s+4))')
13 figure;
14 show_margins(GH,'nyquist')
15 disp("since the point(-1+%i0) is encircled once in
      clockwise and once in anti clockwise direction by
       Nyquist plot ,so N=0 and P=0")
16 N=0;//no. of encirclement of -1+%i0 by G(s)H(s) plot
       anticlockwise
17 P=0;//no. of poles of G(s)H(s) with positive real
      part
18 Z=P-N;//np.of zeros of 1+G(s)H(s)=0 with positive
      real part
19 disp(Z,"Z=")
20 disp("as Z=0,there are no roots of closed loop
      characterstics eq having positive real part,
      hence system is stable.")
```
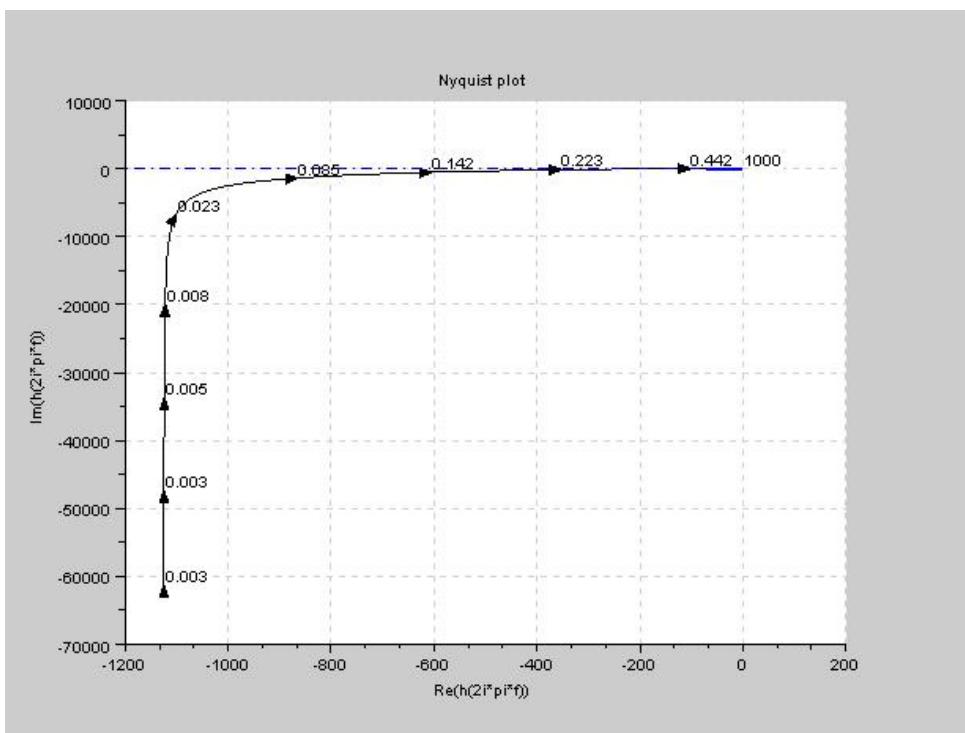
Figure 12.6: Stability Using Nyquist Criterion
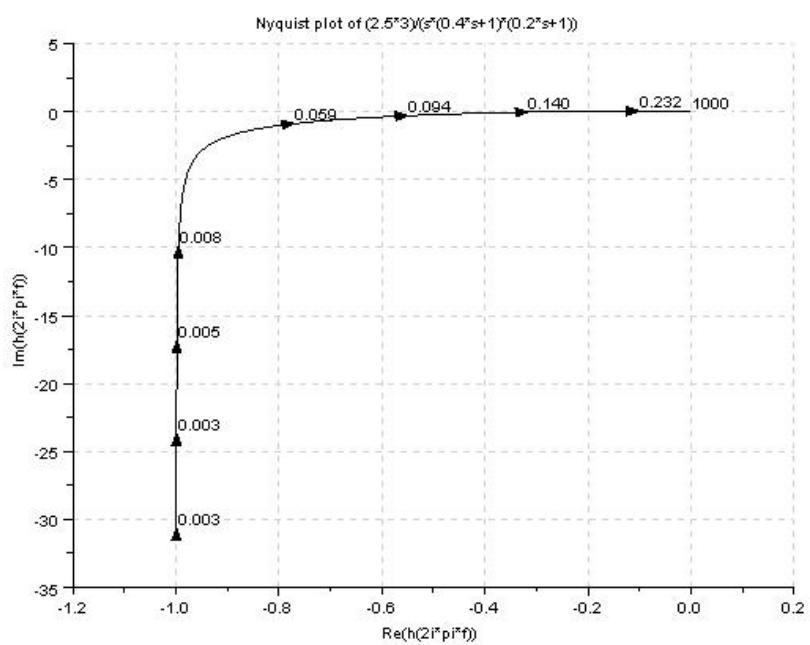
Figure 12.7: Gain and Phase Margin

Scilab code Exa 12.27 Gain and Phase Margin

```
1  //caption:gain_and_phase_margin
2  //example 12_27
3  //page543
4  clf();
5  s=%s;
6  s1=-s;
7  disp("for K=0.5")
8  g=(0.5)/(s*(s+1)^2);
9  GH=syslin('c',g);
10 nyquist(GH);
11 //mtlb_axis([-5 1  -500 500]);
12 xtitle('Nyquist plot of (2.5*3)/(s*(0.4*s+1)*(0.2*s
      +1))')
13 pm=p_margin(GH)
14 disp(pm,"phase margin=")
15 gm=g_margin(GH)
16 disp(gm,"gain margin=")
```

Scilab code Exa 12.33 Determine Close Loop Stability

```
1  //caption:determine_close_loop_stability
2  //example 12_33
3  //page 550
4  s=%s;
5  g=(720*(s+1.25))/(s*(s+10)*(s^2+2*s+9));
6  G=syslin('c',g)
7  fmin=0.1;
8  fmax=100;
9  bode(G, fmin, fmax)
10 [gm,freqGM]=g_margin(G);
11 [pm,freqPM]=p_margin(G);
```
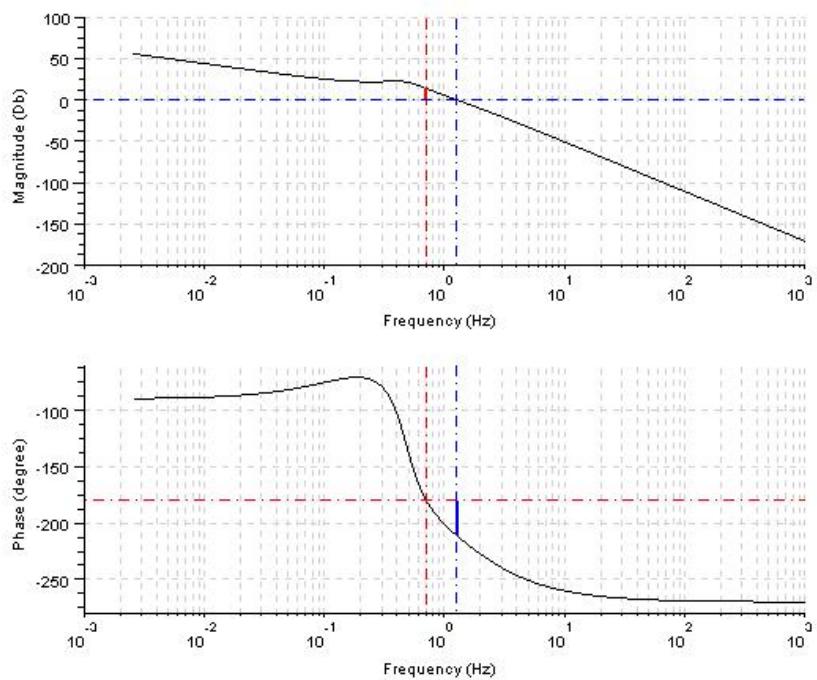
Figure 12.8: Determine Close Loop Stability
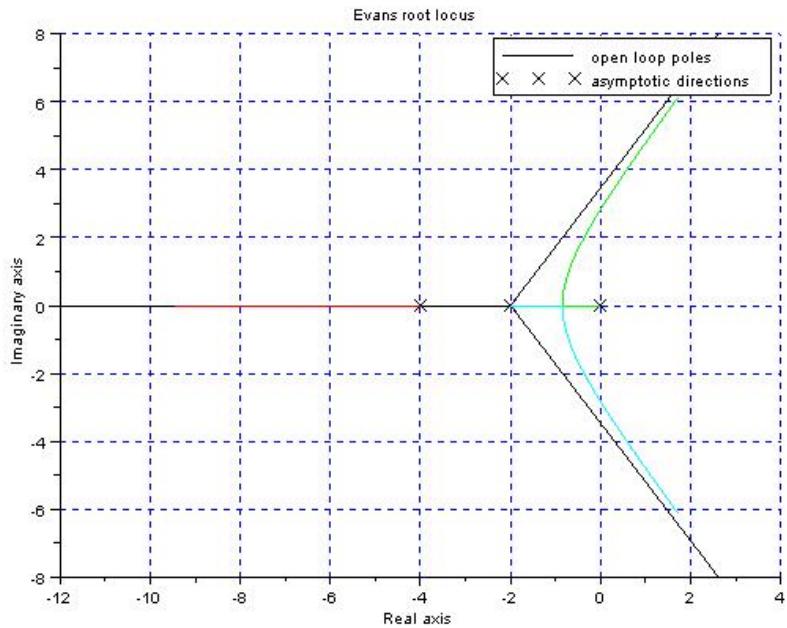
Figure 12.9: Root Locus

```
12  show_margins(G);
13  disp(gm,"gain_margin=");
14  disp((freqGM*2*%pi),"gain_margin_freq=");
15  disp(pm,"phase_margin=");
16  disp((freqPM*2*%pi),"phase_margin_freq=");
17  show_margins(G);
18  disp("since gain and phase margin are both negative
       so system is unstable")
```

Scilab code Exa 12.42 Root Locus

```scilab
1  //caption:root_locus
2  //example 12_42
3  //page 562
4  s=%s;
5  syms K;
6  GH=K/(s*(s+2)*(s+4))
7  //since Mp=40%, so .4=exp((-zeta*%pi)/(sqrt(1-zeta
       ^2))
8  zeta=0.3
9  //from given data
10 disp("the characterstics eq. is determined as:")
11 CH=(s*(s+2)*(s+4))+K
12 K=sym('-(s^3+6*s^2+8*s)')
13 d=diff(K,s)
14 e=-3*s^2-12*s-8
15 r1=roots(e)
16 disp(r1,"roots=")
17 disp("-0.842 is break away point sinc it lies on
       root locus")
18 CH=sym('s^3+6*s^2+8*s+K');
19 disp('=0',CH,"characterstics_eq ,CH=")
20 c0=coeffs(CH,'s',0);
21 c1=coeffs(CH,'s',1);
22 c2=coeffs(CH,'s',2);
23 c3=coeffs(CH,'s',3);
24 b=[c0 c1 c2 c3]
25 n=4;
26 routh=[b([4,2]);b([3,1])];
27 routh=[routh;-det(routh)/routh(2,1),0]
28 t=routh(2:3,1:2)
29 routh=[routh;-det(t)/t(2,1),0]
30 disp(routh,"routh=")
31 disp("for given system to be marginally stable:");
32 disp("(48-K)=0 ");
33 disp("which gives:");
34 disp("K=48");
35 K=48;
36 k=48
```

```
37  a=6*s^2+48//intersection of root locus with
        imaginary plane
38  r=roots(a)
39  g=k/(s*(s+2)*(s+4))
40  G=syslin('c',g)
41  evans(g,8)
42  xgrid(2)
43  disp("the line theta=acos(zeta)=72.5 intersects root
        locus at sa=(-0.5+i1.65)")
44  disp("the value of K at s=sa is find to be 14.87 for
        Mp=40%")
45  K=14.87
46  ts=4/0.5        //ts=4/(zeta*wn)
47  Kv=limit(s*GH,s,0)
48  Kv=eval(Kv)
49  Kv=float(Kv)
50  disp(Kv,"Kv=");
```

Scilab code Exa 12.43 Root Locus and Value of K

```
1  //caption:root_locus_and_value_of_K
2  //example 12.43
3  //page 564
4  s=%s;
5  syms K;
6  GH=K/(s*(s+2)*(s^2+2*s+2))
7  disp("the characterstics eq. is determined as:")
8  CH=(s*(s+2)*(s^2+2*s+2))+K
9  CH=sym('s^4+4*s^3+6*s^2+4*s+K');
10 disp('=0',CH,"characterstics_eq ,CH=")
11 c0=coeffs(CH,'s',0);
12 c1=coeffs(CH,'s',1);
13 c2=coeffs(CH,'s',2);
```
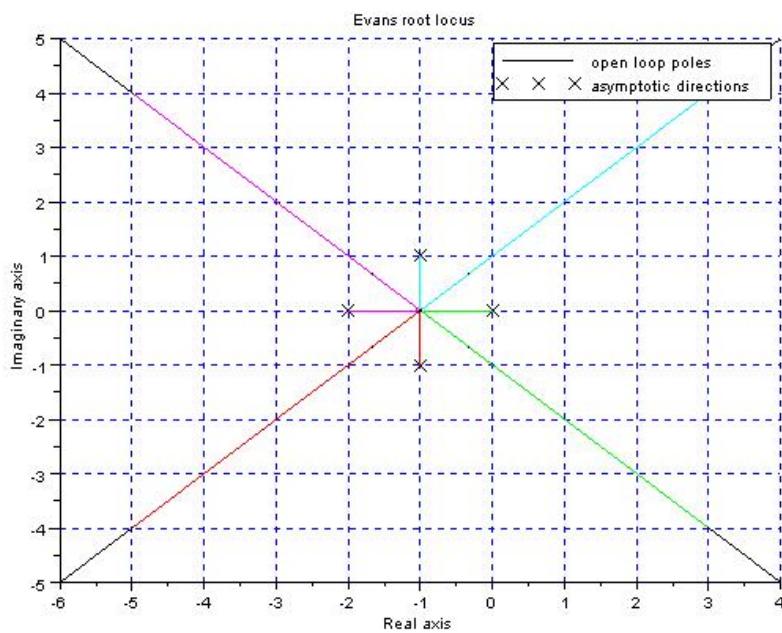
Figure 12.10: Root Locus and Value of K

```
14  c3 = coeffs ( CH , 's ' ,3) ;
15  c4 = coeffs ( CH , 's ' ,4) ;
16  b =[ c0  c1  c2  c3  c4  ]
17  routh =[ b ([5 ,3 ,1]) ; b ([4 ,2]) ,0]
18  routh =[ routh ; - det ( routh (1:2 ,1:2) ) / routh (2 ,1) , routh
       (1 ,3) ,0]
19  routh (3 ,1) = simple ( routh (3 ,1) )
20  t = routh (2:3 ,1:2)
21  l = simple ( - det ( t ) / t (2 ,1) )
22  routh =[ routh ; l ,0 ,0]
23  routh =[ routh ; K ,0 ,0]
24  K = sym ( 's ^4+4* s ^3+6* s ^2+4* s ') 
25  d = diff ( K , s )
26  e = -(4* s ^3+12* s ^2+12* s +4)
27  r = roots ( e )
28  disp ( routh , " routh =")
29  disp (" for  given  system  to  be  marginally  stable :") ;
30  disp ("((20 -4K) /5) =0  ") ;
31  disp (" which  gives :") ;
32  disp (" K =5") ;
33  K =5;
34  k =5
35  a =5* s ^2+5// intersection  of  root  locus  with  s  plane
36  r = roots ( a )
37  disp ( r , " intersection  point  with  imaginary  axis =")
38  g = k /( s *( s +2) *( s ^2+2* s +2) )
39  G = syslin ( 'c ' , g )
40  evans ( g ,200)
41  xgrid (2)
42  disp (" angle  of  departure = -90  and  +90")
43  disp (" breakaway  point  is  -1  and  + j  and  - j ")
44  disp (" on  solving  we  find  K =1")
```
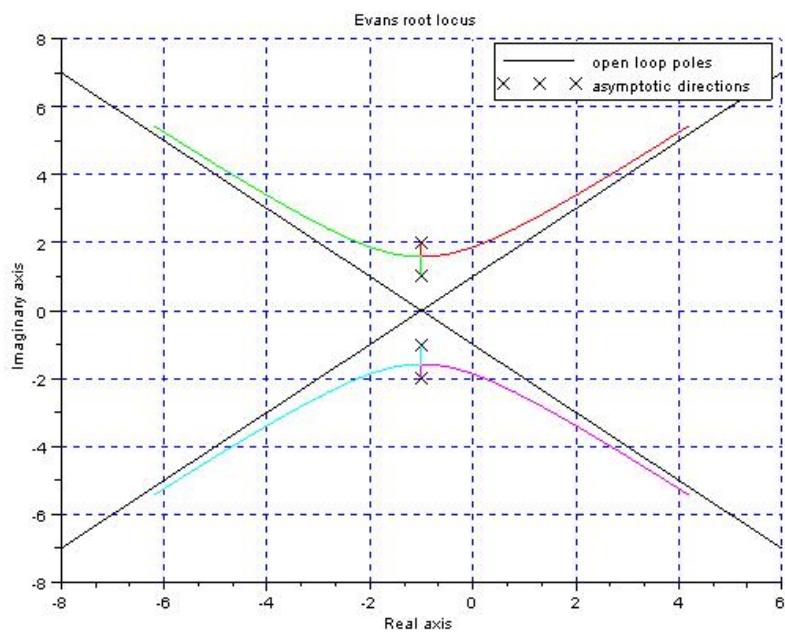
Figure 12.11: Root Locus and Value of K

Scilab code Exa 12.44 Root Locus and Value of K

```scilab
1  //caption:root_locus_and_value_of_K
2  //example 12.44
3  //page 565
4  s=%s;
5  syms K;
6  GH=K/((s^2+2*s+5)*(s^2+2*s+2))
7  disp("the characterstics eq. is determined as:")
8  CH=((s^2+2*s+5)*(s^2+2*s+2))+K
9  CH=sym('((s^2+2*s+5)*(s^2+2*s+2))+K');
10 disp('=0',CH,"characterstics_eq ,CH=")
11 c0=coeffs(CH,'s',0);
12 c1=coeffs(CH,'s',1);
13 c2=coeffs(CH,'s',2);
14 c3=coeffs(CH,'s',3);
15 c4=coeffs(CH,'s',4);
16 b=[c0 c1 c2 c3 c4 ]
17 routh=[b([5,3,1]);b([4,2]),0]
18 routh=[routh;-det(routh(1:2,1:2))/routh(2,1),routh
      (1,3),0]
19 routh(3,1)=simple(routh(3,1))
20 t=routh(2:3,1:2)
21 l=simple(-det(t)/t(2,1))
22 routh=[routh;l,0,0]
23 routh=[routh;K,0,0]
24 K=sym('-(s^2+2*s+5)*(s^2+2*s+2)')
25 d=diff(K,s)
26 e=-(4*s^3+12*s^2+22*s+14)
27 r=roots(e)
28 disp(routh,"routh=")
29 disp("for given system to be marginally stable:");
30 disp("((7.5*14-4(K+10)/7.5)=0 ");
31 disp("which gives:");
32 disp("K=16.25");
33 K=16.25;
34 k=16.25
35 a=7.5*s^2+26.25//intersection of root locus with s
```
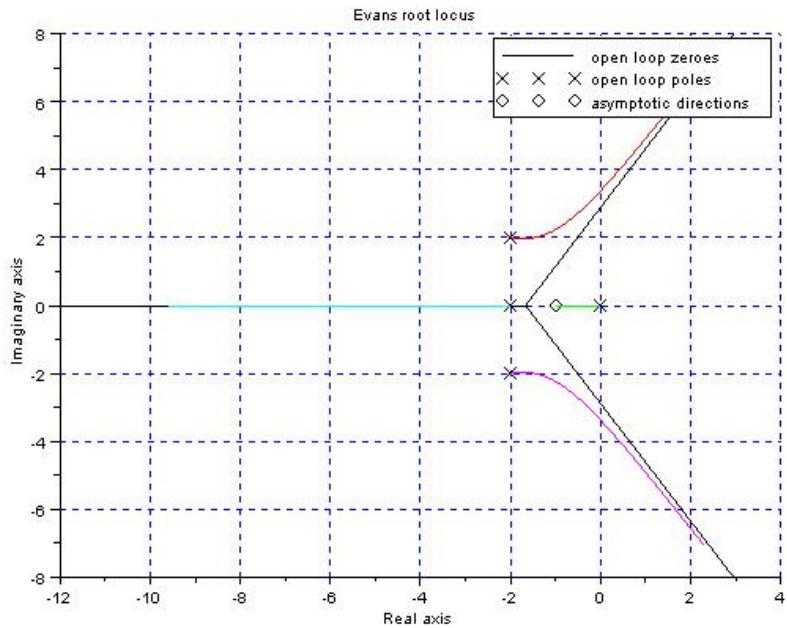
Figure 12.12: Root Locus and Value of K

```
      plane
36  r=roots(a)
37  g=k/((s^2+2*s+5)*(s^2+2*s+2))
38  G=syslin('c',g)
39  evans(g,200)
40  xgrid(2)
41  disp(r,"the point of intersection of root locus with
        imaginary axis =")
```

Scilab code Exa 12.45 Root Locus and Value of K

236

```scilab
 1  //caption:root_locus_and_value_of_K
 2  //example 12.45
 3  //page 567
 4  s=%s;
 5  syms K;
 6  GH=K*(s+1)/(s*(s+2)*(s^2+4*s+8))
 7  disp("the characterstics eq. is determined as:")
 8  CH=((s*(s+2)*(s^2+4*s+8)))+K*(s+1)
 9  CH=sym('((s*(s+2)*(s^2+4*s+8)))+K*(s+1)');
10  disp('=0',CH,"characterstics_eq ,CH=")
11  c0=coeffs(CH,'s',0);
12  c1=coeffs(CH,'s',1);
13  c2=coeffs(CH,'s',2);
14  c3=coeffs(CH,'s',3);
15  c4=coeffs(CH,'s',4);
16  b=[c0 c1 c2 c3 c4 ]
17  routh=[b([5,3,1]);b([4,2]),0]
18  routh=[routh;-det(routh(1:2,1:2))/routh(2,1),routh
        (1,3),0]
19  routh(3,1)=simple(routh(3,1))
20  t=routh(2:3,1:2)
21  l=simple(-det(t)/t(2,1))
22  routh=[routh;l,0,0]
23  routh=[routh;K,0,0]
24  disp(routh,"routh=")
25  disp("for given system to be marginally stable:");
26  disp("(K^2-28*K-1280)=0 ");
27  disp("which gives:");
28  disp("K=52.4 and -24.42");
29  K=52.4;//considering positive value
30  k=52.4
31  a=((80-52.4)/6)*s^2+52.4//intersection of root locus
        with imaginary axis
32  r=roots(a)
33  g=k*(s+1)/(s*(s+2)*(s^2+4*s+8))
34  G=syslin('c',g)
35  clf();
36  evans(g,10)
```
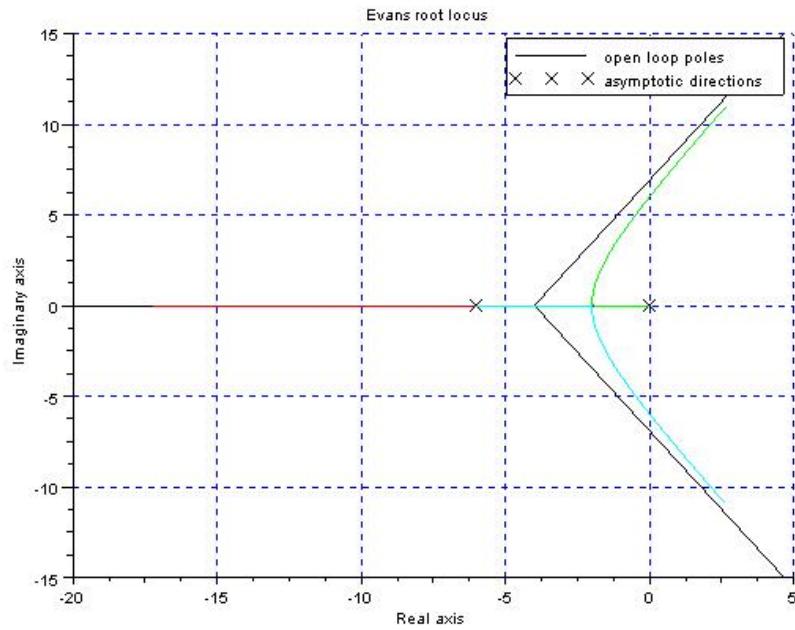
Figure 12.13: Root Locus and Value of K

```
37 xgrid (2)
38 disp (r ," the point of intersection of root locus with
      imaginary axis =")
```

Scilab code Exa 12.46 Root Locus and Value of K

```
1 // caption : root_locus_and_value_of_K
2 // example 12.46
3 // page 570
4 s=%s ;
5 syms K ;
```

```scilab
 6  GH=K/(s*((s+6)^2))
 7  disp("the characterstics eq. is determined as:")
 8  CH=(s*((s+6)^2))+K
 9  CH=sym('(s*((s+6)^2))+K');
10  disp('=0',CH,"characterstics_eq ,CH=")
11  c0=coeffs(CH,'s',0);
12  c1=coeffs(CH,'s',1);
13  c2=coeffs(CH,'s',2);
14  c3=coeffs(CH,'s',3);
15  b=[c0 c1 c2 c3]
16  n=4;
17  routh=[b([4,2]);b([3,1])];
18  routh=[routh;-det(routh)/routh(2,1),0]
19  t=routh(2:3,1:2)
20  routh=[routh;-det(t)/t(2,1),0]
21  K=sym('-(s*((s+6)^2))')
22  d=diff(K,s)
23  e=3*s^2+24*s+36
24  r1=roots(e)
25  disp(r1,"roots=")
26  disp("-2 is break away point sinc it lies on root
       locus")
27  disp(routh,"routh=")
28  disp("for given system to be marginally stable:");
29  disp("(-(K-432)/12)=0 ");
30  disp("which gives:");
31  disp("K=432");
32  K=432;//considering positive value
33  k=432
34  a=12*s^2+k//intersection of root locus with
       imaginary axis plane
35  r=roots(a)
36  g=k/(s*((s+6)^2))
37  G=syslin('c',g)
38  clf();
39  evans(g,5)
40  xgrid(2)
41  disp(r,"the point of intersection of root locus with
```
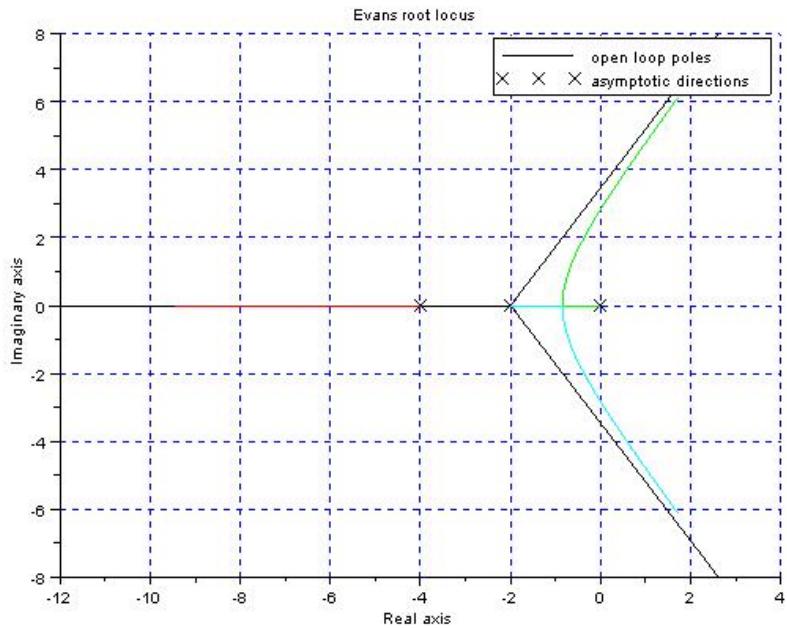
Figure 12.14: Root Locus and Value of K

```
imaginary  axis  =")
```

Scilab code Exa 12.48 Root Locus and Value of K

```
1 // caption : root_locus_and_value_of_K
2 // example  12_48
3 // page  574
4 s=%s ;
5 syms  K;
6 GH=K/(s*(s+2)*(s+4))
7 zeta=0.277
```

```
 8  //from given data
 9  disp("the characterstics eq. is determined as:")
10  CH=(s*(s+2)*(s+4))+K
11  K=sym('-(s^3+6*s^2+8*s)')
12  d=diff(K,s)
13  e=-3*s^2-12*s-8
14  r1=roots(e)
15  disp(r1,"roots=")
16  disp("-0.85 is break away point sinc it lies on root
        locus")
17  CH=sym('s^3+6*s^2+8*s+K');
18  disp('=0',CH,"characterstics_eq ,CH=")
19  c0=coeffs(CH,'s',0);
20  c1=coeffs(CH,'s',1);
21  c2=coeffs(CH,'s',2);
22  c3=coeffs(CH,'s',3);
23  b=[c0 c1 c2 c3]
24  n=4;
25  routh=[b([4,2]);b([3,1])];
26  routh=[routh;-det(routh)/routh(2,1),0]
27  t=routh(2:3,1:2)
28  routh=[routh;-det(t)/t(2,1),0]
29  disp(routh,"routh=")
30  disp("for given system to be marginally stable:");
31  disp("(48-K)=0 ");
32  disp("which gives:");
33  disp("K=48");
34  K=48;
35  k=48
36  a=6*s^2+48//intersection of root locus with
      imaginary plane
37  r=roots(a)
38  g=k/(s*(s+2)*(s+4))
39  G=syslin('c',g)
40  evans(g,8)
41  xgrid(2)
42  disp("the line theta=acos(zeta)=73.9 intersects root
        locus at sa=(-0.5+i1.66)")
```
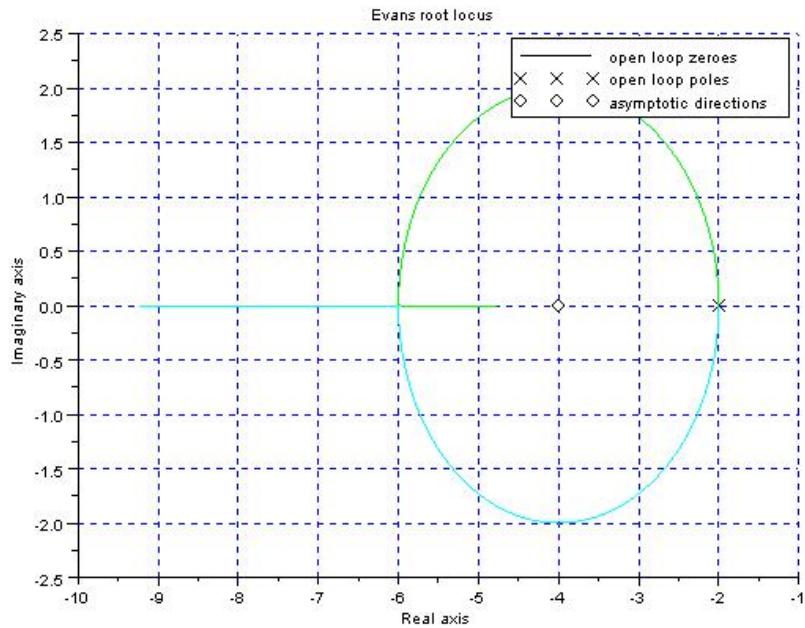
Figure 12.15: Root Locus and Value of K

```
43  disp("the value of K at s=sa is find to be 15 for
        zeta=0.277")
44  K=15
45  k=15
46  disp(r,"the point of intersection of root locus with
        imaginary axis =")
47  g=k/(s*(s+2)*(s+4))
48  cl=g/(1+g)
49  disp(cl,"C(s)/R(s)=")
```

Scilab code Exa 12.49 Root Locus and Value of K

```
1  // caption : root_locus_and_value_of_K
2  // example 12.49
3  // page 576
4  s=%s;
5  syms K;
6  GH=(K*(s+4))/(s+2)^2
7  disp("the characterstics eq. is determined as:")
8  CH=(s+2)^2+(K*(s+4))
9  CH=sym('((s+2)^2)+K*(s+4)');
10 disp('=0',CH,"characterstics_eq ,CH=")
11 K=sym('((s+2)^2/(s+4))')
12 d=diff(K,s)
13 e=(s+2)*(s+6)
14 r1=roots(e)
15 disp(r1,"roots=")
16 disp("-2 and -6 is break away point")
17 g=(s+4)/((s+2)^2)
18 G=syslin('c',g)
19 clf();
20 evans(g,10)
21 xgrid(2)
22 disp("for wd=2rad/sec ,the point on root locus is s
       =-4+j2")
23 disp("the value of K at s=-4+j2 is 4")
24 K=4
25 k=4
26 g=k*(s+4)/((s+2)^2)
27 cl=g/(1+g)
28 disp(cl,"C(s)/R(s)=")
```

Scilab code Exa 12.50 Root Locus and Closed loop Transfer Function

```
1  // caption :
```
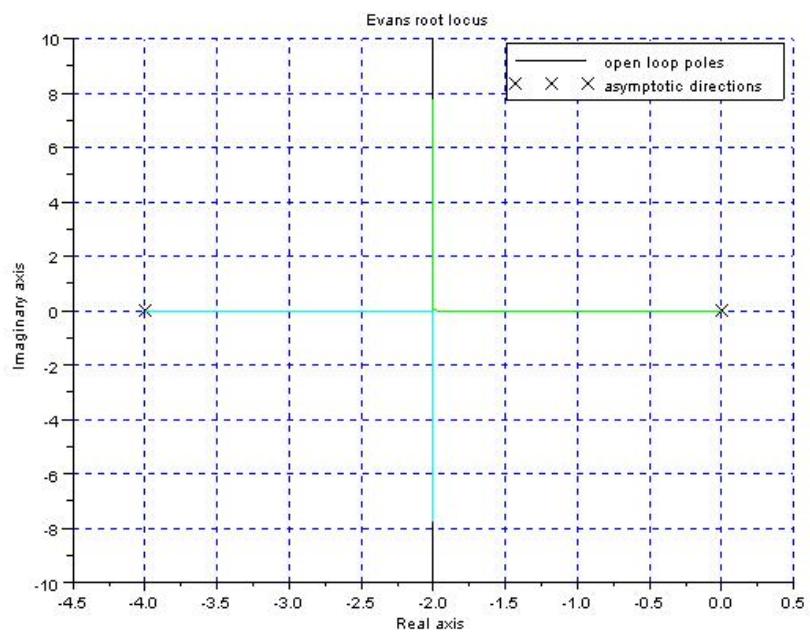
Figure 12.16: Root Locus and Closed loop Transfer Function

```
        root_locus_and_close_loop_transfer_function
2  //example 12.50
3  //page 578
4  s=%s
5  K=8
6  G=K/(s*(s+4))
7  H=1;
8  GH=G*H
9  G=syslin('c',G)
10 evans(G,8)
11 xgrid(2)
12 CH=s*(s+4)+K
13 disp('=0',CH,"characterstics_eq ,CH=")
14 r=roots(CH)
15 disp(r,"the point at which K=8")
16 cl=G/(1+GH)
17 disp(cl,"C(s)/R(s)=")
18
19
20
21 disp("part b")
22 g=K/(s+4)
23 h=1/s
24 gh=g*h
25 CL=g/(1+gh)
26 disp(CL,"C(s)/R(s)=")
```

Scilab code Exa 12.51 Root Locus and Gain and Phase Margin

```
1  //caption:root_locus_and_gain ,phase_margin
2  //example 12.51
3  //page 580
4  s=%s;
```
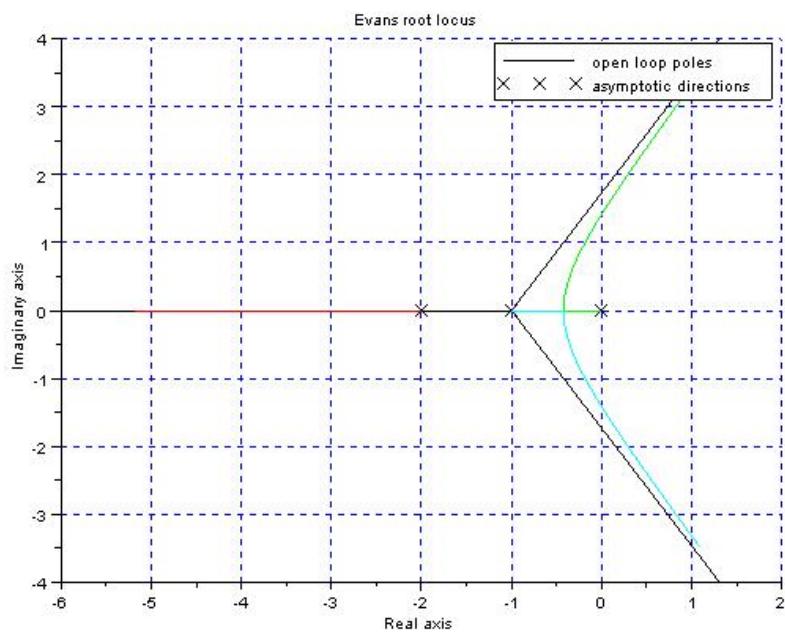
Figure 12.17: Root Locus and Gain and Phase Margin

```
5  K =3.46
6  G=K /( s *( s +1) *( s +2) )
7  G=syslin ( ' c ' ,G)
8  clf () ;
9  evans (G ,20)
10 xgrid (2)
11 [gm , freq_gm ]=g_margin (G)
12 [pm , freq_pm ]=p_margin (G)
13 disp (gm ," gain_margin =" , freq_gm *2*%pi ,"
       gain_margin_freq =" )
14 disp (pm ," phase_margin =" , freq_pm *2*%pi ,"
       phase_margin_freq =" )
```

Scilab code Exa 12.54 Obtain State Matrix

```
1  // caption : obtain_state_matrix
2  // example  12_54
3  // page  583
4  s =%s ;
5  g=5*( s +2) /( s *( s +1) *( s +5) ) ;
6  CL=syslin ( ' c ' ,g) ;
7  disp (CL ," C( s )/R( s )=" ) ;
8  SS=tf2ss (CL)
9  [Ac ,Bc ,U, ind ]=canon (SS (2) ,SS (3) )
10 disp (SS ," state  space  matrix =" )
11 disp (Ac ," Ac" ,Bc ," Bc" ,U ," U" , ind ," ind " )
```

Scilab code Exa 12.55 Obtain State Matrix

```
1  // caption : obtain_state_matrix
2  // example  12_55
3  // page  585
4  s =%s ;
```

```
5  g=50/(s*(s^2+0.5*s+50));
6  CL=syslin('c',g);
7  disp(CL,"C(s)/R(s)=");
8  SS=tf2ss(CL)
9  [Ac,Bc,U,ind]=canon(SS(2),SS(3))
10 disp(SS,"state space matrix=")
11 disp(Ac,"Ac",Bc,"Bc",U,"U",ind,"ind")
```

Scilab code Exa 12.56 Obtain State Transistion Matrix

```
1  //caption:obtain_state_transistion_matrix
2  //example 12_56
3  //page 586
4  s=%s;
5  syms t
6  A=[0 1;0 -3]
7  [r c]=size(A);//size of matrix A
8  p=s*eye(r,c)-A;//s*I-A where I is identity matrix
9  q=det(p)//determinant of sI-A
10 r=inv(p)//inverse of sI-A
11 //for calculating state transistion matrix
12 ip=[0 0;0 0]
13 i=1;
14 j=1;
15 for i=1:2
16   for j=1:2
17     if(i==2 & j==1)
18       else
19     ip(i,j)=ilaplace(r(i,j),s,t);
20     j=j+1;
21     end
22   end
23   i=i+1;
24 end
25 r(2,1)=0
```

```
26  disp(ip,"state transistion matrix,ip(t)=");
```

Scilab code Exa 12.57 Obtain Time Response

```
1  //caption:obtain_time_response
2  //example 12_57
3  //page 586
4  s=%s;
5  syms t
6  A=[0 1;-2 -3]
7  B=[1 0]'
8  x0=[0 0]'
9  u=1/(s+1)
10 [r c]=size(A);//size of matrix A
11 p=s*eye(r,c)-A;//s*I-A where I is identity matrix
12 q=det(p)//determinant of sI-A
13 r=inv(p)//inverse of sI-A
14 m=r*B*(1/(s+1));
15 //for calculating zero state response
16 x=[0;0]
17 x(1,1)=ilaplace(m(1,1),s,t);
18 x(2,1)=ilaplace(m(2,1),s,t);
19 disp(x,"time response of the system,x(t)=");
```

Scilab code Exa 12.59 Obtain Time Response

```
1  //caption:obtain_time_response
2  //example 12_59
3  //page 590
4  s=%s;
5  syms t
6  A=[-1 0;1 -1]
7  B=[0 1]'
```

```
 8  x0 =[1  0] '
 9  [r c]= size (A); // size  of  matrix  A
10  p=s*eye(r,c)-A; //s*I-A  where  I  is  identity  matrix
11  q=det(p) // determinant  of  sI-A
12  r=inv(p) // inverse  of  sI-A
13  m=r*B*(1/s)
14  r1=r*x0
15  X=r1+m
16  // for  calculating  zero  state  response
17  x=[0;0]
18  x(1,1)=ilaplace(X(1,1),s,t);
19  x(2,1)=ilaplace(X(2,1),s,t);
20  disp(x," zero  input  response  of  the  system ,x(t)=");
```

Scilab code Exa 12.61 Obtain Transfer Matrix

```
 1  // caption : obtain_transfer_matrix
 2  // example  12_61
 3  // page  592
 4  s=%s;
 5  syms t
 6  A=[-1  -1;3  -5]
 7  B=[1  1] '
 8  C=[1  2]
 9  [r c]=size(A); // size  of  matrix  A
10  p=s*eye(r,c)-A; //s*I-A  where  I  is  identity  matrix
11  q=det(p) // determinant  of  sI-A
12  r=inv(p) // inverse  of  sI-A
13  G=C*r*B
14  disp(G," transfer_matrix=")
```

Appendix

Scilab code AP 1 SERIES OF TWO FUNCTION

```
1 function [y] = series (sys1,sys2)
2 y =  sys1*sys2
3 endfunction
```

Scilab code AP 2 PARALLEL OF TWO FUNCTION

```
1 function [y]= parallel(sys1,sys2)
2 y=sys1+sys2;
3 endfunction
```