

Scilab Textbook Companion for
Control Systems Engineering
by I. J. Nagrath And M. Gopal ¹

Created by
Anuj Sharma
B.E. (pursuing)
Electrical Engineering
Delhi Technological University
College Teacher
Ram Bhagat, DTU, Delhi
Cross-Checked by
sonanaya tatikola, IITB

May 16, 2016

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

Book Description

Title: Control Systems Engineering

Author: I. J. Nagrath And M. Gopal

Publisher: New Age Publisher, New Delhi

Edition: 3

Year: 2007

ISBN: 81-224-1192-4

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

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

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

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

Contents

List of Scilab Codes	4
2 Mathematical Models of Physical Systems	6
3 Feedback Characteristics of control systems	8
5 Time Response analysis design specifications and performance indices	12
6 Concepts of stability and Algebraic Criteria	18
7 The Root Locus Technique	27
9 Stability in Frequency Domain	39
10 Introduction to Design	58
12 State Variable Analysis and Design	72

List of Scilab Codes

Exa 2.3	signal flow graph	6
Exa 2.4	transfer function	6
Exa 3.2	sensitivity of transfer function	8
Exa 3.3.a	sensitivity of transfer function	8
Exa 3.3.b	steady state error	9
Exa 3.3.c	calculation of slope	9
Exa 3.3.d	calculation of slope	10
Exa 3.3.f	calculation of input	10
Exa 3.3.g	calculation of time	11
Exa 5.2	steady state error	12
Exa 5.2.2	steady state error	13
Exa 5.3	transfer function	13
Exa 5.3.2	transfer function	14
Exa 5.4.1	steady state error	14
Exa 5.4.2	steady state error	15
Exa 5.4.3	steady state error	15
Exa 5.8	steady state error	16
Exa 5.9	state variable analysis	16
Exa 6.1	hurwitz criterion	18
Exa 6.2	routh array	19
Exa 6.3	routh array	19
Exa 6.4	routh array	20
Exa 6.5	routh array	20
Exa 6.6	routh criterion	21
Exa 6.7	routh array	22
Exa 6.8	routh array	23
Exa 6.9	routh array	23
Exa 6.10	routh array	24

Exa 6.11.a routh array	24
Exa 6.11.b routh array	25
Exa 7.1 root locus	27
Exa 7.2 root locus	27
Exa 7.3 root locus	29
Exa 7.4 root locus	30
Exa 7.6 root locus	32
Exa 7.8 root locus	34
Exa 7.9 root locus	36
Exa 7.10 root locus	36
Exa 9.1 nyquist plot	39
Exa 9.2 nyquist plot	39
Exa 9.3 nyquist plot	41
Exa 9.4 nyquist plot	42
Exa 9.5 nyquist plot	44
Exa 9.6 stability using nyquist plot	46
Exa 9.7.a stability using nyquist plot	48
Exa 9.7.b stability using nyquist plot	49
Exa 9.8.a nyquist criterion	50
Exa 9.8.b nyquist criterion	50
Exa 9.10 gm and pm using nyquist plot	50
Exa 9.11 bode plot	52
Exa 9.13.a bode plot	53
Exa 9.13.b bode plot	54
Exa 9.14 m circles	55
Exa 9.15 m circles	56
Exa 10.6 lead compensation	58
Exa 10.7 lead compensation	61
Exa 10.8 lag compnsation	64
Exa 10.9 lag and lead compensation	67
Exa 12.3 state matrix	72
Exa 12.4 modal matrix	72
Exa 12.5 obtain time response	73
Exa 12.6 resolvant matrix	74
Exa 12.7 state transition matrix and state response	74
Exa 12.12 check for controllability	75
Exa 12.13 check for controllability	76
Exa 12.14 check for observability	76

Exa 12.17 design state observer 77

List of Figures

7.1	root locus	28
7.2	root locus	29
7.3	root locus	30
7.4	root locus	31
7.5	root locus	32
7.6	root locus	33
7.7	root locus	35
7.8	root locus	37
9.1	nyquist plot	40
9.2	nyquist plot	41
9.3	nyquist plot	42
9.4	nyquist plot	43
9.5	nyquist plot	44
9.6	nyquist plot	45
9.7	stability using nyquist plot	47
9.8	stability using nyquist plot	48
9.9	stability using nyquist plot	49
9.10	gm and pm using nyquist plot	51
9.11	bode plot	52
9.12	bode plot	53
9.13	bode plot	54
9.14	m circles	55
9.15	m circles	57
10.1	lead compensation	59
10.2	lead compensation	60
10.3	lead compensation	62
10.4	lead compensation	63

10.5 lag compnsation	65
10.6 lag compnsation	66
10.7 lag and lead compensation	68
10.8 lag and lead compensation	69

Chapter 2

Mathematical Models of Physical Systems

Scilab code Exa 2.3 signal flow graph

```
1 s=%s;
2 syms L C R1 R2
3 // forward path denoted by P! and loop by L1,L2 and
  so on
4 // path factor by D1 and graph determinant by D
5 P1=1/(s*L*s*C);
6 L1=-R1/(s*L);
7 L2=-1/(s*R2*C);
8 L3=-1/(s^2*L*C);
9 D1=1;
10 D=1-(L1+L2+L3);
11 Y=(P1*D1)/D;
12 disp(Y," Transfer function=")
```

Scilab code Exa 2.4 transfer function

```

1 syms xv Qf Qo Cf Co V Qw Kv
2 Qo=Qw+Qf;
3 // rate of salt inflow
4 mi=Qf*Cf;
5 // rate of salt outflow
6 mo=Qo*Co;
7 // rate of salt accumulation
8 ma=diff(V*Co,t);
9 mi=ma+mo;
10 Qf*Cf=V*diff(Co,t)+Qo*Co;
11 Qf=Kv*xv;
12 K=Cf*Kv/Qo;
13 G=V/Qo;
14 G*diff(Co,t)+Co=K*xv;
15 // taking laplace
16 G*s*Co+Co=K*xv;
17 // transfer function= Co/xv
18 Co/xv=K/(G*s+1);

```

Chapter 3

Feedback Characteristics of control systems

Scilab code Exa 3.2 sensitivity of transfer function

```
1 syms K;
2 s=%s;
3 G=syslin('c',25*(s+1)/(s+5));
4 p=K;
5 q=s^2+s;
6 J=p/q;
7 F=G*J;
8 T=F/(1+F); // Closed loop transfer function
9 disp(T,"C(s)/R(s)")
10 // sensitivity w.r.t K = dT/dK*K/T
11 S=(diff(T,K))*(K/T)
12 disp(S,"Sensitivity")
```

Scilab code Exa 3.3.a sensitivity of transfer function

```
1 syms K1 K t;
```

```

2 s=%s;
3 p=K1*K;
4 q=t*s+1+(K1*K);
5 T=p/q;
6 disp(T,"V(s)/R(s)")
7 // sensitivity w.r.t K is dT/dK*K/T
8 S=(diff(T,K))*(K/T)
9 // given K1=50 K=1.5
10 s=0
11 S=horner(S,s)
12 K1=50;
13 K=1.5;
14 S=1/(1+K1*K)
15 disp(S,"sensitivity=")

```

Scilab code Exa 3.3.b steady state error

```

1 syms A K K1 t
2 s=%s;
3 p=K1*K*A;
4 q=s*(1+(t*s)+(K1*K));
5 K=1.5;
6 K1=50;
7 V=p/q
8 v=limit(s*V,s,0)
9 // given steady state speed = 60km/hr
10 A=60*(1+(K1*K))/(K1*K)
11 // steady error e(ss)=A-v
12 v=60;
13 e=A-v;
14 disp(e,"e(ss)=")

```

Scilab code Exa 3.3.c calculation of slope

```

1 // under stalled conditions
2 syms Kg K1 D;
3 A=60.8;
4 A*K1=Kg*D;
5 // given Kg=100
6 Kg=100;
7 K1=50;
8 D=(A*K1)/Kg;
9 disp(D," upslope=")

```

Scilab code Exa 3.3.d calculation of slope

```

1 // steady speed=10km/hr
2 syms K Kg D
3 (((A-10)*K1)-(-D*Kg))K=100;
4 A=(60.8*10)/60;
5 K=1.5;
6 Kg=100;
7 D=((100/K)-((A-10)*K))/Kg;
8 disp(D,"Down slope=")

```

Scilab code Exa 3.3.f calculation of input

```

1 // for open loop system
2 // given speed=60km/hr
3 syms R K1 K;
4 (R*K1*K)=60
5 K1=50;
6 K=1.5;
7 R=60/(K1*K)
8 disp(R,"Input open=")
9 // for closed loop
10 R=60(1+(K1*K))/(K1*K)

```

```
11 disp(R,"Input closed=")
```

Scilab code Exa 3.3.g calculation of time

```
1 // for open loop
2 syms t g s;
3 s=%s;
4 K1=50;
5 K=1.5;
6 g=20;
7 V=syslin('c',((K1*K)*0.8)/(s*((g*s)+1)))
8 // taking inverse laplace
9 v=ilaplace(V,s,t)
10 v=60(1-%e^(-t/20))
11 // given v=90%
12 v=0.9;
13 t=-20*log(1-v);
14 disp(t,"time open=")
15 // for closed loop
16 syms K' g'
17 s=%s;
18 V=syslin('c',(60.8*K')/(s*((g'*s)+1)))
19 // taking inverse laplace
20 v=ilaplace(V,s,t)
21 // given
22 K'=75/76;
23 g'=.263;
24 v=60(1-%e^(-t/.263))
25 // at v=90%
26 v=0.9;
27 t=-.263*log(1-(v/60));
28 disp(t,"time closed=")
```

Chapter 5

Time Response analysis design specifications and performance indices

Scilab code Exa 5.2 steady state error

```
1 s=%s
2 syms K J f
3 K=60; // given
4 J=10; // given
5 p=K/J
6 q=K/J+(f/J)*s+s^2
7 G=p/q;
8 disp(G,"Qo(s)/Qi(s)=")
9 zeta=0.3; // given
10 cof1=coeffs(q,'s',0)
11 // on comparing the coefficients
12 Wn=sqrt(cof1)
13 cof2=coeffs(q,'s',1)
14 // 2*zeta*Wn=cof2
15 f/J=2*zeta*Wn
16 r=s^2+f/J
17 s=s^2+f/J+K/J
```



```
18 H=r/s;
19 disp(H,"Qe(s)/Qi(s)=")
```

Scilab code Exa 5.2.2 steady state error

```
1 // given Qi(s)=0.04/s^2
2 Qi=0.04/s^2;
3 e=limit(s*Qi*H,s,0)
4 disp(e,"Steady stste eror=")
```

Scilab code Exa 5.3 transfer function

```
1 s=%s;
2 syms Kp Ka Kt J f
3 // given
4 J=0.4;
5 Kp=0.6;
6 Kt=2;
7 f=2;
8 p=Kp*Ka*Kt
9 q=s^2+f/J+(Kp*Ka*Kt)/J
10 G=p/q;
11 disp(G,"Qm(s)/Qr(s)=")
12 cof_1=coeffs(q,'s',0)
13 // on comparing the coefficients
14 // Wn=sqrt(cof_1)
15 Wn=10;
16 Ka=(Wn)^2*J/(Kp*f)
17 disp(Ka,"Amplifier Constant=")
```

Scilab code Exa 5.3.2 transfer function

```
1 s=%s;
2 syms Kp Ka Kt Kd J f
3 // given
4 J=0.4;
5 Kp=0.6;
6 Kt=2;
7 f=2;
8 Ka=5;
9 p=Kp*Ka*Kt
10 q=s^2+((f+Ka*Kd*Kt)/J)*s+(Kp*Ka*Kt)/J
11 G=p/q;
12 disp(G,"Qm(s)/Qr(s)=")
13 cof_1=coeffs(q,'s',0)
14 // on comparing the coefficients
15 Wn=sqrt(cof_1)
16 zeta=1 // given
17 cof_2=coeffs(q,'s',1)
18 // 2*zeta*Wn=cof_2
19 Kd=(2*zeta*sqrt(Kp*J*Ka*Kt)-f)/(Ka*Kt)
20 disp(Kd,"Tachogenertor constant=")
```

Scilab code Exa 5.4.1 steady state error

```
1 syms Wn zeta Kv Ess
2 s=%s;
3 p=poly([8 2 1],'s','coeff'); // characteristic
    equation
4 z=coeff(p);
5 Wn=sqrt(z(1,1))
6 zeta=z(1,2)/(2*Wn)
7 Kv=z(1,1)/z(1,2)
8 Ess=1/Kv // Steady state error for unit ramp i/p
9 disp(Ess,"Steady state Error=")
```

Scilab code Exa 5.4.2 steady state error

```
1 // with derivative feedback
2 // characteristic equation is
3 syms a
4 s=%s;
5 p=s^2+(2+(8*a))*s+8=0
6 zeta=0.7 // given
7 Wn=2.828;
8 cof_1=coeffs(p, 's', 1)
9 // on comparing 2*zeta*Wn=cof_1
10 a=((2*zeta*Wn)-2)/8
11 disp(a, "Derivative feedback=")
12 cof_2=coeffs(p, 's', 0)
13 cof_1=2+8*0.245;
14 Kv=cof_2/cof_1;
15 Ess=1/Kv
16 disp(Ess, "Steady state error=")
```

Scilab code Exa 5.4.3 steady state error

```
1 // let the char equation be
2 syms Ka
3 s=%s;
4 p=s^2+(2+(a*Ka))*s+Ka=0
5 cof_1=coeffs(p, 's', 0)
6 // Wn^2=cof_1
7 Wn=sqrt(cof_1)
8 cof_2=coeffs(p, 's', 1)
9 // 2*zeta*Wn=cof_2
10 Kv=cof_1/cof_2;
```

```

11 Ess=1/Kv;
12 // given Ess=0.25
13 Ess=0.25;
14 Ka=2/(Ess-a)
15 disp(Ka."Ka=")

```

Scilab code Exa 5.8 steady state error

```

1 s=%s;
2 syms K V
3 p=s^2+(100*K)*s+100=0
4 cof_1=coeffs(p,'s',0)
5 Wn=sqrt(cof_1)
6 zeta=1 // given
7 cof_2=coeffs(p,'s',1)
8 // 2*zeta*Wn=cof_2
9 K=(2*Wn*zeta)/100
10 // For ramp input
11 R=V/s^2
12 E=R/p
13 // steady state error
14 e=limit(s*E(s),s,0)
15 disp(e,"e(ss)=")

```

Scilab code Exa 5.9 state variable analysis

```

1 s=%s;
2 syms t m
3 A=[0 1;-100 -20];
4 B=[0;100];
5 C=[1 0];
6 x=[0;0];
7 [r c]=size(A)

```

```

8 p=s*eye(r,c)-A
9 q=inv(p);
10 disp(q,"phi(s)=") // Resolvant matrix
11 for i=1:r;
12 for j=1:c;
13 q(i,j)=ilaplace(q(i,j),s,t)
14 end
15 end
16 disp(q,"phi(t)=") // State transition matrix
17 t=t-m;
18 q=eval(q)
19 // Integrate q w.r.t m
20 r=integrate(q*B,m)
21 m=0 // Upper limit is t
22 g=eval(r) // Putting upper limit in q
23 m=t // Lower limit is 0
24 h=eval(r) // Putting lower limit in q
25 y=(h-g);
26 disp(y,"y=")
27 printf("x(t)=phi(t)*x(0)+integrate(phi(t-m*B) w.r.t
      m from 0 to t)")
28 y1=(q*x)+y;
29 disp(y1,"x(t)=")
30 // transfer function
31 t=C*q*B;
32 disp(t,"T(s)=")

```

Chapter 6

Concepts of stability and Algebraic Criteria

Scilab code Exa 6.1 hurwitz criterion

```
1 s=%s;
2 p=s^4+8*s^3+18*s^2+16*s+5
3 r=coeff(p)
4 D1=r(4)
5 d2=[r(4) r(5);r(2) r(3)]
6 D2=det(d2);
7 d3=[r(4) r(5) 0;r(2) r(3) r(4);0 r(1) r(2)]
8 D3=det(d3);
9 d4=[r(4) r(5) 0 0;r(2) r(3) r(4) r(5);0 r(1) r(2) r
    (3);0 0 0 r(1)]
10 D4=det(d4);
11 disp(D1,"D1=")
12 disp(D2,"D2=")
13 disp(D3,"D3=")
14 disp(D4,"D4=")
15 printf("Since all the determinants are positive the
    system is stable")
```

Scilab code Exa 6.2 routh array

```
1 s=%s;  
2 p=s^4+8*s^3+18*s^2+16*s+5  
3 r=routh_t(p)  
4 m=coeff(p)  
5 l=length(m)  
6 c=0;  
7 for i=1:l  
8   if (r(i,1)<0)  
9     c=c+1;  
10  end  
11 end  
12 if(c>=1)  
13   printf("System is unstable")  
14 else ("Sysem is stable")  
15 end
```

Scilab code Exa 6.3 routh array

```
1 s=%s;  
2 p=3*s^4+10*s^3+5*s^2+5*s+2  
3 r=routh_t(p)  
4 m=coeff(p)  
5 l=length(m)  
6 c=0;  
7 for i=1:l  
8   if (r(i,1)<0)  
9     c=c+1;  
10  end  
11 end  
12 if(c>=1)
```

```

13 printf("System is unstable")
14 else ("Sysem is stable")
15 end

```

Scilab code Exa 6.4 routh array

```

1 s=%s;
2 syms Kv Kd Kp Kt
3 p=s^3+(1+(Kv*Kd))*s^2+(Kv*Kp)*s+(Kp*Kt)
4 cof_a_0=coeffs(p,'s',0);
5 cof_a_1=coeffs(p,'s',1);
6 cof_a_2=coeffs(p,'s',2);
7 cof_a_3=coeffs(p,'s',3);
8 r=[cof_a_0 cof_a_1 cof_a_2 cof_a_3]
9 n=length(r);
10 routh=[r([4,2]);r([3,1])]
11 routh=[routh;-det(routh)/routh(2,1),0]
12 t=routh(2:3,1:2);
13 routh=[routh;-det(t)/routh(3,1),0]
14 disp(routh,"routh=");
15 // for stability r(:,1)>0
16 // for the given table
17 b=routh(3,1)
18 disp("for stability"b,">0")

```

Scilab code Exa 6.5 routh array

```

1 s=%s;
2 syms K
3 p=(1+K)*s^2+((3*K)-0.9)*s+(2K-1)
4 cof_a_0=coeffs(p,'s',0);
5 cof_a_1=coeffs(p,'s',1);
6 cof_a_2=coeffs(p,'s',2);

```



```

7 r=[cof_a_0 cof_a_1 cof_a_2]
8 n=length(r);
9 routh=[r([3,1]);r(2),0];
10 routh=[routh;-det(routh)/routh(2,1),0];
11 disp(routh,"routh=")
12 // for no root in right half
13 // routh(1,1),routh(2,1),routh(3,1)>0
14 routh(1,1)=0
15 routh(2,1)=0
16 routh(3,1)=0
17 // combining the result
18 K=0.9/3;
19 disp(K,"For no roots in right half=")
20 // for 1 pole in right half i.e. one sign change
21 //routh(1,1)>0 n routh(3,1)<0
22 disp("For one pole in right half, -1<K<0.05")
23 // for 2 poles in right half
24 // routh(2,1)<0 n routh(3,1)>0
25 disp("For 2 poles in right half, 0.05<K<0.3")

```

Scilab code Exa 6.6 routh criterion

```

1 s=%s;
2 syms K
3 p=s^2-(K+2)*s+((2*K)+5)
4 cof_a_0=coeffs(p,'s',0);
5 cof_a_1=coeffs(p,'s',1);
6 cof_a_2=coeffs(p,'s',2);
7 r=[cof_a_0 cof_a_1 cof_a_2]
8 n=length(r);
9 routh=[r([3,1]);r(2),0];
10 routh=[routh;-det(routh)/routh(2,1),0];
11 disp(routh,"routh=")
12 // for system to be stable
13 routh(2,1)>0

```

```

14 K<-2;
15 routh(3,1)>0
16 K>-2.5;
17 disp("For stable system , -2>K>-2.5")
18 // for limited stability
19 routh(2,1)=0
20 K=-2
21 routh(3,1)=0
22 K=-2.5
23 disp("For limited stable system K=-2 and K=-2.5")
24 // for unstable system
25 disp("For unstable system K<-2 or K>-2.5")
26 roots(p) // gives the roots of the polynomial m
27 // for critically damped case
28 g=(K+2)^2-4*((2*K)+5)
29 roots(g)
30 // for stability K=6.47 is unstable
31 // for critical damping K=-2.47
32 disp("For underdamped case , -2>K>-2.47")
33 disp("for overdamped case , -2.47>K>-2.5")

```

Scilab code Exa 6.7 routh array

```

1 s=%s;
2 syms eps
3 p=s^5+s^4+2*s^3+2*s^2+3*s+5
4 r=coeff(p);
5 n=length(r);
6 routh=[r([6,4,2]);r([5,3,1])]
7 syms eps;
8 routh=[routh;eps,-det(routh(1:2,2:3))/routh(2,2),0];
9 routh=[routh;-det(routh(2:3,1:2))/routh(3,1),-det(
    routh(2:3,2:3))/routh(4,2),0];
10 routh=[routh;-det(routh(4:5,1:2))/routh(5,1),0,0];
11 disp(routh," routh=")

```

```

12 // to check stability
13 routh(4,1)=8-limit(5/eps,eps,0);
14 disp(routh(4,1),"routh(4,1)=")
15 routh(5,1)=limit(routh(5,1),eps,0);
16 disp(routh(5,1),"routh(5,1)=")
17 printf("There are two sign changes of first column
        hence the system is unstable")

```

Scilab code Exa 6.8 routh array

```

1 s=%s;
2 p=s^6+2*s^5+8*s^4+12*s^3+20*s^2+16*s+16
3 r=routh_t(p)
4 roots(p)
5 disp(0,"the number of real part of roots lying in
        the right half")
6 printf("System is stable")

```

Scilab code Exa 6.9 routh array

```

1 s=%s;
2 syms K a
3 p=s^4+10*s^3+32*s^2+(K+32)*s+(K*a)
4 cof_a_0=coeffs(p,'s',0);
5 cof_a_1=coeffs(p,'s',1);
6 cof_a_2=coeffs(p,'s',2);
7 cof_a_3=coeffs(p,'s',3);
8 cof_a_4=coeffs(p,'s',4);
9 r=[cof_a_0 cof_a_1 cof_a_3 cof_a_4]
10 n=length(r);
11 routh=[r([5,3,1]);r([4,2]),0]
12 routh=[routh;-det(routh(1:2,1:2))/routh(2,1),-det(
        routh(1:2,2:3))/routh(2,2),0];

```

```

13 routh=[routh;-det(routh(2:3,1:2))/routh(3,1),-det(
    routh(2:3,2:3))/routh(3,2),0];
14 routh=[routh;-det(routh(3:4,1:2))/routh(4,1),0,0];
15 disp(routh,"routh=")
16 // for the given system to be stable
17 routh(3,1)>0
18 K<288;
19 routh(4,1)>0
20 (288-K)*(K+32)-100(K*a)>0
21 // let K=200
22 K=200;
23 a=((288-K)*(K+32))/(100*K)
24 // velocity error
25 Kv=(K*a)/(4*2*4);
26 // % velocity error
27 Kvs=100/Kv
28 disp(a,"control parameter=")
29 disp(K,"Gain=")

```

Scilab code Exa 6.10 routh array

```

1 s=%s;
2 p=s^3+7*s^2+25*s+39
3 // to check if the roots lie left of s=-1
4 // substitute s=s-1
5 p=(s-1)^3+7*(s-1)^2+25*(s-1)+20
6 r=routh_t(p)
7 printf("All the signs of elements first column are
    positive hence the roots lie left of s=-1")

```

Scilab code Exa 6.11.a routh array

```

1 s=%s;

```

```

2 syms K
3 // the system characteristic eq can be written as
4 p=s^3+8.5*s^2+20*s+12.5(1+K)
5 cof_a_0=coeffs(p,'s',0);
6 cof_a_1=coeffs(p,'s',1);
7 cof_a_2=coeffs(p,'s',2);
8 cof_a_3=coeffs(p,'s',3);
9 r=[cof_a_0 cof_a_1 cof_a_2 cof_a_3]
10 n=length(r);
11 routh=[r([4,2]);r([3,1])]
12 routh=[routh;-det(routh)/routh(2,1),0]
13 t=routh(2:3,1:2);
14 routh=[routh;-det(t)/routh(3,1),0]
15 disp(routh,"routh=");
16 // for limiting value of K
17 routh(3,1)=0
18 K=12.6;
19 disp(K,"Limiting value of K")

```

Scilab code Exa 6.11.b routh array

```

1 syms zeta Wn ts z
2 // settling time ts=4/zeta*Wn
3 // given ts=4sec
4 ts=4;
5 zeta*Wn=ts/4
6 printf("now the real part of dominant root should be
       -1 or more")
7 // substituting s=s-1
8 p=(s-1)^3+8.5*(s-1)^2+20*(s-1)+12.5*(1+K)
9 cof_a_0=coeffs(p,'s',0);
10 cof_a_1=coeffs(p,'s',1);
11 cof_a_2=coeffs(p,'s',2);
12 cof_a_3=coeffs(p,'s',3);
13 r=[cof_a_0 cof_a_1 cof_a_2 cof_a_3]

```

```

14 n=length(r);
15 routh=[r([4,2]);r([3,1])]
16 routh=[routh;-det(routh)/routh(2,1),0]
17 t=routh(2:3,1:2);
18 routh=[routh;-det(t)/routh(3,1),0]
19 disp(routh,"routh=");
20 // for limiting value of K
21 routh(3,1)=0
22 K=2.64
23 disp(K,"Limiting value of K for settling time of 4s="
      ")
24 // roots of char eq at K=2.64
25 g=s^3+8.5*s^2+20*s+12.5*(1+2.64)
26 roots(g)

```

Chapter 7

The Root Locus Technique

Scilab code Exa 7.1 root locus

```
1 s=%s;  
2 syms k  
3 H=syslin('c', (k*(s+1)*(s+2))/(s*(s+3)*(s+4)));  
4 evans(H,5)  
5 printf("There are three branches of root locus  
   starting with K=0 and poles s=0,-3,-4.")  
6 printf("As k increases two branches terminate at  
   zeros s=-1,-2 and one at infinity")
```

Scilab code Exa 7.2 root locus

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

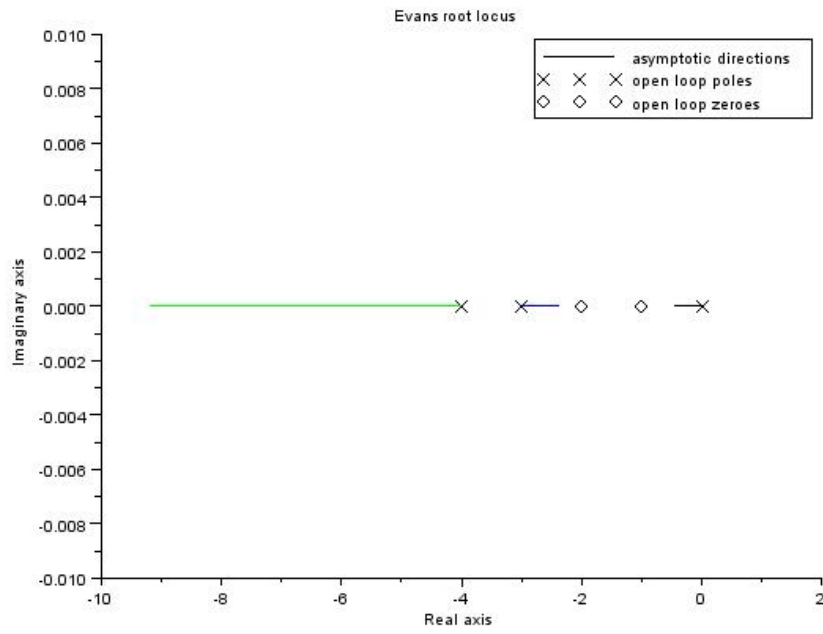


Figure 7.1: root locus

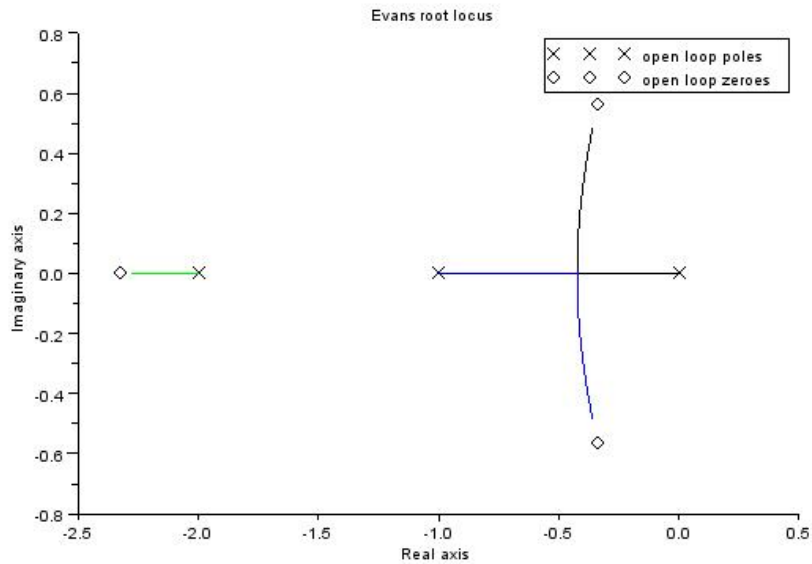


Figure 7.2: root locus

```

5 printf("The branches of root locus starts with K=0
    and poles s=0,-1,-2.")
6 printf("Since there is no open loop zero the
    branches terminate at infinity")

```

Scilab code Exa 7.3 root locus

```

1 s=%s;
2 syms k
3 H=syslin('c',1+(k/(s*(s+1)*(s+2))))
4 evans(H,5)
5 d=derivat(H)
6 p=numer(d)

```

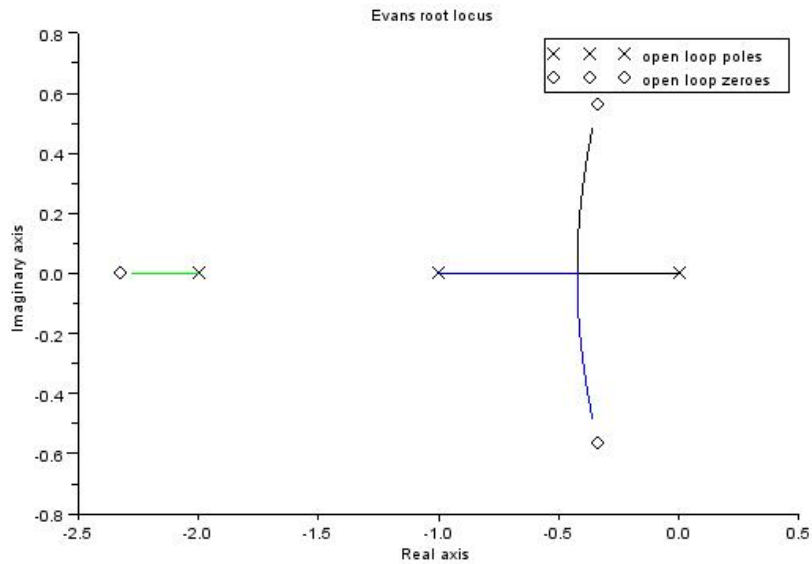


Figure 7.3: root locus

```
7 a=roots(p) // a=breakaway point
8 disp(a,"breakaway ppoint=")
```

Scilab code Exa 7.4 root locus

```
1 s=%s;
2 syms k
3 H=syslin('c',k/(s*(s+4)*(s^2+(4*s)+20)))
4 evans(H,1000)
5 printf("Since there are no open loop zeros all
        branches terminate at infimty")
```

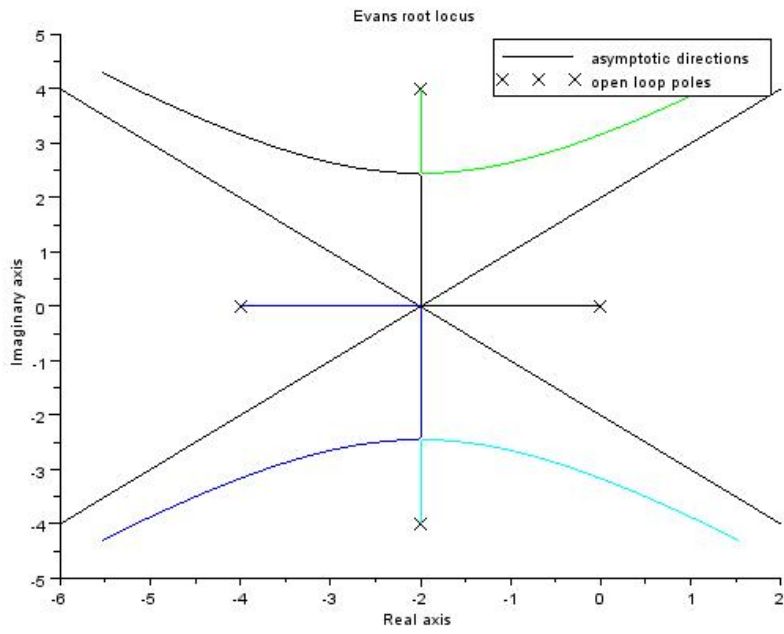


Figure 7.4: root locus

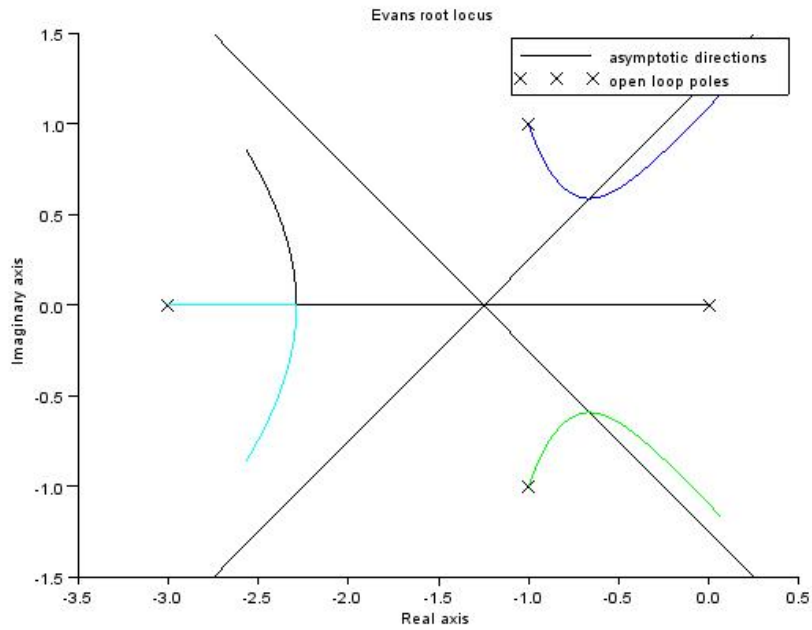


Figure 7.5: root locus

Scilab code Exa 7.6 root locus

```

1 s=%s;
2 syms k
3 H=syslin('c',k/(s*(s+3)*(s^2+(2*s)+2)))
4 evans(H,10)

```

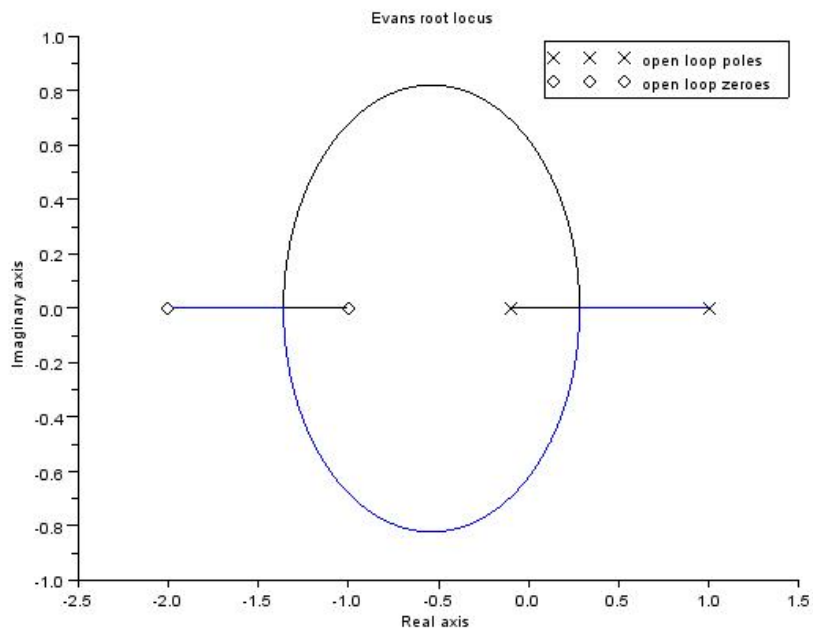


Figure 7.6: root locus

Scilab code Exa 7.8 root locus

```
1 syms K
2 s=%s;
3 G=syslin('c', (K*(s+1)*(s+2))/((s+0.1)*(s-1)))
4 evans(G)
5 n=2;
6 disp(n,"no of poles=")
7 m=2;
8 disp(m,"no of zeroes=")
9 K=kpure(G)
10 disp(K,"value of K where RL crosses jw axis=")
11 d=derivat(G)
12 p=numer(d)
13 a=roots(p); // a=breakaway points
14 disp(a,"breakaway points=")
15 for i=1:2
16     K=-(a(i,1)+0.1)*(a(i,1)-1)/((a(i,1)+1)*(a(i,1)
17         +2))
18     disp(a(i,1),"s=")
19     disp(K,"K=")
20 end
21 printf("zeta=1 is achieved when the two roots are
22     equal and negative(real).This happens at the
23     breakaway point in the left half s-plane/n")
24 zeta=1;
25 wn=0.6;
26 sgrid(zeta,wn)
27 K=-1/real(horner(G,[1 %i]*locate(1)));
28 disp(K,"The corresponding value of gain is=")
```

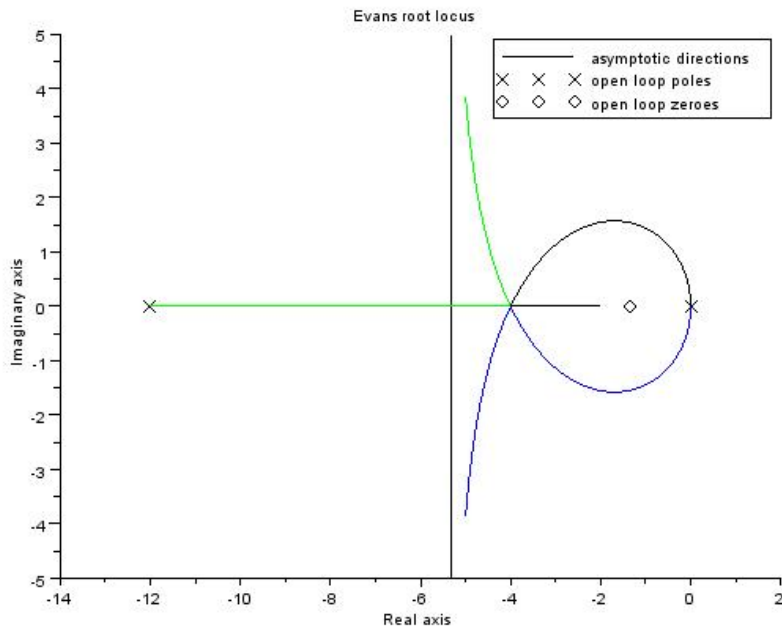


Figure 7.7: root locus

Scilab code Exa 7.9 root locus

```
1 syms K
2 s=%s;
3 G=syslin('c', (K*(s+4/3))/(s^2*(s+12)))
4 evans(G,60)
5 d=derivat(G)
6 p=numer(d)
7 a=roots(p) // a=breakaway points
8 disp(a,"Breakaway points=")
9 printf("Equal roots are at s=-4")
10 printf("/n Value of K at s=-4=")
11 K=4*4*8/(4-(4/3))
12 disp(K)
```

Scilab code Exa 7.10 root locus

```
1 syms Kh
2 s=%s;
3 G=syslin('c', 10*Kh*(s+0.04)*(s+1)/((s+0.5)*(s
   ^2-(0.4*s)+0.2)*(s+8)));
4 evans(G,3)
5 Kh=kpure(G)
6 K=10*Kh
7 zeta=1/(2)^(1/2);
8 wn=.575;
9 sgrid(zeta,wn)
10 K=-1/real((2*horner(G,[1 %i]*locate(1))));
11 printf("The zeta=1/(2)^1/2 line intersects the root
   locus at two points with K1=1.155 and K2=0.79")
```

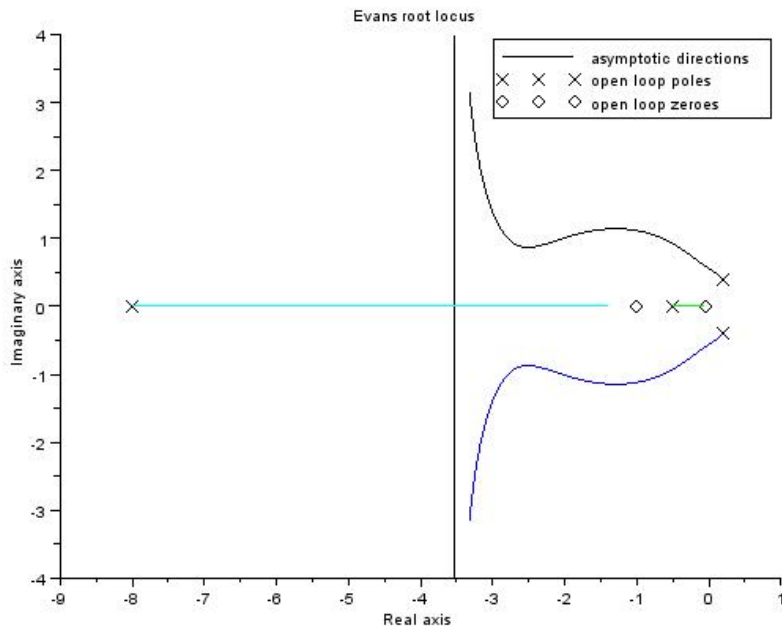



Figure 7.8: root locus

```
12 Kh1=0.156;
13 Kh2=0.079;
14 // from the block diagram
15 Td(s)=1/s;
16 E(s)=C(s)=G/(1+(G*Kh*(s+1))/(s+8))*Td(s);
17 // substituting value of G
18 F=s*E(s)=10*Kh/(1+(10*Kh));
19 // steady state error
20 ess=limit(F,s,0)
21 // for Kh1=0.156
22 ess=0.609;
23 // for Kh2=0.079
24 ess=0.44;
```

Chapter 9

Stability in Frequency Domain

Scilab code Exa 9.1 nyquist plot

```
1 s=%s;  
2 syms K T1 T2  
3 H=syslin('c',K/((T1*s+1)*(T2*s+1)));  
4 nyquist(H)  
5 show_margins(H,'nyquist')  
6 printf("Since P=0(no of poles in RHP) and the  
   nyquist contour does not encircle the point -1+j0  
   ")  
7 printf("System is stable")
```

Scilab code Exa 9.2 nyquist plot

```
1 s=%s;  
2 H=syslin('c',(s+2)/((s+1)*(s-1)))  
3 nyquist(H)
```

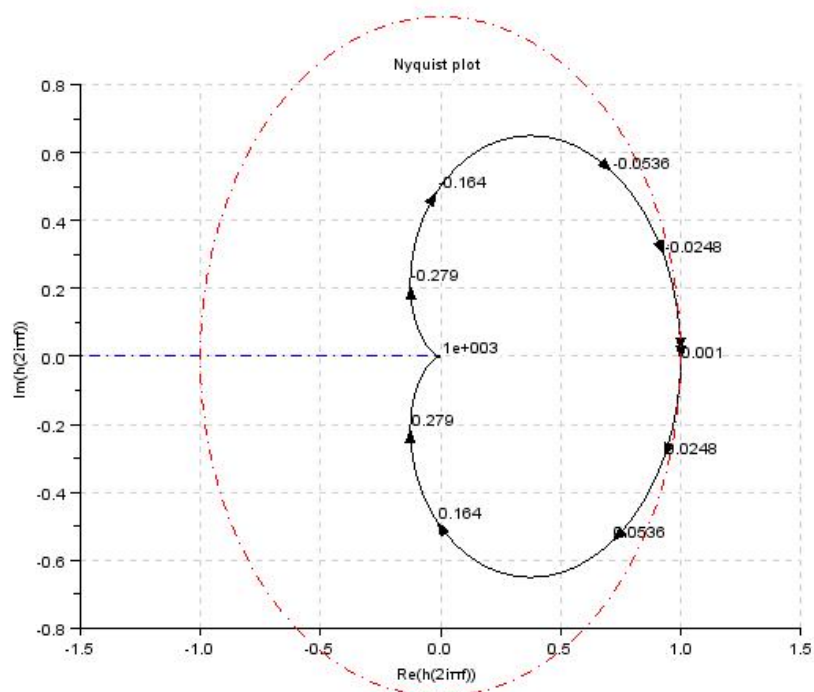


Figure 9.1: nyquist plot

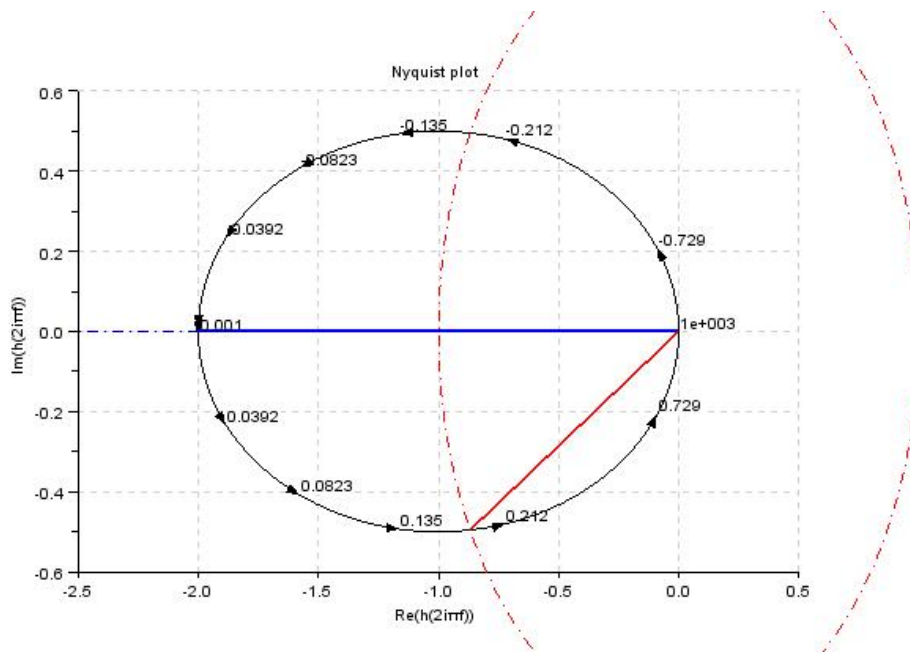


Figure 9.2: nyquist plot

```

4 show_margins(H, 'nyquist')
5 printf("Since P=1 and the pt. -1+j0 is encircled
   once by the locus")
6 printf("Hence N=1 therefore , Z=0(no of zeros in RHP)
   ")
7 printf("System is stable")

```

Scilab code Exa 9.3 nyquist plot

```

1 s=%s;
2 syms K T
3 H=syslin('c',K/(s*(T*s+1)))
4 nyquist(H)

```

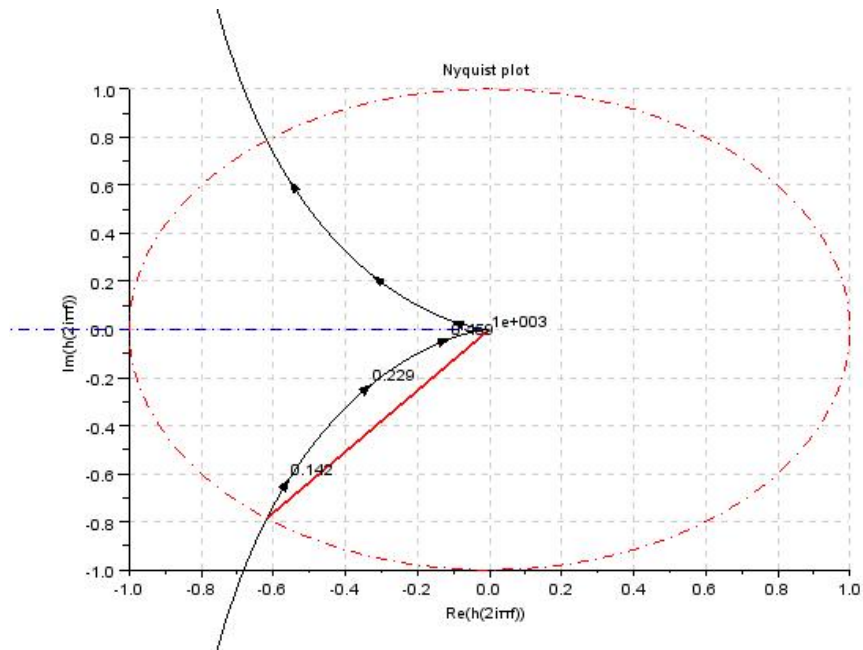


Figure 9.3: nyquist plot

```

5 show_margins(H, 'nyquist')
6 mtlb_axis([-1 1 -1 1])
7 printf("Since P=0(no of poles in RHP) and the
      nyquist contour does not encircle the point -1+j0
      ")
8 printf("System is stable")

```

Scilab code Exa 9.4 nyquist plot

```

1 s=%s;
2 H=syslin('c', (4*s+1)/(s^2*(s+1)*(2*s+1)))
3 nyquist(H)
4 show_margins(H, 'nyquist')

```

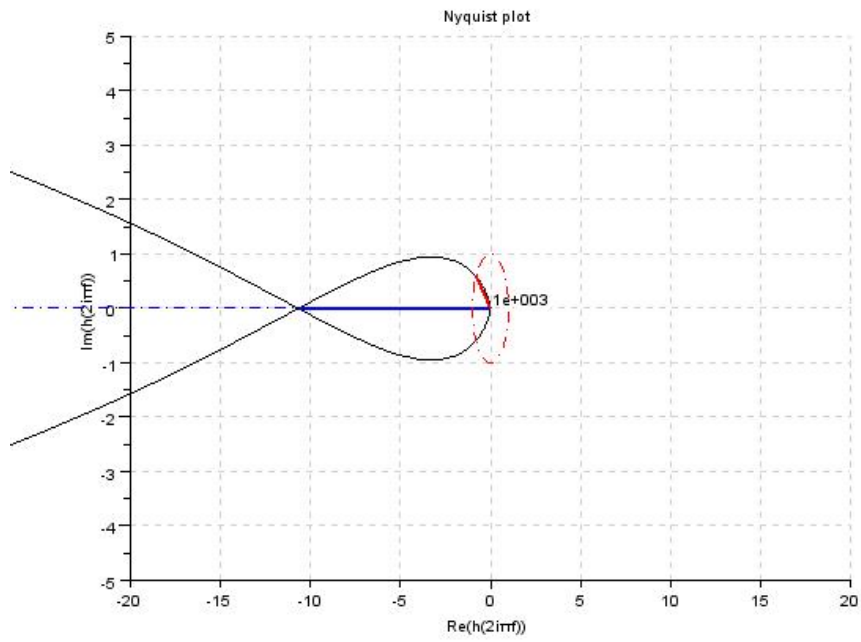


Figure 9.4: nyquist plot

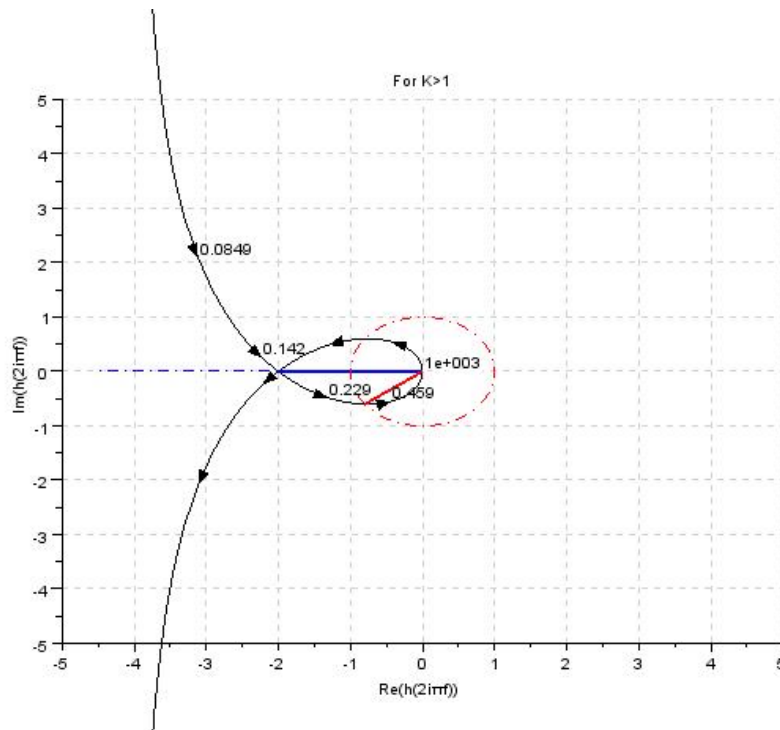


Figure 9.5: nyquist plot

```

5 mtlb_axis([-20 20 -5 5])
6 ("We see from the locus that the point  $-1+j0$  is
   encircled twice, hence  $N=2$  and  $P=0$ .")
7 printf("Therefore  $Z=2$ , hence two zeros lie in RHP")
   // N=P-Z
8 printf("System is unstable")

```

Scilab code Exa 9.5 nyquist plot

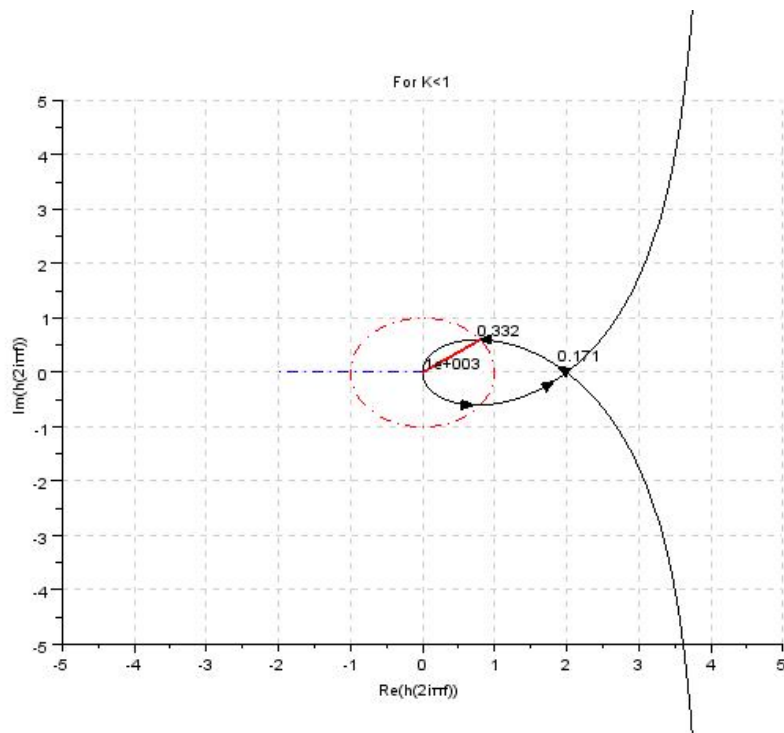


Figure 9.6: nyquist plot

```

1 s=%s;
2 syms K a
3 H=syslin('c',(K*(s+a))/(s*(s-1)))
4 // for K>1
5 nyquist(H)
6 show_margins(H,'nyquist')
7 mtlb_axis([-5 5 -5 5])
8 xtitle("For K>1")
9 printf("P=1(pole in RHP)")
10 printf("Nyquist plot encircles the the point -1+j0
        once anti-clockwise i.e.,N=1")
11 printf("Hence Z=0") // N=P-Z
12 printf("System is stable")
13 // for K<1
14 H=syslin('c',(-2*(s+1))/(s*(s-1)))
15 show_margins(H,'nyquist')
16 mtlb_axis([-5 5 -5 5])
17 xtitle("For K<1")
18 printf("The point -1+j0 lie beyond -K(the crossing
        point of the plot).So N=-1,P=1")
19 printf("Hence Z=2,zeros in RHP=2")
20 printf("System is unstable")

```

Scilab code Exa 9.6 stability using nyquist plot

```

1 s=%s;
2 syms k
3 H=syslin('c',(k*(s-2))/(s+1)^2)
4 // for K/2>-1 or K>-2
5 nyquist(H)
6 show_margins(H,'nyquist')
7 printf("P=0(poles in RHP)")
8 printf("N=-1,hence Z=1")

```

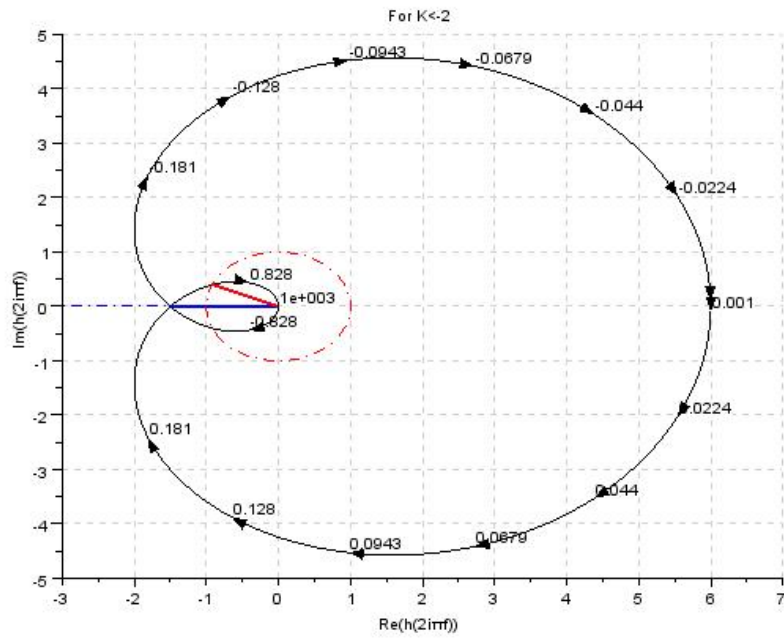


Figure 9.7: stability using nyquist plot

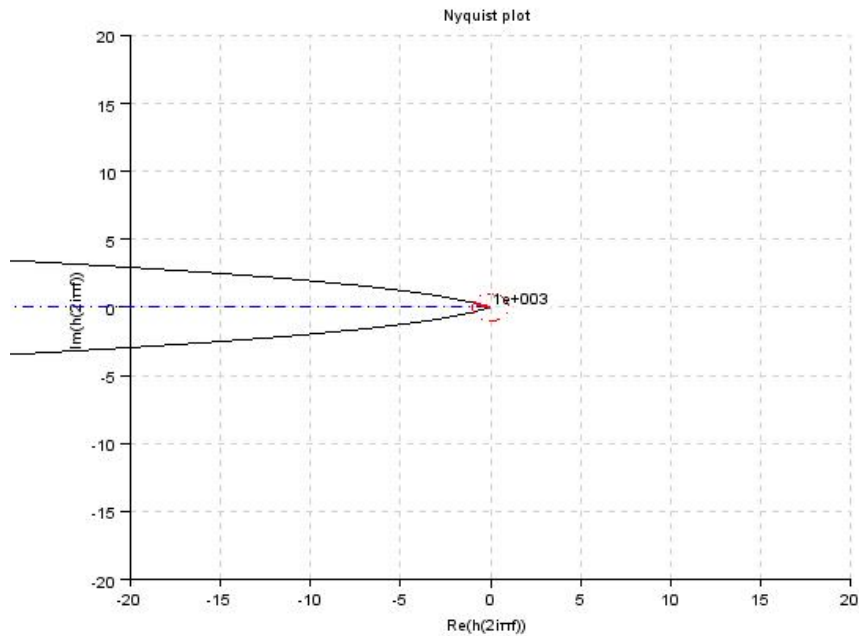


Figure 9.8: stability using nyquist plot

9 `printf("Therefore ,System is unstable")`

Scilab code Exa 9.7.a stability using nyquist plot

```

1 s=%s;
2 syms k
3 H=syslin('c',(K*(s+2))/(s^2*(s+1)))
4 nyquist(H)
5 show_margins(H,'nyquist')
6 mtlb_axis([-20 20 -20 20])
7 printf("P=0 and the locus does not encircle the

```

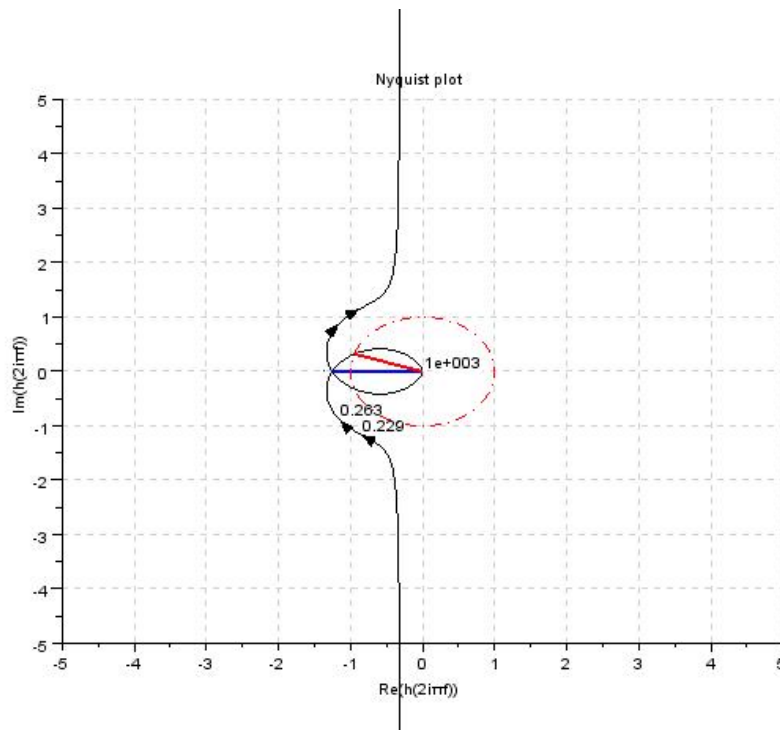


Figure 9.9: stability using nyquist plot

```

point -1+j0")
8 printf("System is stable")

```

Scilab code Exa 9.7.b stability using nyquist plot

```

1 H=syslin('c',k/(s*(s^2+s+4)))
2 nyquist(H)
3 show_margins(H,'nyquist')
4 mtlb_axis([-5 5 -5 5])
5 // nyquist plot crosses the axis of reals with
   intercept of -k/4

```

```
6 // for  $k/4 > 1$  or  $k > 4$ 
7 printf("N=-2 as it encircles the point twice in
      clockwise direction")
8 printf("P=0 and hence Z=2")
9 printf("System is unstable for  $k > 4$ ")
```

Scilab code Exa 9.8.a nyquist criterion

```
1 // from the nyquist plot
2 N=-2; // no of encirclements
3 P=0; // given
4 Z=P-N
5 printf("Since Z=2 therefore two roots of the
      characteristic equation lies in the right half of
      s-plane, hence the system is unstable")
```

Scilab code Exa 9.8.b nyquist criterion

```
1 // from the nyquist plot
2 N=0; // one clockwise and one anticlockwise
      encirclement
3 P=0; // given
4 Z=N-P
5 printf("Since Z=0 no root of the characterisic
      equation lies in the right half hence the system
      is stable")
```

Scilab code Exa 9.10 gm and pm using nyquist plot

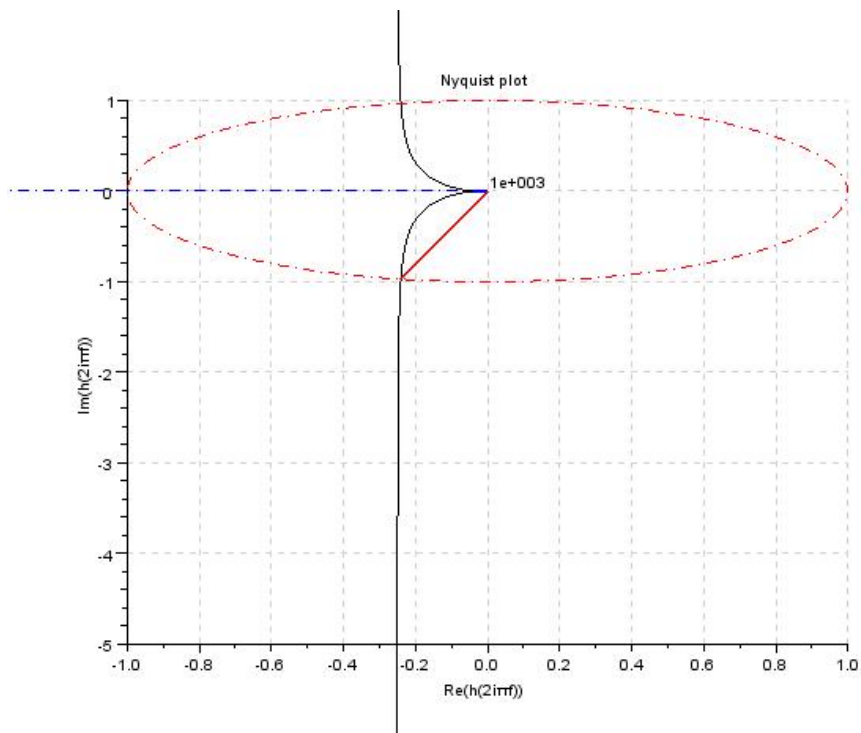


Figure 9.10: gm and pm using nyquist plot

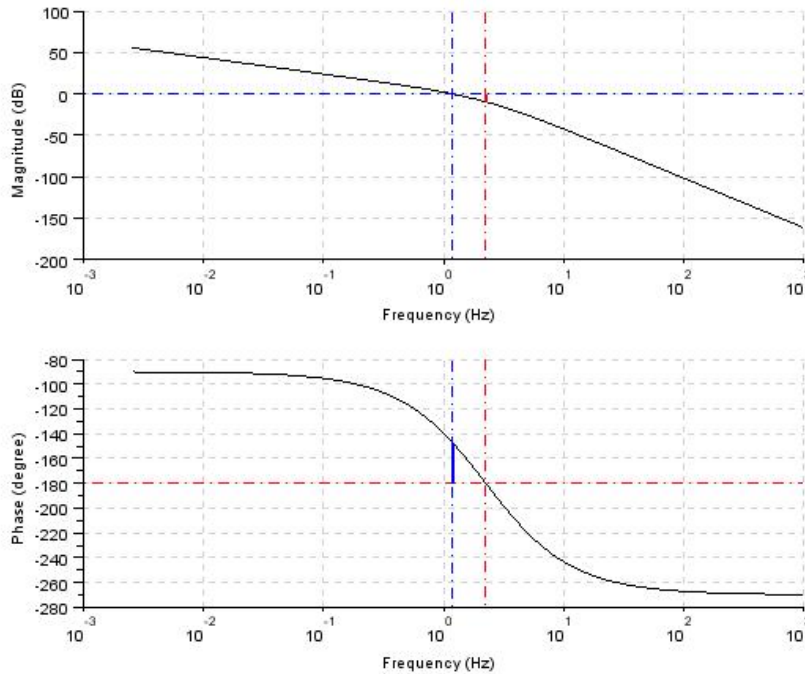


Figure 9.11: bode plot

```

1 s=%s;
2 syms K
3 H=syslin('c',K/(s*(0.2*s+1)*(0.05*s+1)))
4 nyquist(H)
5 show_margins(H,'nyquist')
6 mtlb_axis([-1 1 -5 1])
7 gm=g_margin(H) // gain margin
8 pm=p_margin(H) // phase margin

```

Scilab code Exa 9.11 bode plot

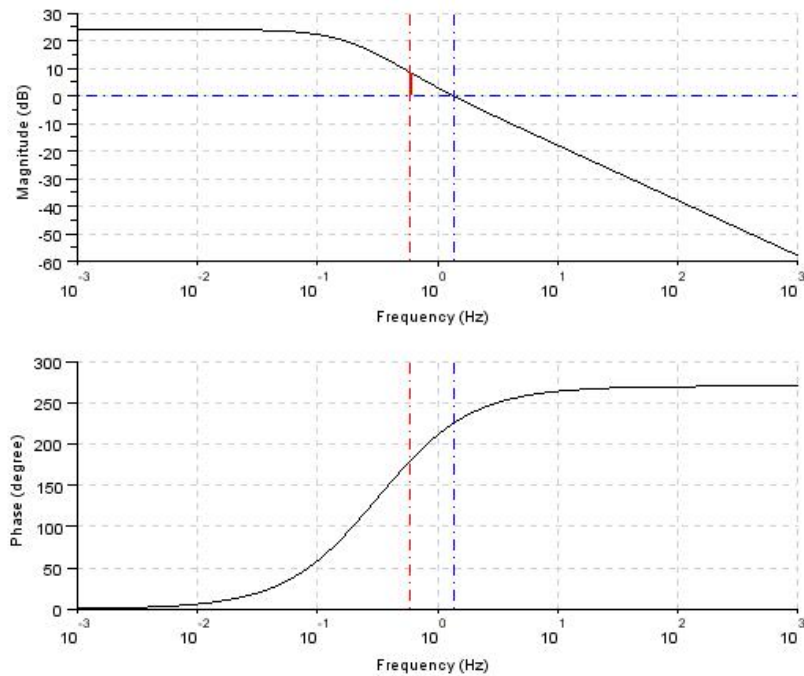


Figure 9.12: bode plot

```

1 s=%s;
2 H=syslin('c',10/(s*(0.1*s+1)*(0.05*s+1)))
3 fmin=0.1;
4 fmax=100;
5 bode(H,fmin,fmax)
6 show_margins(H)
7 gm=g_margin(H)
8 pm=p_margin(H)

```

Scilab code Exa 9.13.a bode plot

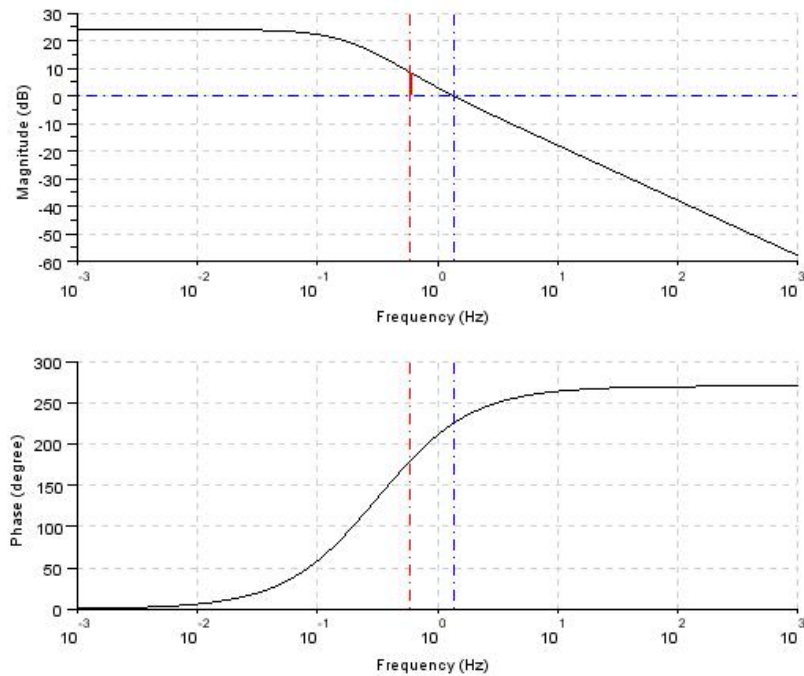


Figure 9.13: bode plot

```

1 s=%s;
2 H=syslin('c', (8*(s+4))/((s-1)*(s-2)))
3 fmin=0.1;
4 fmax=100;
5 bode(H, fmin, fmax)
6 show_margins(H)
7 gm=g_margin(H)
8 pm=p_margin(H)

```

Scilab code Exa 9.13.b bode plot

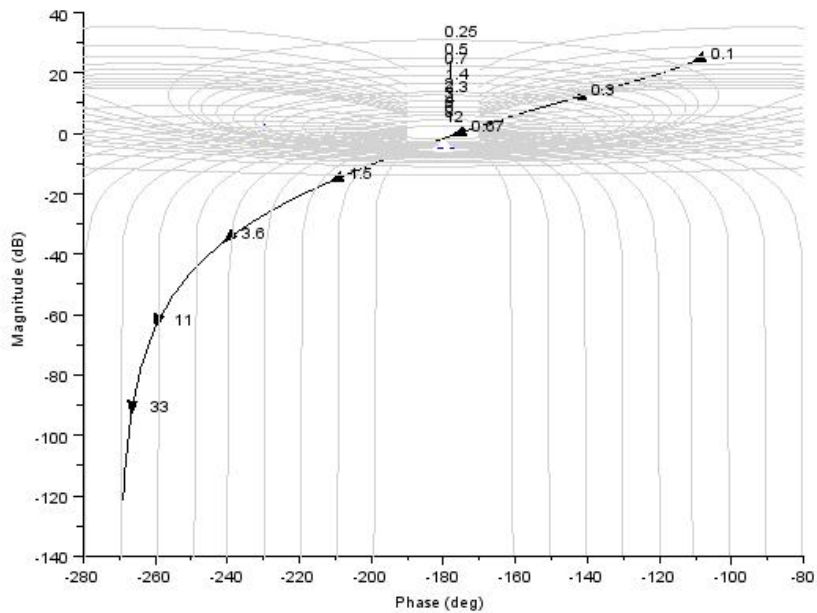


Figure 9.14: m circles

```

1 syms K
2 H=syslin('c',(K*(s+4))/((s-1)*(s-2)))
3 fmin=0.1;
4 fmax=100;
5 bode(H,fmin,fmax)
6 show_margins(H)
7 // for phase margin =30
8 printf("From bode plot it can be seen that gain
        should be reduced by 4db")

```

Scilab code Exa 9.14 m circles

```

1 s=%s;
2 H=syslin('c',10/(s*((0.1*s)+1)*((0.5*s)+1)))
3 fmin=0.1;
4 fmax=100;
5 clf()
6 black(H,0.1,100)
7 chart(list(1,0))
8 gm=g_margin(H)
9 pm=p_margin(H)
10 printf("For gain margin of 20 db plot is shifted
        downwards by 8 db and a phase margin of 24
        degrees is obtained if curve is shifted upwards
        by 3.5 db")

```

Scilab code Exa 9.15 m circles

```

1 s=%s;
2 H=syslin('c',10/(s*((0.1*s)+1)*((0.05*s)+1)))
3 fmin=0.1;
4 fmax=100;
5 clf()
6 black(H,0.1,100)
7 chart(list(1,0))
8 gm=g_margin(H)
9 pm=p_margin(H)
10 printf("For gain margin of 20 db plot is shifted
        downwards by 8 db and a phase margin of 24
        degrees is obtained if curve is shifted upwards
        by 3.5 db")

```

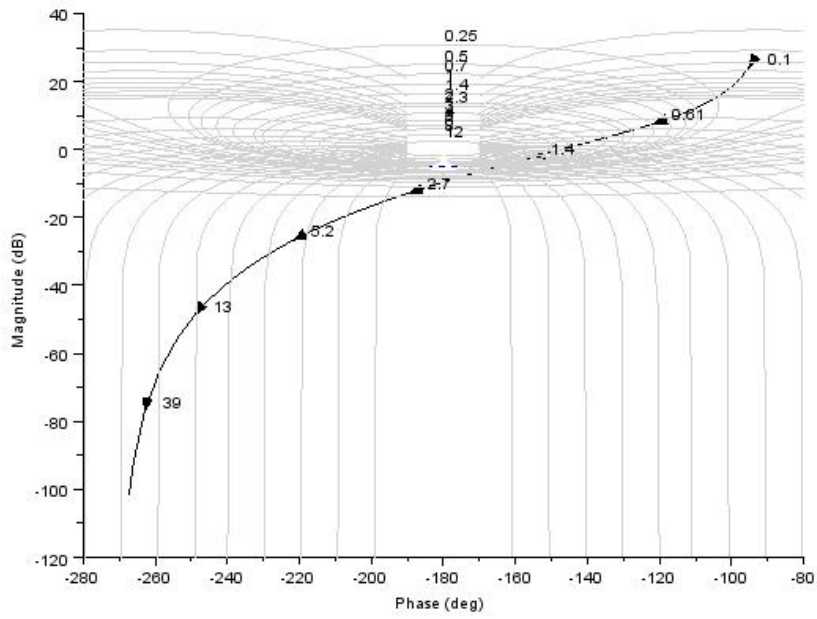


Figure 9.15: m circles

Chapter 10

Introduction to Design

Scilab code Exa 10.6 lead compensation

```
1 s=%s;
2 syms Kv;
3 g=(Kv/(s*(s+1)));
4 // given Kv=12
5 Kv=12;
6 g=(12/(s*(s+1)));
7 G=syslin('c',g)
8 fmin=0.01;
9 fmax=100;
10 bode(G,fmin,fmax)
11 show_margins(G)
12 xtitle("uncompensated system")
13 [gm,freqGM]=g_margin(G)
14 [pm,freqPM]=p_margin(G)
15 disp(gm,"gain margin=")
16 disp((freqGM*2*%pi),"gain margin freq=");
17 disp(pm,"phase margin=")
```

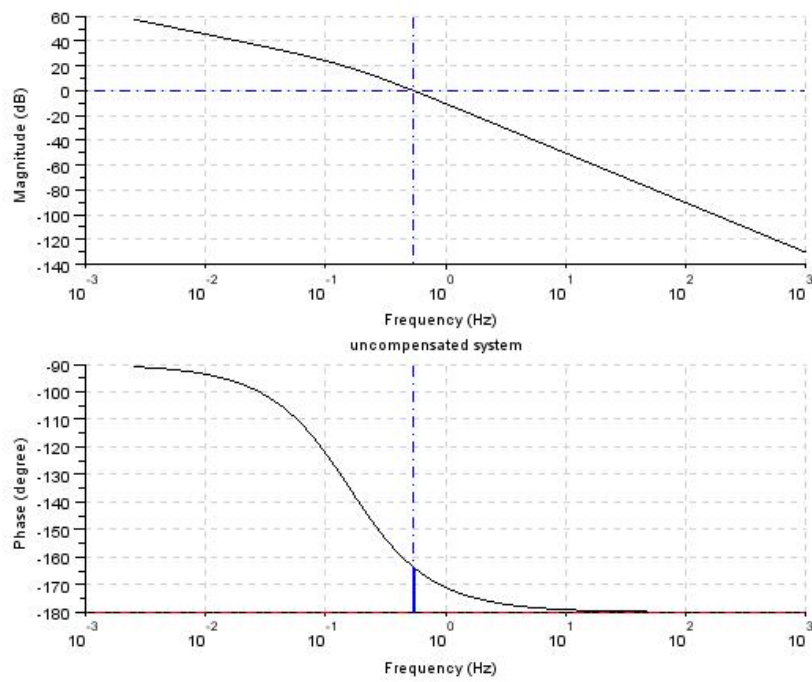


Figure 10.1: lead compensation

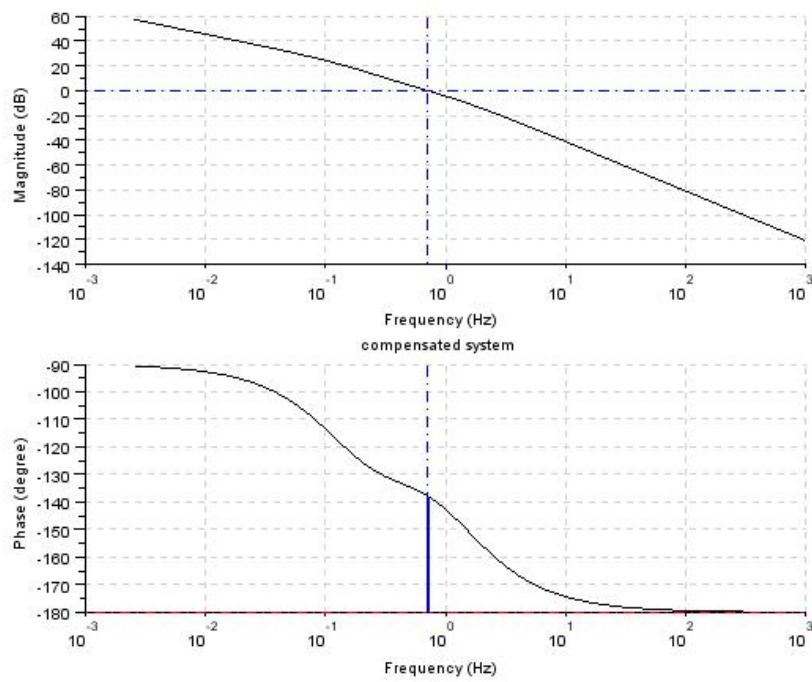


Figure 10.2: lead compensation


```

18 disp((freqPM*2*%pi),"phase margin freq=");
19 printf("since P.M is less than desired value so we
    need phase lead network")
20 disp("selecting zero of lead compensting network at
    w=2.65rad/sec and pole at w=7.8rad/sec and
    applying gain to account attenuation factor.")
21 gc=(1+0.377*s)/(1+0.128*s)
22 Gc=syslin('c',gc)
23 disp(Gc,"transfer function of lead compensator=");
24 G1=G*Gc
25 disp(G1,"overall transfer function=");
26 fmin=0.01;
27 fmax=100;
28 bode(G1,fmin,fmax);
29 show_margins(G1)
30 xtitle("compensated system")
31 [gm,freqGM]=g_margin(G1);
32 [pm,freqPM]=p_margin(G1);
33 disp(pm,"phase margin of compensated system=")
34 disp((freqPM*2*%pi),"gain cross over frequency=")

```

Scilab code Exa 10.7 lead compensation

```

1 s=%s;
2 syms Ka;
3 g=(Ka/(s^2*(1+0.2*s)));
4 // given Ka=10
5 Ka=10;
6 g=(10/(s^2*(1+0.2*s)));
7 G=syslin('c',g)
8 fmin=0.01;

```

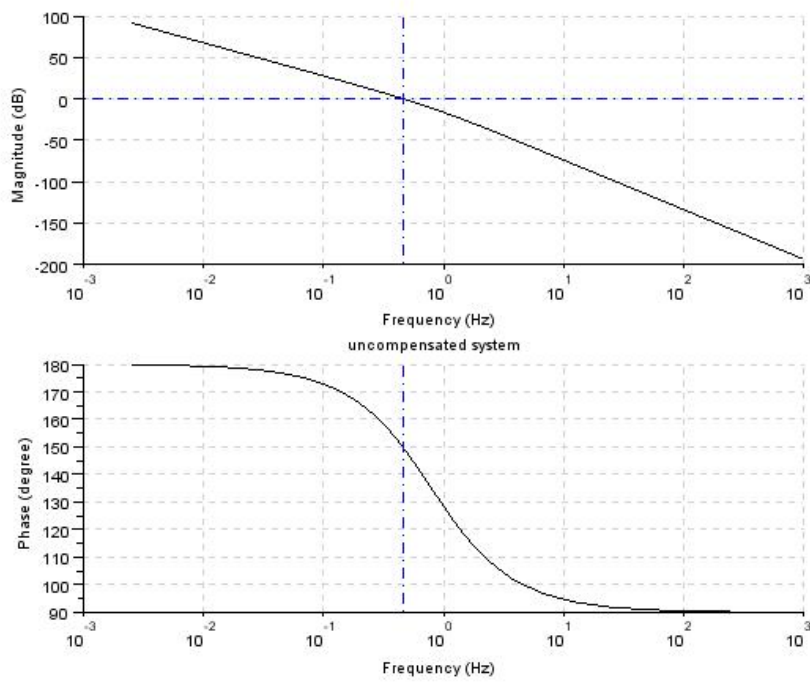


Figure 10.3: lead compensation

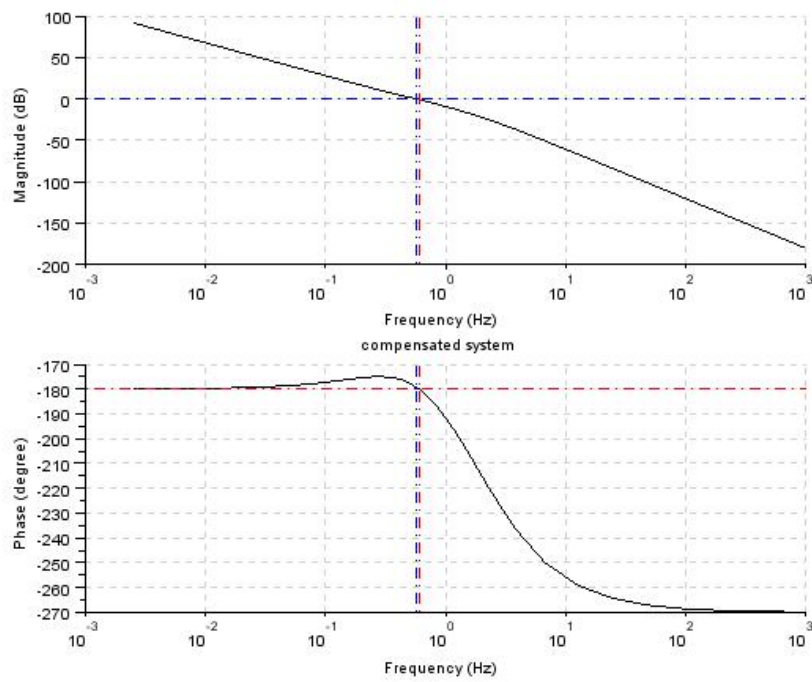


Figure 10.4: lead compensation

```

 9 fmax=100;
10 bode(G,fmin,fmax)
11 show_margins(G)
12 xtitle("uncompensated system")
13 [gm,freqGM]=g_margin(G)
14 [pm,freqPM]=p_margin(G)
15 disp(gm,"gain_margin=")
16 disp((freqGM*2*%pi),"gain margin freq=");
17 disp(pm,"phase margin=")
18 disp((freqPM*2*%pi),"phase margin freq=");
19 disp("since P.M is negative so system is unstable")
20 disp("selecting zero of lead compensating network at
      w=2.8 rad/sec and pole at w=14 rad/sec and
      applying gain to account attenuation factor.")
21 gc=(1+0.358*s)/(1+0.077*s)
22 Gc=syslin('c',gc)
23 disp(Gc,"transfer function of lead compensator=");
24 G1=G*Gc
25 disp(G1,"overall transfer function=");
26 fmin=0.01;
27 fmax=100;
28 bode(G1,fmin,fmax);
29 show_margins(G1)
30 xtitle("compensated system")
31 [gm,freqGM]=g_margin(G1);
32 [pm,freqPM]=p_margin(G1);
33 disp(pm,"phase margin of compensated system=")
34 disp((freqPM*2*%pi),"gain cross over frequency=")

```

Scilab code Exa 10.8 lag compnsation

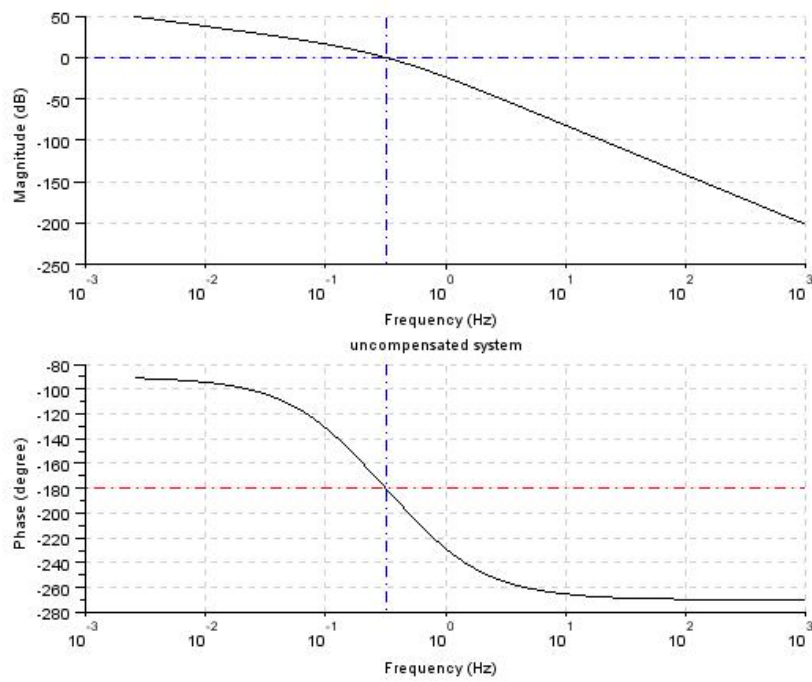


Figure 10.5: lag compensation

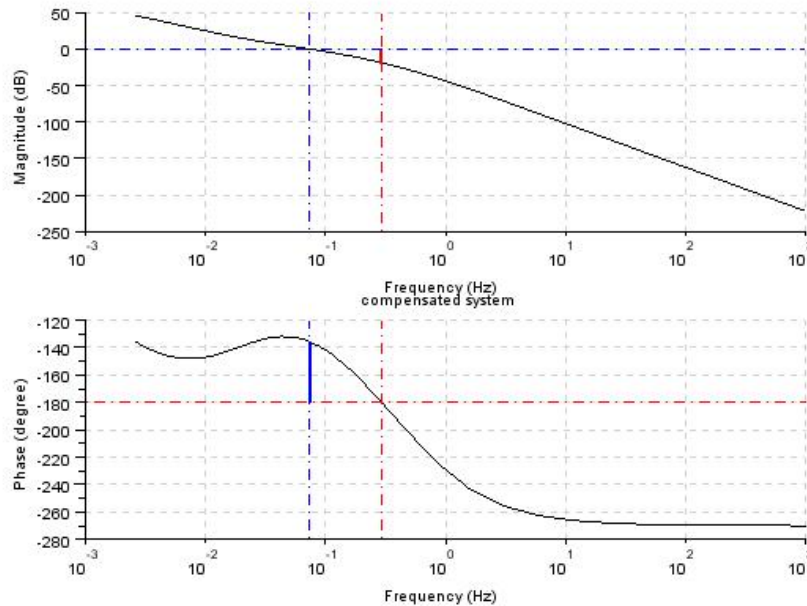


Figure 10.6: lag compensation

```

1 s=%s;
2 syms K;
3 g=(K/(s*(s+1)*(s+4)));
4 g=((K/4)/(s*(s+1)*(0.25*s+1)))
5 // given Kv=5 : velocity error constant
6 K=20;
7 g=(5/(s*(s+1)*(0.25*s+1)))
8 G=syslin('c',g)
9 fmin=0.01;
10 fmax=100;
11 bode(G,fmin,fmax)
12 show_margins(G)
13 xtitle("uncompensated system")
14 [gm,freqGM]=g_margin(G)
15 [pm,freqPM]=p_margin(G)
16 disp(gm,"gain_margin=")
17 disp((freqGM*2*%pi),"gain margin freq=");
18 disp(pm,"phase_margin=")

```

```

19 disp((freqPM*2*%pi),"phase margin freq=");
20 disp("since P.M is negative so system is unstable")
21 disp("selecting zero of phase lag network at w=0.013
      rad/sec and pole at w=0.13 rad/sec and applying
      gain to account attenuation factor")
22 gc=((s+0.13)/(10*(s+0.013)))
23 Gc=syslin('c',gc)
24 disp(Gc,"transfer function of lag compensator=");
25 G1=G*Gc
26 disp(G1,"overall transfer function=");
27 fmin=0.01;
28 fmax=100;
29 bode(G1,fmin,fmax);
30 show_margins(G1)
31 xtitle("compensated system")
32 [gm,freqGM]=g_margin(G1);
33 [pm,freqPM]=p_margin(G1);
34 disp(pm,"phase margin of compensated system=")
35 disp((freqPM*2*%pi),"gain cross over frequency=")

```

Scilab code Exa 10.9 lag and lead compensation

```

1 s=%s;
2 syms K;
3 g=(K/(s*(0.1*s+1)*(0.2*s+1)));
4 // given Kv=30 :velocity error constnt
5 K=30;
6 g=(30/(s*(0.1*s+1)*(0.2*s+1)))
7 G=syslin('c',g)
8 fmin=0.01;
9 fmax=100;

```

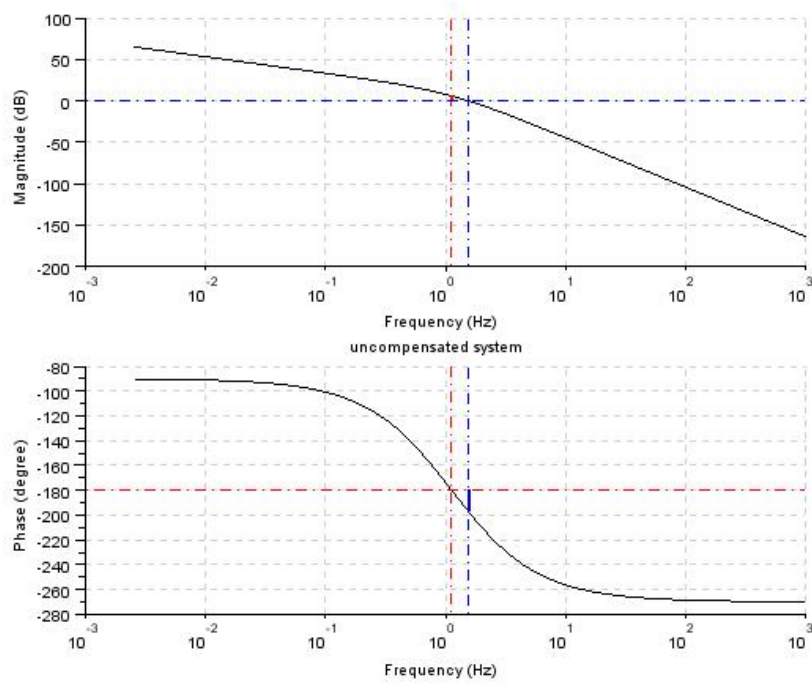


Figure 10.7: lag and lead compensation

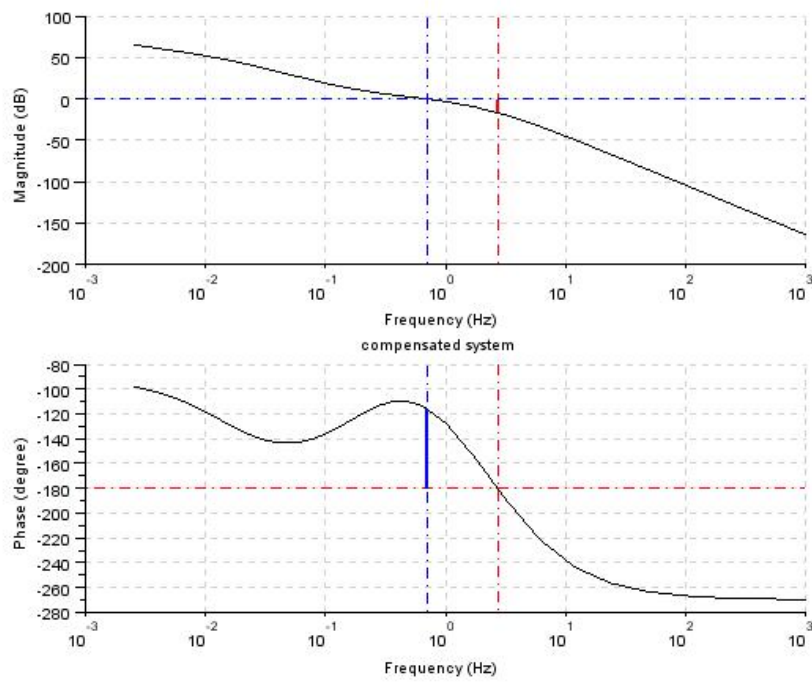


Figure 10.8: lag and lead compensation

```

10 bode(G,fmin,fmax)
11 show_margins(G)
12 xtitle("uncompensated system")
13 [gm,freqGM]=g_margin(G)
14 [pm,freqPM]=p_margin(G)
15 disp(gm,"gain_margin=")
16 disp((freqGM*2*%pi),"gain margin freq=");
17 disp(pm,"phase margin=")
18 disp((freqPM*2*%pi),"phase margin freq=");
19 disp("since P.M is negative so system is unstable")
20 disp("If lead compenstion is used bandwidth will
      increase resulting in undesirable system
      sensitive to noise. If lag compensation is used
      bandwidth decreases so as to fall short of
      specified value of 12 rad/sec resulting in
      sluggish system")
21 disp("/n hence we use a lag-lead compensator")
22 // lag compensator
23 disp("selecting zero of phase lag network w=1 rad/
      sec and pole at w=0.1 rad/sec and applying gain
      to account attenuation factor")
24 gc1=((s+1)/(10*s+1));
25 Gc1=syslin('c',gc1)
26 disp(Gc1,"transfer function of lag compensator")
27 // lead compensator
28 disp("selecting zero of lead compensator at w=0.425
      rad/sec and pole at w=0.0425rad/sec ")
29 gc2=((0.425*s+1)/(0.0425*s+1));
30 Gc2=syslin('c',gc2)
31 disp(Gc2,"transfer function of lead compensator")
32 Gc=Gc1*Gc2 // transfer function of lag and lead
      sections
33 disp(Gc,"transfer function of lag and lead sections"
      )
34 G1=G*Gc
35 disp(G1,"overall transfer function=");
36 fmin=0.01;
37 fmax=100;

```

```
38 bode(G1,fmin,fmax);
39 show_margins(G1)
40 xtitle("compensated system")
41 [gm,freqGM]=g_margin(G1);
42 [pm,freqPM]=p_margin(G1);
43 disp(pm,"phase margin of compensated system=")
44 disp((freqPM*2*%pi),"gain cross over frequency=")
```

Chapter 12

State Variable Analysis and Design

Scilab code Exa 12.3 state matrix

```
1 s=%s;  
2 H=syslin('c',(2*s^2+6*s+7)/((s+1)^2*(s+2)))  
3 SS=tf2ss(H)  
4 [Ac,Bc,U,ind]=canon(SS(2),SS(3))
```

Scilab code Exa 12.4 modal matrix

```
1 syms m11 m12 m13 m21 m22 m23 m31 m32 m33 ^  
2 s=%s;  
3 poly(0,"l");  
4 A=[0 1 0;3 0 2;-12 -7 -6]  
5 [r c]=size(A)  
6 I=eye(r,c);  
7 p=1*I-A;  
8 q=det(p); // determinant of li-p  
9 // roots of q are
```

```

10 l1=-1;
11 l2=-2;
12 l3=-3;
13 x1=[m11;m21;m31];
14 q1=(l1*I-A)*1
15 // on solving we find m11=1 m21=-1 31=-1
16 m11=1;m21=-1;m31=-1;
17 x2=[m12;m22;m32];
18 q2=(l2*I-A)*1
19 // on solving we find m12=2 m22=-4 m32=1
20 m12=2;m22=-4;m32=1;
21 x3=[m13;m23;m33];
22 q3=(l3*I-A)*1
23 // on solving we get m13=1 m23=-3 m33=3
24 m13=1;m23=-3;m33=3;
25 // modal matrix is
26 M=[m11 m12 m13;m21 m22 m23;m31 m32 m33]

```

Scilab code Exa 12.5 obtain time response

```

1 syms t m
2 s=%s;
3 A=[1 0;1 1];
4 B=[1;1];
5 x=[1;0];
6 [r c]=size(A)
7 p=s*eye(r,c)-A // s*I-A
8 q=inv(p)
9 for i=1:r
10 for j=1:c
11 // inverse laplace of each element of Matrix q
12 q(i,j)=ilaplace(q(i,j),s,t);
13 end
14 end
15 disp(q,"phi(t)=") // State Transition Matrix

```

```

16 t=t-m;
17 q=eval(q)
18 // Integrate q w.r.t m
19 r=integrate(q*B,m)
20 m=0 // Upper limit is t
21 g=eval(r) // Putting upper limit in q
22 m=t // Lower limit is 0
23 h=eval(r) // Putting lower limit in q
24 y=(h-g);
25 disp(y,"y=")
26 printf("x(t)=phi(t)*x(0)+integrate(phi(t-m*B) w.r.t
      m from 0 to t)")
27 y1=(q*x)+y;
28 disp(y1,"x(t)=")

```

Scilab code Exa 12.6 resolvent matrix

```

1 syms t
2 s=%s;
3 A=[1 0;1 1];
4 [r c]=size(A)
5 p=s*eye(r,c)-A
6 // resolvent matrix
7 q=inv(p)
8 disp(q,"phi(s)")
9 for i=1:r
10 for j=1:c
11 q(i,j)=ilaplace(q(i,j),s,t)
12 end
13 end
14 disp(q,"phi(t)") // state transition matrix

```

Scilab code Exa 12.7 state transition matrix and state response

```

1 syms t m
2 s=%s;
3 A=[0 1;-2 -3];
4 B=[0;2];
5 x=[0;1];
6 [r c]=size(A)
7 p=s*eye(r,c)-A
8 q=inv(p)
9 for i=1:r
10 for j=1:c
11 q(i,j)=ilaplace(q(i,j),s,t)
12 end
13 end
14 disp(q,"phi(t)=") // state transition matrix
15 t=t-m;
16 q=eval(q)
17 // Integrate q w.r.t m
18 r=integrate(q*B,m)
19 m=0 // Upper limit is t
20 g=eval(r) // Putting upper limit in q
21 m=t // Lower limit is 0
22 h=eval(r) // Putting lower limit in q
23 y=(h-g);
24 disp(y,"y=")
25 printf("x(t)=phi(t)*x(0)+integrate(phi(t-m*B) w.r.t
        m from 0 to t)")
26 y1=(q*x)+y;
27 disp(y1,"x(t)=")

```

Scilab code Exa 12.12 check for controllability

```

1 A=[0 1 0;0 0 1;-6 -11 -6];
2 B=[0;0;1];
3 P=cont_mat(A,B);
4 disp(P,"Controllability Matrix=")

```

```

5 d=det(P)
6 if d==0
7     printf("matrix is singular , so system is
            uncontrollable");
8 else
9     printf("system is controllable");
10 end;

```

Scilab code Exa 12.13 check for controllability

```

1 A=[0 1;-1 -2];
2 B=[1;-1];
3 P=cont_mat(A,B);
4 disp(P,"Controllability Matrix=")
5 d=determ(P)
6 if d==0
7     printf("matrix is singular , so system is
            uncontrollable");
8 else
9     printf("system is controllable");
10 end;

```

Scilab code Exa 12.14 check for observability

```

1 A=[0 1 0;0 0 1;0 -2 -3];
2 B=[0;0;1];
3 C=[3 4 1];
4 P=obsv_mat(A,C);
5 disp(P,"Observability Matrix=");
6 d=det(P)
7 if d==0
8     printf("matrix is singular , so system is
            unobservable");

```



```

9 else
10     printf("system is observable");
11 end;

```

Scilab code Exa 12.17 design state observer

```

1 syms g1 g2 g3
2 poly(0,"l");
3 A=[1 2 0;3 -1 1;0 2 0];
4 C=[0;0;1];
5 G=[g1;g2;g3];
6 p=A-G*C;
7 [r c]=size(A);
8 I=eye(r,c);
9 q=1I-p; // 1I-(A-G*C) where I is identity matrix
10 r=det(q) // determinant of 1I-(A-G*C)
11 // on equating r=0 we get
12 // characteristic equation
13 l^3+g3*(l)^2+(2*g2-9)l+2+6*g1-2*g2-7*g3=0;
14 printf("desired characteristic equation given is\n")
15 l^3+10*(l)^2+34*l+40=0;
16 // on comparing the coefficients of the two
    equations
17 // we get g1=25.2 g2=21.5 g3=10
18 g1=25.2;
19 g2=21.5;
20 g3=10;
21 disp(G)

```
