Scilab Manual for
Control System Engineering
by Prof Priyen S. Patel
Electrical Engineering
Swarrnim Institute Of Technology[1]

Solutions provided by
Prof Priyen S. Patel
Electrical Engineering
Swarrnim Institute Of Technology

May 31, 2026

[1]Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Scilab Manual and Scilab codes written in it can be downloaded from the "Migrated Labs" section at the website http://scilab.in

Contents

List of Experiments

List of Figures

Experiment: 1

To study open loop control system.

This code can be downloaded from the website wwww.scilab.in

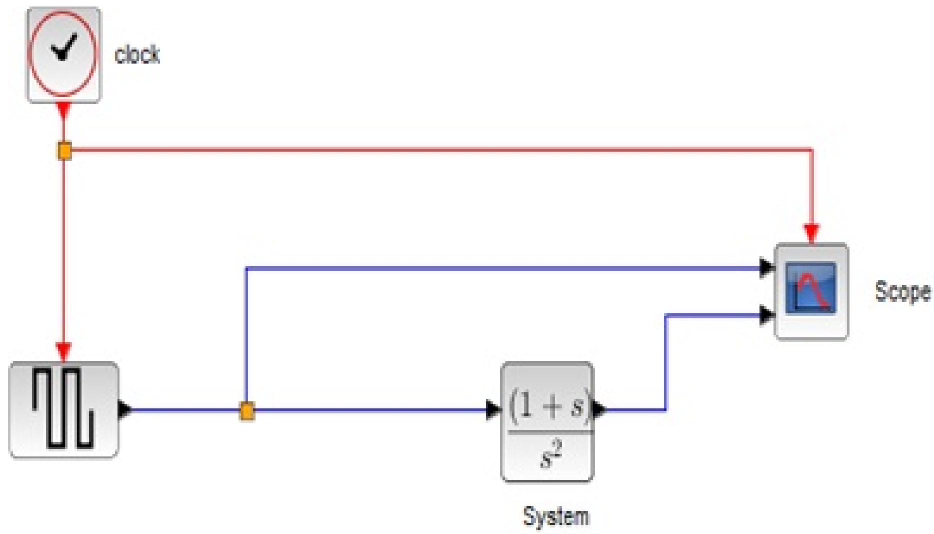

Figure 1.1: Open Loop system

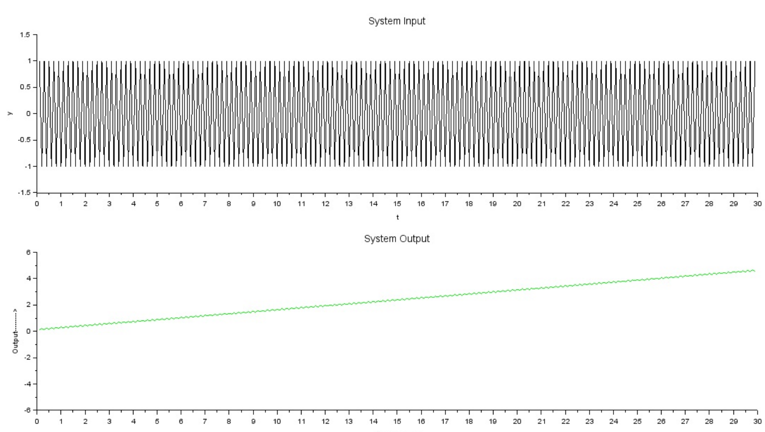

Figure 1.2: Open Loop system

Experiment: 2

To study closed loop control system.

This code can be downloaded from the website wwww.scilab.in

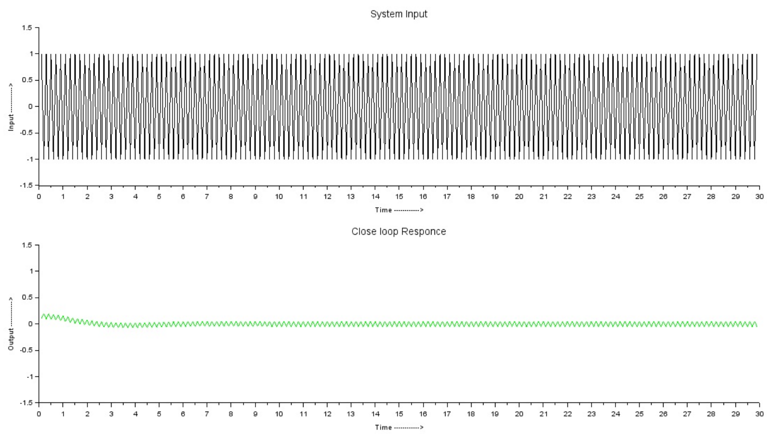

Figure 2.1: Close Loop system

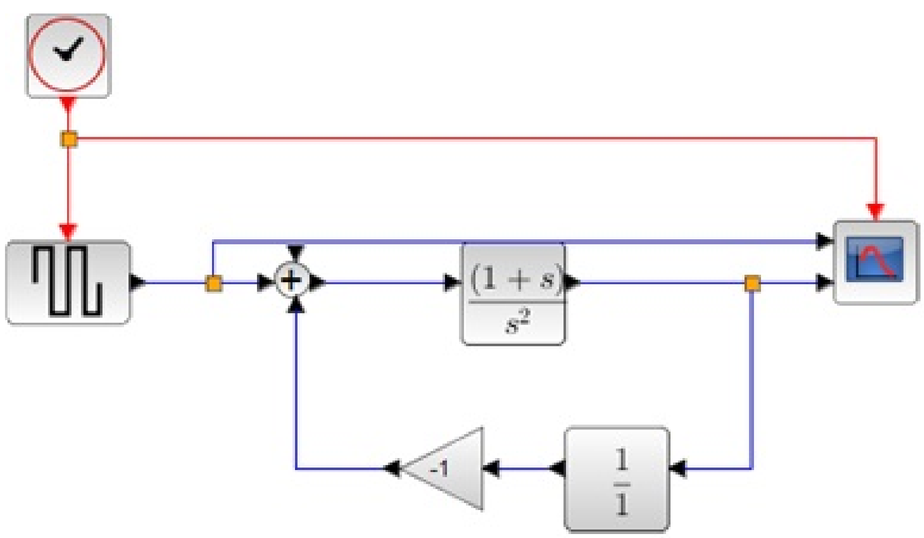

Figure 2.2: Close Loop system

Experiment: 3

Develop a program to study transfer function from poles and zeros.

Scilab code Solution 3.01 Obtain Transfer Function

```
// OS : Windows 7
// Scilab : 6.0.1

// To find Transfer function of given system

clc
close
s=%s;
// From Pole & Zero
 // For First order transfer function
z1=input('enter the value of z1 = ')//z1=0
p1=input('enter the value of p1 = ')//p1=-5
tf1=(s-z1)/((s-p1))
disp(tf1,"Transfer function of first order system  =
    ")


```

```
 // For Second order transfer function
z1=input('enter the value of z1 = ')//z1=-2
p1=input('enter the value of p1 = ')//p1=-5
p2=input('enter the value of p2 = ')//p2=1
tf2=(s-z1)/((s-p1)*(s-p2))
disp(tf2,"Transfer function of Second order system = ")

// From Numerator & Denominator

numm = input('enter the Co-efficient of numerator:'); //[1]
denn = input('enter the Co-efficient of denominator:');//[-7 3 3 1]

A = poly([numm], 's','c')
B = poly([denn], 's','c')

tff = syslin('c',A,B)
disp(tff, 'Transfer Function is:')
```

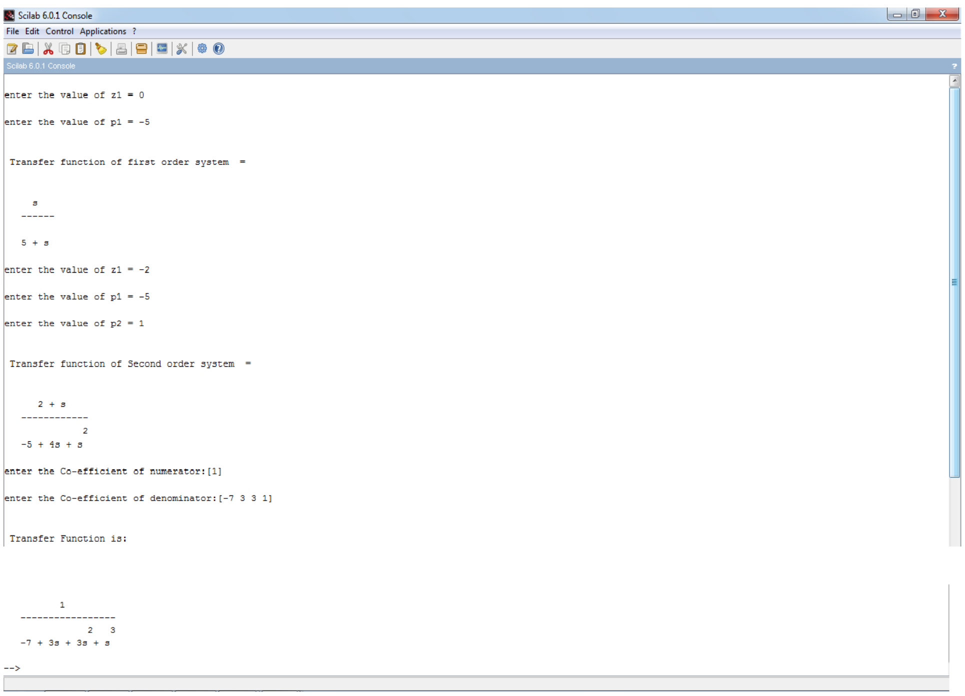

Figure 3.1: Obtain Transfer Function

Experiment: 4

Develop a program to study transient response for given system.

Scilab code Solution 4.01 Transient Response

```
//
   ________________________________________________________________________________
// To study Transient Response of given system.
//
   ________________________________________________________________________________


// OS : Windows 7
// Scilab : 6.0.1

clc
```

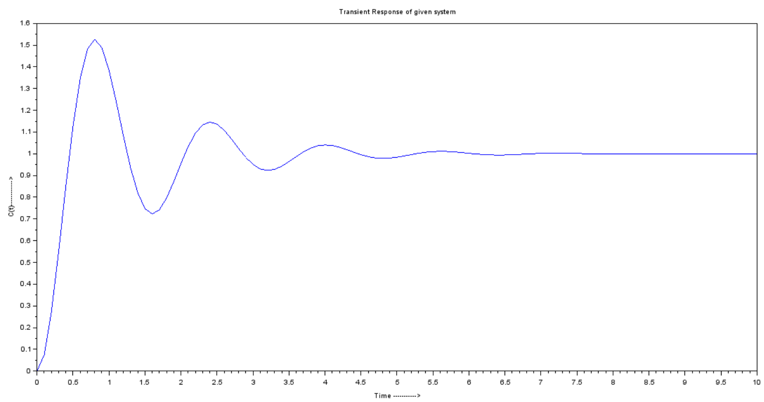

Figure 4.1: Transient Response

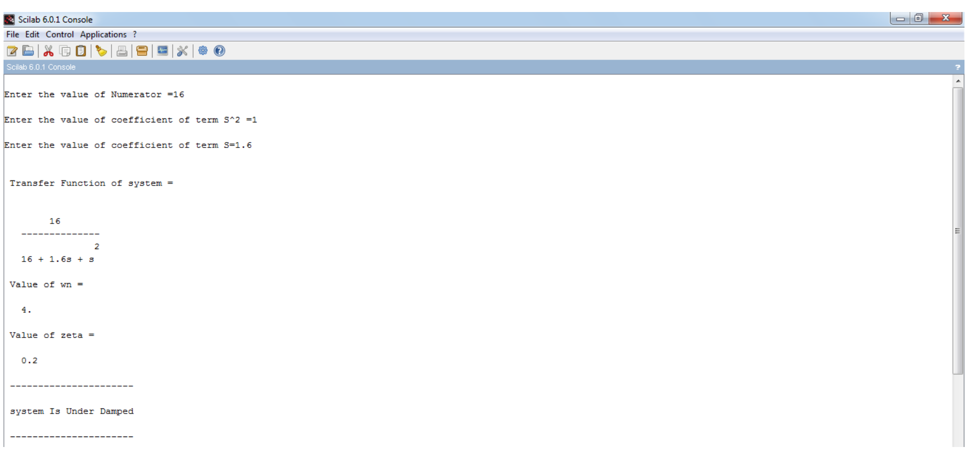

Figure 4.2: Transient Response

```
close
s=%s
// To form Numerator in form of wn^2
num = input('Enter the value of Numerator =')//num=16
wn=sqrt(num)
// To form denominator in form of S^2+(2*zeta*wn)*s+wn^2
a= input('Enter the value of coefficient of term S^2 =')//a=1
b= input('Enter the value of coefficient of term S=')//b=1.6
den= [a*s^2+b*s+wn^2]
TF = syslin('c',num,den)//transfer function
disp(TF,"Transfer Function of system = ")
t=0:0.1:10;
y1 = csim('step', t, TF);//time response
title('Transient Response of given system');
xlabel('Time ------------->');
ylabel('C(t)------------->');
plot(t, y1)
disp(wn,"Value of wn =")
z=b/(2*wn)
disp(z,"Value of zeta =")
if z<1 then
    disp("-------------------------------")
    disp("system Is Under Damped")
    disp("-------------------------------")
elseif z==1 then
    disp("-------------------------------")
    disp("system Is Critically Damped")
    disp("-------------------------------")
else
    disp("-------------------------------")
    disp("system Is Critically Damped")
    disp("-------------------------------")
    end
wd=wn*(sqrt(1-z^2));
```

Figure 4.3: Transient Response

```
disp(wd,"Damped frequency =")
theta=atan((sqrt(1-z^2)/z));
disp(theta, "Angle in radians=")
Tr = ((%pi-theta)/wd)
disp(Tr,"Rise TIme =")
Tp=%pi/wd
disp(Tp,"Peak TIme =")
Ts=4/(z*wn)
disp(Ts,"Settling Time =")
Mp=100*%e^((-z*%pi)/(sqrt(1-z^2)))
disp(Mp,"Peak overshoot % =")
```

Experiment: 5

Develop a program to study the controllability and observability of a given system.

Scilab code Solution 5.01 Controllability and Observability

```

//____________________________________________________________________________
// To Study the controllability and observability of
    a given system.
//____________________________________________________________________________

// OS : Windows 7
// Scilab : 6.0.1
clc;
clear all;

// State space representation

```

```
A=input('Enter the value of Matrix A =');//[0 0 0; 1
    0 -3; 0 1 -4];
B=input('Enter the value of Matrix B =');//[40; 10;
   0];
C=input('Enter the value of Matrix C =');//[0 0 1];
D=input('Enter the value of Matrix D =');// [0];
sys=syslin('c',A,B,C,D)

// For Controllability
n=cont_mat(sys)
mprintf('Controllability matrix is')
disp(n)

if rank(n)==3 then
    disp('System is controllable as rank is 3')
else
    disp('System is uncontrollable')
end

// For Observability
m=obsv_mat(sys)
mprintf('Observability matrix is')
disp(m)

if rank(m)==3 then
    disp('System is observable as rank is 3')
else
    disp('System is unobservable')
end
```

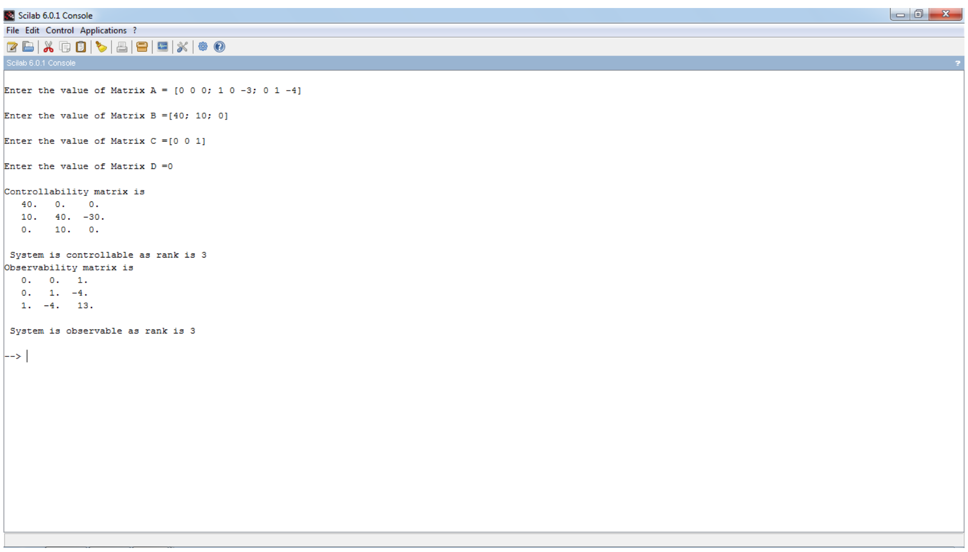

Figure 5.1: Controllability and Observability

Experiment: 6

Develop a program to study routh stability criterion.

Scilab code Solution 6.01 Routh Stability Criterion

```

//
    ------------------------------------------------------------------------
// To Study the Routh-Hurwitz Criterion
//
    ------------------------------------------------------------------------

// OS : Windows 7
// Scilab : 6.0.1

clc;
clear all;

D=input('Input coefficients of characteristic
   equation,i.e:[a0 an+1 an+2_ _ _ _ _an]= ');
```

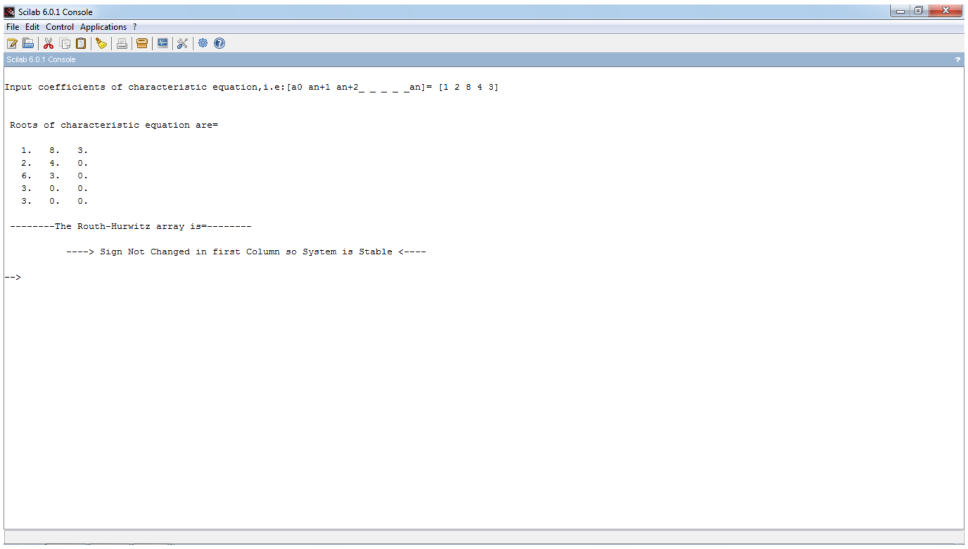

Figure 6.1: Routh Stability Criterion

```
//Case- 1 enter cvalue of D = [1 2 8 4 3]
//Case- 2 enter cvalue of D = [1 1 3 6 6]
l=length (D);

disp('Roots of characteristic equation are=')
roots(D)
if modulo(l,2)= =0
    m=zeros(l,l/2);
    [cols,rows]=size(m);
    for i=1:rows
        m(1,i)=det(1,(2*i)-1);
        m(2,i)=det(1,(2*i));
    end
else
    m=zeros(l,(l+1)/2);
    [cols,rows]=size(m);
    for i=1:rows
        m(1,i)=D(1,(2*i)-1);
    end
    for i=1:((l-1)/2)
```

```
        m(2,i)=D(1,(2*i));
    end
end

for j=3:cols

    if m(j-1,1)==0
        m(j-1,1)=0.001;
    end

    for i=1:rows-1
        m(j,i)=(-1/m(j-1,1))*det([m(j-2,1) m(j-2,i+1);m(j-1,1) m(j-1,i+1)]);
    end
end

disp('--------The Routh-Hurwitz array is=--------',m)
//--------------------End of Bulding array--------------------------------

//Checking for sign change
Temp=sign(m);a=0;
for j=1:cols
    a=a+Temp(j,1);
end
if a==cols
    disp('          ----> Sign Not Changed in first Column so System is Stable <----')
else
    disp('          ----> Sign  Changed in first Column so System is Unstable <----')
end
```

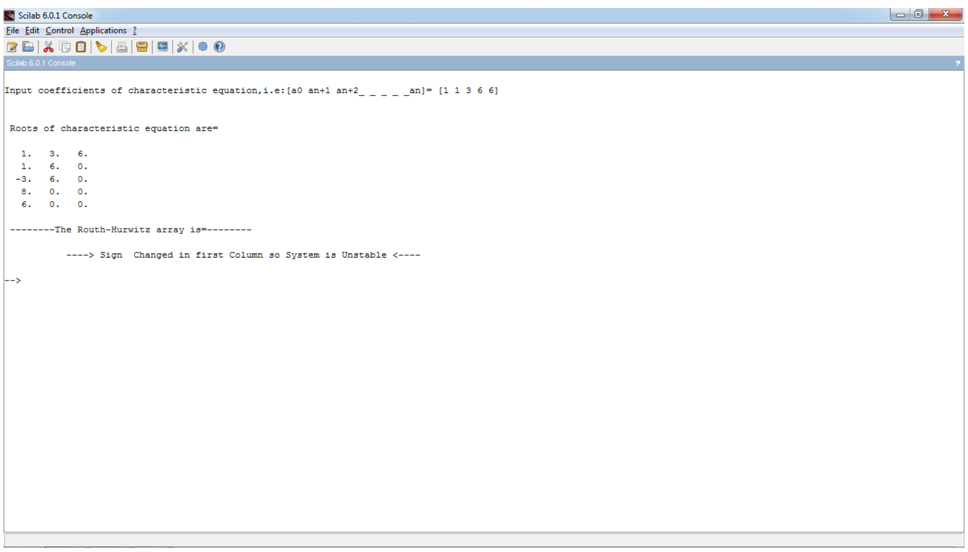

Figure 6.2: Routh Stability Criterion

Experiment: 7

Develop a program to study step responses of a 2nd order system.

Scilab code Solution 7.01 Step Response of 2nd order system

```
//
    ________________________________________________________________________________

// TO STUDY STEP RESPONSES OF A 2ND ORDER SYSTEM
//
    ________________________________________________________________________________

// OS : Windows 7
// Scilab : 6.0.1
clc;
clear all;
t=0:0.000001:0.0002;
zeta=input('Enter the values for zeta =')//[0.4 1
    1.6];
cv=[1 2 3];
```

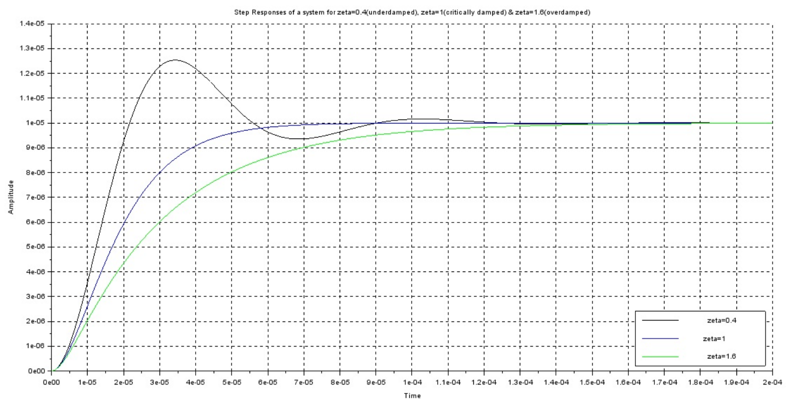

Figure 7.1: Step Response of 2nd order system

```
s=%s;
wn = input('Enter the value for wn =')//10^5;
for n=1:3
den = s^2 + 2*zeta(n)*(wn)*s+(wn^2);
P = syslin('c',wn,den);
Ps=csim('step',t,P);
plot2d(t,Ps,style=cv(n));
end;
xgrid;
xtitle(['Step Responses of a system for zeta=0.4(
    underdamped), zeta=1(critically damped) & zeta
    =1.6(overdamped)'],'Time', 'Amplitude' );
legends(['zeta=0.4';'zeta=1';'zeta=1.6'],[1,2,3],opt
    =4);


//—————— OUTPUT——————

//Enter the values for zeta =[0.4 1 1.6]

//Enter the value for wn =10^5
```

Experiment: 8

Develop a program to obtain Root locus using Scilab.

Scilab code Solution 8.01 Root Locus

```

//
    ------------------------------------------------------------------------
// To Study the Root Locus
//
    ------------------------------------------------------------------------

// OS : Windows 7
// Scilab : 6.0.1

clc
close
s=%s
num=input('Enter the Numerator =')
// Case - 1  Enter the Numerator = (s+1)
// Case - 2 Enter the Numerator = (s+1)
// Case - 3 Enter the Numerator = 1
```

```
den=input('Enter the Denominator =')
// Case - 1 Enter the Denominator = (s^2*(s+3)*(s+5))
// Case - 2 Enter the Denominator = (s*(s+2)*(s^2+2*s+5))
// Case - 3  Enter the Denominator =(s*(s+2)*(s+5))
TF = syslin('c',num,den)//Transfer function
disp(TF,"Transfer Function of system = ")
h=syslin('c',num,den)
evans(h,100)

//-------------Output----------------
//Case 1
//----------------------------------------
//Enter the Numerator =s+1

//Enter the Denominator =(s^2*(s+3)*(s+5))


//  Transfer Function of system =


//        1 + s
//    -------------
//        2     3    4
//    15s + 8s + s
//------------------------------------------------------------
//Case 2
//------------------------------------------------------------
//Enter the Numerator =s+1

//Enter the Denominator =s*(s+2)*(s^2+2*s+5)


//Transfer Function of system =


//             1 + s
```

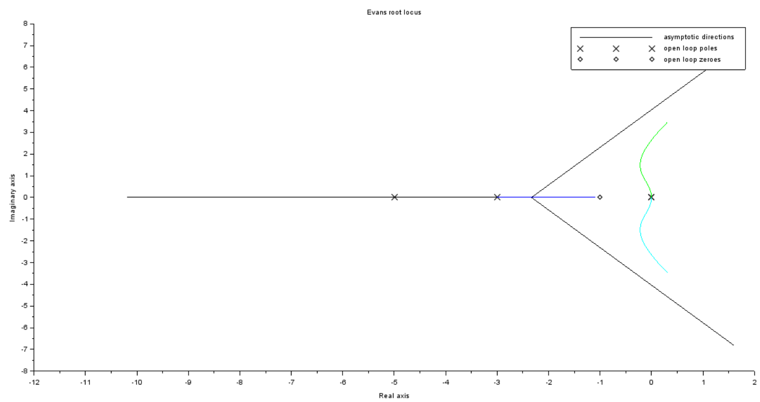

Figure 8.1: Root Locus

```
//     ____________________________
//                     2       3      4
//     10s  +  9s  +  4s  +  s
//____________________________________________________________
//Case  3
//____________________________________________________________
//Enter  the  Numerator  =1

//Enter  the  Denominator  =(s*(s+2)*(s+5))


//  Transfer  Function  of  system  =


//              1
//     ____________________
//                     2      3
//     10s  +  7s  +  s
```

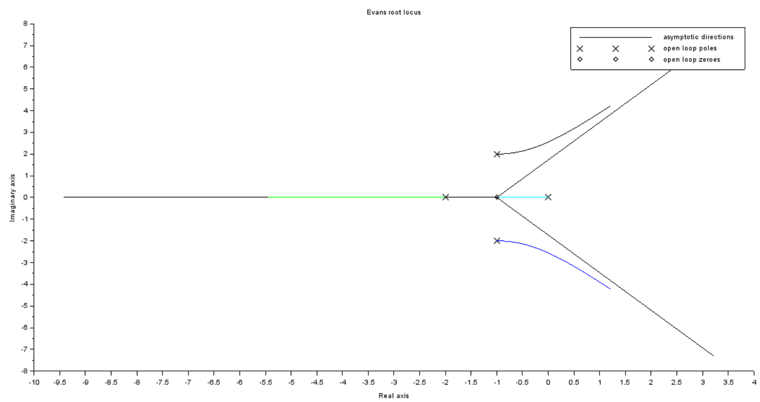

Figure 8.2: Root Locus

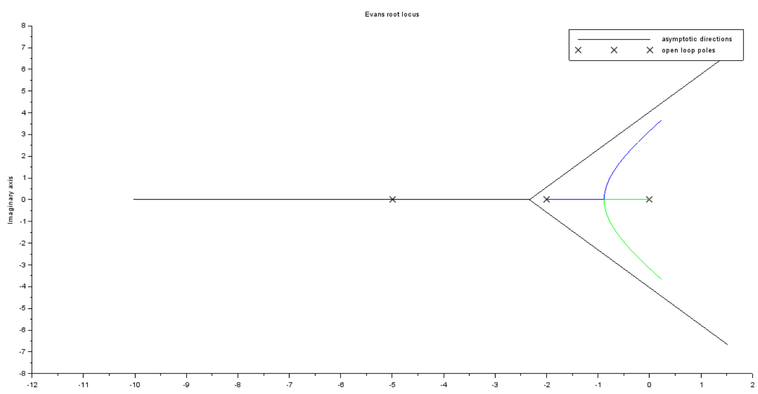

Figure 8.3: Root Locus

Experiment: 9

Develop a program to obtain Nyquist Plot using Scilab.

Scilab code Solution 9.01 Nyquist plot

```

//  ------------------------------------------------------------------------
// To Study the Nyquist plot
//  ------------------------------------------------------------------------

// OS : Windows 7
// Scilab : 6.0.1
// Toolboxes : Maxima 5.20.1 and ActivePerl 5.20.2
clc
close

s=poly(0,'s')
num=input('Enter the Numerator =')
// Case - 1  Enter the Numerator = 1
// Case - 2 Enter the Numerator = (5+s)*(s+40)
```

```
// Case - 3 Enter the Numerator = (s+1)
den=input('Enter the Denominator =')
// Case - 1 Enter the Denominator = (s*(s+1)*(2*s+1))
// Case - 2 Enter the Denominator = (s^3)*(s+200)*(s+1000)
// Case - 3  Enter the Denominator =(s^2)*(s+5)*(s+10)
TF = syslin('c',num,den)//Transfer function
disp(TF,"Transfer Function of system = ")
h=syslin('c',num,den)
nyquist(h);

//-------------Output------------------
//Case 1
//-----------------------------------------
//Enter the Numerator =1

//Enter the Denominator = (s*(s+1)*(2*s+1))


// Transfer Function of system =


//           1
//     ---------------
//            2      3
//     s + 3s + 2s

//------------------------------------------------------
//Case 2
//------------------------------------------------------

//Enter the Numerator =(5+s)*(s+40)

//Enter the Denominator =(s^3)*(s+200)*(s+1000)


```

```
// Transfer Function of system =

//                         2
//        200 + 45s + s
//    -------------------------
//             3          4    5
//    200000s + 1200s + s

//---------------------------------------------------
//Case 3
//---------------------------------------------------
//Enter the Numerator =(s+1)

//Enter the Denominator =(s^2)*(s+5)*(s+10)


// Transfer Function of system =


//         1 + s
//    -----------------
//        2       3    4
//    50s + 15s + s
```

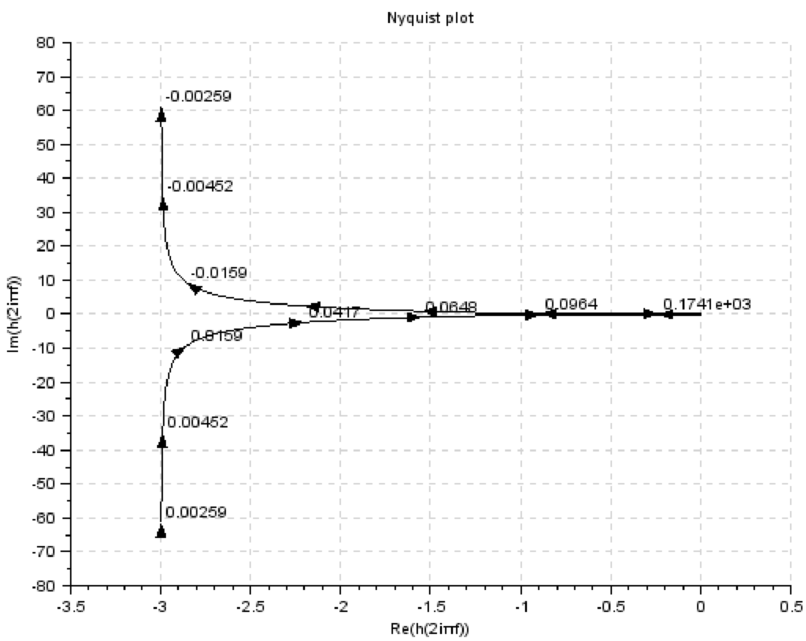

Figure 9.1: Nyquist plot

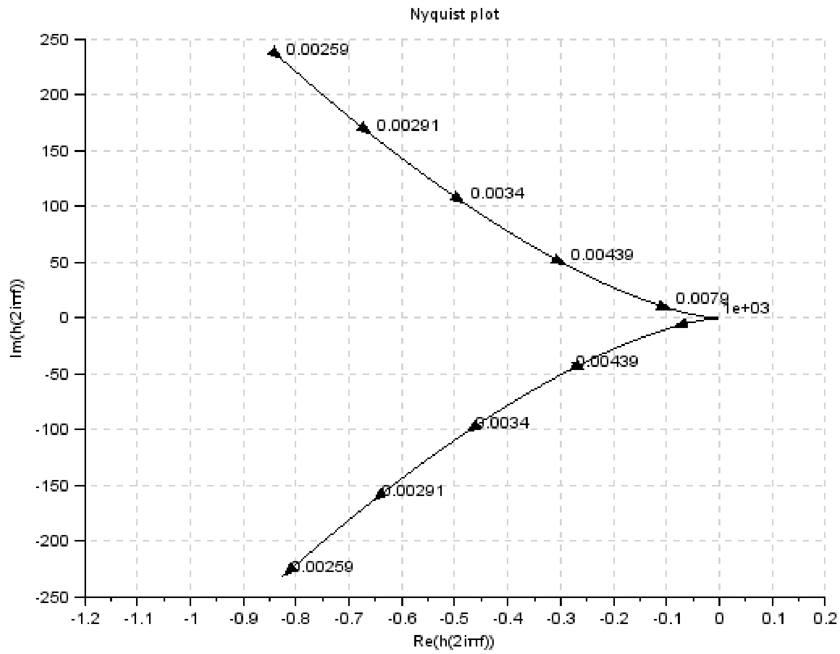

Figure 9.2: Nyquist plot

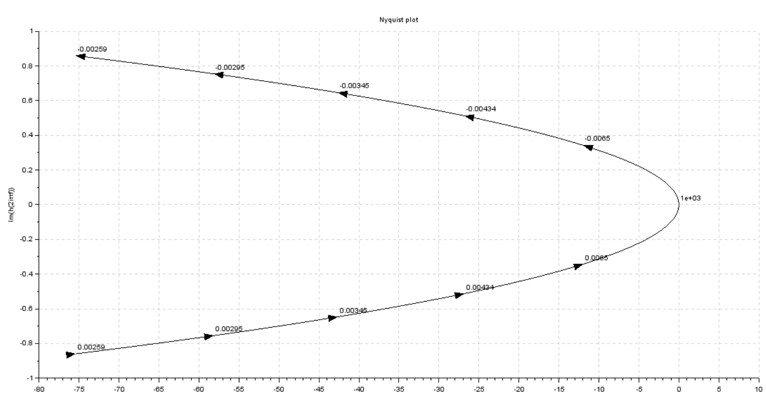

Figure 9.3: Nyquist plot

Experiment: 10

Develop a program to obtain Bode Plot using Scilab.

Scilab code Solution 10.01 Bode Plot

```
// ________________________________________________________________________________
// To Study the Bode plot
// ________________________________________________________________________________

// OS : Windows 7
// Scilab : 6.0.1

clc
close

s=poly(0,'s')
num=input('Enter the Numarator =')
// Case - 1  Enter the Numarator = 20
// Case - 2 Enter the Numarator = (s+1)
// Case - 3 Enter the Numarator = 1
```

```
den=input('Enter the Denominator =')
// Case - 1 Enter the Denominator = (s)*(1+s)*(1+0.5*s)
// Case - 2 Enter the Denominator = (s)*(s+2)*(s+5)
// Case - 3  Enter the Denominator =(s)*(s+1)*(s+5)*(s+10)
TF = syslin('c',num,den)//Transfer function
disp(TF,"Transfer Function of system = ")
h=syslin('c',num,den)
clf();
bode(h,0.1,100)
g_margin(h)
show_margins(h)
p_margin(h)
show_margins(h)

//----------Output-------------
//Case 1
//--------------------------------
//Enter the Numarator =20

//Enter the Denominator =(s)*(1+s)*(1+0.5*s)


// Transfer Function of system =


//          20
//    --------------------
//             2        3
//    s + 1.5s + 0.5s
//-----------------------------------------------
//Case 2
//-----------------------------------------------
//Enter the Numarator =s+1

//Enter the Denominator =(s)*(s+2)*(s+5)

```

```
//Transfer Function of system =


//          1 + s
//    ---------------
//              2    3
//    10s + 7s + s
//------------------------------------------------
//Case 3
//------------------------------------------------
//Enter the Numarator =1

//Enter the Denominator = (s)*(s+1)*(s+5)*(s+10)


// Transfer Function of system =


//               1
//    -----------------------
//               2       3    4
//    50s + 65s + 16s + s
```

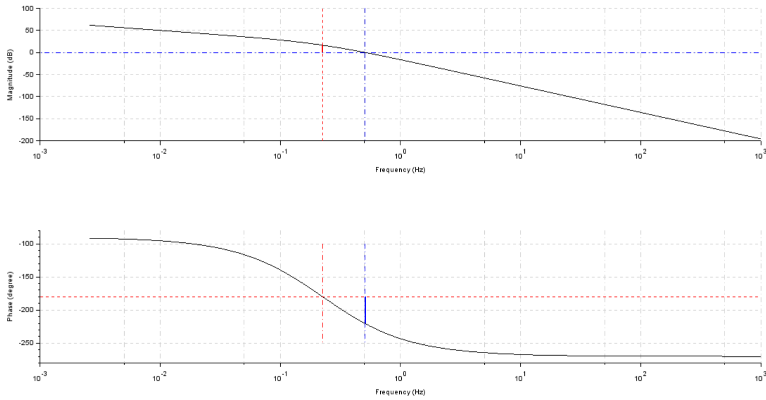

Figure 10.1: Bode Plot

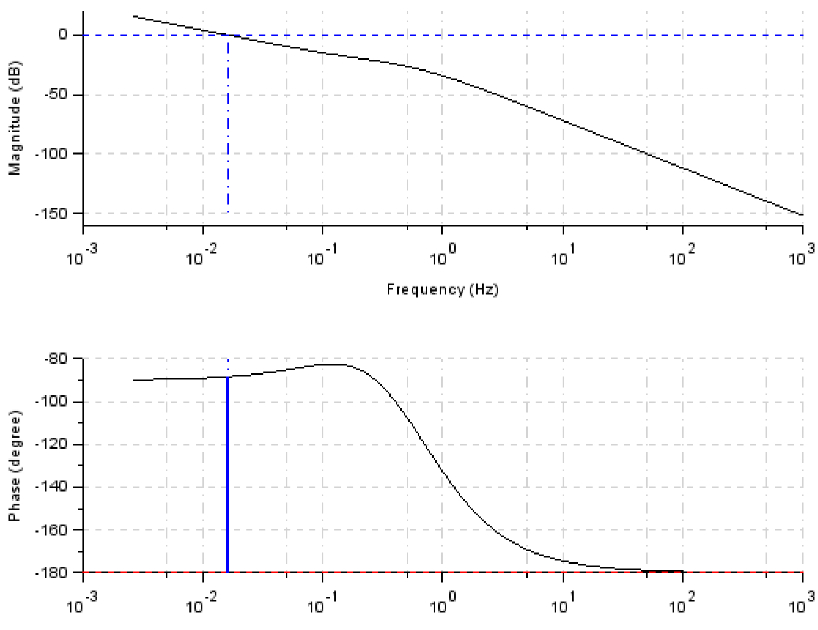

Figure 10.2: Bode Plot

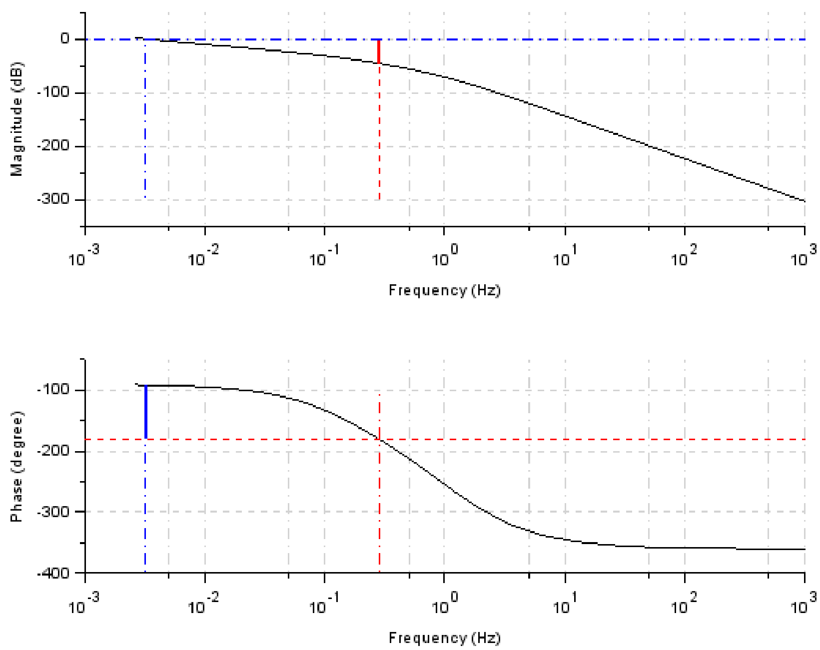

Figure 10.3: Bode Plot