Scilab Manual for Control Systems and Simulation by Prof Kiran Babu Electrical Engineering Vignan's Institute of Technology & Aeronautical Engg.[1]

Solutions provided by
Prof Kiran Babu
Electrical Engineering
Vignan'sInstituteOfTechnologyAndAeronauticalEngg.

June 20, 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

Linear System Analysis (Time domain analysis, Error analysis)

Scilab code Solution 1.1 Linear System Analysis

```
1 //Created using Ubuntu 14.04 and Scilab 5.5.0
2 //Time Domain Analysis
3 clear
4 num=poly([12.811 18 6.3223],"s","coeff"); //
    Defines the numerator of the transfer function
5 den=poly([12.811 18 11.3223 6 1],"s","coeff"); //
    Defines the denominator of the transfer function
6 sl=syslin('c',num,den); //
    Defines the transfer function
7 t=[0:0.001:25]; //The
     time of simulation is set from 0 to 25 seconds
8 plot2d(t,csim('step',t,sl)) //It
    plots the step response of the transfer function
    sl
9 xgrid(5,1,7)
10 xtitle('Time Domain Analysis','Time(sec)',' C (t) ')
11
```

```
12
13
14 //Error Analysis
15 clear
16 clc
17 num=poly([240 120],"s","coeff"); //Defines the
     numerator of G(s)
18 den=poly([12 7 1],"s","coeff"); //Defines the
     denominator of G(s)
19 G=num/den;                          //Defines G(s)
20 Ess=1/(1+horner(G,0));              //Evaluates Steady
     state error for step input
21 mprintf('The steady state error is')
22 disp(Ess)
```

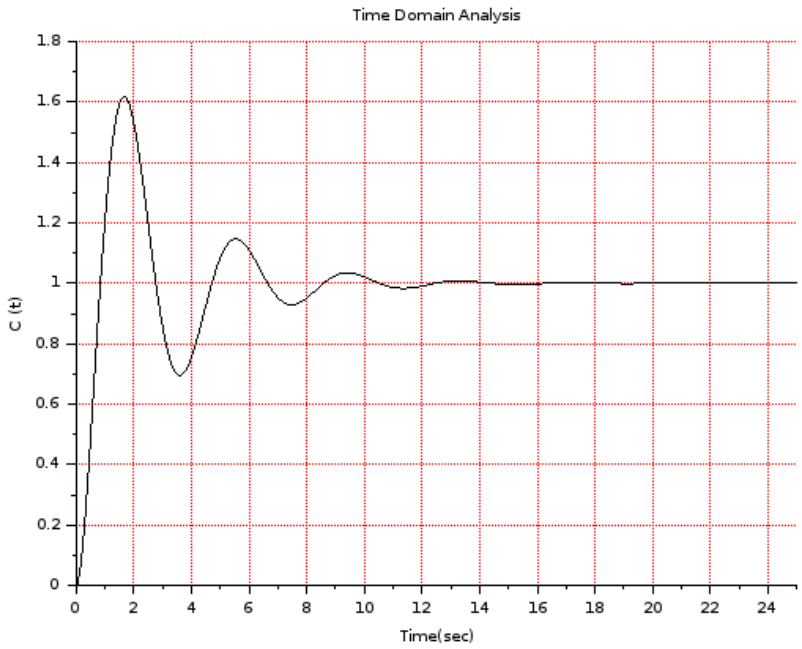

Figure 1.1: Linear System Analysis

Experiment: 2

Stability analysis (Bode, Root Locus, Nyquist) of Linear Time Invariant System

Scilab code Solution 2.2 Stability Analysis

```
1 //Created using Ubuntu 14.04 and Scilab 5.5.0
2 //Bode Plot
3 clear
4 num=poly([9 1.8 9],"s","coeff");    //Defines the
     numerator of G(s)
5 den=poly([0 9 1.2 1],"s","coeff"); //Defines the
     denominator of G(s)
6 sl=syslin('c',num,den);             //Defines G(s)
7 subplot(2,2,1)
8 bode(sl,0.01,100)                  //It plots the
     Bode plot from 0.01 Hz to 100 Hz
9
10
11 //Root Locus
12 clear
```

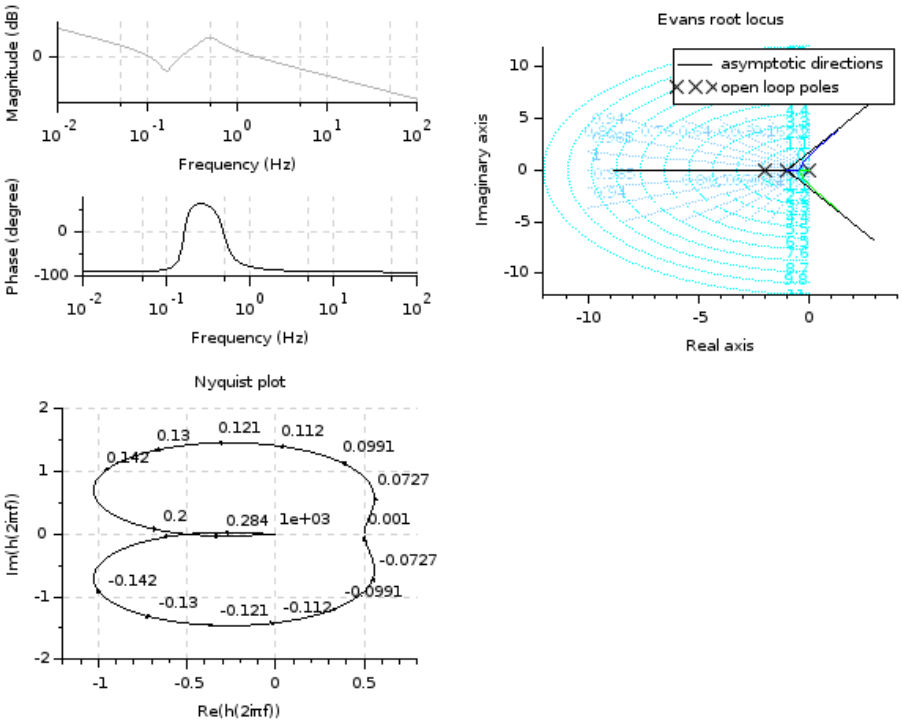

Figure 2.1: Stability Analysis

```
13 num=1;                                  //Defines the
     numerator of G(s)
14 den=poly([0 -1 -2],"s","roots"); //Defines the
     denominator of G(s)
15 sl=syslin('c',num,den);                 //Defines G(s)
16 subplot(2,2,2)
17 evans(sl,100)                          //Plots the root
     locus for a maximum gain of 100
18 sgrid()
19
20
21 //Nyquist Plot
22 clear
23 num=poly([0.5 1],"s","coeff");    //Defines the
     numerator of G(s)
24 den=poly([1 0 1 1],"s","coeff"); //Defines the
     denominator of G(s)
25 sl=syslin('c',num,den);                 //Defines G(s)
26 subplot(2,2,3)
27 nyquist(sl)                           //Plots the Nyquist
     plot for G(s)
```

Experiment: 3

State space model for classical transfer function - Verification

Scilab code Solution 3.3 Statespace model for Classical transfer function

```
1 //Created using Ubuntu 14.04 and Scilab 5.5.0
2 clear
3 clc
4 num=poly([10 10],"s","coeff");     //Defines the
     numerator of the transfer function
5 den=poly([10 5 6 1],"s","coeff"); //Defines the
     denominator of the transfer function
6 sl=syslin('c',num,den);            //Defines the
     transfer function
7 sys=tf2ss(sl);                     //Converts
     transfer function to space model and stores the
     result in "sys"
8 disp(sys)
```