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.

April 25, 2024

[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

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

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

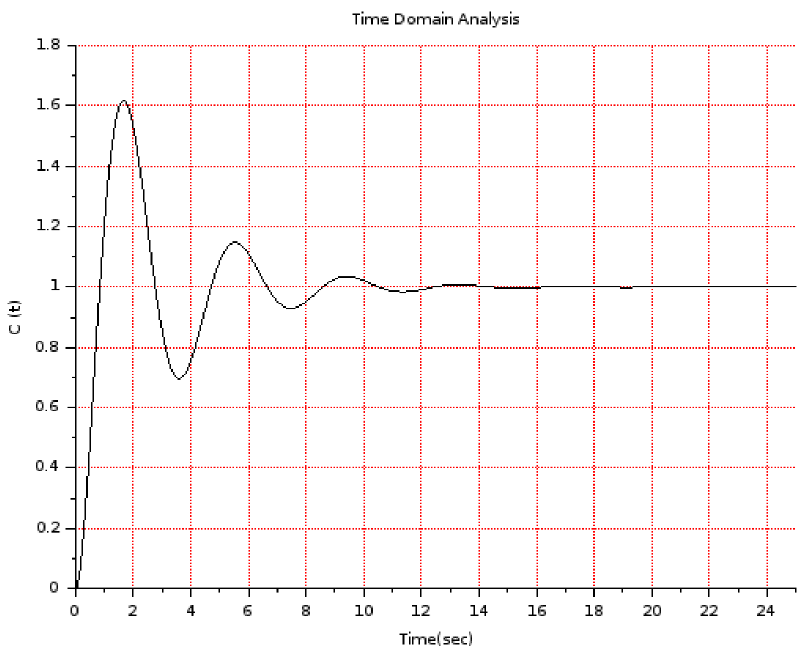

Figure 1.1: Linear System Analysis

Experiment: 2

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

Scilab code Solution 2.2 Stability Analysis

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


//Root Locus
clear
```

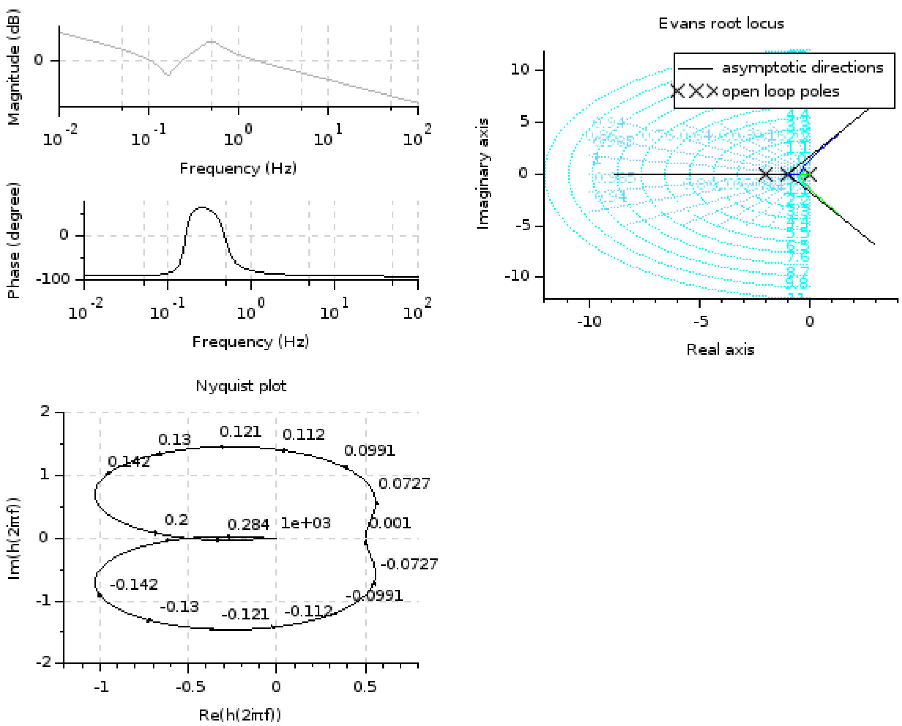

Figure 2.1: Stability Analysis

```
num=1;                          //Defines the numerator of G(s)
den=poly([0 -1 -2],"s","roots"); //Defines the denominator of G(s)
sl=syslin('c',num,den);         //Defines G(s)
subplot(2,2,2)
evans(sl,100)                  //Plots the root locus for a maximum gain of 100
sgrid()


//Nyquist Plot
clear
num=poly([0.5 1],"s","coeff");   //Defines the numerator of G(s)
den=poly([1 0 1 1],"s","coeff"); //Defines the denominator of G(s)
sl=syslin('c',num,den);         //Defines G(s)
subplot(2,2,3)
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

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