Scilab Manual for
Lab Practices II (M.E. Instrumentation & Control)

by Prof Deepti Khimani
Instrumentation Engineering
VESIT[1]

Solutions provided by
Prof Mrs. Deepti Khimani
Instrumentation Engineering
Mumbai University

May 19, 2022

[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 various built in functions for finding determinant, inverse, eigen values and eigen vectors.

Scilab code Solution 1.1 Lab01

```
// Lab01: To study various built in functions for
   finding determinant,
// inverse, eigen values and eigen vectors.

//              scilab - 5.5.2
//              Operating System : Windows 7, 32-bit

//
   ____________________________________________________________________________


//Clean the environment
close;
clear;
clc;
```

```
//
-------------------------------------------------------------------------------

// Ex1.
//Declaration of given matrix
A=[3 2 0 ; 2 -1 7; 5 4 9];
disp(A,"Matirx A=","Ex1.:")

//Built-in fuctions to compute the determinant and
   inverse of a given matrix.
detA=det(A);
disp(detA, 'Determinant of A is=')
invA=inv(A);
disp(invA, 'Inverse of A is=')

//Built-in fuction to find eigen values of a given
   matrix.
eig_val=spec(A)
disp(eig_val, 'Eigen values of matrix A are=')

[eig_vec eig_val]=spec(A)
disp(eig_vec,'Eigen vector form of matrix A is')
disp(eig_val,'Diagonal form of eigen values of
   matrix A')

//
-------------------------------------------------------------------------------

// Ex.2
//Declaration of given matrix
B=[-4 2 5; 7 -1 6; 2 3 7];
disp(B,"Matirx B=","Ex2.:")

//Built-in fuction to compute the determinant and
   inverse of a given matrix.
```

```
detB=det(B);
disp(detB, 'Determinant of B is=')
invB=inv(B);
disp(invB, 'Inverse of B is=')

//Built-in fuction to find eigen values of a given matrix.
eig_val=spec(B)
disp(eig_val, 'Eigen values of matrix B are=')

[eig_vec eig_val]=spec(B)
disp(eig_vec,'Eigen vector form of matrix B is')
disp(eig_val,'Diagonal form of eigen values of matrix B')

//
```

Experiment: 2

Obtain 1, 2, infinity norm and reduced row echeolon form of a given matrix.

Scilab code Solution 2.1 Lab02

```
// Lab02: Obtain 1, 2, infinity norm and reduced row echelon form
// of a given matrix.

//              scilab - 5.5.2
//              Operating System : Windows 7, 32-bit
// ------------------------------------------------------------------------

//Clean the environment
close;
clear;
clc;

// ------------------------------------------------------------------------
```

```
// Ex1.
//Declaration for given matrix
A=[0 1 0; 3 0 2; -12 7 -6];
disp(A,"Matirx A=","Ex1.:")

//1-norm of matrix
Nrm_1=norm(A,1);
disp(Nrm_1,'1-norm of a given matrix is:')

//2-norm of matrix
Nrm_2=norm(A,2);
disp(Nrm_2,'2-norm of a given matrix is:')

//inf. norm of matrix
Nrm_inf=norm(A,'inf');
disp(Nrm_inf,'Infinity norm of a given matrix is')

// Reduced row Echeolon Form of a matrix
Re=rref(A);
disp(Re,'Reduced row Echeolon Form of A is =')

//
    ----------------------------------------------------------------------------------------------------

// Ex2.
//Declaration for given matrix
B=[2 3 4 5; -7 4 3 -2; 0 0 1 3; 1 0 -5 -4];
disp(B,"Matirx B=","Ex2.:")

//1-norm of matrix
Nrm_1=norm(B,1);
disp(Nrm_1,,'Norm 1 of a given matrix is')

//2-norm of matrix
Nrm_2=norm(B,2);
```

```
disp(Nrm_2,,'Norm 2 of a given matrix is')

//inf. norm of matrix
Nrm_inf=norm(B,'inf');
disp(Nrm_inf,,'Infinity norm of a given matrix is')

// Reduced row Echeolon Form of a matrix
Re=rref(B);
disp(Re,'Reduced row Echeolon Form of B is =')

//
    ____________________________________________________________________
```

Experiment: 3

To perform LU Decomposition on a given matrix.

Scilab code Solution 3.1 Lab03

```
// Lab03: To perform LU Decomposition on a given
   matrix.

//                  scilab - 5.5.2
//                  Operating System : Windows 7, 32-bit
//
   ---------------------------------------------------------------------------------

//Clean the environment

close;
clear;
clc;

//
   ---------------------------------------------------------------------------------


// Ex1.
```

```
//Declaration for given matrix
A=[1 2 3; 2 3 4; 3 4 6];
disp(A,"Matirx A=","Ex1.:")

//LU decomposition
[La Ua]=lu(A);
disp(La,'Lower tringular matrix La=')
disp(Ua,'Upper tringular matrix Ua=')

//
    ------------------------------------------------------------------------------------------------

// Ex2.
//Declaration for given matrix
B=[3 5 7 7; 98 7 42 122; 100 25 90 160; 234 67 91
    97];
disp(B,"Matirx B=","Ex2.:")

//LU decomposition
[Lb Ub]=lu(B);
disp(Lb,'Lower tringular matrix Lb=')
disp(Ub,'Upper tringular matrix Ub=')

//
    ------------------------------------------------------------------------------------------------
```

Experiment: 4

To perform QR Decomposition on a given matrix.

Scilab code Solution 4.1 Lab04

```
// Lab04: To perform QR Decomposition on a given
   matrix.

//                 scilab - 5.5.2
//                 Operating System : Windows 7, 32-bit
//
   ----------------------------------------------------------------------------------------


//Clean the environment
close;
clear;
clc;
//
   ----------------------------------------------------------------------------------------


//Ex1.
// Declaration for given matrix
```

```
A=[3 6; -2 5];
disp(A,"Matirx A=","Ex1.:")

//QR decomposition
[Qa Ra]=qr(A);
disp(Qa,'Orthogonal matrix Qa=')
disp(Ra,'Upper tringular matrix Ra=')

//------------------------------------------------------------------------------

//Ex2.
// Declaration for given matrix
B=[1 2 3 4; 1 3 3 5; 1 2 3 7];
disp(B,"Matirx B=","Ex2.:")

//QR decomposition
[Qb Rb]=qr(B);
disp(Qb,'Orthogonal matrix Qb=')
disp(Rb,'Upper tringular matrix Rb=')

//------------------------------------------------------------------------------
```

Experiment: 5

To perform Single Value Decomposition on a given matrix.

Scilab code Solution 5.1 Lab05

```
// Lab05: To perform Singular Value Decomposition on
    a given matrix.

//                scilab - 5.5.2
//                Operating System : Windows 7, 32-bit
//
    ------------------------------------------------------------------------


//Clean the environment
close;
clear;
clc;

//
    ------------------------------------------------------------------------
```

```
//Ex1.
// Declaration for given matrix
A=[3 6; -2 5];
disp(A,"Matirx A=","Ex1.:")

//Singular value decomposition
[Ua Sa Va]=svd(A);
disp(Ua,'Orthogonal matrix Ua=')
disp(Va,'Orthogonal matrix Va=')
disp(Sa,'Singular value matrix Sa=')

//
    ------------------------------------------------------------------------------------

//Ex2.
// Declaration of another matrix
B=[1 2 3 4; 1 3 3 5; 1 2 3 7];
disp(B,"Matirx B=","Ex2.:")

//Singular value decomposition
[Ub Sb Vb]=svd(B);
disp(Ub,'Orthogonal matrix Ub=')
disp(Vb,'Orthogonal matrix Vb=')
disp(Sb,'Singular value matrix Sb=')

//
    ------------------------------------------------------------------------------------
```

Experiment: 6

To study Gram Schmidt Orthogonalization.

Scilab code Solution 6.1 Lab06

```
// Lab06: To study Gram Schmidt Orthogonalization.

//                 scilab - 5.5.2
//                 Operating System : Windows 7, 32-bit

//
    ------------------------------------------------------------------------

//Clean the environment
close;
clear;
clc;

//
    ------------------------------------------------------------------------

// Declaration of given matrix
```

```
a=[1 2 1; 0 2 1; 2 3 0; 1 1 1];
[m n]=size(a);
q=zeros(m,n);
r=zeros(n,n);

for j=1:n
    v=a(:,j)
    for i=1:j-1
        r(i,j)=q(:,i)'*a(:,j)
        v=v-r(i,j)*q(:,i)
    end
    r(j,j)=norm(v)
    q(:,j)=v/r(j,j)
end

disp(v,'v=')
disp(r,'r=')
disp(q,'q=')

//
```

Experiment: 7

To compare closed loop response of a given system using P, PI and PID controller.

Scilab code Solution 7.1 Lab07

```
// Lab07: To compare closed loop response of a given
    system using
// P, PI and PID controller.

//                scilab - 5.5.2
//                Operating System : Windows 7, 32-bit
//
    ------------------------------------------------------------

//Clean the environment
xdel(winsid());//close all graphics Windows clear ;
clear;
clc;

//
    ------------------------------------------------------------
```

```
//// Transfer function
s=%s; // or
s=poly(0,'s');
sys=syslin('c',(1)/(s^2+10*s+20))

//step response
t=0:0.05:2.5;
v=csim('step',t,sys);
plot2d(t,v,2)

//Title, labels and grid to the figure
//figure handel settings
f=get("current_figure"); //Current figure handle
f.background=8; //make the figure window background white
l=f.children(1);
l.background=8 ;//make the text background white
id=color('grey');
xgrid(id);
//custom script for setting figure properties
title('Response of given system to a step','fontsize',3)
xlabel('Time t (sec.)','fontsize',2)
ylabel('Amplitude','fontsize',2)

//------------------------------------------------------------------------------------------------------------


// Response of the system with proportional controller
Kp=10;
sys_P=syslin('c',(Kp)/(s^2+10*s+20+Kp));

//step response
figure;
t=0:0.05:2.5;
```

```
v=csim('step',t,sys_P);
plot2d(t,v,2)

//Title, labels and grid to the figure
//figure handel settings
f=get("current_figure"); //Current figure handle
f.background=8; //make the figure window background white
l=f.children(1);
l.background=8 ;//make the text background white
id=color('grey');
xgrid(id);
title('Response of given system with Proportional controller','fontsize',3)
xlabel('Time t (sec.)','fontsize',2)
ylabel('Amplitude','fontsize',2)
xstring(1,0.31,"Steady state error=(1-0.335)*100=66.5%")

//
------------------------------------------------------------------------------
// Response of the system with proportional plus integral (PI) controller
Kp=25;
Ki=60;
sys_PI=syslin('c',(Kp*s+Ki)/(s^3+10*s^2+(20+Kp)*s+Ki));

//step response
figure;
t=0:0.05:2.5;
v=csim('step',t,sys_PI);
plot2d(t,v,2)

//Title, labels and grid to the figure
//figure handel settings
f=get("current_figure"); //Current figure handle
```

```
f.background=8; //make the figure window background
   white
l=f.children(1);
l.background=8 ;//make the text background white
id=color('grey');
xgrid(id);
title('Response of given system with Proportional
   plus integral...
 controller','fontsize',3)
xlabel('Time t (sec.)','fontsize',2)
ylabel('Amplitude','fontsize',2)
xstring(1,0.93,"Steady state error=(1−1)∗100=0%")

//
   ________________________________________________________________

// Response of the system with proportional plus
   integral plus
//derivative (PID) controller
Kp=25;
Ki=60;
Kd=1;
sys_PID=syslin('c',(Kd*s^2+Kp*s+Ki)/(s^3+(10+Kd)*s
   ^2+(20+Kp)*s+Ki));

//step response
figure;
t=0:0.05:2.5;
v=csim('step',t,sys_PID);
plot2d(t,v,2)

//Title, labels and grid to the figure
//figure handel settings
f=get("current_figure"); //Current figure handle
f.background=8; //make the figure window background
   white
l=f.children(1);
l.background=8 ;//make the text background white
```

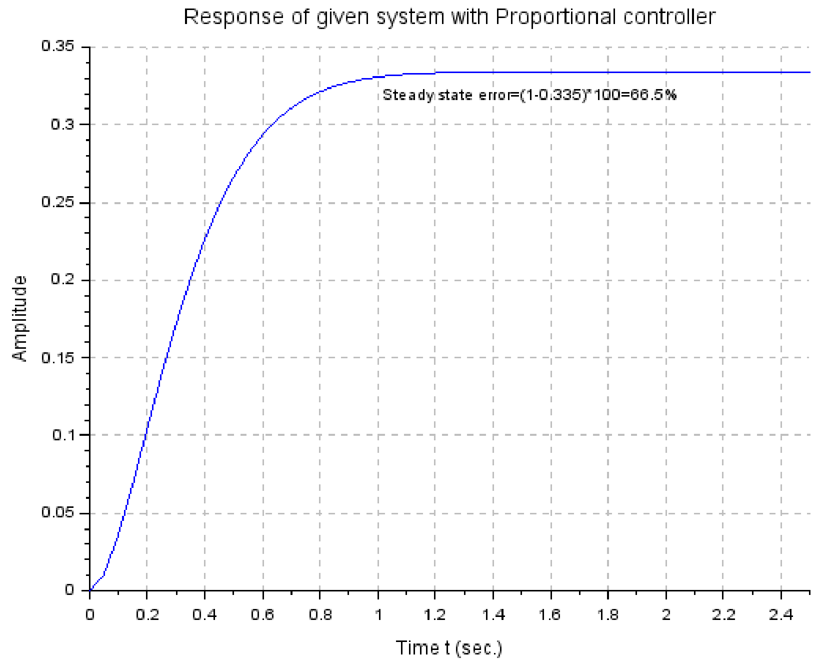

Figure 7.1: Lab07

```
id=color('grey');
xgrid(id);

title('Response of given system with Proportional plus integral...
 plus derivative controller','fontsize',3)
xlabel('Time t (sec.)','fontsize',2)
ylabel('Amplitude','fontsize',2)
xstring(1,0.93,"Settling time improved to 1.6 sec.")
//
```

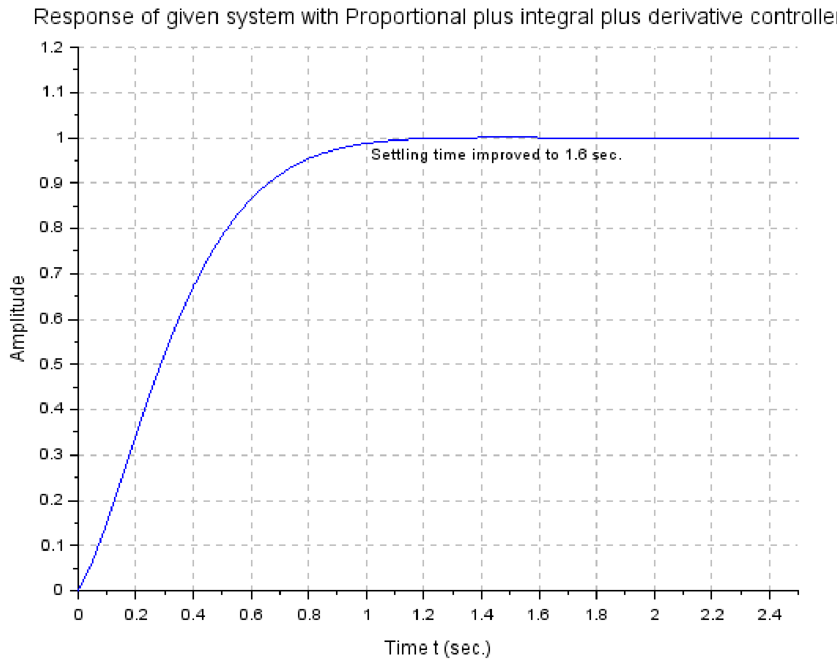

Figure 7.2: Lab07

Experiment: 8

Design the controller for a given system using Model Reference Adaptive Control approach.

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

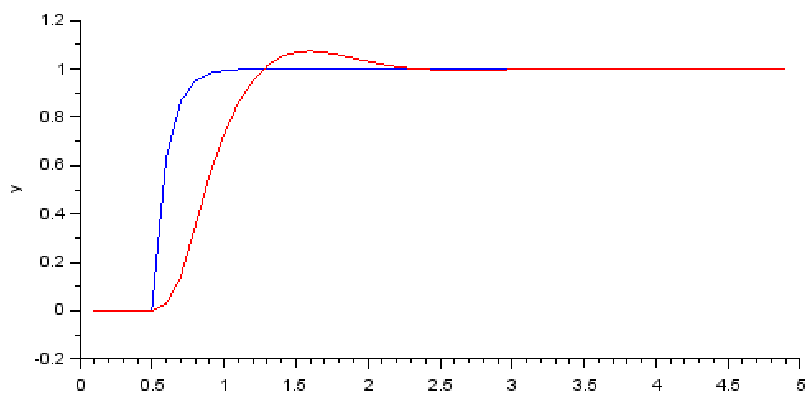

Figure 8.1: Lab08

Experiment: 9

Design the controller for a given system using Lyapunov design approach.

Scilab code Solution 9.1 Lab09

```
// Lab09: Design the controller for a given system
   using
// Lyapunov design approach.


//              scilab - 5.5.2
//              Operating System : Windows 7, 32-bit
//
   ------------------------------------------------------------

//Clean the environment
xdel(winsid());//close all graphics Windows clear ;
clear;
clc;

```

```
//
    ------------------------------------------------------------------------

//system dicription
A=[0 1;0.5 -0.5]
B=[0 1]';
C=[1 0];
D=0;

//system model

sys=syslin ('c',A,B,C,D);
//
    ------------------------------------------------------------------------


//controller design
//here alpha1=1.5, alpha2=2;
alpha1=1.5;
alpha2=2;
b2=2;
//by design
k1=1+alpha1;
k2=abs(alpha2)+3 // condition is k2>|alpha2|
K=inv(b2)*[k1 k2];

//
    ------------------------------------------------------------------------

//closed loop system model

Ac=A-B*K
sysc=syslin ('c',A-B*K,B,C,D);

//
    ------------------------------------------------------------------------


```

```
//Simulation

x0=[1 0.5]'; //initial Condition
t=0:0.1:5; //simulation time
u=zeros(1,length(t)); //no input
[y x]=csim(u,t,sys,x0); //open loop system

tc=0:0.1:20; //simulation time
uc=zeros(1,length(tc)); //no input
[yc xc]=csim(uc,tc,sysc,x0); //closed loop system

//

//Open and closed loop state responses
plot(t',x'); //open loop

//Title, labels and grid to the figure

//figure handel settings
f=get("current_figure"); //Current figure handle
f.background=8; //make the figure window background white
l=f.children(1);
l.background=8 ;//make the text background white
id=color('grey');
xgrid(id);

title('Open loop state responses','fontsize',3)
xlabel('$t (sec.)$','fontsize',3)
ylabel (["$x_1(t)$","$x_2(t)$"] , 'fontsize',3)
h=legend("$x_1(t)$","$x_2(t)$",4);
h.font_size=3;
h.fill_mode='off'

figure,
plot(tc',xc'); //closed loop
```

```
//Title, labels and grid to the figure
exec .\fig_settings.sci; // custom script for setting figure properties
title('Closed loop state responses','fontsize',3)
xlabel('$t (sec.)$','fontsize',3)
ylabel (["$x_1(t)$","$x_2(t)$"] , 'fontsize',3)
h=legend("$x_1(t)$","$x_2(t)$");
h.font_size=3;
h.fill_mode='off'

//----------------------------------------------------------------------------

//Open and closed loop state trajectories
figure,
plot(x(1,:),x(2,:),'r--');
plot(xc(1,:),xc(2,:));
zoom_rect([-1 -1 2 3])

//Title, labels and grid to the figure

//figure handel settings
f=get("current_figure"); //Current figure handle
f.background=8; //make the figure window background white
l=f.children(1);
l.background=8 ;//make the text background white
id=color('grey');
xgrid(id);
title('Open and Closed loop state trajectories','fontsize',3)
xlabel("$x_1(t)$",'fontsize',3);
ylabel ("$x_2(t)$", 'fontsize',3);
f=gca();
f.x_location = "origin"
f.y_location = "origin"
h=legend("Oen loop","Closed loop",4);
```

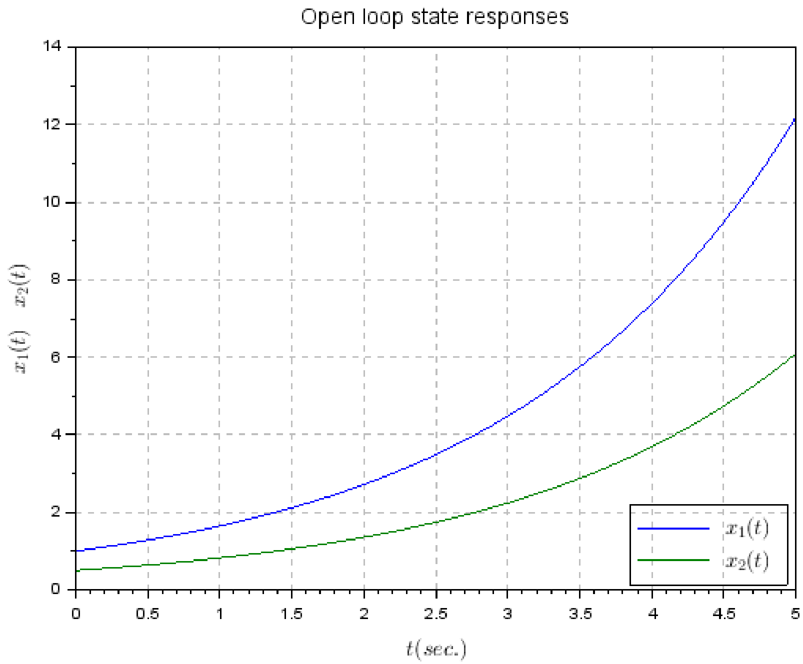

Figure 9.1: Lab09

```
h.font_size=2;
h.fill_mode='off'

//
```

check Appendix AP 1 for dependency:

`fig_settings.sci`

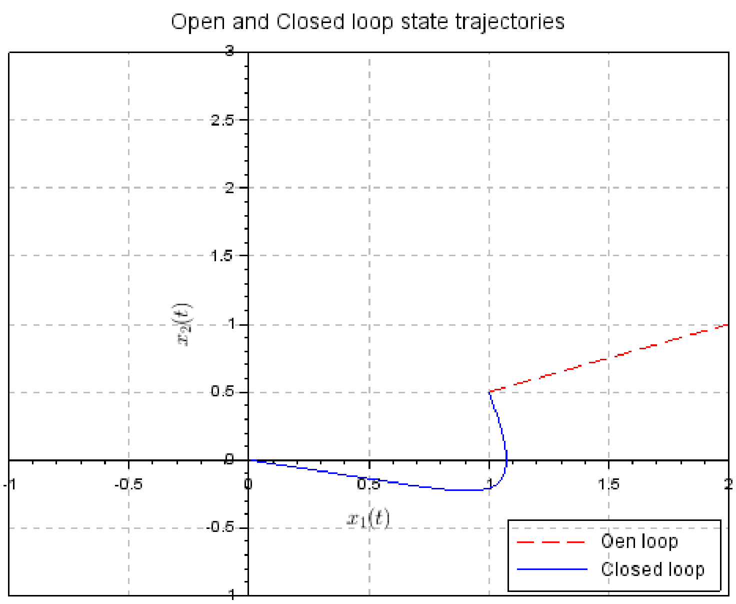

Figure 9.2: Lab09

Experiment: 10

Design the controller for a given system using variable structure control scheme.

Scilab code Solution 10.1 Lab10

```
// Lab10: Design the controller for a given system
   using
// variable structure control scheme.


//                 scilab - 5.5.1
//                 Operating System : Windows 7, 32-bit
//
   ____________________________________________________________


//Clean the environment
xdel(winsid());//close all graphics Windows clear ;
clear;
clc ;

//
```

```
    ------------------------------------------------------------

//system dicription
xi=0.7;
alpha=1;
c=-xi/2+sqrt(xi^2/4+alpha)

//
    ------------------------------------------------------------

//Simulation
importXcosDiagram("Lab10.xcos")
xcos_simulate(scs_m,4);
scs_m.props.context

//
    ------------------------------------------------------------

//State responses
figure,
plot(x1.time,x1.values,2);
plot(x2.time,x2.values,'r--');
//Title, labels and grid to the figure
exec .\fig_settings.sci; // custom script for
    setting figure properties
title('State responses of the system with VSC','
    fontsize',3)
xlabel("$t sec.$",'fontsize',3);
ylabel ("$x_1(t),\,x_2(t)$", 'fontsize',3);
h=legend("Oen loop","Closed loop",1);
h.font_size=2;
h.fill_mode='off'

//
    ------------------------------------------------------------
```

```

//State trajectory
figure,
plot(x1.values,x2.values);
//Title, labels and grid to the figure
exec .\fig_settings.sci; // custom script for setting figure properties
title('State trajectory of the system with VSC','fontsize',3)
xlabel("$x_1$",'fontsize',3);
ylabel ("$x_2$", 'fontsize',3);
f=gca();
f.x_location = "origin"
f.y_location = "origin"

//
```

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

check Appendix AP 1 for dependency:

`fig_settings.sci`

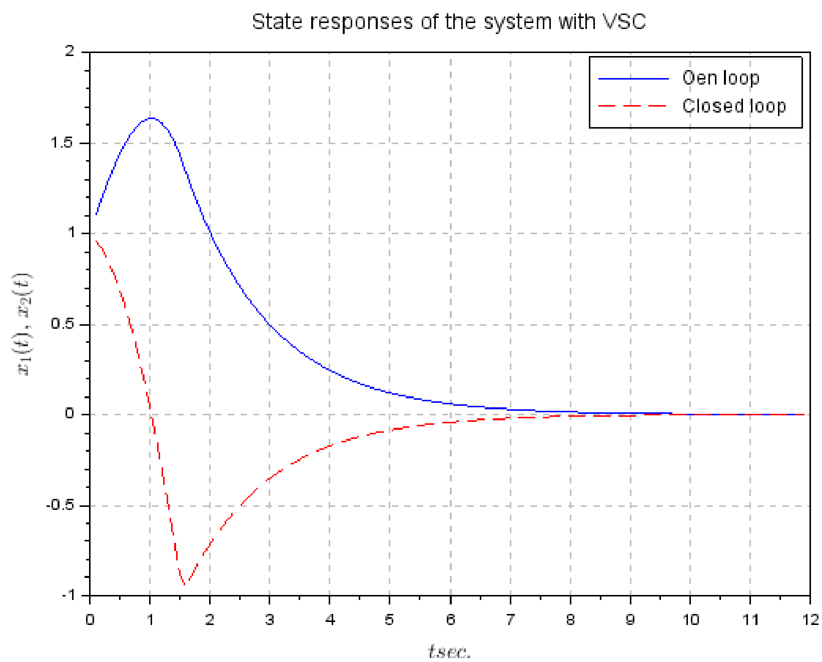

Figure 10.1: Lab10

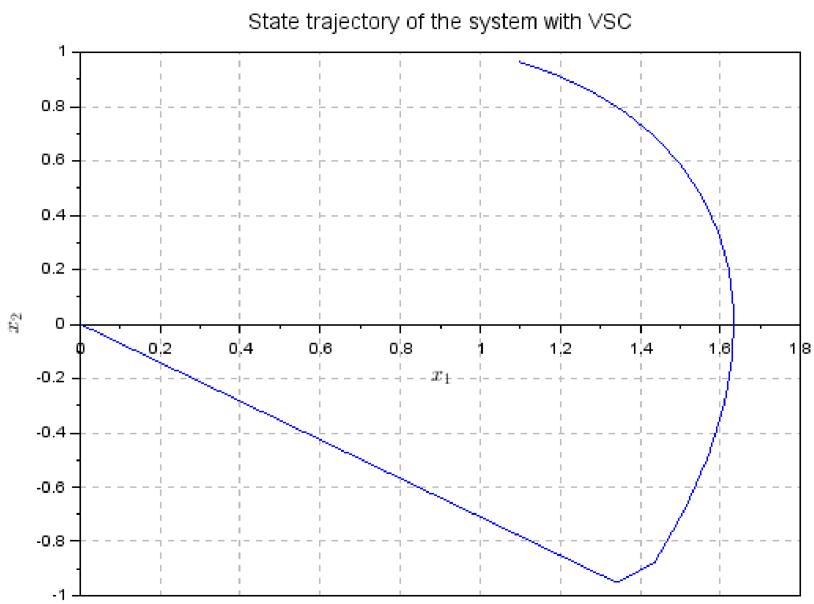

Figure 10.2: Lab10

Experiment: 11

To identify the parameters of a discrete time system using non recursive LS method.

Scilab code Solution 11.1 Lab11

```
// Lab.11: Identify the parameters of a discrete
   time system using
// non-recursive LS method.

xdel(winsid()); //close all graphics Windows clear ;
clear;
clc ;
//
   ------------------------------------------------------------

//system model
s=poly(0,'s');
num=s+1;
den=s^2+1.6*s+1;
sys=syslin('c', num/den);
//
   ------------------------------------------------------------
```

```
//Discretization of a given system
Ts=0.2;    //Sampling time
sysd=dscr(sys,Ts);
sysd_tf=ss2tf(sysd);
//------------------------------------------------------------------------
//System Identification
t=0:Ts:2;
//ramp inputis used as with the step input psi becomes singular,
//because input columns in psi are repeated
u=t;
y=flts(u,sysd_tf);

for i=1:1:5
Y(i)=y(i+3);
psi(i,1:4)=[-y(i+2) -y(i+1) u(i+2) u(i+1)];
end
theta=inv(psi'*psi)*psi'*Y;
//identified system
num=[theta([4,3])];
den=[theta([2,1]); 1];
num_z=poly(num,'z','coeff')
den_z=poly(den,'z','coeff')
sysd_id=num_z/den_z;
disp(sysd_tf,'sysd_tf=','The discrete model of the system is');
disp(sysd_tf,'sysd_id=','The identified system is');
//------------------------------------------------------------------------
```

Appendix

Scilab code AP 1

```
//figure handel settings
f=get("current_figure"); //Current figure handle
f.background=8; //make the figure window background
    white
l=f.children(1);
l.background=8 ;//make the text background white
id=color('grey');
xgrid(id);
```

fig settings