Appendix

Matlab Programs

P-1 : Matlab Code for the Computation of Controlled Trajectories and Steering Control of Semi-linear non autonomous discrete time system

Chapter 3 Example 3.6.1 is solved using this program.

```
%%------------------------------------------------------------------
%%   System Considered :
%%                   x(t+1) =  A(t)x(t) + B(t)u(t) + f(t, x(t)),
%%                              t \in N_0 = {0, 1, 2,....}
%%   A(t) : Sequence of real  n x n matrices defined in function A.m
%%   B(t) : Sequence of real n x m matrices defined in function B.m
%%   f    : nonlinear function
%%   x0   : Initial State - column vector of size n
%%   x1   : Final State - column vector of size n
```

```
%%   N     : Final time (system should reach from x0 to x1 in interval
%%           [0,N])
%%   W_r  : Reachability  Grammian
%%-------------------------------------------------------------------%
clear all
clc
k0=input('Enter initial point k0 ')
k1=input(' Enter final point k1 ')
nd=input('Enter number of points where solution is required nd ')
h=(k1-k0)/nd;
x0=input('Enter initial State x0 ')
x1=input('Enter final State x1 ')
u0=input('Enter initial control u0 ')
%-------------------------------------------------------------------%
%The following values are taken for the computation for example 3.6.1

%k0=0;
%k1=1;
%nd=10;
%u0=0;
%x0=[4 -4]';
%x1=[-4 4]';

%---------------------------------------------------------------------%

d=1;
for i=k0:h:k1
```

```
  u(:,d)=u0;
  d=d+1;
end t=1;
for i=k0:h:k1
  x(:,t)=x0;
  t=t+1;
end
prevx=x;
prevu=u;

%-----------------------------------------------------------------%
% Following part computes reachability Gramian W_r as per equation
% 3.1.6 and checks whether the  system is controllable or  not using
% W_r. Computation of transition matrix is done in function trans1.m
% according to equation 2.4.3. If system is not controllable  then
% procedure will be stopped.
%-------------------------------------------------------------------%
REASUM=0;
for i=k0:h:k1-h
    TM=trans1(i+h,k1,h);
    REASUM=REASUM + TM *B(i)*B(i)'*TM';
end
WR=REASUM;
r=det(WR);
if r==0
  disp('Linear system is not controllable');
  error('');
```

```
else
  disp('Linear system is controllable');
end
INVWR=inv(WR);
TMfix=trans1(k0,k1,h);
fix1=x1-TMfix*x0;

%Iteration loop

for count=1:1:2
   t=1;
%-------------------------------------------------------------------%
% Following part of the program computes the state x according to the
% formula mentioned in equation 3.1.3
%---------------------------------------------------------------------%
    for i=k0+h:h:k1
        t=t+1;
        d=1;
        TM1=trans1(k0,i,h);
        sum1=0;
    for j=k0:h:i-h
        TM2=trans1(j+h,i,h);
        sum1=sum1+(TM2*B(j))*prevu(:,d);
        d=d+1;
    end
    sum2=0;
    d=1;
```

```
    for l=k0:h:i-h
        tempf=[sin(prevx(:,d)).^2 cos(prevx(:,d)).^2];
        f=(1/5)*[tempf(1) tempf(4)]';
        TM3= trans1(l+h,i,h);
        sum2=sum2+TM3 *f;
        d=d+1;
    end
    x(:,t)=TM1*x0+sum1+sum2;
    end
%-----------------------------------------------------------------%
% Following part of the program computes the control u  according
% to the algorithm given in equation 3.1.5
%-----------------------------------------------------------------%
    p=1;
    for i=k0:h:k1-h
        d=1;
        TM4=trans1(i+h,k1,h);
        sum3=0;
        for j=k0:h:k1-h
            tempf=[sin(x(:,d)).^2 cos(x(:,d)).^2];
            f=(1/5)*[tempf(1) tempf(4) ]';
            TM5=trans1(j+h,k1,h);
            sum3=sum3+TM5*f;
            d=d+1;
        end
        tcon=B(i)'*TM4'*INVWR*(fix1-sum3);
        u(:,p)=tcon;
```

```
        p=p+1;
    end
    iteration=count
    prevu= u;
    x
    pause
    prevx=x;
end
%-----------------------------------------------------------------------%
% Plotting the graph of the solution with computed control verses time :
%-----------------------------------------------------------------------%
t = k0:h:k1;
plot(t,x,'linewidth',2,'color',[0,.6,0])
title('Controlled States in nonlinear nonautonomous case')
xlabel('Time (t) ')
ylabel('State (x)')
grid

%--------------------------------------------------------------------------
%Function file for matrix A used for computation in example 3.6.1
%--------------------------------------------------------------------------
function [ funA ] = A(t)
    funA  = 1/4* [cos(2*t) 1; t^2 cos(t)^2];
return;
%---------------------------------------------------------------------------
%Function file for matrix B  used for computation in example 3.6.1
%---------------------------------------------------------------------------
```

```
function funB = B(t)

    funB = [.5 .5*t]';

return;

%-----------------------------------------------------------------------------

%Following function computes state transition matrix according to formula 2.4.

%-----------------------------------------------------------------------------

function phi = trans1(k0,k1,h)

    phi=eye(2);

    if (k0==k1)

        phi=eye(2);

    else

        for l=k1-h:-h:k0

            phi=phi*A(i);

        end

    end
```

P-2 : Matlab Code for the Computation of Controlled Trajectories and Steering Control of Semi-linear Autonomous Discrete-time System

Chapter 3 Example 3.6.2 is solved using this program.

```
%%---------------------------------------------------------------------%

%%   System Considered : x(t+1) =  Ax(t) + Bu(t) + f(t, x(t)), t \in N_0

%%   N_0 = {0, 1, 2,....}

%%   A : Constant n x n real matrix
```

```
%%  B : Constant n x m  real matrix
%%  f   : nonlinear function
%%  x0 : Initial State - column vector of size n
%%  x1 : Final State - column vector of size n
%%  N : Final time (system should reach from x0 to x1 in interval [0,N])
%%  W_r : Controllability Grammian
%%  trans() functtion is used to compute transition matrix
%%-------------------------------------------------------------------%

k0=input('Enter initial point k0 ') k1=input(' Enter final point k1 ')
nd=input('Enter number of points where solution is required nd ')
 h=(k1-k0)/nd;
x0=input('Enter initial State x0 ')
x1=input('Enter final State x1 ')
u0=input('Enter initial control u0 ')
A=input('Enter matrix A ' )
B=input('Enter matrix B')
%----------------------------------------------------------%
%%Following data are taken for computation in example 3.6.2
%----------------------------------------------------------%
%k0=0;
%k1=1;
%nd=10;
%u0=[0];
%x0=[50 -50]';
%x1=[-50 50]';
%A=[1.3511 0.0239; 0.1195 1.0524];
```

```
%B=[.0237 .0319]';
%%-------------------------------------------------------------------
[n,d]=size(A);
d=1;
for i=k0:h:k1
    u(:,d)=u0;
    d=d+1;
end
t=1;
for i=k0:h:k1
    x(:,t)=x0;
    t=t+1;
end
prevx=x;
prevu=u;

%--------------------------------------------------------------------%
% Following part computes reachability Gramian W_r as per equation 3.1.6
% using A and B as constant matrices and checks whether the
% system is controllable or  not using W_r. Computation of transition
% matrix is done in function trans.m according to equation 2.4.4.
%If system is not controllable  then procedure will be stopped.
%--------------------------------------------------------------------%
REASUM=0;
for i=k0:h:k1-h
    TM=trans(A,i+h,k1,h);
    REASUM=REASUM + TM *B*B'*TM';
```

```
end
WR=REASUM;
r=det(WR);

if r==0
    disp('Linear system is not controllable');
    error('');
else
    disp('Linear system is controllable');
end

INVWR=inv(WR);
TMfix=trans(A,k0,k1,h);
fix1=x1-TMfix*x0;

%Iteration loop

for count=1:1:2
    t=1;
%-----------------------------------------------------------------%
% Following part of the program computes the state x according to the
% formula mentioned in equation 3.1.3
%-----------------------------------------------------------------%

    for i=k0+h:h:k1
        t=t+1;
        d=1;
```

```
        TM1=trans(A,k0,i,h);
        sum1=0;
        for j=k0:h:i-h
            TM2=trans(A,j+h,i,h);
            sum1=sum1+(TM2*B)*prevu(:,d);
            d=d+1;
        end
        sum2=0;
        d=1;
        for l=k0:h:i-h
            tempf=(1/5)*[sin(prevx(:,d)).^2 cos(prevx(:,d)).^2];
            f= [tempf(1) tempf(4)]';
            TM3=trans(A,l+h,i,h);
            sum2=sum2+TM3 *f;
            d=d+1;
        end
        x(:,t)=TM1*x0+sum1+sum2;
 end
%-------------------------------------------------------------------
% Following part of the program computes the control u  according
% to the algorithm given in equation 3.1.5
%-------------------------------------------------------------------
p=1;
for i=k0:h:k1-h
    d=1;
    TM4=trans(A,i+h,k1,h);
    sum3=0;
```

```
    for j=k0:h:k1-h
        tempf=[sin(x(:,d)).^2 cos(x(:,d)).^2];
        f=(1/5)*[tempf(1) tempf(4) ]';
        TM5=trans(A,j+h,k1,h);
        sum3=sum3+TM5*f;
        d=d+1;
    end
    tcon=B'*TM4'*INVWR*(fix1-sum3);
    u(:,p)=tcon;
    p=p+1;
end

iteration=count
prevu= u;
prevx=x;
end
%-----------------------------------------------------------------------
% Plotting the graph of the solution with computed control verses time :
%------------------------------------------------------------------------
t = k0:h:k1
plot(t,x,'LineWidth', 2, 'Color', [.6, 0, 0])
title('Controlled States of nonlinear autonomous system')
xlabel('Time (t) ')
ylabel('State (x)')
grid
%---------------------------------------------------------------------
%Following function computes the transition matrix using formula 2.2.4
```

```
%-----------------------------------------------------------------------
function phi = trans(A, k0,k1,h)


phi=eye(2);
if (k0==k1)
    phi=eye(2);
else
    for l=k1-h:-h:k0
        phi=phi*A;
    end
end
```

P-3 : Matlab Code for the Computation of Controlled Trajectories and Steering Control of discrete-time linear Volterra system using controller defined in equation 4.2.4

Chapter 4 Example 4.4.1 is solved using this program.

```
%%----------------------------------------------------------------------%
% System Considered : x(t+1) = \sum_{i=0}^{t}A(t)x(t-i )+ B(t)u(t), t \in N_0
%%   N_0 = {0, 1, 2,....}
%%   A(t) : Sequence of real  n x n matrices defined in function A.m
%%   B(t) : Sequence of real n x m matrices defined in function B.m
%%   u(t): Sequence of control vectors - column vector of size m
%%   f   : nonlinear function
%%   x0 : Initial State - column vector of size n
```

```
%%   x1 : Final State - column vector of size n
%%   N : Final time (system should reach from x0 to x1 in interval [0,N])
%%   W_c : Controllability Matrix
%% RREF(): produces row echelon form of argument
%%------------------------------------------------------------------%

%-------------------------------------------------------------------%
% This program checks whether the linear Volterra system is controllable
% or not using the Kalman type rank condition. It also computes the
% steering control and shows  that for  linear controllable system, any
% initial state can be steered to final state.
%-------------------------------------------------------------------%
clear all
clc
nd=input('Enter number of iterations  ')
x0=input('Enter initial state  ')
x1=input('Enter initial state  ')

%---------------------------------------------------------------%
% Following data are taken to compute the example 4.4.1
%---------------------------------------------------------------%
nd=5;
x0=[1 -1]';
x1=[-20  1]';
x(:,1)=x0;
tQ(:,:,1)=eye(2);
W = [];
```

```
%-------------------------------------------------------------------%
%following loop computes values of operator Q_t defined in 4.1.3
%and controllability matrix W stated in equation 4.2.1
%-------------------------------------------------------------------%
for t = 1 : 1: nd
    sum=zeros(2);
    W=[W  tQ(:,:,t)*B(nd-t)];
    for i = 0:1:t-1
        sum = sum + A(i)*tQ(:,:,t-i);
    end
    tQ(:,:,t+1)=sum;
end
W [row,col]=size(W);
rankValueW=rank(W);
%-------------------------------------------------------------------%
%Following part checks Kalman type rank condition for controllability
%-------------------------------------------------------------------%
if rankValueW == 2 %size of A or n
    disp('Given linear Volterra system is controllable');
else
    disp('Given linear Volterra system is not controllable');
    error('');
end
[R,jb]=rref(W);
C=W(:,jb)
[crow,ccol]=size(C);
[jbrow,jbcol]=size(jb);
```

```
%------------------------------------------------------------------
% Following portion computes the control u defined by equation 4.2.4
%------------------------------------------------------------------
u =inv(C)*( x1 - tQ(:,:,nd+1)*x0)
d=1;
for t=1:1:col
    if d <= jbcol & jb(d)==t
        U(:,t)=u(d);
        d=d+1;
    else
        U(:,t)=0;
    end
end
U
%--------------------------------------------------------------------%
% following part steers initial state to final state in given time steps
% using the control obtained above.
%--------------------------------------------------------------------%
for t=1:1:nd
    sum=zeros(2,1);
    for i=1:1:t
        sum = sum + tQ(:,:,i)*B(t-i)*U(:,i);
    end;
    x(:,t+1)= sum + tQ(:,:,t+1)*x0;
end;
% Plotting the graph of the solution with computed control verses time :
```

```
t = 1:1:nd+1

plot(t,x,'LineWidth', 2, 'Color', [.6, 0, 0])

title('Controlled States using controller defined in formula 4.2.4')

xlabel('Time (t) ')

ylabel('State (x)')

grid


%--------------------------------------------------------------------%

%Following matrix function is used for A

%--------------------------------------------------------------------%

function [ funA ] = A(n)

    funA  = [cos(n) sin(n); n/5 2*cos(n)];

return;

%---------------------------------------------------------------------%

%Following matrix function is used for matrix B

%---------------------------------------------------------------------%

function funB = B(n)

    funB = [.5 .5*n]';

return;

%-----------------------------------------------------------------%

%Following matrix function is used for transpose of matrix B

%-----------------------------------------------------------------%

function funB =tB(n)

    funB = [.5 .5*n];

return;
```

P-4 : Matlab Code for the Computation of Controlled Trajectories and Steering Control of discrete-time linear Volterra system using controller defined in equation 4.2.5

Chapter 4 Example 4.4.2 is solved using this program.

```
% Here all the system data are same as in example 4.4.1
clear all
clc
n=input('Enter number of time steps ')
%-------------------------------------------------------------------%
%Following values are taken to plot graph in example 4.4.2
%-------------------------------------------------------------------%
n=5;
x0=[1 -1]';
x1=[-20  1]';
x(:,1)=x0;
tQ(:,:,1)=eye(2);
%-------------------------------------------------------------------%
%Following loop computes values of Q_t defined in 4.1.3 and
% reachability Grammian matrix W_cg is computed using formula defined
% in 4.1.8
%-------------------------------------------------------------------%
for t = 1 : 1: n
    sum=zeros(2);
    for i = 0:1:t-1
```

```
        sum = sum + A(i)*tQ(:,:,t-i);
    end
    tQ(:,:,t+1)=sum;
end
sum1=zeros(2);
for i = 1 : 1: n
    sum1 = sum1 + tQ(:,:,i)*B(n-i)*tB(n-i)*tQ(:,:,i)';
end
W_cg=sum1;
detW_cg=det(W_cg)
[row,col]=size(W_cg);
%-------------------------------------------------------------------%
%Following part checks invertibility of controllability
%Grammian matrix.
%--------------------------------------------------------------------%

if detW_cg ~= 0
    disp('Given linear Volterra system is controllable');
else
    disp('Given linear Volterra system is not controllable');
    error('');
end
i=1;
%---------------------------------------------------------------------%
%Following part computes steering Control.
%----------------------------------------------------------------------%
for i=1:1:n
```

```
    U(:,i) = tB(n-i)*tQ(:,:,i)'*inv(W_cg)*(x1 - tQ(:,:,n+1)*x0);
end
control=U
for t=1:1:n
    sum=zeros(2,1);
    for i=1:1:t
        sum = sum + tQ(:,:,i)*B(t-i)*U(:,i);
    end;
    x(:,t+1)= sum + tQ(:,:,t+1)*x0;
end;
state = x;
%-------------------------------------------------------------------%
% Plotting the graph of time verses the solution using computed
%control
%-------------------------------------------------------------------%
t = 1:1:n+1
plot(t,x,'r')
title('Controlled States using equation 4.2.5')
xlabel('Time (t) ')
ylabel('State (x)')
grid
```

P-5 : Matlab Code to decide whether the given matrix is (sp) matrix or not (refer section 2.6.3 for the algorithm of (sp) matrix)

Chapter 5 Section 5.1

```
% Some of the examples are given here to check the functionality of
%the given function.

%a=[0 0 0 6/7;0 0  4/5  0 ; 2/3 0 0 0 ;0 1 0 0];               -- a (sp) Matrix
%a=[7/8 0 0 0 ;1/3 2/5 0 0 ;2/7 3/7 1/7 0;1/2 1/4 1/8 1/8]; -- a (sp) Matrix
%a=[0 1/5 0 0;0 2/3 0 0 ; 0 0 2/3 0;1/4 0 0 0];      -- a trivial (sp) matrix
%a=[0 0 4/7 1/3;1/3 0 2/3 0;2/5 3/5 0 0;0 0 1 0];              -- a (sp) Matrix
%a=[1/3 2/3 0 0;0 1/3 1/4 0;0 0 1/3 0;1 0 0 0];                -- a (sp) matrix
%a = [0 1 0 0;0 1/3 0 0;0 0 1/3 0;1 0 0 0];                    -- a (sp) matrix
%a=[0 0 0 1;0 1/3 0 0;0 0 1/3 0;1 0 0 0];                 -- Not a (Sp) matrix
%a=[0 1/3 0 0;0 1/3 0 0;0 0 1/3 0; 1/3 0 0 0];       -- a trivial (sp) matrix

%------------------------------------------------------------------------%
% The function displays following messages
% "True - a (sp) matrix", if the given matrix is a (sp) matrix.
% "True - a trivial (sp) matrix", if the given matrix is a  trivial (sp) matri
% "False - Not  a (sp) matrix", if the given matrix is not a (sp) matrix.
%------------------------------------------------------------------------%

function flag_sp = issp(a)
[n,d]=size(a);
```

```
i1=0;
i2=0;

rowsum=0;
for i=1:1:n
    for j=1:1:n
        if a(i,j) < 0
            disp('False - Not a (sp) matrix');
            return;
        end
    end
end
k=0;
l=0;
for i=1:1:n
    rowsum=0;
    for j=1:1:n
        rowsum = rowsum + a(i,j);
    end
    if rowsum < 1
        k=k+1;
        i1(k)= i;
    elseif rowsum==1
        l=l+1;
        i2(l)=i;
    else
        disp('False - Not a (sp) matrix');
```

```
        return;
    end
end
if k == 0
    disp('False - Not a (sp) matrix');
    return;
end
if l == 0
    disp('True - a trivial (sp) matrix');
    return;
end

ne=1;
flag_sp = 0;
while (flag_sp == 0 ) & (ne  <= n-1)
    i1size=k;
    i2size=l;
    k=0;
    for j=1:1:i2size
        t1 = i2(j);
        tk=0;
        for i = 1:1: i1size
            t2 =i1(i);
            if a(t1,t2) ~= 0
                tk=tk+1 ;
            end
        end
```

```
        if tk > 0
            k = k + 1;
            reci1(k) = t1;
        end
    end

if k == 0
    disp('False - Not a (sp) matrix');
    return;
end

l=0;
for j=1:1:i2size
    t1 = i2(j);
    t=0 ;
    for i = 1:1:i1size
        t2 =i1(i);
        if a(t1,t2) == 0
            t=t+1 ;
        end
    end
    if  t == i1size
        l = l + 1;
        reci2(l) = t1;
    end
end
```

```
recr1size = k;
recr2size = l;
if recr1size == i2size
    m=0;
    for i = 1:1: recr1size
        t1=reci1(i);
        for j = 1:1:i2size
            t2 = i2(j);
            if t1 == t2
                m = m+1;
            end
        end
    end

    if m == i2size & recr2size == 0
            flag_sp = 1;
    else
        i1 = reci1;
        i2= reci2 ;
        ne=ne+1;
        recr1size = 0;
        recr2size = 0;
    end
else
    i1 = reci1;
    i2 = reci2 ;
    ne=ne+1;
```

```
    recr1size = 0;
    recr2size = 0;
end
end

if flag_sp == 1
    disp('True - a (sp) matrix')
else
    disp('False - Not a (sp) matrix')
end
```

P-6 : Matlab code for asymptotic behavior of null solution

Chapter 5 Example 5.1.1 is solved using this program.

```
%-----------------------------------------------------------------------
%%   System Considered : x(t+1) = Ax(t) + f(t, x(t)), t \in N_0
%%   N_0 = \{0, 1, 2,...\}
%%   A : Constant Matrix of order n x n
%%   f : Nonlinear function
%%   n : order of A
%%   x0 : Initial State - column vector of size n
%% issp() : Function checks whether the given matrix is (sp) matrix or not.
%%-------------------------------------------------------------------%
clc
clear all
A = input('Enter matrix A')
```

```
flag=issp(A)
if flag==0
    disp('Not a (sp) matrix')
    exit();
end
x0=input('Enter the initial state ')
n=input('Enter no.iterations to reach to null solution ')
%-------------------------------------------------------------------%
%Graph is drawn using the following data
%-------------------------------------------------------------------%
%a=[0 1 0 0;0 1/3 0 0;0 0 1/3 0;1 0 0 0];
%x0=[1 -1 1 -1]' ;
%n=12;
t=1;
for i=1:n
    x(:,t)=x0;
    t=t+1;
end
t=1;
for i= 1 :n
    t=t+1;
    f=1/i^2 * [5*x(1,i)*sin(x(1,i))/4  x(2,i)*cos(x(2,i))/4
    x(3,i)*sin(x(1,i))*cos(x(3,i))/4  x(4,i)*cos(2*x(2,i))/4]';
    x(:,t)= A*x(:,t-1)+ f;
end
x
%-------------------------------------------------------------------%
```

```
%Plot showing the asymptotic behavior of null solution
%-------------------------------------------------------------------%
t=1:1:n+1
plot(t,x)
title('Asymptotic behavior of State')
xlabel('Time (t) ')
ylabel('State (x)')
```

P-7 : Matlab Code for the Computation of asymptotic behavior of null solution

Chapter 5 Example 5.2.1 is solved using this program.

```
%%------------------------------------------------------------------%
%% Systems Considered :
%% Nonlinear system
%%   y(t+1) = A(t)y(t) + \sum_{r=t_0}^{t}B(t-r)y(r)+ f(t, y(t)), t \in N_0
%% and Linear System
%%   x(t+1) = A(t)x(t) + \sum_{r=t_0}^{t}B(t-r)x(r)
%%   A(t) : Sequence of real  n x n matrices defined in function A.m
%%   B(t) : Sequence of real n x m matrices defined in function B.m
%%   f    : nonlinear function
%%   x0 : Initial State - column vector of size n
%%   N : Final time
%%------------------------------------------------------------------%
%-------------------------------------------------------------------%
```

```
%Following program checks whether the linear system satisfies the
%Dichotomy condition or not and computes the solution of linear and
% nonlinear Volterra systems.
%--------------------------------------------------------------------%
clear all
clc
x0=[1 -1]';
x(:,1)=x0;
Phi(:,:,1)=eye(2);
nd=5;
prevx=x;
infy=5;
%------------------------------------------------------------------%
%following parts computes fundamental matrix Phi.
%------------------------------------------------------------------%
for t = 1 : 1: infy
    sumr=zeros(2);
    for r = 0:1:t-1
        sumr = sumr + B(t-1-r)*Phi(:,:,r+1);
    end
    Phi(:,:,t+1)=A(t)*Phi(:,:,t)+sumr;
end
Phi;
%------------------------------------------------------------------%
% By the observation of the values of Phi, we can see that matrix P
% can be defined as follows.
%------------------------------------------------------------------%
```

```
P=[1 0;0 0];
P1=eye(2)-P;
d=1;
for n=1:1:nd
    d=1;
    for m=1:1:n
        c1=Phi(:,:,n+1)*P*inv(Phi(:,:,m+1));
        con1(:,d)= norm(c1);
        d=d+1;
        if m==n
            minCon1(:,n)=max(con1);
        end
    end
end
minCon1
for m=1:1:nd
    d=1;
    for n=1:1:m
        con2(:,d)=norm(Phi(:,:,n+1)*P1*inv(Phi(:,:,m+1)));
        d=d+1;
        if n==m
            minCon2(:,m)=max(con2);
        end
    end
end
minCon2
pause
```

```
%-----------------------------------------------------------------------%
%In this example we can find M=1. Hence the given linear system
%satisfies the dichotomy condition with M=1. Folowing part computes
%the solution x of semi-linear system.
%-----------------------------------------------------------------------%
y0=x0;
y(:,1)=y0;
prevy=y;
d=1;
eps=0.004
for t=1:1:infy
    sumr=0;
    for r=0:1:t-1
        tempf=eps/(r+1)^5*[sin(prevy(:,r+1))  (1-cos(prevy(:,r+1)))]';
        f=[tempf(1) tempf(4)]';
        sumr=sumr + Phi(:,:,t-r)*f;
    end
    y(:,t+1)=Phi(:,:,t+1)*y0+sumr;
    d=d+1;
    prevy=y;
end
y
pause
%-----------------------------------------------------------------------%
% Following part shows the solution of the linear system
%-----------------------------------------------------------------------%
d=1;
```

```
for t=1:1:infy
    x(:,d+1)=Phi(:,:,t+1)*x0;
    d=d+1;
end
x
pause

%-----------------------------------------------------------------
%
function [ Mat_A ] = A(n)
    Mat_A=[1/exp(n) 0; 0 (n+1)];
return;
%-----------------------------------------------------------------
function [ Mat_B ] = B(n)
    Mat_B=[4^(-n-1) 0; 0 3^(-n-1)];
return;
%-----------------------------------------------------------------
```