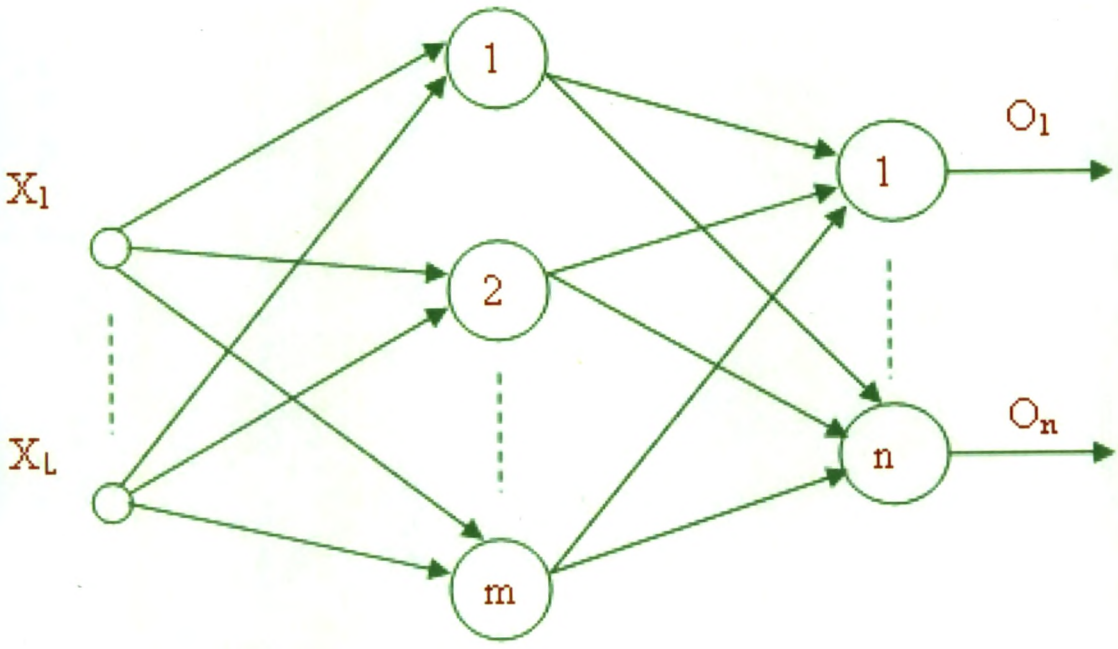
Appendix - A



APPENDIX - A

1. mt_nl_eqm.m (computed for Chapter 4.)----------

% % % % This program finds the equilibrium for the mixtank process

```
% syms x y alpha
% There are several ways to address the syms x y alpha
% There are several ways to address the output of solve.
% One is to use a two-output call [x,y] = solve(x^2*y^2, x-y/2-alpha)
% output of solve. One is to use a two-output call [x,y] = solve(x^2*y^2, x-y/2-alpha)
clc
Q1 = 10
Q2 = 20
C1 = 9
C2 = 18
syms V C
S = solve( V + Q1 + Q2 - sqrt(V/4)-3595, C + ((C1 - C)*Q1/V + (C2 - C)*Q2/V)-14)
% S = solve(Q1 + Q2 - sqrt(V/4), (C1 - C)*Q1/V + (C2 - C)*Q2/V)
S.V
S.C
```

% % % % The equilibrium is (3600, 15)----------

2. nnctrlc.m (computed for Chapter 4.)----------

```
% % % % This program demonstrates neural network controller for the
% % % % CONTINUOUS LINEARIZED system about the equilibrium (3600,15).
% % % % It is trained for the error of 50 in the first component and that of 5 units
% % % % in the 2nd component. To test the controller another neural network is
% % % % designed which verifies whether the NN controller will bring the
% % % % perturbed state to the equilibrium state.
% %  CONTINUOUS LINEARIZED SYSTEM FOR MIXTANK PROCESS -
% %  TWO NETWORKS DESIGNED 1. CONTROLLER 2. FINAL STATE.
% %
% %
% clear;
% Q10 = 10;
% Q20 = 20;
% C1 = 9;
% C2 = 18;
%
% V0 = 3600;
%
%
```

```
% Q0 = Q10 + Q20;
% C0 = (C1*Q10+C2*Q20)/Q0;
% tau = V0/Q0;
% disp(' Continuous version has evolution and control matrix as :');
% A = [-1/(2*tau) 0;0 -1/tau]
% B = [ 1 1; (C1-C0)/V0  (C2-C0)/V0]
% [M N] = size(B);

% disp(' From the perturbed state we want to reach to the desired state [3600 15] in 5
seconds');
% T = 5;
%
% syms('t');
% syms('s');
%
% disp(' The Gramian matrix is :');
% W =  int(expm(-A*t)*B*B'*expm(-A'*t),t,0,T);
% subs(W)
% V = inv(W);
% r = rank(V);
% if r == N
%    fprintf(' The system is controllable ');
% else
%    fprintf(' The system is not controllable ');
%    return;
% end
%
%
%
% X0 = [3600+100*rand(1)-50 15+10*rand(1)-5]';
% for i = 1:99
%    X0 = [X0  [3600+100*rand(1)-50 15+10*rand(1)-5]'];
% end
%
% dX1 = [3600 15]';
% X1 = dX1;
% for i = 1:99
%    X1 = [X1 dX1];
% end
%
%
% TU =  zeros(size(X1));
% for i = 1 :100
%    u = - B'*expm(-A'*t)*inv(W)*(X0(:,i) - expm(-A*T)*X1(:,i)) ;
%    U = subs(u,t,s);
%    s = 5;
```

```
%    TU(:,i) = eval(U);
%    syms('s');
%    CX1(:,i) = eval(expm(A*T)*X0(:,i) + int(expm(A*(T-s))*B*U,s,0,T));
% end
% disp(' The computed control is :');
% TU
% disp(' The computed state is :');
% CX1
% % % % % %
%
% % Design two NN one for controller and other for final state train them and verify
% % !!!!!!!!!!!!!!!!!!!!!!!!!!!!!
%
%
% % % %----------------NEURAL NETWORK IMPLEMENTATIONS----------------
% % % %--------1111111----------The data here is generated using the definition---------
% % % %-----------------------------of the controller for the time invarient system---------
%
% Pu = [X0;X1];
% Tu = TU/(max(max(abs(TU))));
% netu = newff([min(X0(1,:)) max(X0(1,:));min(X0(2,:)) max(X0(2,:));0 max(X1(1,:));0
max(X1(2,:))],[10 5 2],{'tansig' 'tansig' 'tansig'}, 'trainlm', 'learngdm','mse');
% netu.trainParam.epochs = 100000;
% netu.trainParam.goal = 0.0000000001;
% netu = train(netu,Pu,Tu);
%
% Yu = sim(netu,Pu);
% Yu = Yu*(max(max(abs(TU))))
%
%
%
% Px1 = [X0;Yu];
% Tx1 = X1/(max(max(abs(X1))));
% netx1 = newff([min(X0(1,:)) max(X0(1,:));min(X0(2,:)) max(X0(2,:));min(Yu(1,:))
max(Yu(1,:)); min(Yu(2,:)) max(Yu(2,:))],[10 5 2],{'tansig' 'tansig' 'tansig'}, 'trainlm',
'learngdm','mse');
% netx1.trainParam.epochs = 100000;
% netx1.trainParam.goal = 0.0000000001;
% netx1 = train(netx1,Px1,Tx1);
%
% Yx1 = sim(netx1,Px1);
% Yx1 = Yx1*(max(max(abs(X1))))
%
%
load('final_datac.mat');
ans = input( ' Validate the ANN ');
```

```
while ans == 'y'
   TestX0 = input(' Enter the initial state in neighborhood of [3600 15]');
   fprintf(' \n The control signal will be generated by the ANN to bring the given state to
[3600 15] \n\n');

   TestPu = [TestX0 3600 15]';
   fprintf(' The control vector is ');
   TestYu = sim(netu,TestPu)*(max(max(abs(TU))))


   disp(' Validation shows the final state as :');
   TestPx1 = [TestX0'; TestYu];
   TestX1 = sim(netx1,TestPx1)*(max(max(abs(X1))))
   ans = input( ' Validate the ANN ');
end
```

3. NNgDMT.m (computed for Chapter 5.)----------

```
% % % %This program implements the ANN controller for the discrete linearized
% % % % mixtank proces system.
% % % %The equilibrium of the mixtank process with the understated parameter
% % % %values
% % % % is calculated using mt_nl_eqm.m. The equilibrium is (3600, 15).


% clear
% Q10 = 10;
% Q20 = 20;
% C1 = 9;
% C2 = 18;
%
% V0 = 3600;
%
%
% Q0 = Q10 + Q20;
% C0 = (C1*Q10+C2*Q20)/Q0;
% tau = V0/Q0;
% %disp(' Continuous version has evolution and control matrix as :');
% A = [-1/(2*tau) 0;0 -1/tau];
% B = [ 1 1; (C1-C0)/V0  (C2-C0)/V0];
%
%
% [N M] = size(B);
% % disp(' Discrete version has evolution and control matrix as :');
% ST = 1;
```

```
% F = expm(A*ST);
% G = (F-eye(size(F)))*inv(A)*B;
%
% % disp(' The Controllability matrix is :');
% N1 = 2;
% Un = G;
% for i = 1:N1-1
%    Un = [F^i*G Un ];
% end
% Un;
% r = rank(Un);
%
% iUn = pinv(Un);
%
% %fprintf(' The Control signal');

% %----------------DATA generation----------------------------------------
% The data is randomly generated such that the initial state lies in the interval
{(3585,3625), (10,20)}
% The control signal is obtained using the definition, which becomes the output for the
training patterns
% for the ANN controller. 100 such patterns are generated.

% pat = 100;
% P = zeros(N,pat);
% Tu = zeros(N*N1,pat);
% X1 = [3600 15]';
% for p = 1:pat
%   X0 = [3600+50*rand(1)-25 15+10*rand(1)-5]' ;
% % X0 = [3580 14]';
%    P(:,p) = X0;
%    dU = iUn*(X1 - F^N1*X0)
%    Tu(:,p) = dU;
%    pdU = dU(1:M);
%    for j = 2:N1
%       pdU = [ pdU dU((j-1)*M+1:j*M)];
%    end
%    CX1 = zeros(N,N1);
%    for i = 1:N1
%       sum = zeros(N,1);
%       for k = 1:i
%          sum =  sum + Un(:,(k-1)*M+1:k*M)*pdU(:,k);
%
%       end
%       CX1(:,i) = sum + F^i*X0 ;
%    end
```

```
% %    CX1
% %    pause
% end
%
% %%%%%----------NN training----------
%
% maxT = max(max(abs(Tu)));
% T = Tu/maxT;
% [PN PM] = size(P);
% MM = [min(P(1,:)) max(P(1,:))];
% for i = 2 : PN
%    MM = [MM ; min(P(i,:)) max(P(i,:))];
% end
%
%
% netu = newff(MM,[10 20 N*N1],{'tansig' 'tansig' 'tansig'}, 'trainlm', 'learngdm','mse');
% netu.trainParam.epochs = 1*10^6;
% netu.trainParam.goal = 1/10^10;
% netu = train(netu,P,T);
%
% Y = sim(netu,P)*maxT
% Tu
load('NNgDMT_data.mat');

%%%%----------NN Validation ----------

ans = input( ' Validate the ANN ');
while ans == 'y'
   TestX0 = input(' Enter the initial state in neighborhood of [3600 15]');
   TestX0 = TestX0';
   mydU = zeros(N*N1,1);
   mynnCX1 = zeros(N,1);
   count = 1;
   while 1

      nndU = sim(netu,TestX0)*maxT;
      mydU = [ mydU ; nndU];
      dU = iUn*(X1 - F^N1*TestX0);

      nnpdU = nndU(1:M);
      for j = 2:N1
         nnpdU = [ nnpdU nndU((j-1)*M+1:j*M)];
      end
      nnCX1 = zeros(N,N1);
      for i = 1:N1
         sum = zeros(N,1);
```

```
      for k = 1:i
        sum =  sum + Un(:,(k-1)*M+1:k*M)*nnpdU(:,k);

      end
      nnCX1(:,i) = sum + F^i*TestX0 ;
    end
    nnCX1;
    mynnCX1 = [mynnCX1 nnCX1];
    if abs(nnCX1(:,N1) - X1) <0.1
      [mCX1 nCX1] = size(mynnCX1);
      [mdU ndU] = size(mydU);
      mydU(5:mdU)
      mynnCX1 = mynnCX1(:,2:nCX1)
      break
    else
      TestX0 = nnCX1(:,N1);
      count = count + 1;
      if count >10
        disp('\n The state diverges');
        nCX1 = 10;
        mynnCX1 = mynnCX1(:,2:nCX1)
        break
      end
    end
  end

  ans = input( ' Validate the ANN ');
end

% figure
% plot(1:N1,CX1(1,:),1:N1,CX1(2,:))
```

4. arb_u_NN.m (computed for Chapter 5.)----------

```
% % % %This program implements the ANN controller for the discrete linearized
% % % % mixtank proces system.
% % % % The program differs from the previous one in the way that the formula
% % % % for control is not used instead the training patterns by randomly
% % % % generating the control signal U, and its effect on the state is calculated
% % % % Input to the network is X0 and X1 and Output is U.
% % % %
% % % % This program illustrates that the mathematical form of dynamical
% % % % system is not required for training the Neural Network as controller.

clear
clc

% Q10 = 10;
% Q20 = 20;
% C1 = 9;
% C2 = 18;
% V0 = 3600;
%
%
% Q0 = Q10 + Q20;
% C0 = (C1*Q10+C2*Q20)/Q0;
% tau = V0/Q0;
%
% A = [-1/(2*tau) 0;0 -1/tau]
% B = [ 1 1; (C1-C0)/V0  (C2-C0)/V0]
%
% disp(' The Controllability matrix is :');
% C = [B A*B]
%
% disp(' The rank of controllability Grammian is :');
% z = rank(C)
% ST = .01;
% F = expm(A*ST);
% G = (F-eye(size(F)))*inv(F)*B
%
%
% F = exp(A*5)
% G = (F-eye(size(F)))*inv(F)*B
%
% [N M] = size(B)
% t = 1;
%
% disp(' Continuous version has evolution and control matrix as :');
```

```
% A
% B
% ST = .05;
%
% disp(' Discrete version has evolution and control matrix as :');
% F = expm(A*ST)
% G = (F-eye(size(F)))*pinv(A)*B
% Un = G;
% for i = 1 : N-1
%    Un = [Un F^i*G];
% end
% disp(' The rank of the controllability matrix is ');
% rank(Un)
% disp(' Hence the system is controllable ');
% Un = Un(:,1:rank(Un))
%
% NOS = 500;
% X0 = 10*rand(1)-5;
% for i = 1:N-1
%    X0 = [X0 100*rand(1)-50];
% end
% X0 = X0';
% for j = 1:NOS-1
%    rr = 100*rand(1)-50;
%    for i = 1:N-1
%       rr = [rr 100*rand(1)-50];
%    end
%    X0 = [X0 rr'];
% end
%
% X0
%
% U = 1000*rand(1)-500;
% for i = 1:N-1
%    U = [U 1000*rand(1)-500];
% end
% U = U';
% for j = 1:NOS-1
%    rr1 = 1000*rand(1)-500;
%    for i = 1:N-1
%       rr1 = [rr1 1000*rand(1)-500];
%    end
%    U = [U rr1'];
% end
% U
%
```

```
% X1 = zeros(N,NOS);
% for i = 1:NOS
%
%    X1(:,i) = F^N*X0(:,i) + Un*U(:,i);
% end
% X1
%
% % % %----------------------NEURAL NETWORK IMPLEMENTATIONS----------------
% % % %--------1111111-------The data here is generated using the definition-------------
% % % %--------------------------of the controller for the time invarient system-------------
%
% UU = U;
% T = UU/max(max(abs(UU)));
% P = [X0;X1];
% [NR NC] = size(P);
% MM = [min(P(1,:)) max(P(1,:))];
% for j = 2:NR
%    MM = [MM ; [min(P(j,:)) max(P(j,:))]];
% end
% net = newff(MM,[20 15 10 N],{ 'tansig' 'tansig' 'tansig' 'tansig'}, 'trainlm',
'learngdm','mse');
% net.trainParam.epochs = 100000;
% net.trainParam.goal = 0.00000001;
% net = train(net,P,T);
% Y = sim(net,P);
% Y = Y*(max(max(abs(UU))));
% N
% for i = 1:NOS
%     NNX1(:,i) = F^N*X0(:,i) + Un*Y(:,i);
% end
% [X0;X1;NNX1]
load 'NN_lin_mt.mat';
ans = input( ' Validate the ANN ');
while ans == 'y'
   TX0 = input(' Enter the initial state 2-D in ');
   TX1 = input(' Enter the final state 2-D in  ');
   TP = [TX0'; TX1'];
   CCU = inv(Un)*(TX1' - F^N*TX0')
   fprintf(' The control vector is ');
   TY2 = sim(net,TP);
   TY2 = TY2*(max(max(abs(UU))))
   nnX1 = F^N*TX0' + Un*TY2
   ans = input(' Validate again ');
end
```

5. example.m (computed for Chapter 5.)----------

```
% % % % data is generated using the definition for the state and the control
% % % % variables. The definitions are stored in the file data.m. The data is
% % % % imported in the variables x1, x2,x3 and u. Here we have taken the
% % % % feedback controller u = x1. Using these I/O pairs the feedforward Neural
% % % % Networks with the architecture 3,3,1 is designed and is trained. For the
% % % % complete data the network is validated.
%
% x1 = [.1500 -0.5775 0.4423 0.3071 0.1619 -0.0804 0.0058 0.0058 0.0002
-0.0000];
% x2 = [-0.2500 0.1687 -0.5288 0.5592 0.2213 0.1440 -0.0746 0.0060
0.0058 0.0002];
% x3 = [0.3500 0.0600 -0.6403 0.4916 0.3279 0.1447 -0.0804 0.0059
0.0058 0.0002];
% u = x1;
% datau = [x1;x2;x3];
% targetu = u;
%
% netu = newff([0 1;0 1;0 1],[10, 7, 1],{'tansig' 'tansig' 'tansig'}, 'traingd', 'learngd',
'mse');
% netu = init(netu);
% Pu = datau;
% Tu = targetu
% netu.trainParam.epochs = 1000000;
% netu.trainParam.goal = 0.00001;
% netu = train(netu,Pu,Tu);
%
% yu = sim(netu,Pu)
% % datax = [x1(1:8);x2(1:8);x3(1:8);u(1:8)]
% % targetx1 = [x1(2:9)];
% % targetx2 = [x2(2:9)];
% % targetx3= [x3(2:9)];
load('nnu_x1.mat');
ans = input( ' Validate the ANN ');
while ans == 'y'
   TestX0 = input(' Enter the initial state in neighborhood of origin');
   fprintf(' \n The control signal will be generated by the ANN to bring the given state to
origin \n\n');

   TestX0 = TestX0'
   TestXN = zeros(size(TestX0));
   fprintf(' The control vector is ');
   while 1
      TestY2 = sim(netu,TestX0)
      TestXN(1)= TestX0(2)-TestX0(3)+TestY2^2;
```

```
    TestXN(2)= 2*TestX0(1)-(1+0.5*TestX0(2))*TestY2;
    TestXN(3)= TestX0(1)*(TestX0(2)-TestX0(3))+TestY2;
    TestXN
    pause;
    if (abs(TestX0 - TestXN) > .001)
       TestX0 = TestXN;
    else
       break
    end
  end
end



%    disp(' Validation shows the final state as :');
%
%    nnX1 = F^2*TestX0' + Un*TestY2
%
  ans = input( ' Validate the ANN ');
```

6. my_sim_data.m (computed for Chapter 5.)----------

% % % % This program generates the data required for training the Neural
% % % % Network for the semilinear system. Then nonlinear feedback controller
% % % % u(k) = -sin(x1+x3) steers the state to the desired equilibrium zero.

```
% x1(1)= -.01;
% x2(1)= .5;
% x3(1)= -.4;

% % x1 = [.1500  -0.5775   0.4423   0.3071   0.1619  -0.0804   0.0058   0.0058
0.0002  -0.0000];
% % x2 = [-0.2500   0.1687  -0.5288   0.5592   0.2213   0.1440  -0.0746   0.0060
0.0058   0.0002];
% % x3 = [0.3500   0.0600  -0.6403   0.4916   0.3279   0.1447  -0.0804   0.0059
0.0058   0.0002];
% x1(1) = .15;
% x2(1) = -0.25;
% x3(1) = .35;
x1(1) = rand(1);
x2(1) = rand(1);
x3(1) = rand(1);
u(1) = -sin(x1(1)+x3(1));
```

```
for i = 1:7,
   x1(i+1)= u(i)-x2(i)*u(i)
   x2(i+1)= -x1(i)+0.5*x1(i)*x2(i)
   x3(i+1)= x1(i)-x2(i)-u(i)*u(i)
   u(i+1) = -sin(x1(i+1)*x3(i+1))
end
% subplot(2,2,1)
% plot(x1)
% subplot(2,2,2)
% plot(x2)
% subplot(2,2,3)
% plot(x3)
% subplot(2,2,4)
% plot(u)
% % X = zeros(3,5);
% % X(1,1) = rand(1);
% % X(2,1) = rand(1);
% % X(3,1) = rand(1);
% %
```

7. exampleu.m (computed for Chapter 5.)----------

```
% % % % The feedback ANN steering controller for the semilinear system.
% % % % The steering control as well as the state components are implemented
% % % % using ANN. Thus the use of ANN as controller in completely automated
% % % % system can be authorized.
clear
my_sim_data
datau = [x1;x2;x3];
targetu = u;
for i = 1:4
   my_sim_data
   datau = [datau [x1;x2;x3]];
   targetu = [targetu u];
end
netu = newff([0 1;0 1;0 1],[20, 15, 5, 1],{'tansig' 'tansig' 'tansig' 'tansig'}, 'traingd',
'learngd', 'mse');
netu = init(netu);
Pu = datau;
Tu = targetu
netu.trainParam.epochs = 100000;
netu.trainParam.goal = 0.000001;
netu = train(netu,Pu,Tu);
yu = sim(netu,Pu)
```

```
pause

% datax = [x1(1:8);x2(1:8);x3(1:8);u(1:8)]
%
% %----------targetx1 = [x1(2:9)];
% targetx1 = [x1(2:9)];
% netx1 = newff([0 1;0 1;0 1;0 1],[3,1],{'tansig' 'tansig'}, 'traingd', 'learngd', 'mse')
% netx1 = init(netx1);
% Px = datax;
% Tx1 = targetx1
% netx1.trainParam.epochs = 100000  ;
% netx1.trainParam.goal = 0.009;
% netx1 = train(netx1,Px,Tx1)
% yx1 = sim(netx1,Px)
%
% pause
%
% % ----------targetx2 = [x2(2:9)];
% targetx2 = [x2(2:9)];
% netx2 = newff([0 1;0 1;0 1;0 1],[3,1],{'tansig' 'tansig'}, 'traingd', 'learngd', 'mse')
% netx2 = init(netx2);
% Px = datax;
% Tx2 = targetx2
% netx2.trainParam.epochs = 100000  ;
% netx2.trainParam.goal = 0.009;
% netx2 = train(netx2,Px,Tx2)
% yx2 = sim(netx2,Px)
%
% pause
%
% % % ----------targetx3 = [x3(2:9)];
% targetx3 = [x3(2:9)];
% netx3 = newff([0 1;0 1;0 1;0 1],[3,1],{'tansig' 'tansig'}, 'traingd', 'learngd', 'mse')
% netx3 = init(netx3);
% Px = datax;
% Tx3 = targetx3
% netx3.trainParam.epochs = 100000  ;
% netx3.trainParam.goal = 0.009;
% netx3 = train(netx3,Px,Tx3)
% yx3 = sim(netx3,Px)
% pause
load('semi_controller.mat');

% %----------------validation
% checku = zeros(50);
% checkx1 = zeros(50);
```

```
% checkx2 = zeros(50);
% checkx3 = zeros(50);
% checku(1) = sim(netu,[x1(1);x2(1);x3(1)]);
% checkx1(1) = x1(1);
% checkx2(1) = x2(1);
% checkx3(1) = x3(1);
%
% for i = 2 : 10
%    checkx1(i) = sim(netx1,[checkx1(i-1);checkx2(i-1);checkx3(i-1);checku(i-1)]);
%    checkx2(i) = sim(netx2,[checkx1(i-1);checkx2(i-1);checkx3(i-1);checku(i-1)]);
%    checkx3(i) = sim(netx3,[checkx1(i-1);checkx2(i-1);checkx3(i-1);checku(i-1)]);
%    checku(i) = sim(netu,[checkx1(i);checkx2(i);checkx3(i)]);
% end
% M = 10
% checku = checku(1:7)
% checkx1 = checkx1(1:7)
% checkx2 = checkx2(1:7)
% checkx3 = checkx3(1:7)

disp(' Validating the network ');
X = input(' Enter the perturbed state (3-D Vector):' );
X = X'
i = 1;
while 1 == 1
   NNU = sim(netu,X)
   NNX1 = sim(netx1,[X;NNU]);
   NNX2 = sim(netx2,[X;NNU]);
   NNX3 = sim(netx3,[X;NNU]);

   X1 = [NNX1;NNX2;NNX3]
   if abs(X)<0.001
      break;
   else
      X = X1;
      i = i+1;
      pause;
   end
end
fprintf(' Converged in %d steps ',i);
```

8. NNgDMT_nl.m (computed for Chapter 5.)----------

% % % % The example constructed , where evolution matrix A is nonzero,
% % % % linear part of the system is controllable and the nonlinear
% % % % function in the system is Lipschitz.
% % % % The data for such system is generated by solving the coupled nonlinear
% % % % integral equations representing the state and control.
% % % % The ANN steering control takes the perturbed initial state to the
% % % % equilibrium.

```
% clear
% Q10 = 10;
% Q20 = 20;
% C1 = 9;
% C2 = 18;
%
% V0 = 3600;
%
%
% Q0 = Q10 + Q20;
% C0 = (C1*Q10+C2*Q20)/Q0;
% tau = V0/Q0;
% %disp(' Continuous version has evolution and control matrix as :');
% A = [-1/(2*tau) 0;0 -1/tau];
% B = [ 1 1; (C1-C0)/V0  (C2-C0)/V0];
%
%
% [N M] = size(B);
% % disp(' Discrete version has evolution and control matrix as :');
% ST = 1;
% F = expm(A*ST)
% G = (F-eye(size(F)))*inv(A)*B
% % N = 2;
% % F = [1 0 ; 0 1]
% % G = [1 1 ; 0 0]
% % disp(' The Controllability matrix is :');
% N1 = 2;
% Un = G;
% fUn = eye(N);
% for i = 1:N1-1
%    Un = [F^i*G Un ];
%    fUn = [F^i fUn];
% end
% Un;
%
```

```
% r = rank(Un)
% % fprintf(' the rank of controllability matrix is %d',r);
% % fprintf(' Hence the system is controllable in %d steps',r);
%
% % ------Checking the system----------------------------------------------
% % disp(' The initial state :')
% % X0 = [3580 14]';
% % disp(' The final state :')
% %
% % X1 = [ 3600 15]';
% % disp(' We want to reach final distribution ');
%
% iUn = pinv(Un);
%
% %fprintf(' The Control signal');
% %-----------------DATA generation-----------------------------------------
% pat = 10;
% P = zeros(N,pat);
% Tu = zeros(M*N1,pat);
% X1 = [3600 15]';
% for p = 1:pat
%  X0 = [3600+10*rand(1)-5 15+5*rand(1)-2.5]'    ;
% % %    X0 = [3590 15]';
%    xold = ones(N,N1);
%    P(:,p) = X0;
%    fsum = zeros(N,1);
%
%    iter = 1;
%    while 1
%       for k = 1:N1
%          fsum =  fsum + fUn(:,(k-1)*N+1:k*N)*(sin(xold(:,k))/10);
%       end
%       dU = iUn*(X1 - F^N1*X0 - fsum)
%       Tu(:,p) = dU;
%       pdU = dU(1:M);
%       for j = 2:N1
%          pdU = [ pdU dU((j-1)*M+1:j*M)];
%       end
%       CX1 = zeros(N,N1);
%       for i = 1:N1
%          sum = zeros(N,1);
%          for k = 1:i
%             sum =  sum + Un(:,(k-1)*M+1:k*M)*pdU(:,k) + fUn(:,(k-
1)*N+1:k*N)*sin(xold(:,k)/10);
%          end
%          CX1(:,i) = sum + F^i*X0 ;
```

```
%       end
%       iter
%       CX1
%
%       if abs(xold - CX1)<0.1
%          break;
%       else
%          xold = CX1;
%          iter = iter+1;
%       end
%    end
%    pause
% end
%
% %%%%%----------NN training----------
%
% % maxT = max(max(abs(Tu)));
% % T = Tu/maxT;
% % [PN PM] = size(P);
% % MM = [min(P(1,:)) max(P(1,:))];
% % for i = 2 : PN
% %    MM = [MM ; min(P(i,:)) max(P(i,:))];
% % end
% %
% %
% % net = newff(MM,[15 N*N1],{'tansig' 'tansig'}, 'trainlm', 'learngdm','mse');
% % net.trainParam.epochs = 1*10^6;
% % net.trainParam.goal = 1/10^10;
% % net = train(net,P,T);
% %
% % Y = sim(net,P)*maxT;
% %
% % %%%%%----------NN Validation ----------
% %
% % ans = input( ' Validate the ANN ');
% % while ans == 'y'
% %    TestX0 = input(' Enter the initial state in neighborhood of [3600 15]');
% %
% %    nndU = sim(net,TestX0')*maxT;
% %    nnpdU = nndU(1:M);
% %    for j = 2:N1
% %       nnpdU = [ nnpdU nndU((j-1)*M+1:j*M)];
% %    end
% %    nnCX1 = zeros(N,N1);
% %    for i = 1:N1
% %       sum = zeros(N,1);
```

```
% %       for k = 1:i
% %          sum =  sum + Un(:,(k-1)*M+1:k*M)*nnpdU(:,k);
% %
% %       end
% %       nnCX1(:,i) = sum + F^i*TestX0' ;
% %    end
% %    nnCX1
% %    ans = input( ' Validate the ANN ');
% % end
% %
% %
% %
% %
% %
% %
% %
% %
% % %  figure
% % %  plot(1:N1,CX1(1,:),1:N1,CX1(2,:))
```

9. NNZONAL.m (computed for Chapter 6.)----------

```
% % % %This program is for designing of the ANN steering control for the 2 zone
% % % % control. The system is approximated on the five dimensional space.
% % % % The desired temperature profile is y(x) = x (1-x^2).
% % % % Sampling rate is 0.1 sec
% % % % The system is brought to the desired temperature in seven time steps
% % % % That is 0.7 seconds.

%
% clear
% alpha = 0.5;
% N = 5;
% p = 2;
% x = [.388 .733]'
% lambda = zeros(N);
% for i = 1 : N
%    lambda(i) = - alpha*i^2*pi^2;
% end
% A = zeros(N,N);
% for i = 1:N
%    A(i,i) = lambda(i);
% end
% A
```

```
% B = zeros(N,p);
% for i = 1:N
%    for j = 1:p
%       B(i,j) = 2^1.5/(i*pi)*sin(i*pi/10)*sin(i*pi*x(j));
%    end
% end
% B
%
% [N M] = size(B);
% % disp(' Discrete version has evolution and control matrix as :');
% ST = .1;
% F = expm(A*ST);
% G = (F-eye(size(F)))*inv(A)*B;
%
% % disp(' The Controllability matrix is :');
% N1 = 7;
% Un = G;
% for i = 1:N1-1
%    Un = [F^i*G Un ];
% end
% Un;
% r = rank(Un);
%
%
% iUn = pinv(Un);
%
%
% % % % ----------DATA generation----------
% pat = 100;
% P = zeros(N,pat);
% Tu = zeros(M*N1,pat);
% X1 = [ 0.3870  -0.0484  0.0143   -0.0060  0.0031]';
% for p = 1:pat
%    X0 = [rand(1) rand(1) rand(1) rand(1) rand(1)]'   ;
%    P(:,p) = X0;
%    dU = iUn*(X1 - F^N1*X0);
%    Tu(:,p) = dU;
%    pdU = dU(1:M);
%    for j = 2:N1
%       pdU = [ pdU dU((j-1)*M+1:j*M)];
%    end
%    CX1 = zeros(N,N1);
%    for i = 1:N1
%       sum = zeros(N,1);
%       for k = 1:i
%          sum =  sum + Un(:,(k-1)*M+1:k*M)*pdU(:,k);
```

```
%
%       end
%       CX1(:,i) = sum + F^i*X0 ;
%    end
% %    CX1
% %    pause
% end
%
% % % %----------------------------------NN training----------------------------------------
%
% maxT = max(max(abs(Tu)));
% T = Tu/maxT;
% [PN PM] = size(P);
% MM = [min(P(1,:)) max(P(1,:))];
% for i = 2 : PN
%    MM = [MM ; min(P(i,:)) max(P(i,:))];
% end
%
%
% netu = newff(MM,[30 M*N1],{'tansig' 'tansig' 'tansig' 'tansig'}, 'trainlm',
'learngdm','mse');
% netu.trainParam.epochs = 1*10^6;
% netu.trainParam.goal = 1/10^10;
% netu = train(netu,P,T);
%
% Y = sim(netu,P)*maxT
% Tu
load('NNZONALdata.mat');

% % % %----------------------------------NN Validation ----------------------------------------

ans = input( ' Validate the ANN ');
while ans == 'y'
  TestX0 = input(' Enter the initial state in neighborhood of [ 0.3870  -0.0484  0.0143   -
0.0060  0.0031]');

  nndU = sim(netu,TestX0')*maxT
  dU = iUn*(X1 - F^N1*TestX0')
  pause
  nnpdU = nndU(1:M);
  for j = 2:N1
    nnpdU = [ nnpdU nndU((j-1)*M+1:j*M)];
  end
  nnCX1 = zeros(N,N1);
  for i = 1:N1
    sum = zeros(N,1);
```

```
    for k = 1:i
      sum =  sum + Un(:,(k-1)*M+1:k*M)*nnpdU(:,k);

    end
    nnCX1(:,i) = sum + F^i*TestX0' ;
  end
  nnCX1(:,N1)

  X = linspace(0,1,100);
  nnZd = zeros(N1,length(X));
  for k = 1:N1
    for i = 1 : length(X)
      for j = 1 : N
        nnZd(k,i) = nnZd(k,i)+nnCX1(j,k)*sin(j*pi*X(i));
      end
    end
  end
  Z = zeros(1,length(X));
  for i = 1:length(X)
    Z(i) = X(i)*(1-X(i)^2);
  end
  figure
  plot(X,nnZd(N1,:),X,Z,'*')
  figure
  surf(nnZd)
  ans = input( ' Validate the ANN ');
end
```

10. NNgDZONAL_nl.m (computed for Chapter 6.)----------

```
% % % %This program is for designing of the ANN steering control for the 2 zone
% % % % control. The system is approximated on the five dimensional space.
% % % % The desired temperature profile is y(x) = x (1-x^2). The nonlinear
% % % % function in the system is sin(y(x))/6.
% % % % Sampling rate is 0.2 sec
% % % % The system is brought to the desired temperature in five time steps
% % % % That is, 1 second.
% % % % The coupled equations for the state and control takes 4 iterations


%
% clear
% alpha = 0.5;
% N = 5;
```

```
% p = 2;
% x = [.388 .733]'
% lambda = zeros(N);
% for i = 1 : N
%    lambda(i) = - alpha*i^2*pi^2;
% end
% A = zeros(N,N);
% for i = 1:N
%    A(i,i) = lambda(i);
% end
% A
% B = zeros(N,p);
% for i = 1:N
%    for j = 1:p
%       B(i,j) = 2^1.5/(i*pi)*sin(i*pi/10)*sin(i*pi*x(j));
%    end
% end
% B
%

% [N M] = size(B);
% % disp(' Discrete version has evolution and control matrix as :');
% ST = .2;
% F = expm(A*ST);
% G = (F-eye(size(F)))*inv(A)*B;
%
% % disp(' The Controllability matrix is :');
% N1 = 5;
% Un = G;
% fUn = eye(N);
% for i = 1:N1-1
%    Un = [F^i*G Un ];
%    fUn = [F^i fUn];
% end
% Un;
%
% r = rank(Un);

% iUn = pinv(Un);
%
% %fprintf(' The Control signal');

% % % %----------DATA generation----------
% pat = 10;
% P = zeros(N,pat);
% Tu = zeros(4*M*N1,pat);
```

```
% %X1 = [3600 15]';
% X1 = [ 0.3870  -0.0484  0.0143   -0.0060  0.0031]';
% for p = 1:pat
%   X0 = [rand(1) rand(1) rand(1) rand(1) rand(1)]'   ;
% %   X0 = [3600+200*rand(1)-100 15+20*rand(1)-10]'   ;
%    xold = ones(N,N1);
%    P(:,p) = X0;
%    T = 0;
%    iter = 1;
%    while 1
%       close all
%       fsum = zeros(N,1);
%       for k = 1:N1
%          fsum =  fsum + fUn(:,(k-1)*N+1:k*N)*(sin(xold(:,k)/6));
%       end
%       dU = iUn*(X1 - F^N1*X0 - fsum);
%       T = [T;dU];
%       pdU = dU(1:M);
%       for j = 2:N1
%          pdU = [ pdU dU((j-1)*M+1:j*M)];
%       end
%       CX1 = zeros(N,N1);
%       for i = 1:N1
%          sum = zeros(N,1);
%          for k = 1:i
%             sum =  sum + Un(:,(k-1)*M+1:k*M)*pdU(:,k) + fUn(:,(k-
1)*N+1:k*N)*sin(xold(:,k)/6);
%          end
%          CX1(:,i) = sum + F^i*X0;
%       end
% %       iter
% %       CX1(:,N1)
%       X = linspace(0,1,100);
%       Zd = zeros(N1,length(X));
%       for k = 1:N1
%          for i = 1 : length(X)
%             for j = 1 : N
%                Zd(k,i) = Zd(k,i)+CX1(j,k)*sin(j*pi*X(i));
%             end
%          end
%
%       end
%       Z = zeros(1,length(X));
%       for i = 1:length(X)
%          Z(i) = X(i)*(1-X(i)^2);
%       end
```

```
% %       figure
% %       plot(X,Zd(N1,:),X,Z,'*')
%
%       if abs(xold - CX1)<0.01
%          Tu(:,p) = T(2:41);
%          break;
%       else
%          xold = CX1;
%          iter = iter+1;
%       end
%    end
%  % pause
% end
%
% % % %----------------------------------------NN training----------------------------------------
%
% maxT = max(max(abs(Tu)));
% T = Tu/maxT;
% [PN PM] = size(P);
% MM = [min(P(1,:)) max(P(1,:))];
% for i = 2 : PN
%    MM = [MM ; min(P(i,:)) max(P(i,:))];
% end
%
%
% net = newff(MM,[10 40],{'tansig' 'tansig'}, 'trainlm', 'learngdm','mse');
% net.trainParam.epochs = 1*10^6;
% net.trainParam.goal = 1/10^10;
% net = train(net,P,T);
%
% Y = sim(net,P)*maxT;
load('NNZONALdataNL.mat');
%
% % % %----------------------------------NN Validation ----------------------------------------

ans = input( ' Validate the ANN ');
while ans == 'y'
   TestX0 = input(' Enter the initial state ');
   xold = ones(N,N1);
   nnCX1 = zeros(N,N1);
   X = linspace(0,1,100);
   mysurf = zeros(4,length(X));
   nndU = sim(net,TestX0')*maxT
   for kk = 1:4
      nnpdU = nndU(kk*10-9:kk*10);
      nnpdU = nndU(1:M);
```

```
    for j = 2:N1
      nnpdU = [ nnpdU nndU((j-1)*M+1:j*M)];
    end

    for i = 1:N1
      sum = zeros(N,1);
      for k = 1:i
        sum =  sum + Un(:,(k-1)*M+1:k*M)*pdU(:,k) + fUn(:,(k-
1)*N+1:k*N)*sin(xold(:,k)/6);
      end
      nnCX1(:,i) = sum + F^i*X0;
    end
%     mysurf(k) =  nnCX1(:,N1);
    for i = 1 : length(X)
      for j = 1 : N
        mysurf(kk,i) = mysurf(kk,i)+nnCX1(j,kk)*sin(j*pi*X(i));
      end
    end
    xold = nnCX1
  end
  mysurf
  nnCX1(:,N1)


  nnZd = zeros(N1,length(X));
  k = N1;
    for i = 1 : length(X)
      for j = 1 : N
        nnZd(k,i) = nnZd(k,i)+nnCX1(j,k)*sin(j*pi*X(i));
      end
    end
    Z = zeros(1,length(X));
  for i = 1:length(X)
    Z(i) = X(i)*(1-X(i)^2);
  end
  figure
  plot(X,nnZd(N1,:),X,Z,'*')
  figure
  surf(mysurf)
  ans = input( ' Validate the ANN ');
end


% figure
% plot(1:N1,CX1(1,:),1:N1,CX1(2,:))
```

11. NNgPARABOLIC_b.m (computed for Chapter 7.)----------

```
% % % % This program illustrates the use of Neural Network as the BOUNDARY
% % % % steering control to the linear part of thermal system. It is spatially
% % % % discretized giving 5-dimensional system. Since the system is tridiagonal,
% % % % the calculation for the Grammian matrix for the higher dimensional
% % % %  system is complicated.
% % % %  The desired final temperature distribution is y(x) = x(1-x).

% close all
% clear
% clc
% n = 4;
% s = .5;
% z = .0118;
% % % %----------Continuous version of the dynamical system----------
% disp(' Continuous version has evolution and control matrix as :');
%
% A = [-(s+z)      s      0     0     0 ;
%       s      -(2*s+z)  s     0     0 ;
%       0        s    -(2*s+z)  s     0 ;
%       0        0      s   -(2*s+z)  s ;
%       0        0      0     s  -(2*s+z);]
% B = [0 0 0 0 s]'
%
%
% [N M] = size(B);
% % disp(' Discrete version has evolution and control matrix as :');
% ST = 1;
% F = expm(A*ST);
% G = (F-eye(size(F)))*inv(A)*B;
%
% % disp(' The Controllability matrix is :');
% N1 = 10;
% Un = G;
% for i = 1:N1-1
%    Un = [F^i*G Un ];
% end
% Un;
% r = rank(Un);
%
%
% iUn = pinv(Un);
%
%
% % % % ----------DATA generation----------
```

```
% pat = 200;
% P = zeros(N,pat);
% Tu = zeros(M*N1,pat);
% X1 = [0    0.1875    0.2500    0.1875         0 ]';
%
% for p = 1:pat
%    X0 = [0 rand(1) rand(1) rand(1) 0]'    ;
%    P(:,p) = X0;
%    dU = iUn*(X1 - F^N1*X0);
%    Tu(:,p) = dU;
%    pdU = dU(1:M);
%    for j = 2:N1
%       pdU = [ pdU dU((j-1)*M+1:j*M)];
%    end
%    CX1 = zeros(N,N1);
%    for i = 1:N1
%       sum = zeros(N,1);
%       for k = 1:i
%          sum =  sum + Un(:,(k-1)*M+1:k*M)*pdU(:,k);
%
%       end
%       CX1(:,i) = sum + F^i*X0 ;
%    end
% %    CX1
% %   pause
% end
%
% % % %----------------------------------NN training----------------------------------
%
% maxT = max(max(abs(Tu)));
% T = Tu/maxT;
% [PN PM] = size(P);
% MM = [min(P(1,:)) max(P(1,:))];
% for i = 2 : PN
%    MM = [MM ; min(P(i,:)) max(P(i,:))];
% end
%
%
% netu = newff(MM,[30 M*N1],{'tansig' 'tansig' 'tansig' 'tansig'}, 'trainlm',
'learngdm','mse');
% netu.trainParam.epochs = 1*10^6;
% netu.trainParam.goal = 1/10^10;
% netu = train(netu,P,T);
%
% Y = sim(netu,P)*maxT
% Tu
```

```
load('NNgPARADdata.mat');
% % % %----------------------------------NN Validation ----------------------------------

ans = input( ' Validate the ANN ');
X = linspace(0,1,100);
Z = zeros(1,length(X));
for i = 1:length(X)
  Z(i) = X(i)*(1-X(i));
end
while ans == 'y'
  TestX0 = input(' Enter the initial state in neighborhood of [0 0.1875 0.2500 0.1875
0]');

  nndU = sim(netu,TestX0')*maxT
  dU = iUn*(X1 - F^N1*TestX0')
  pause
  nnpdU = nndU(1:M);
  for j = 2:N1
    nnpdU = [ nnpdU nndU((j-1)*M+1:j*M)];
  end
  nnCX1 = zeros(N,N1);
  for i = 1:N1
    sum = zeros(N,1);
    for k = 1:i
      sum =  sum + Un(:,(k-1)*M+1:k*M)*nnpdU(:,k);

    end
    nnCX1(:,i) = sum + F^i*TestX0' ;
  end
  nnCX1(:,N1)
  iy = interp(interp(TestX0',5,2,.2),5,2,.2);
  fy = interp(interp(nnCX1(:,N1),5,2,.2),5,2,.2);
  plot(X,iy(1:100),X,Z,X,fy(1:100),'+')
  ans = input( ' Validate the ANN ');

end

% figure
% plot(1:N1,CX1(1,:),1:N1,CX1(2,:))
```

12. NNgPARABOLIC_d.m (computed for Chapter 7.)----------

```
% % % % This program illustrates the use of Neural Network as the DISTRIBUTED
% % % % steering control to the linear part of thermal system. It is spatially
% % % % discretized giving 5-dimensional system. Since the system is tridiagonal,
% % % % the calculation for the Grammian matrix for the higher dimensional
% % % %  system is complicated.
% % % %  The desired final temperature distribution is y(x) = x(1-x).

% close all
% clear
% clc
% n = 4;
% s = .5;
% z = .0118;
% % % %----------Continuous version of the dynamical system----------
% disp(' Continuous version has evolution and control matrix as :');
%
% A = [-(s+z)      s      0     0     0 ;
%       s      -(2*s+z)  s     0     0 ;
%       0        s    -(2*s+z)  s     0 ;
%       0        0      s   -(2*s+z)  s ;
%       0        0      0     s  -(2*s+z);]
% B = [z z z z z]'
%
%
% [N M] = size(B);
% % disp(' Discrete version has evolution and control matrix as :');
% ST = 1;
% F = expm(A*ST);
% G = (F-eye(size(F)))*inv(A)*B;
%
% % disp(' The Controllability matrix is :');
% N1 = 10;
% Un = G;
% for i = 1:N1-1
%    Un = [F^i*G Un ];
% end
% Un;
% r = rank(Un);
% % fprintf(' the rank of controllability matrix is %d',r);
% % fprintf(' Hence the system is controllable in %d steps',r);
%
% % ------Checking the system----------
% % disp(' The initial state :')
% % X0 = [3580 14]';
```

```
% % disp(' The final state :')
% %
% % X1 = [ 3600 15]';
% % disp(' We want to reach final distribution ');
%
% iUn = pinv(Un);
%
% %fprintf(' The Control signal');
% %----------------DATA generation----------------------------------------
% pat = 200;
% P = zeros(N,pat);
% Tu = zeros(M*N1,pat);
% X1 = [0    0.1875    0.2500    0.1875         0 ]';
%
% for p = 1:pat
%    X0 = [0 rand(1) rand(1) rand(1) 0]'    ;
%    P(:,p) = X0;
%    dU = iUn*(X1 - F^N1*X0);
%    Tu(:,p) = dU;
%    pdU = dU(1:M);
%    for j = 2:N1
%       pdU = [ pdU dU((j-1)*M+1:j*M)];
%    end
%    CX1 = zeros(N,N1);
%    for i = 1:N1
%       sum = zeros(N,1);
%       for k = 1:i
%          sum =  sum + Un(:,(k-1)*M+1:k*M)*pdU(:,k);
%
%       end
%       CX1(:,i) = sum + F^i*X0 ;
%    end
% %    CX1
% %   pause
% end
%
% % % %----------------------------------NN training-----------------------------------
%
% maxT = max(max(abs(Tu)));
% T = Tu/maxT;
% [PN PM] = size(P);
% MM = [min(P(1,:)) max(P(1,:))];
% for i = 2 : PN
%    MM = [MM ; min(P(i,:)) max(P(i,:))];
% end
%
```

```
%
% netu = newff(MM,[30 M*N1],{'tansig' 'tansig' 'tansig' 'tansig'}, 'trainlm',
'learngdm','mse');
% netu.trainParam.epochs = 1*10^6;
% netu.trainParam.goal = 1/10^10;
% netu = train(netu,P,T);
%
% Y = sim(netu,P)*maxT
% Tu
load('NNgPARADdata.mat');
% % % %----------------------------------NN Validation ----------------------------------
ans = input( ' Validate the ANN ');
X = linspace(0,1,100);
Z = zeros(1,length(X));
for i = 1:length(X)
   Z(i) = X(i)*(1-X(i));
end
while ans == 'y'
   TestX0 = input(' Enter the initial state in neighborhood of [0 0.1875 0.2500 0.1875
0]');

   nndU = sim(netu,TestX0')*maxT
   dU = iUn*(X1 - F^N1*TestX0')
   pause
   nnpdU = nndU(1:M);
   for j = 2:N1
      nnpdU = [ nnpdU nndU((j-1)*M+1:j*M)];
   end
   nnCX1 = zeros(N,N1);
   for i = 1:N1
      sum = zeros(N,1);
      for k = 1:i
         sum =  sum + Un(:,(k-1)*M+1:k*M)*nnpdU(:,k);
            end
      nnCX1(:,i) = sum + F^i*TestX0' ;
   end
   nnCX1(:,N1)
   iy = interp(interp(TestX0',5,2,.2),5,2,.2);
   fy = interp(interp(nnCX1(:,N1),5,2,.2),5,2,.2);
   plot(X,iy(1:100),X,Z,X,fy(1:100),'+')
   ans = input( ' Validate the ANN ');
end
```

13. NNgPARABOLIC_nlb.m (computed for Chapter 7.)----------

```
% % % % This program illustrates the use of Neural Network as the BOUNDARY
% % % % steering control to the semilinear thermal system. It is spatially
% % % % discretized giving 5-dimensional system.
% % % %  The desired final temperature distribution is y(x) = x(1-x).
% % % %  The coupled equation converges in 8 iterations and in 7 time steps each
% % % % of 0.5 seconds

% close all
% clear
% clc
% n = 4;
% s = .5;
% z = .0118;
% % % %----------Continuous version of the dynamical system----------
% disp(' Continuous version has evolution and control matrix as :');
%
% A = [-(s+z)      s      0     0     0 ;
%       s      -(2*s+z)  s     0     0 ;
%       0        s    -(2*s+z)  s     0 ;
%       0        0      s   -(2*s+z)  s ;
%       0        0      0     s  -(2*s+z);]
% B = [0 0 0 0 s]'
%
%
% [N M] = size(B);
% % disp(' Discrete version has evolution and control matrix as :');
% ST = 1;
% F = expm(A*ST);
% G = (F-eye(size(F)))*inv(A)*B;
%
% % disp(' The Controllability matrix is :');
% N1 = 7;
% Un = G;
% fUn = eye(N);
% for i = 1:N1-1
%    Un = [F^i*G Un ];
%    fUn = [F^i fUn];
% end
% Un;
%
% r = rank(Un);
%
% iUn = pinv(Un);
%
```

```
% %fprintf(' The Control signal');
% % % % ----------DATA generation----------
% pat = 10;
% P = zeros(N,pat);
% Tu = zeros(8*M*N1,pat);
% X1 = [0   0.1875   0.2500   0.1875        0 ]';
% for p = 1:pat
%    X0 = [0 rand(1) rand(1) rand(1) 0]'   ;
%    xold = zeros(N,N1);
%    P(:,p) = X0;
%    T = 0;
%    iter = 1;
%    while 1
%       fsum = zeros(N,1);
%       for k = 1:N1
%          fsum =  fsum + fUn(:,(k-1)*N+1:k*N)*(sin(xold(:,k)/6));
%       end
%       dU = iUn*(X1 - F^N1*X0 - fsum);
%       T = [T;dU];
%       pdU = dU(1:M);
%       for j = 2:N1
%          pdU = [ pdU dU((j-1)*M+1:j*M)];
%       end
%       pdU;
%       CX1 = zeros(N,N1);
%       for i = 1:N1
%          sum = zeros(N,1);
%          for k = 1:i
%             sum =  sum + Un(:,(k-1)*M+1:k*M)*pdU(:,k) + fUn(:,(k-
1)*N+1:k*N)*sin(xold(:,k)/6);
%          end
%          CX1(:,i) = sum + F^i*X0;
%       end
%
%
%       if abs(xold - CX1)<0.001
%         Tu(:,p) = T(2:57);
%          break;
%       else
%          xold = CX1;
%          iter = iter+1;
%       end
%    end
% % %       iter
% % %       CX1(:,N1)
% % %       pause
```

```
% %
% %       X = linspace(0,1,100);
% %       Z = zeros(1,length(X));
% %       for i = 1:length(X)
% %          Z(i) = X(i)*(1-X(i));
% %       end
% %       CX1(:,N1)
% %       y = interp(interp(CX1(:,N1),5,2,.2),5,2,.2);
% %       figure
% %       plot(X,Z,X,y(1:100),'+')
% %       pause
% end
%
% % % %----------------------------------NN training-----------------------------------------
%
% maxT = max(max(abs(Tu)));
% T = Tu/maxT;
% [PN PM] = size(P);
% MM = [min(P(1,:)) max(P(1,:))];
% for i = 2 : PN
%    MM = [MM ; min(P(i,:)) max(P(i,:))];
% end
%
%
% net = newff(MM,[10 56],{'tansig' 'tansig'}, 'trainlm', 'learngdm','mse');
% net.trainParam.epochs = 1*10^6;
% net.trainParam.goal = 1/10^10;
% net = train(net,P,T);
%
% Y = sim(net,P)*maxT;
load('NNPARABdataNL.mat');
%
% % % %----------------------------------NN Validation ----------------------------------------

ans = input( ' Validate the ANN ');
while ans == 'y'
   TestX0 = input(' Enter the initial state ');
   xold = ones(N,N1);
   nnCX1 = zeros(N,N1);
   mysurf = zeros(N,4);
   nndU = sim(net,TestX0')*maxT
   for kk = 1:8
      nnpdU = nndU(kk*(N1-1)+1:kk*N1);
      nnpdU = nndU(1:M);
      for j = 2:N1
         nnpdU = [ nnpdU nndU((j-1)*M+1:j*M)];
```

```
    end

    for i = 1:N1
      sum = zeros(N,1);
      for k = 1:i
        sum =  sum + Un(:,(k-1)*M+1:k*M)*pdU(:,k) + fUn(:,(k-
1)*N+1:k*N)*sin(xold(:,k)/6);
      end
      nnCX1(:,i) = sum + F^i*X0;
    end
    mysurf(:,kk) =  nnCX1(:,N1);
    xold = nnCX1
  end
  mysurf
  nnCX1(:,N1)
  X = linspace(0,1,100);
  nnZd = zeros(N1,length(X));
  nnyi = interp(interp(TestX0',5,2,.2),5,2,.2);
  nnyf = interp(interp(nnCX1(:,N1),5,2,.2),5,2,.2);
  Z = zeros(1,length(X));
  for i = 1:length(X)
    Z(i) = X(i)*(1-X(i));
  end
  figure
  plot(X,nnyi(1:100), X, nnyf(1:100),X,Z,'*')
  figure
  surf(mysurf)
  ans = input( ' Validate the ANN ');
end

%
% % figure
% % plot(1:N1,CX1(1,:),1:N1,CX1(2,:))
```

14. NNgPARABOLIC_nld.m (computed for Chapter 7.)----------

```
% % % % This program illustrates the use of Neural Network as the DISTRIBUTED
% % % % steering control to the semilinear thermal system. It is spatially
% % % % discretized giving 5-dimensional system.
% % % %  The desired final temperature distribution is y(x) = x(1-x).
% % % %  The coupled equation converges in 8 iterations and in 7 time steps each
% % % % of 1 seconds

close all
clear
clc
n = 4;
s = .5;
z = .0118;
% % % %----------------Continuous version of the dynamical system----------
disp(' Continuous version has evolution and control matrix as :');

A = [-(s+z)      s      0     0     0 ;
      s      -(2*s+z)  s     0     0 ;
      0        s    -(2*s+z)  s     0 ;
      0        0      s   -(2*s+z)  s ;
       0       0      0     s  -(2*s+z);]
B = [z z z z z]'


[N M] = size(B);

ST = 1;
F = expm(A*ST);
G = (F-eye(size(F)))*inv(A)*B;

N1 = 7;
Un = G;
fUn = eye(N);
for i = 1:N1-1
   Un = [F^i*G Un ];
   fUn = [F^i fUn];
end
Un;

r = rank(Un);

iUn = pinv(Un);

pat = 10;
```

```
P = zeros(N,pat);
Tu = zeros(12*M*N1,pat);
X1 = [0   0.1875   0.2500   0.1875        0 ]';
for p = 1:pat
   X0 = [0 rand(1) rand(1) rand(1) 0]'   ;
   xold = zeros(N,N1);
   P(:,p) = X0;
   T = 0;
   iter = 1;
   while 1
      fsum = zeros(N,1);
      for k = 1:N1
         fsum =  fsum + fUn(:,(k-1)*N+1:k*N)*(sin(xold(:,k)/6));
      end
      dU = iUn*(X1 - F^N1*X0 - fsum);
      T = [T;dU];
      pdU = dU(1:M);
      for j = 2:N1
         pdU = [ pdU dU((j-1)*M+1:j*M)];
      end
      pdU;
      CX1 = zeros(N,N1);
      for i = 1:N1
         sum = zeros(N,1);
         for k = 1:i
            sum =  sum + Un(:,(k-1)*M+1:k*M)*pdU(:,k) + fUn(:,(k-
1)*N+1:k*N)*sin(xold(:,k)/6);
         end
         CX1(:,i) = sum + F^i*X0;
      end


      if abs(xold - CX1)<0.001
         Tu(:,p) = T(2:85);
         break;
      else
         xold = CX1;
         iter = iter+1;
      end
   end
      iter
      CX1(:,N1)
      pause
%
%       X = linspace(0,1,100);
%       Z = zeros(1,length(X));
```

```
%       for i = 1:length(X)
%          Z(i) = X(i)*(1-X(i));
%       end
%       CX1(:,N1)
%       y = interp(interp(CX1(:,N1),5,2,.2),5,2,.2);
%       figure
%       plot(X,Z,X,y(1:100),'+')
%       pause
end

% % % %----------------------------------NN training----------------------------------------
%
maxT = max(max(abs(Tu)));
T = Tu/maxT;
[PN PM] = size(P);
MM = [min(P(1,:)) max(P(1,:))];
for i = 2 : PN
   MM = [MM ; min(P(i,:)) max(P(i,:))];
end


net = newff(MM,[10 84],{'tansig' 'tansig'}, 'trainlm', 'learngdm','mse');
net.trainParam.epochs = 1*10^6;
net.trainParam.goal = 1/10^10;
net = train(net,P,T);

Y = sim(net,P)*maxT;
% load('NNPARADdataNL.mat');
% %
% % % %----------------------------------NN Validation ----------------------------------------
%
% ans = input( ' Validate the ANN ');
% while ans == 'y'
%    TestX0 = input(' Enter the initial state ');
%    xold = ones(N,N1);
%    nnCX1 = zeros(N,N1);
%    mysurf = zeros(N,4);
%    nndU = sim(net,TestX0')*maxT
%    for kk = 1:8
%       nnpdU = nndU(kk*(N1-1)+1:kk*N1);
%       nnpdU = nndU(1:M);
%       for j = 2:N1
%          nnpdU = [ nnpdU nndU((j-1)*M+1:j*M)];
%       end
%
%       for i = 1:N1
```

```
%           sum = zeros(N,1);
%           for k = 1:i
%              sum =  sum + Un(:,(k-1)*M+1:k*M)*pdU(:,k) + fUn(:,(k-
1)*N+1:k*N)*sin(xold(:,k)/6);
%           end
%           nnCX1(:,i) = sum + F^i*X0;
%        end
%        mysurf(:,kk) =  nnCX1(:,N1);
%        xold = nnCX1
%     end
%     mysurf
%     nnCX1(:,N1)
%     X = linspace(0,1,100);
%     nnZd = zeros(N1,length(X));
%     for i = 1 : length(X)
%        nny = interp(interp(nnCX1(:,N1),5,2,.2),5,2,.2);
%     end
%     Z = zeros(1,length(X));
%     for i = 1:length(X)
%        Z(i) = X(i)*(1-X(i));
%     end
%     figure
%     plot(X,nny(1:100),X,Z,'*')
%     figure
%     surf(mysurf)
%     ans = input( ' Validate the ANN ');
% end
%
% %
% %
% %
% % % figure
% % % plot(1:N1,CX1(1,:),1:N1,CX1(2,:))
```