**Program 9: Develop a MATLAB program to perform adaptive prediction with adaline.**

clear;

clc;

f1=2;

ts=1/(40\*f1);

N=100;

t1=(0:N)\*4\*ts;

t2=(0:2\*N)\*ts+4\*(N+1)\*ts;

t=[t1 t2];

N=size(t,2);

xt=[sin(2\*pi\*f1\*t1) sin(2\*pi\*2\*f1\*t2)];

plot(t,xt),grid,title('Signal to be predicted')

p=4;

X=convmtx(xt,p);X=X(:,1:N);

d=xt;

y=zeros(size(d));

eps=zeros(size(d));

eta=0.4;

w=rand(1,p);

for n=1:N

y(n)=w\*X(:,n);

eps(n)=d(n)-y(n);

w=w+eta\*eps(n)\*X(:,n)';

end

figure(1)

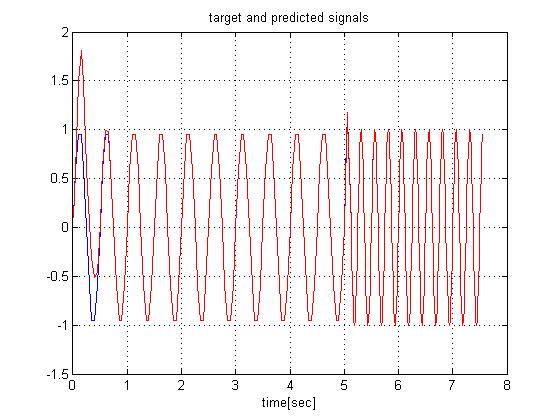
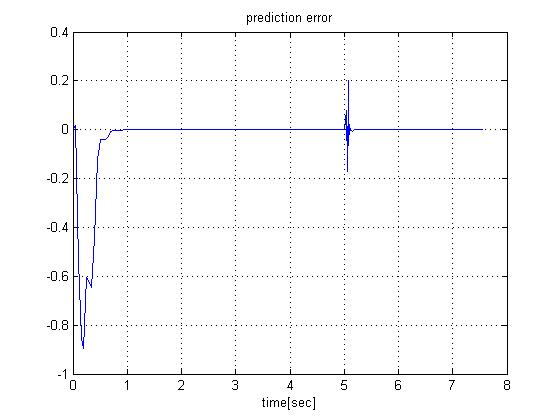
plot(t,d,'b',t,y,'-r'),grid,...

title('target and predicted signals'),xlabel('time[sec]')

figure(2)

plot(t,eps),grid,title('prediction error'),xlabel('time[sec]')

**Output:**



**Program 10: Write a M-file for adaptive system identification using adaline network.**

clear;

clc;

f=0.8;

ts=0.005;

N1=800;

N2=400;

N=N1+N2;

t1=(0:N1-1)\*ts;

t2=(N1:N-1)\*ts;

t=[t1 t2];

xt=sin(3\*t.\*sin(2\*pi\*f\*t));

p=3;

b1=[1 -0.6 0.4];

b2=[0.9 -0.5 0.7];

[d1,stt]=filter(b1,1,xt(1:N1));

d2=filter(b2,1,xt(N1+1:N),stt);

dd=[d1 d2];

x=convmtx(xt,p);

x=x(:,1:N);

d=[b1\*x(:,1:N1) b2\*x(:,N1+1:N)];

y=zeros(size(d));

eps=zeros(size(d));

eta=0.2;

w=2\*(rand(1,p) -0.5);

for n=1:N

y(n)=w\*x(:,n);

eps(n)=d(n)-y(n);

w=w+eta\*eps(n)\*x(:,n)';

if n==N1-1,w1=w;

end

end

figure(1)

subplot(2,1,1)

plot(t,xt),grid,title('Input Signal,x(t)'),xlabel('time sec')

subplot(2,1,2)

plot(t,d,'b',t,y,'-r'),grid,....

title('target and predicted signals'),....

Xlabel('time[sec]')

figure(2)

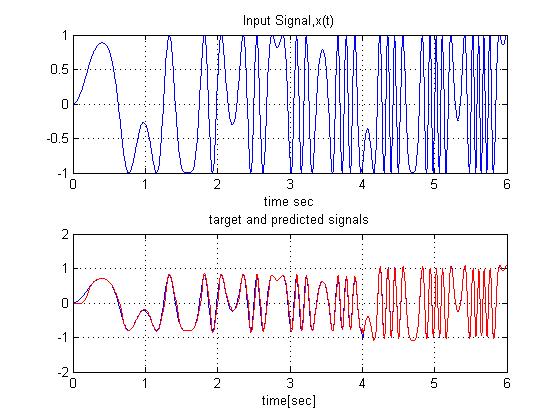
plot(t,eps),grid,title(['prediction error for eta=',num2str(eta)]),....

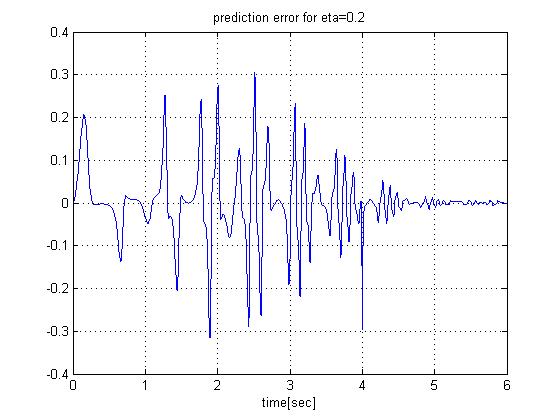
xlabel('time[sec]')

[b1; w1]

[b2; w]

**Output:**





**Program 11: Develop a MATLAB program for adaptive noise cancellation using adaline** **network.**

clear;

clc;

f=4e3;

fm=300;

fa=200;

ts=2e-5;

N=400;

t=(0:N-1)\*ts;

ut=(1+0.2\*sin(2\*pi\*fa\*t)).\*sin(2\*pi\*f\*(1+0.2\*cos(2\*pi\*fm\*t)).\*t);

xt=sawtooth(2\*pi\*1e3\*t,0.7);

b=[1 -0.6 -0.3];

vt=filter(b,1,xt);

dt=ut+vt;

figure(1)

subplot(2,1,1)

plot(1e3\*t, ut, 1e3\*t, dt),grid,title('Input u(t) and noisy input signal d(t)'),xlabel('time -- msec')

subplot(2,1,2)

plot(1e3\*t, xt, 1e3\*t, vt),grid,title('Noise x(t) and coloured noise v(t)'),xlabel('time --- msec')

p=4;

X=convmtx(xt,p); X=X(:,1:N);

y=zeros(1,N);

eps=zeros(1,N);

eta=0.05;

w=2\*(rand(1,p)-0.5);

for c=1:4

for n=1:N

y(n)=w\*X(:,n);

eps(n)=dt(n)-y(n);

w=w+eta\*eps(n)\*X(:,n)';

end

eta=0.8\*eta;

end

figure(2)

subplot(2,1,1)

plot(1e3\*t,ut,1e3\*t,eps),grid,...

title('Input signal u(t) and estimated signal uh(t)'),...

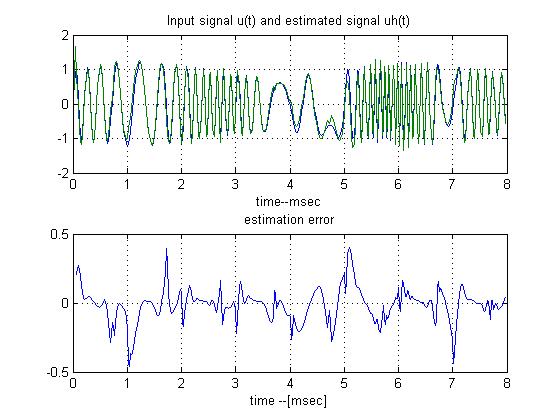
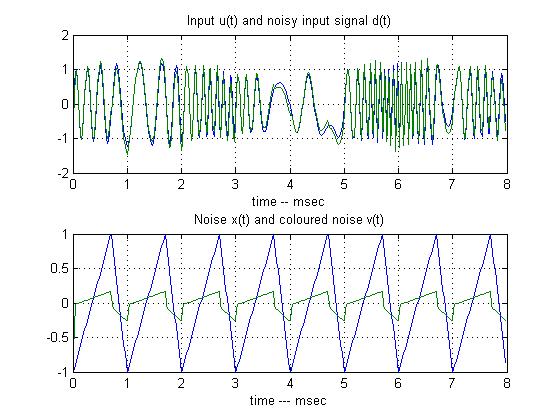
xlabel('time--msec')

subplot(2,1,2)

plot(1e3\*t(p:N),ut(p:N)-eps(p:N)),grid,...

title('estimation error'),xlabel('time --[msec]')

**Output:**



**Program 12: Write a MATLAB program for calculating the weights for the following pattern using hetro associative neural net for mapping four input vectors to two output vectors**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S1** | **S2** | **S3** | **S4** | **T1** | **T2** |
| **1** | **1** | **0** | **0** | **1** | **0** |
| **1** | **0** | **1** | **0** | **1** | **0** |
| **1** | **1** | **1** | **0** | **0** | **1** |
| **0** | **1** | **1** | **0** | **0** | **1** |

clc;

clear;

x=[1 1 0 0;1 0 1 0 ;1 1 1 0;0 1 1 0];

t=[1 0;1 0;0 1;0 1];

w=zeros(4,2);

for i=1:4

w=w+x(i,1:4)'\*t(i,1:2);

end

disp('weight matrix');

disp(w);

**Output:**

>> weight matrix

2 1

1 2

1 2

0 0

**Program 13: Write a MATLAB program to find the weight matrix of an auto associative net to store the vector (1 1 -1 -1).Test the response of the network by presenting the same pattern and recognize whether it is a known vector.**

clc;

clear;

x=[1 1 -1 -1];

w=zeros(4,4);

w=x'\*x;

yin=x\*w;

for i=1:4

if yin(i)>0

y(i)=1;

else

y(i)=-1;

end

end

disp('weight matrix');

disp(w);

if x==y

disp('the vector is a known vector')

else

disp('the vector is unknown vector');

end

**Output:**

>> weight matrix

1 1 -1 -1

1 1 -1 -1

-1 -1 1 1

-1 -1 1 1

the vector is a known vector

**Program 14: Write a MATLAB program to store the vectors (-1 -1 -1 -1) and (-1 -1 1 1) in an auto associative net. Find the weight matrix. Test the net with (1 1 1 1) as input.**

clc;

clear;

x=[-1 -1 -1 -1;-1 -1 1 1];

t=[1 1 1 1];

w=zeros(4,4);

for i=1:2

w=w+x(i,1:4)'\*x(i,1:4);

end

yin=t\*w;

for i=1:4

if yin(i)>0

y(i)=1;

else

y(i)=-1;

end

end

disp('the calculated weight matrix');

disp(w);

if x(1,1:4)==y(1:4)| x(2,1:4)==y(1:4)

disp('the vector is a known vector');

else

disp('the vector is a unknown vector');

end

Output:

>> the calculated weight matrix

2 2 0 0

2 2 0 0

0 0 2 2

0 0 2 2

the vector is a unknown vector

**Program 15: Write a MATLAB program for approximating two 2-dimensional functions using Back Propagation in batch mode.**

clear;

clc;

p=3;

L=12;

m=2;

na=16; N=na^2; nn=0:na-1;

X1=nn\*4/na-2;

[X1 X2]=meshgrid(X1);

R=(X1.^2+X2.^2+1e-5);

D1=X1.\*exp(-R); D=(D1(:))';

D2=0.25\*sin(2\*R)./R; D=[D;(D2(:))'];

Y=zeros(size(D));

X=[X1(:)'; X2(:)'; ones(1,N)];

figure(1), clf reset, hold off

surfc([X1-2 X1+2], [X2 X2], [D1 D2]), title('Two 2-D target functions'), grid on , drawnow

Wh=randn(L-1, p)/p;

Wy=randn(m, L)/L;

C=100;

J=zeros(m, C);

eta=[0.005 0.2];

figure(2), clf reset, hold off

tic

for c=1:C

H=ones(L-1, N)./(1+exp(-Wh\*X));

Hp=H.\*(1-H);

H=[H; ones(1,N)];

Y=tanh(Wy\*H);

Yp=1-Y.^2;

Ey=D-Y;

JJ=(sum((Ey.\*Ey)'))';

J(:, c)=JJ;

delY=Ey.\*Yp;

dWy=delY\*H';

Eh=Wy(:,1:L-1)'\*delY;

delH=Eh.\*Hp;

dWh=delH\*X';

Wy=Wy+eta(1)\*dWy; Wh=Wh+eta(2)\*dWh;

D1(:)=Y(1, :)'; D2(:)=Y(2, :)';

surfc([X1-2 X1+2], [X2 X2], [D1 D2]), grid on, title(['epoch:',num2str(c),'.error:',num2str(JJ'),',eta:',num2str(eta)]), drawnow

end

toc

figure(3)

clf reset

plot(J'), grid

title('The approximation error')

xlabel('number of training epochs')

**Output:**

