Table of Contents

Devon Quaternik

```
%ELEN 431
%Midterm

close all;
clear;
clc;
```

Problem 3

```
%Part b
figure;
itr = 1000;
rns = 50;
mu = 0.1;

sign2 = 0.0001;
sigx2 = 1.0;
M = 8;
w = zeros(M,1);
num = [0.8 -.31 -.45 -.8 .25 .55 .1 .9]';
den = [1];
WM = [];
JM = [];


for j=1:rns
    x = sqrt(sigx2)*randn(itr+2*M, 1);
    n = sqrt(sign2)*randn(itr, 1);

    g1=filter(num,den,x);
    g2=g1(M:itr+M-1,1);
    d = g2 + n;
        for i = 1:(itr-M+1)
            u = x(i+M-1:-1:i,1);
            y(i,1) = w'*u;
            e(i,1) = d(i,1)-y(i,1);
            J(i,1) = e(i,1)*conj(e(i,1));
            w = w+mu*u*e(i,1);
        end
        WM = [WM w];
        JM = [JM J];
end
```

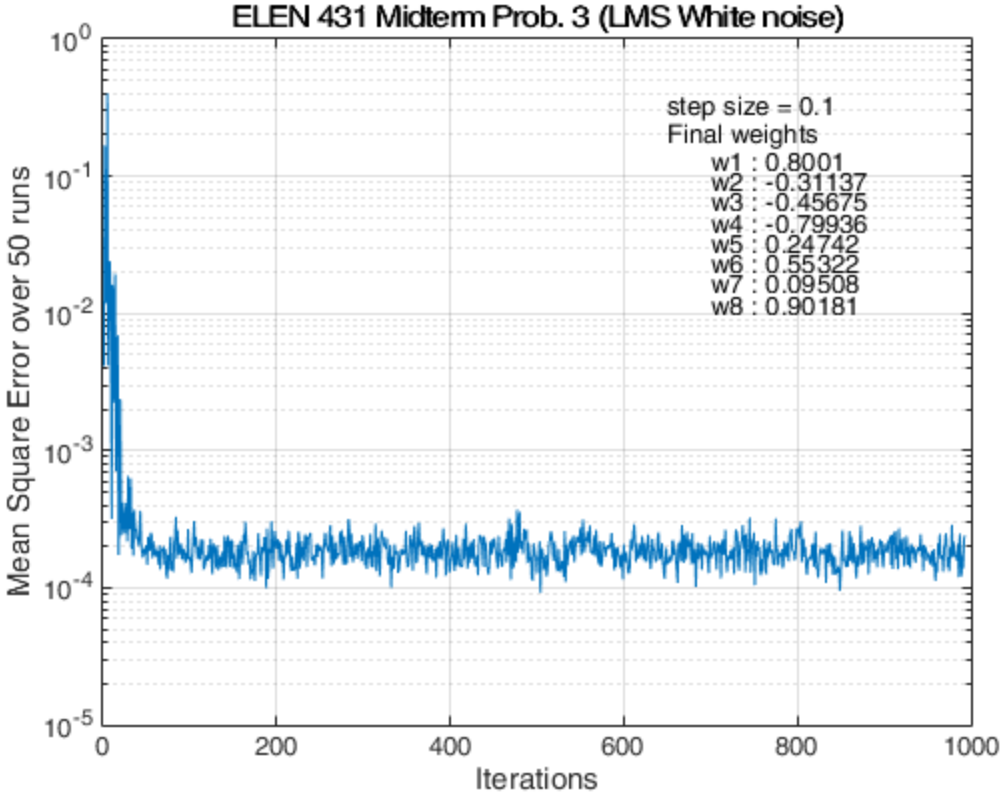
```
W = mean(WM');
LMS = [num W' W'-num]
Y = mean(JM');


tstr='ELEN 431 Midterm Prob. 3 (LMS White noise)';
ystr=['Mean Square Error over ',num2str(rns),' runs'];
xstr='Iterations';


semilogy(Y)
text(0.65,0.90,['step size = ',num2str(mu)],'sc');
text(0.65,0.86,['Final weights'],'sc');
for i = 1:M
    text(0.7,(0.85-i*0.03),['w',num2str(i),' : ',num2str(w(i))],'sc')
end
title(tstr)
xlabel(xstr)
ylabel(ystr)
grid
%print LMS
save LMS.mat;

% Leaky LMS
figure;
itr = 1000;
rns = 50;
mu = 0.1;

sign2 = 0.0001;
sigx2 = 1.0;
M = 8;
w = zeros(M,1);
num = [0.8 -.31 -.45 -.8 .25 .55 .1 .9]';
den = [1];
WM = [];
JM = [];
alph = 0.01;

for j=1:rns
    x = sqrt(sigx2)*randn(itr+2*M, 1);
    n = sqrt(sign2)*randn(itr, 1);

    g1=filter(num,den,x);
    g2=g1(M:itr+M-1,1);
    d = g2 + n;
        for i = 1:(itr-M+1)
            u = x(i+M-1:-1:i,1);
            y(i,1) = w'*u;
            e(i,1) = d(i,1)-y(i,1);
            %The difference is here
            J(i,1) = e(i,1)*conj(e(i,1)) + alph.*(w.'*w);
            w = w+mu*u*e(i,1);
        end
```

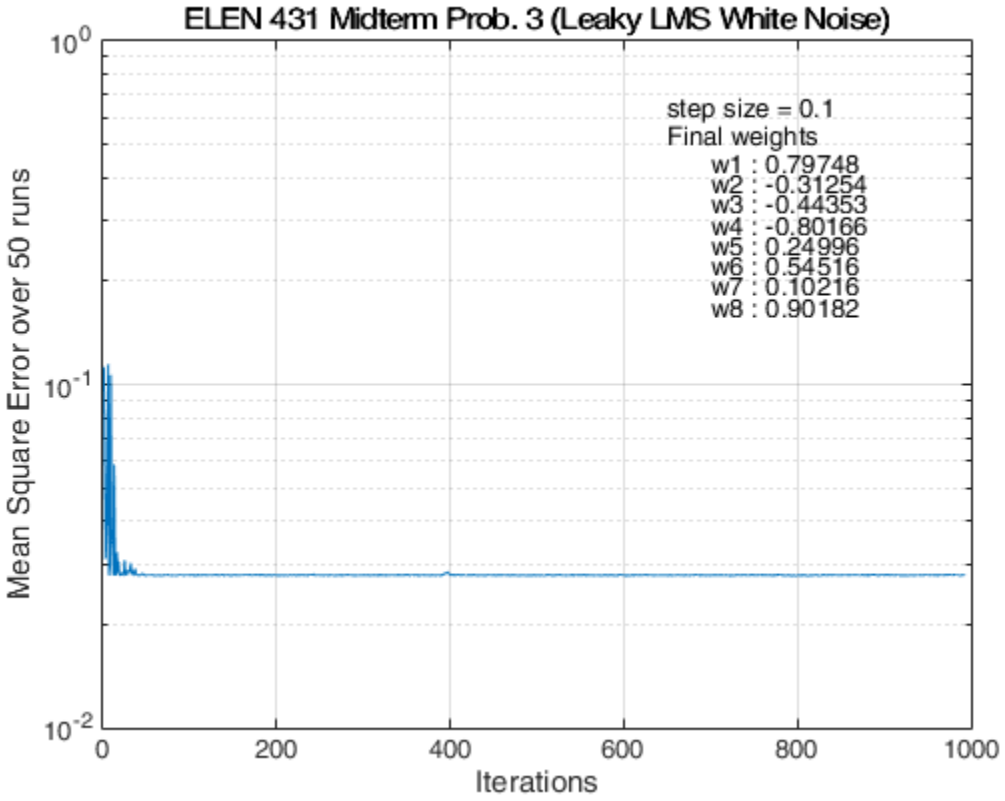
```
        WM = [WM w];
        JM = [JM J];
end

W = mean(WM');
leaky = [num W' W'-num]
Y = mean(JM');


tstr='ELEN 431 Midterm Prob. 3 (Leaky LMS White Noise)';
ystr=['Mean Square Error over ',num2str(rns),' runs'];
xstr='Iterations';


semilogy(Y)
text(0.65,0.90,['step size = ',num2str(mu)],'sc');
text(0.65,0.86,['Final weights'],'sc');
for i = 1:M
    text(0.7,(0.85-i*0.03),['w',num2str(i),' : ',num2str(w(i))],'sc')
end
title(tstr)
xlabel(xstr)
ylabel(ystr)
grid
%print leaky lms
save LeakyLMS.mat;

disp('The Leaky LMS has a much larger Mean square error, orders of
 magnitude larger, but it is far more consistent in achieving the
 lowest possible. There is almost no noise after a few iterations,
 where LMS has many');


%Part c
%LMS
figure;
iter = 1000;
num = 20;
mu = 0.01; %Mu must be decrease to account for the coloration or it
 will not converge

xpower = 1.0;
npower = 0.0001;
M = 8;

w = zeros(M,1);
y = zeros(iter,1);
e = zeros(iter,1);
G=[1 0 -0.9375 0 0.3281 0 0.0244];
B=[0.8 -0.31 -0.45 -0.8 0.25 0.55 0.1 0.9]';
A=[1];
WM = [];
JM = [];
```

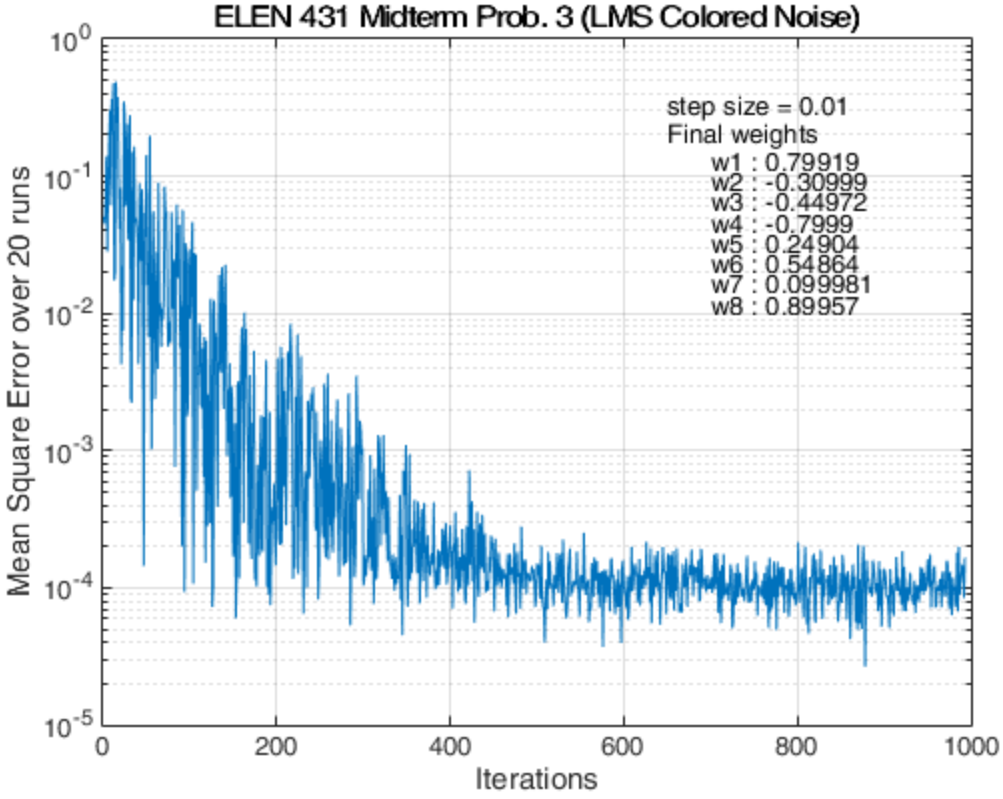
```
for j = 1:num;

x1 = sqrt(xpower) * randn(iter+2*M, 1);
n = sqrt(npower) * randn(iter, 1);

x = filter(G,A,x1);
g = filter(B,A,x);
g = g(M:iter+M-1, 1);
d = g + n;
 for n = 1:(iter-M+1)
  u = x(n+M-1:-1:n,1);
  y(n,1) = w' * u;
  e(n,1) = d(n,1) - y(n,1);
  J(n,1) = e(n,1) * conj(e(n,1));
  w = w + mu * u * e(n,1);
 end
 WM = [WM w];
 JM = [JM J];
end
W = mean(WM');
[B  W'  W'-B]
Y = mean(JM');
tstr='ELEN 431 Midterm Prob. 3 (LMS Colored Noise)';
ystr=['Mean Square Error over ',num2str(num),' runs'];
xstr='Iterations';

semilogy(Y)
text(0.65,0.90,['step size = ',num2str(mu)],'sc');
text(0.65,0.86,['Final weights'],'sc');
for i = 1:M
   text(0.7,(0.85-i*0.03),['w',num2str(i),' : ',num2str(w(i))],'sc')
end
title(tstr)
xlabel(xstr)
ylabel(ystr)
grid
%print LMS colored
save LMScolored.mat;


%Leaky LMS
figure;
iter = 1000;
num = 20;
mu = 0.01; %Mu must be decrease to account for the coloration or it
 will not converge

xpower = 1.0;
npower = 0.0001;
M = 8;

w = zeros(M,1);
y = zeros(iter,1);
e = zeros(iter,1);
```

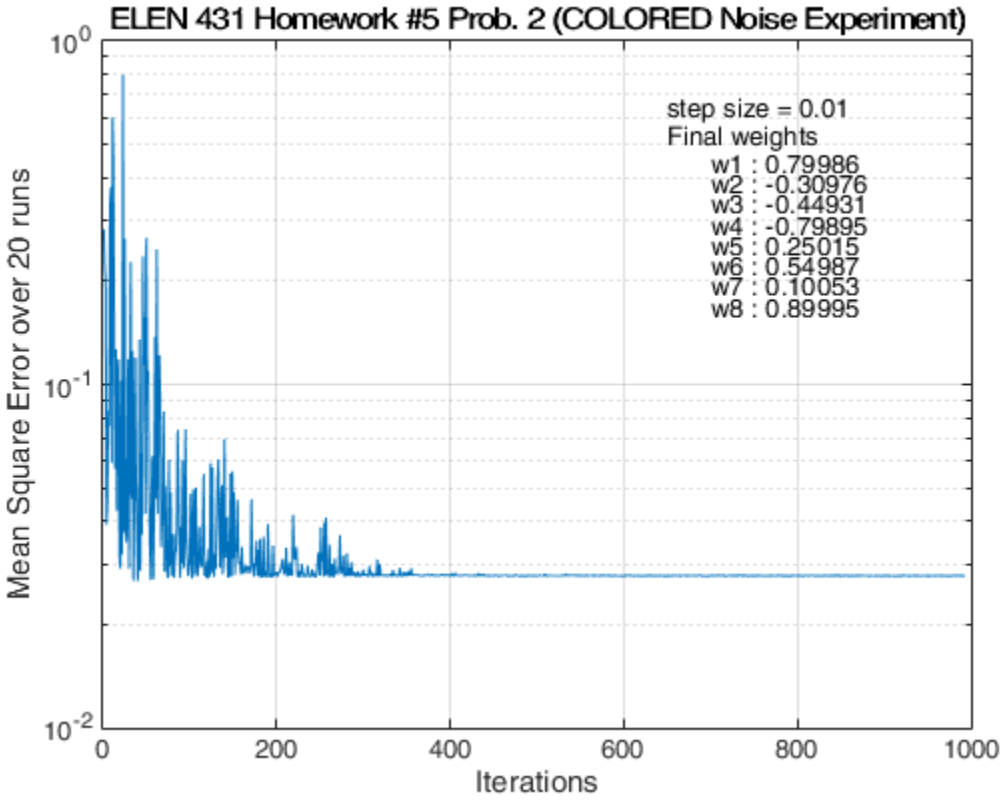
```
G=[1 0 -0.9375 0 0.3281 0 0.0244];
B=[0.8 -0.31 -0.45 -0.8 0.25 0.55 0.1 0.9]';
A=[1];
WM = [];
JM = [];
alph = 0.01;

for j = 1:num;

x1 = sqrt(xpower) * randn(iter+2*M, 1);
n = sqrt(npower) * randn(iter, 1);

x = filter(G,A,x1);
g = filter(B,A,x);
g = g(M:iter+M-1, 1);
d = g + n;
 for n = 1:(iter-M+1)
  u = x(n+M-1:-1:n,1);
  y(n,1) = w' * u;
  e(n,1) = d(n,1) - y(n,1);
  J(n,1) = e(n,1) * conj(e(n,1)) + alph.*(w.'*w);
  w = w + mu * u * e(n,1);
 end
 WM = [WM w];
 JM = [JM J];
end
W = mean(WM');
[B  W'  W'-B]
Y = mean(JM');
tstr='ELEN 431 Homework #5 Prob. 2 (COLORED Noise Experiment)';
ystr=['Mean Square Error over ',num2str(num),' runs'];
xstr='Iterations';

semilogy(Y)
text(0.65,0.90,['step size = ',num2str(mu)],'sc');
text(0.65,0.86,['Final weights'],'sc');
for i = 1:M
   text(0.7,(0.85-i*0.03),['w',num2str(i),' : ',num2str(w(i))],'sc')
end
title(tstr)
xlabel(xstr)
ylabel(ystr)
grid
%print hw5_p2
save hw5_p2.mat;

disp('As with the regular LMS, leaky is slower to converge with
 colored noise. The leaky is still much faster, converging in about
 half the number of iterations, and is again much more stable once
 converged. The mean square error remains higher by orders of
 magnitude.');


LMS =
```

```
    0.8000    0.8003    0.0003
   -0.3100   -0.3104   -0.0004
   -0.4500   -0.4506   -0.0006
   -0.8000   -0.8004   -0.0004
    0.2500    0.2497   -0.0003
    0.5500    0.5497   -0.0003
    0.1000    0.0995   -0.0005
    0.9000    0.8996   -0.0004


leaky =

    0.8000    0.7998   -0.0002
   -0.3100   -0.3092    0.0008
   -0.4500   -0.4504   -0.0004
   -0.8000   -0.8010   -0.0010
    0.2500    0.2499   -0.0001
    0.5500    0.5500   -0.0000
    0.1000    0.1001    0.0001
    0.9000    0.9004    0.0004

The Leaky LMS has a much larger Mean square error, orders of magnitude
 larger, but it is far more consistent in achieving the lowest
 possible. There is almost no noise after a few iterations, where LMS
 has many

ans =

    0.8000    0.8001    0.0001
   -0.3100   -0.3102   -0.0002
   -0.4500   -0.4499    0.0001
   -0.8000   -0.8000   -0.0000
    0.2500    0.2499   -0.0001
    0.5500    0.5499   -0.0001
    0.1000    0.0998   -0.0002
    0.9000    0.8998   -0.0002


ans =

    0.8000    0.7999   -0.0001
   -0.3100   -0.3102   -0.0002
   -0.4500   -0.4503   -0.0003
   -0.8000   -0.7998    0.0002
    0.2500    0.2499   -0.0001
    0.5500    0.5497   -0.0003
    0.1000    0.0996   -0.0004
    0.9000    0.8999   -0.0001

As with the regular LMS, leaky is slower to converge with colored
 noise. The leaky is still much faster, converging in about half the
 number of iterations, and is again much more stable once converged.
 The mean square error remains higher by orders of magnitude.
```


ELEN 431 Midterm Prob. 3 (LMS White noise)
step size = 0.1
Final weights
w1 : 0.8001
w2 : -0.31137
w3 : -0.45675
w4 : -0.79936
w5 : 0.24742
w6 : 0.55322
w7 : 0.09508
w8 : 0.90181
Mean Square Error over 50 runs
Iterations


ELEN 431 Midterm Prob. 3 (Leaky LMS White Noise)
step size = 0.1
Final weights
w1 : 0.79748
w2 : -0.31254
w3 : -0.44353
w4 : -0.80166
w5 : 0.24996
w6 : 0.54516
w7 : 0.10216
w8 : 0.90182
Mean Square Error over 50 runs
Iterations


ELEN 431 Midterm Prob. 3 (LMS Colored Noise)
step size = 0.01
Final weights
w1 : 0.79919
w2 : -0.30999
w3 : -0.44972
w4 : -0.7999
w5 : 0.24904
w6 : 0.54864
w7 : 0.099981
w8 : 0.89957
Mean Square Error over 20 runs
Iterations


ELEN 431 Homework #5 Prob. 2 (COLORED Noise Experiment)
step size = 0.01
Final weights
w1 : 0.79986
w2 : -0.30976
w3 : -0.44931
w4 : -0.79895
w5 : 0.25015
w6 : 0.54987
w7 : 0.10053
w8 : 0.89995
Mean Square Error over 20 runs
Iterations

Problem 4

```
%Part i
hnum = [0.8 -0.31 -0.45 -0.8 0.25 0.55 0.1 0.9];
hden = [1];
g1num = [0.1 0.2 0.3 0.4 0.4 0.2 0.1];
g1den = [1];
g2num = [0.1 -0.2 -0.3 0.4 0.4 -0.2 -0.1];
g2den = [1];
g3num = [0.1 0.2 0.3 -0.4 0.4 -0.2 0.1];
g3den = [1];

[h1,w1] = freqz(hnum,hden,1000);
[h2,w2] = freqz(g1num,g1den,1000);
[h3,w3] = freqz(g2num,g2den,1000);
[h4,w4] = freqz(g3num,g3den,1000);

figure;
subplot(2,2,1), plot(w1,abs(h1));
title('Frequency respons of H'); xlabel('Frequency (0:2*PI)');
 ylabel('Magnitude')
subplot(2,2,2), plot(w2,abs(h2));
title('Frequency respons of G1'); xlabel('Frequency (0:2*PI)');
 ylabel('Magnitude')
subplot(2,2,3), plot(w3,abs(h3));
title('Frequency respons of G2'); xlabel('Frequency (0:2*PI)');
 ylabel('Magnitude')
subplot(2,2,4), plot(w4,abs(h4));
title('Frequency respons of G3'); xlabel('Frequency (0:2*PI)');
 ylabel('Magnitude')

disp('H is a multi-band pass, G1 is LPF, G2 is bandpass over the
 middle of the spectrum, and G3 is a HPF.');

%Part ii
nIterations = 1000;
ensemble = 20;
enserror = [];
ensw     = [];

for j=1:ensemble
    u=wgn(1,nIterations+2*8,0);
    e0=wgn(1,nIterations+2*8,10^-4,'linear');
    input1=filter(g1num,g2den,u);
    desired1=(filter(hnum,hden,input1)+e0).';


    field1='step'; value1=.02;
    field2='filterOrderNo'; value2=7;
    field3='initialCoefficients'; value3=zeros(8,1);
    field4='gamma'; value4=0.01;
    field5='alpha'; value5=0.03;
    field6='initialPower'; value6=1;
```

```
    S =
 struct(field1,value1,field2,value2,field3,value3,field4,value4,field5,value5,fiel

    %DFT
    %G1 Plant
    %   Initialization Procedure
    nCoefficients       =   S.filterOrderNo+1;

    %   Pre Allocations
    errorVector             =   zeros(nIterations   ,1);
    outputVector            =   zeros(nIterations   ,1);
    coefficientVectorDFT    =   zeros(nCoefficients ,(nIterations+1));

    %   Initial State
    coefficientVectorDFT    =   fft(S.initialCoefficients)/
sqrt(nCoefficients);
    powerVector             =   S.initialPower*ones(1,nCoefficients);

    %   Improve source code regularity
    prefixedInput           =   [zeros(1,nCoefficients-1) input1];

    %   Body
    for it = 1:nIterations,

        regressorDFT        =   fft(prefixedInput(it
+(nCoefficients-1):-1:it))/...
                                sqrt(nCoefficients);

        %   Summing two column vectors
        powerVector         =
 S.alpha*(regressorDFT.*conj(regressorDFT))+...
                                (1-S.alpha)*(powerVector);

        outputVector(it,1)  =
 (coefficientVectorDFT(:,it)')*regressorDFT.';

        errorVector(it,1)   =   desired1(it)-outputVector(it,1);

        %   Vectorized
        extra=(S.step*conj(errorVector(it,1))*regressorDFT)./(S.gamma
+powerVector);

        coefficientVectorDFT(:,it+1)=
 coefficientVectorDFT(:,it)+(extra.');
        J(it,1) = errorVector(it,1) * conj(errorVector(it,1));

    end

    coefficientVector =
 ifft(coefficientVectorDFT)*sqrt(nCoefficients);
    enserror = [enserror J];
    ensw     = [ensw coefficientVector(:,1001)];
end
```

```
figure;
W = mean(ensw');
[B  W'  W'-B]
Y = mean(enserror');
tstr='ELEN 431 Midterm Prob. 4 (G1)';
ystr=['Mean Square Error over ',num2str(num),' runs'];
xstr='Iterations';

semilogy(Y)
text(0.65,0.90,['step size = ',num2str(S.step)],'sc');
text(0.65,0.86,['Final weights'],'sc');
for i = 1:M
   text(0.7,(0.85-i*0.03),['w',num2str(i),' :
 ',num2str(coefficientVector(i,1001))],'sc')
end
title(tstr)
xlabel(xstr)
ylabel(ystr)
grid
%print G1
save G1.mat;


%G2
enserror = [];
ensw     = [];

for j=1:ensemble
    u=wgn(1,nIterations+2*8,0);
    e0=wgn(1,nIterations+2*8,10^-4,'linear');
    input2=filter(g2num,g2den,u);
    desired2=(filter(hnum,hden,input2)+e0).';


    field1='step'; value1=.02;
    field2='filterOrderNo'; value2=7;
    field3='initialCoefficients'; value3=zeros(8,1);
    field4='gamma'; value4=0.01;
    field5='alpha'; value5=0.03;
    field6='initialPower'; value6=1;

    S =
 struct(field1,value1,field2,value2,field3,value3,field4,value4,field5,value5,fiel

    %DFT
    %G1 Plant
    %   Initialization Procedure
    nCoefficients       =   S.filterOrderNo+1;

    %   Pre Allocations
    errorVector             =   zeros(nIterations   ,1);
    outputVector            =   zeros(nIterations   ,1);
    coefficientVectorDFT    =   zeros(nCoefficients ,(nIterations+1));
```

```
    %   Initial State
    coefficientVectorDFT    =   fft(S.initialCoefficients)/
sqrt(nCoefficients);
    powerVector             =   S.initialPower*ones(1,nCoefficients);

    %   Improve source code regularity
    prefixedInput           =   [zeros(1,nCoefficients-1) input2];

    %   Body
    for it = 1:nIterations,

        regressorDFT        =   fft(prefixedInput(it
+(nCoefficients-1):-1:it))/...
                                sqrt(nCoefficients);

        %   Summing two column vectors
        powerVector         =
 S.alpha*(regressorDFT.*conj(regressorDFT))+...
                                (1-S.alpha)*(powerVector);

        outputVector(it,1)  =
 (coefficientVectorDFT(:,it)')*regressorDFT.';

        errorVector(it,1)   =   desired2(it)-outputVector(it,1);

        %   Vectorized
        extra=(S.step*conj(errorVector(it,1))*regressorDFT)./(S.gamma
+powerVector);

        coefficientVectorDFT(:,it+1)=
 coefficientVectorDFT(:,it)+(extra.');
        J(it,1) = errorVector(it,1) * conj(errorVector(it,1));

    end

    coefficientVector =
 ifft(coefficientVectorDFT)*sqrt(nCoefficients);
    enserror = [enserror J];
    ensw     = [ensw coefficientVector(:,1001)];
end

figure;
W = mean(ensw');
[B  W'  W'-B]
Y = mean(enserror');
tstr='ELEN 431 Midterm Prob. 4 (G2)';
ystr=['Mean Square Error over ',num2str(num),' runs'];
xstr='Iterations';

semilogy(Y)
text(0.65,0.90,['step size = ',num2str(S.step)],'sc');
text(0.65,0.86,['Final weights'],'sc');
for i = 1:M
```

```
   text(0.7,(0.85-i*0.03),['w',num2str(i),' :
 ',num2str(coefficientVector(i,1001))],'sc')
end
title(tstr)
xlabel(xstr)
ylabel(ystr)
grid
%print G2
save G2.mat;

%G3
enserror = [];
ensw     = [];

for j=1:ensemble
    u=wgn(1,nIterations+2*8,0);
    e0=wgn(1,nIterations+2*8,10^-4,'linear');
    input3=filter(g3num,g3den,u);
    desired3=(filter(hnum,hden,input3)+e0).';


    field1='step'; value1=.02;
    field2='filterOrderNo'; value2=7;
    field3='initialCoefficients'; value3=zeros(8,1);
    field4='gamma'; value4=0.01;
    field5='alpha'; value5=0.03;
    field6='initialPower'; value6=1;

    S =
 struct(field1,value1,field2,value2,field3,value3,field4,value4,field5,value5,fiel

    %DFT
    %G1 Plant
    %   Initialization Procedure
    nCoefficients       =   S.filterOrderNo+1;

    %   Pre Allocations
    errorVector             =   zeros(nIterations   ,1);
    outputVector            =   zeros(nIterations   ,1);
    coefficientVectorDFT    =   zeros(nCoefficients ,(nIterations+1));

    %   Initial State
    coefficientVectorDFT    =   fft(S.initialCoefficients)/
sqrt(nCoefficients);
    powerVector             =   S.initialPower*ones(1,nCoefficients);

    %   Improve source code regularity
    prefixedInput           =   [zeros(1,nCoefficients-1) input3];

    %   Body
    for it = 1:nIterations,

        regressorDFT        =   fft(prefixedInput(it
+(nCoefficients-1):-1:it))/...
```

```
                                sqrt(nCoefficients);

        %   Summing two column vectors
        powerVector         =
 S.alpha*(regressorDFT.*conj(regressorDFT))+...
                                (1-S.alpha)*(powerVector);

        outputVector(it,1)  =
 (coefficientVectorDFT(:,it)')*regressorDFT.';

        errorVector(it,1)   =   desired3(it)-outputVector(it,1);

        %   Vectorized
        extra=(S.step*conj(errorVector(it,1))*regressorDFT)./(S.gamma
+powerVector);

        coefficientVectorDFT(:,it+1)=
 coefficientVectorDFT(:,it)+(extra.');
        J(it,1) = errorVector(it,1) * conj(errorVector(it,1));

    end

    coefficientVector =
 ifft(coefficientVectorDFT)*sqrt(nCoefficients);
    enserror = [enserror J];
    ensw     = [ensw coefficientVector(:,1001)];
end

figure;
W = mean(ensw');
[B  W'  W'-B]
Y = mean(enserror');
tstr='ELEN 431 Midterm Prob. 4 (G3)';
ystr=['Mean Square Error over ',num2str(num),' runs'];
xstr='Iterations';

semilogy(Y)
text(0.65,0.90,['step size = ',num2str(S.step)],'sc');
text(0.65,0.86,['Final weights'],'sc');
for i = 1:M
   text(0.7,(0.85-i*0.03),['w',num2str(i),' :
 ',num2str(coefficientVector(i,1001))],'sc')
end
title(tstr)
xlabel(xstr)
ylabel(ystr)
grid
%print G3
save G3.mat;


%DCT
%G1
nIterations = 1000;
```

```
ensemble = 20;
enserror = [];
ensw     = [];

for j=1:ensemble
    u=wgn(1,nIterations+2*8,0);
    e0=wgn(1,nIterations+2*8,10^-4,'linear');
    input1=filter(g1num,g2den,u);
    desired1=(filter(hnum,hden,input1)+e0).';


    field1='step'; value1=.02;
    field2='filterOrderNo'; value2=7;
    field3='initialCoefficients'; value3=zeros(8,1);
    field4='gamma'; value4=0.01;
    field5='alpha'; value5=0.03;
    field6='initialPower'; value6=1;

    nCoefficients       =   S.filterOrderNo+1;
    T                   =   dctmtx(nCoefficients);

    %   Pre Allocations
    errorVector             =   zeros(nIterations   ,1);
    outputVector            =   zeros(nIterations   ,1);
    coefficientVectorDCT    =   zeros(nCoefficients ,(nIterations+1));

    %   Initial State
    coefficientVectorDCT    =   T*(S.initialCoefficients);
    powerVector             =   S.initialPower*ones(nCoefficients,1);

    %   Improve source code regularity
    prefixedInput           =   [zeros(nCoefficients-1,1)
                                 transpose(input1)];

    %   Body
    for it = 1:nIterations,

        regressorDCT        =   T*(prefixedInput(it
+(nCoefficients-1):-1:it));

        %   Summing two column vectors
        powerVector         =
 S.alpha*(regressorDCT.*conj(regressorDCT))+...
                                (1-S.alpha)*(powerVector);

        outputVector(it,1)  =
 (coefficientVectorDCT(:,it)')*regressorDCT;

        errorVector(it,1)   =   desired1(it)-outputVector(it,1);

        %   Vectorized
        extra = S.step*conj(errorVector(it,1)*regressorDCT)./(S.gamma
+powerVector);
```

```
        coefficientVectorDCT(:,it+1)=
 coefficientVectorDCT(:,it)+(extra);

        J(it,1) = errorVector(it,1) * conj(errorVector(it,1));

    end

    coefficientVector = T'*(coefficientVectorDCT);
    enserror = [enserror J];
    ensw     = [ensw coefficientVector(:,1001)];
end

figure;
W = mean(ensw');
[B  W'  W'-B]
Y = mean(enserror');
tstr='ELEN 431 Midterm Prob. 4 (G1 DCT)';
ystr=['Mean Square Error over ',num2str(num),' runs'];
xstr='Iterations';

semilogy(Y)
text(0.65,0.90,['step size = ',num2str(S.step)],'sc');
text(0.65,0.86,['Final weights'],'sc');
for i = 1:M
   text(0.7,(0.85-i*0.03),['w',num2str(i),' :
 ',num2str(coefficientVector(i,1001))],'sc')
end
title(tstr)
xlabel(xstr)
ylabel(ystr)
grid
%print G1
save G1DCT.mat;


%G2
nIterations = 1000;
ensemble = 20;
enserror = [];
ensw     = [];

for j=1:ensemble
    u=wgn(1,nIterations+2*8,0);
    e0=wgn(1,nIterations+2*8,10^-4,'linear');
    input2=filter(g1num,g2den,u);
    desired2=(filter(hnum,hden,input2)+e0).';


    field1='step'; value1=.02;
    field2='filterOrderNo'; value2=7;
    field3='initialCoefficients'; value3=zeros(8,1);
    field4='gamma'; value4=0.01;
    field5='alpha'; value5=0.03;
    field6='initialPower'; value6=1;
```

```
    nCoefficients       =   S.filterOrderNo+1;
    T                   =   dctmtx(nCoefficients);

    %   Pre Allocations
    errorVector             =   zeros(nIterations   ,1);
    outputVector            =   zeros(nIterations   ,1);
    coefficientVectorDCT    =   zeros(nCoefficients ,(nIterations+1));

    %   Initial State
    coefficientVectorDCT    =   T*(S.initialCoefficients);
    powerVector             =   S.initialPower*ones(nCoefficients,1);

    %   Improve source code regularity
    prefixedInput           =   [zeros(nCoefficients-1,1)
                                 transpose(input2)];


    %   Body
    for it = 1:nIterations,

        regressorDCT        =   T*(prefixedInput(it
+(nCoefficients-1):-1:it));

        %   Summing two column vectors
        powerVector         =
 S.alpha*(regressorDCT.*conj(regressorDCT))+...
                                (1-S.alpha)*(powerVector);

        outputVector(it,1)  =
 (coefficientVectorDCT(:,it)')*regressorDCT;

        errorVector(it,1)   =   desired2(it)-outputVector(it,1);

        %   Vectorized
        extra = S.step*conj(errorVector(it,1)*regressorDCT)./(S.gamma
+powerVector);

        coefficientVectorDCT(:,it+1)=
 coefficientVectorDCT(:,it)+(extra);

        J(it,1) = errorVector(it,1) * conj(errorVector(it,1));

    end

    coefficientVector = T'*(coefficientVectorDCT);
    enserror = [enserror J];
    ensw     = [ensw coefficientVector(:,1001)];
end

figure;
W = mean(ensw');
[B  W'  W'-B]
Y = mean(enserror');
tstr='ELEN 431 Midterm Prob. 4 (G2 DCT)';
```

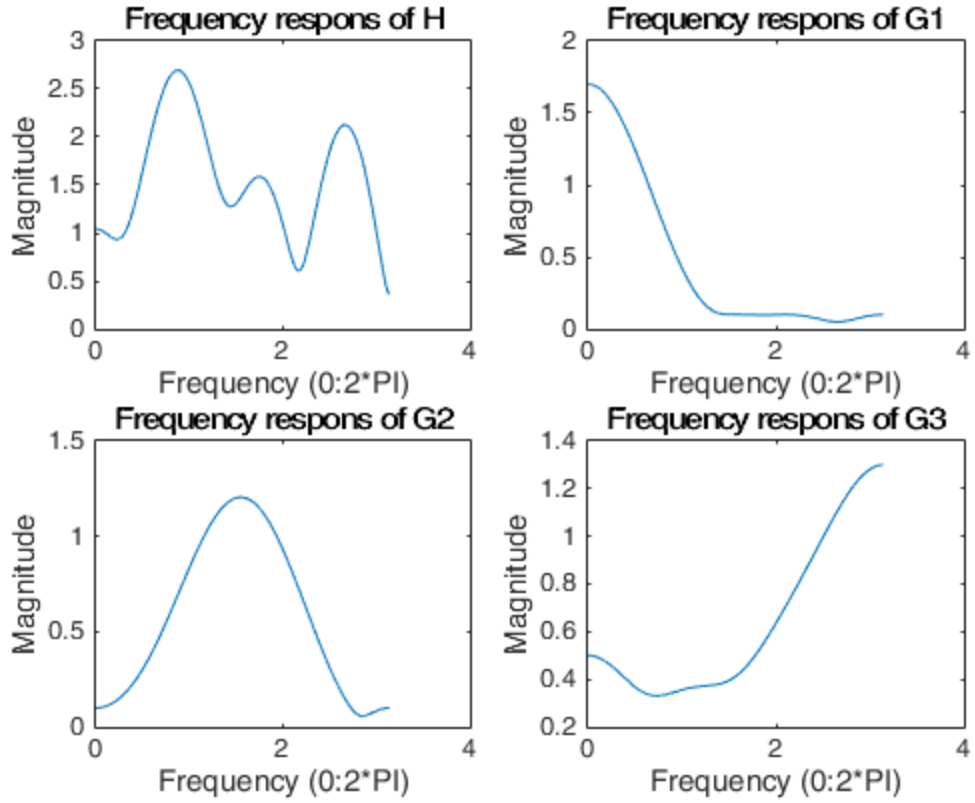
```
ystr=['Mean Square Error over ',num2str(num),' runs'];
xstr='Iterations';

semilogy(Y)
text(0.65,0.90,['step size = ',num2str(S.step)],'sc');
text(0.65,0.86,['Final weights'],'sc');
for i = 1:M
   text(0.7,(0.85-i*0.03),['w',num2str(i),' :
 ',num2str(coefficientVector(i,1001))],'sc')
end
title(tstr)
xlabel(xstr)
ylabel(ystr)
grid
%print G2
save G2DCT.mat;

%G3
nIterations = 1000;
ensemble = 20;
enserror = [];
ensw     = [];

for j=1:ensemble
    u=wgn(1,nIterations+2*8,0);
    e0=wgn(1,nIterations+2*8,10^-4,'linear');
    input3=filter(g3num,g3den,u);
    desired3=(filter(hnum,hden,input3)+e0).';


    field1='step'; value1=.02;
    field2='filterOrderNo'; value2=7;
    field3='initialCoefficients'; value3=zeros(8,1);
    field4='gamma'; value4=0.01;
    field5='alpha'; value5=0.03;
    field6='initialPower'; value6=1;

    nCoefficients       =   S.filterOrderNo+1;
    T                   =   dctmtx(nCoefficients);

    %   Pre Allocations
    errorVector             =   zeros(nIterations   ,1);
    outputVector            =   zeros(nIterations   ,1);
    coefficientVectorDCT    =   zeros(nCoefficients ,(nIterations+1));

    %   Initial State
    coefficientVectorDCT    =   T*(S.initialCoefficients);
    powerVector             =   S.initialPower*ones(nCoefficients,1);

    %   Improve source code regularity
    prefixedInput           =   [zeros(nCoefficients-1,1)
                                 transpose(input3)];

    %   Body
```

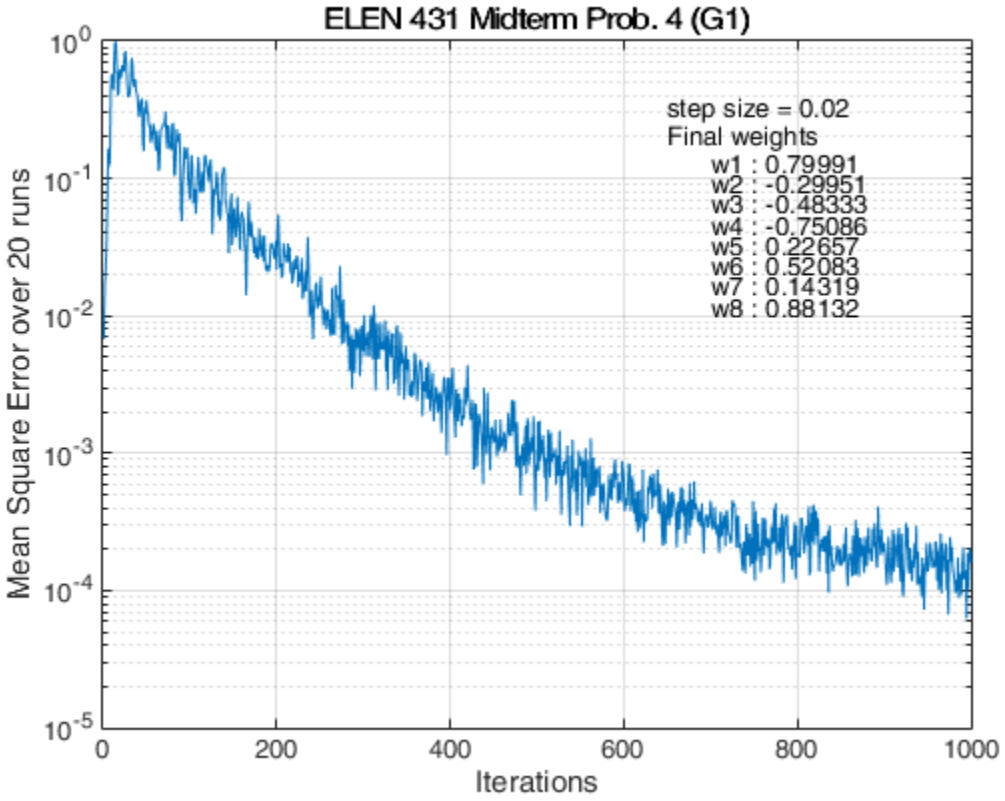
```
    for it = 1:nIterations,

        regressorDCT        =   T*(prefixedInput(it
+(nCoefficients-1):-1:it));

        %   Summing two column vectors
        powerVector         =
 S.alpha*(regressorDCT.*conj(regressorDCT))+...
                                (1-S.alpha)*(powerVector);

        outputVector(it,1)  =
 (coefficientVectorDCT(:,it)')*regressorDCT;

        errorVector(it,1)   =   desired3(it)-outputVector(it,1);

        %   Vectorized
        extra = S.step*conj(errorVector(it,1)*regressorDCT)./(S.gamma
+powerVector);

        coefficientVectorDCT(:,it+1)=
 coefficientVectorDCT(:,it)+(extra);

        J(it,1) = errorVector(it,1) * conj(errorVector(it,1));

    end

    coefficientVector = T'*(coefficientVectorDCT);
    enserror = [enserror J];
    ensw     = [ensw coefficientVector(:,1001)];
end

figure;
W = mean(ensw');
[B  W'  W'-B]
Y = mean(enserror');
tstr='ELEN 431 Midterm Prob. 4 (G3 DCT)';
ystr=['Mean Square Error over ',num2str(num),' runs'];
xstr='Iterations';

semilogy(Y)
text(0.65,0.90,['step size = ',num2str(S.step)],'sc');
text(0.65,0.86,['Final weights'],'sc');
for i = 1:M
   text(0.7,(0.85-i*0.03),['w',num2str(i),' :
 ',num2str(coefficientVector(i,1001))],'sc')
end
title(tstr)
xlabel(xstr)
ylabel(ystr)
grid
%print G3
save G3DCT.mat;

%Part iii
```

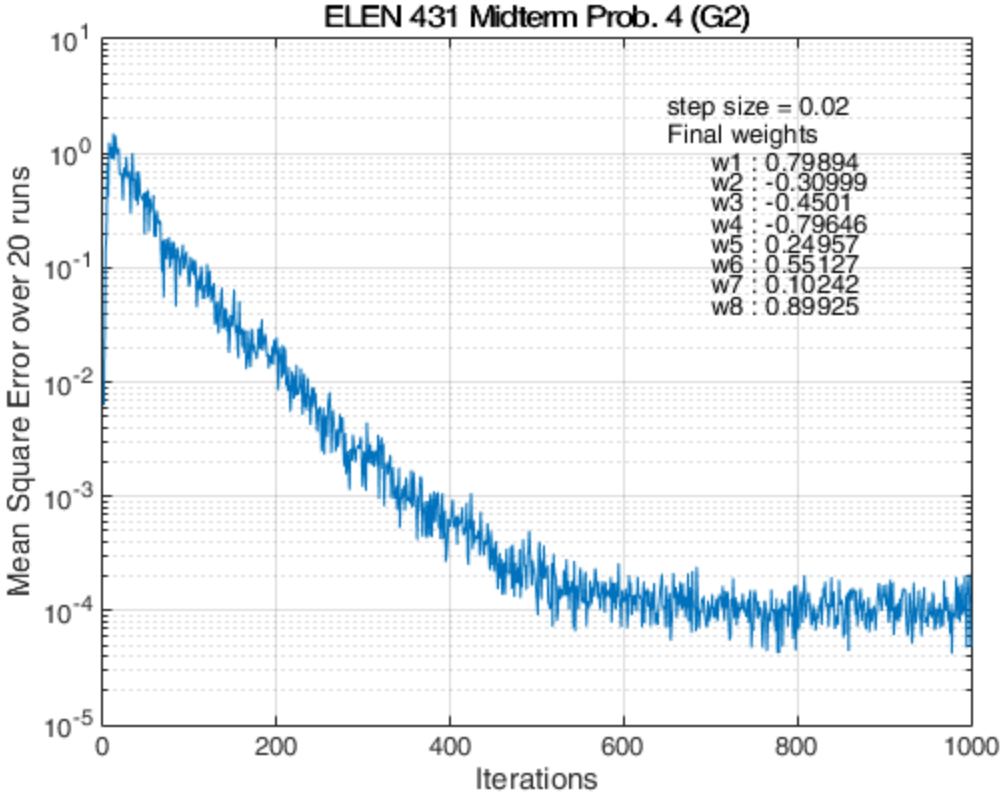
```
disp('The DCT performs best with a high pass filter, as in G3. It
 consistently converges faster on low and band-pass as well.');

%Part iv
disp('The reason DCT performs better because of the type of input
 used. Since it stationary, the DCT can take advantage of symmetries
 that DFT cannot. In doing this, it uses new data that allows it
 converge more quickly. The DCT generally works better when used on an
 even function, but any special symmetry helps.')
```

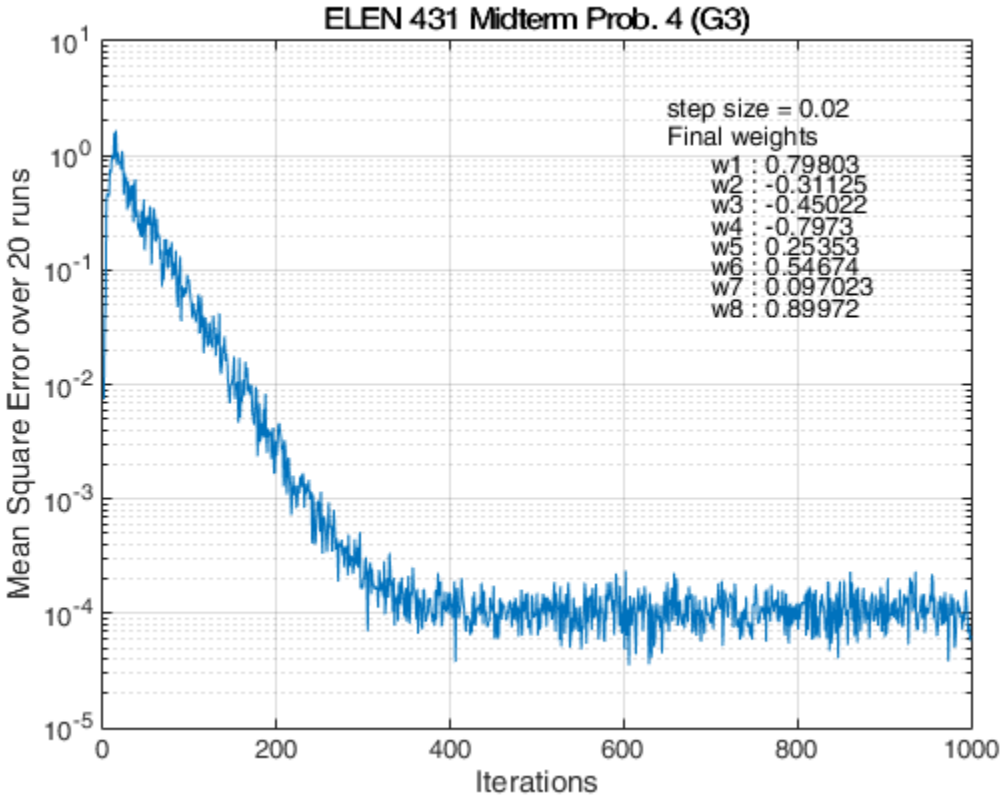
```
H is a multi-band pass, G1 is LPF, G2 is bandpass over the middle of
 the spectrum, and G3 is a HPF.

ans =

    0.8000    0.7956   -0.0044
   -0.3100   -0.2980    0.0120
   -0.4500   -0.4755   -0.0255
   -0.8000   -0.7632    0.0368
    0.2500    0.2303   -0.0197
    0.5500    0.5263   -0.0237
    0.1000    0.1420    0.0420
    0.9000    0.8800   -0.0200


ans =

    0.8000    0.7995   -0.0005
   -0.3100   -0.3088    0.0012
   -0.4500   -0.4513   -0.0013
   -0.8000   -0.7988    0.0012
    0.2500    0.2491   -0.0009
    0.5500    0.5509    0.0009
    0.1000    0.1001    0.0001
    0.9000    0.8997   -0.0003


ans =

    0.8000    0.8003    0.0003
   -0.3100   -0.3097    0.0003
   -0.4500   -0.4495    0.0005
   -0.8000   -0.7996    0.0004
    0.2500    0.2501    0.0001
    0.5500    0.5493   -0.0007
    0.1000    0.0994   -0.0006
    0.9000    0.9005    0.0005


ans =

    0.8000    0.7998   -0.0002
   -0.3100   -0.3091    0.0009
   -0.4500   -0.4511   -0.0011
```

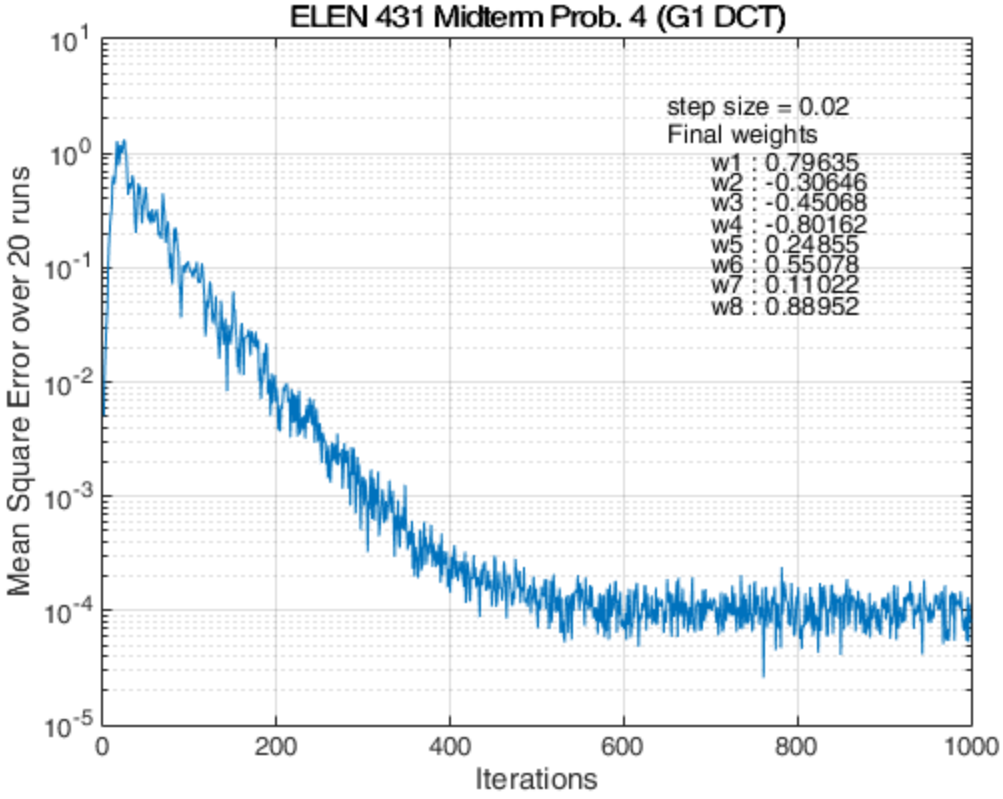
```
   -0.8000   -0.8015   -0.0015
    0.2500    0.2522    0.0022
    0.5500    0.5503    0.0003
    0.1000    0.1002    0.0002
    0.9000    0.8997   -0.0003


ans =

    0.8000    0.7990   -0.0010
   -0.3100   -0.3094    0.0006
   -0.4500   -0.4495    0.0005
   -0.8000   -0.7991    0.0009
    0.2500    0.2475   -0.0025
    0.5500    0.5497   -0.0003
    0.1000    0.1023    0.0023
    0.9000    0.8990   -0.0010


ans =

    0.8000    0.8001    0.0001
   -0.3100   -0.3106   -0.0006
   -0.4500   -0.4499    0.0001
   -0.8000   -0.7994    0.0006
    0.2500    0.2499   -0.0001
    0.5500    0.5494   -0.0006
    0.1000    0.0993   -0.0007
    0.9000    0.8996   -0.0004

The DCT performs best with a high pass filter, as in G3. It
 consistently converges faster on low and band-pass as well.
The reason DCT performs better because of the type of input used.
 Since it stationary, the DCT can take advantage of symmetries that
 DFT cannot. In doing this, it uses new data that allows it converge
 more quickly. The DCT generally works better when used on an even
 function, but any special symmetry helps.
```

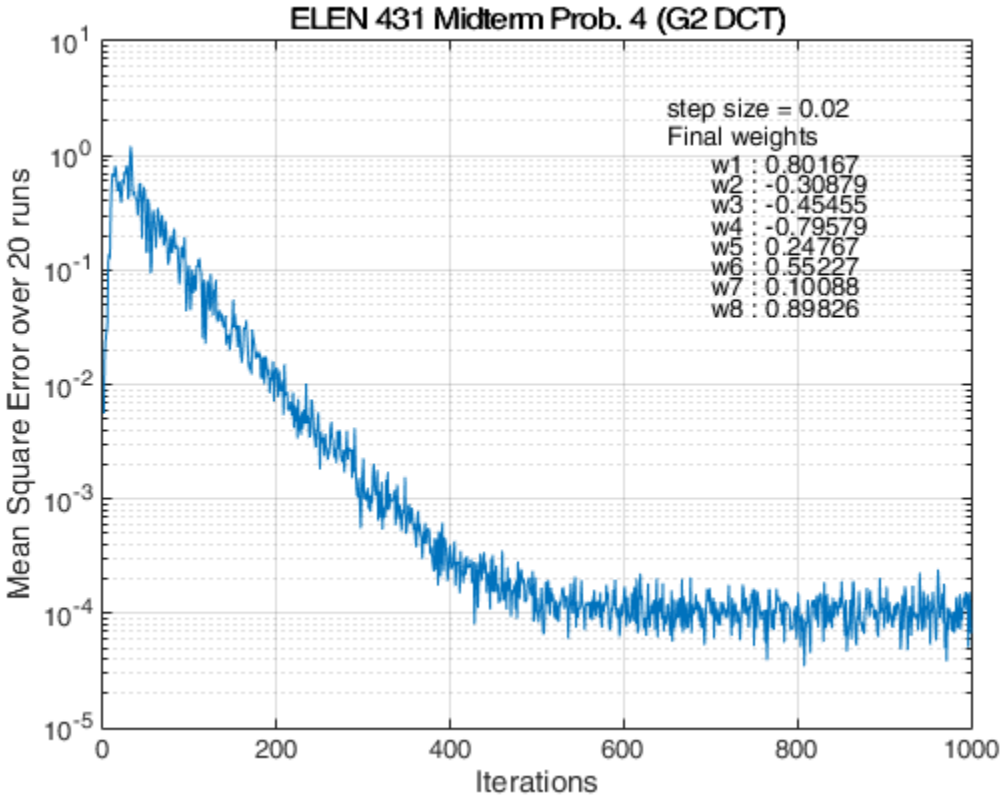

Frequency respons of H
Magnitude
Frequency (0:2*PI)
Frequency respons of G1
Magnitude
Frequency (0:2*PI)
Frequency respons of G2
Magnitude
Frequency (0:2*PI)
Frequency respons of G3
Magnitude
Frequency (0:2*PI)

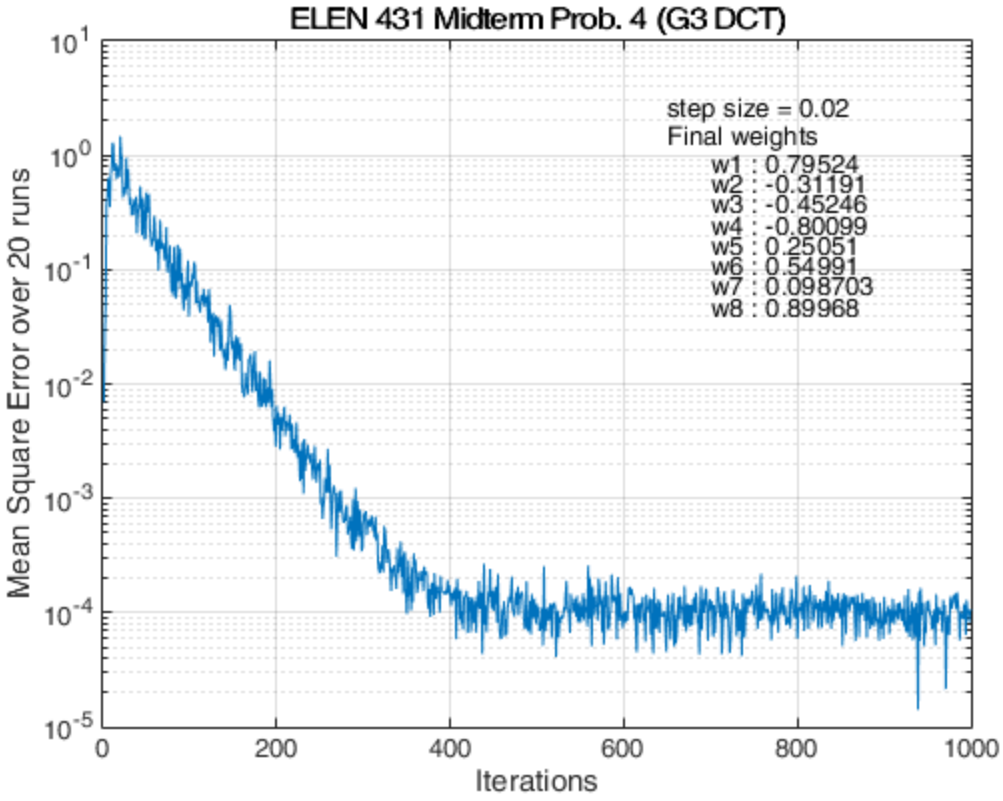

ELEN 431 Midterm Prob. 4 (G1)
step size = 0.02
Final weights
w1 : 0.79991
w2 : -0.29951
w3 : -0.48333
w4 : -0.75086
w5 : 0.22657
w6 : 0.52083
w7 : 0.14319
w8 : 0.88132
Mean Square Error over 20 runs
Iterations


ELEN 431 Midterm Prob. 4 (G2)
step size = 0.02
Final weights
w1 : 0.79894
w2 : -0.30999
w3 : -0.4501
w4 : -0.79646
w5 : 0.24957
w6 : 0.55127
w7 : 0.10242
w8 : 0.89925
Mean Square Error over 20 runs
Iterations


ELEN 431 Midterm Prob. 4 (G3)
step size = 0.02
Final weights
w1 : 0.79803
w2 : -0.31125
w3 : -0.45022
w4 : -0.7973
w5 : 0.25353
w6 : 0.54674
w7 : 0.097023
w8 : 0.89972
Mean Square Error over 20 runs
Iterations


ELEN 431 Midterm Prob. 4 (G1 DCT)
step size = 0.02
Final weights
w1 : 0.79635
w2 : -0.30646
w3 : -0.45068
w4 : -0.80162
w5 : 0.24855
w6 : 0.55078
w7 : 0.11022
w8 : 0.88952
Mean Square Error over 20 runs
Iterations


ELEN 431 Midterm Prob. 4 (G2 DCT)
step size = 0.02
Final weights
w1 : 0.80167
w2 : -0.30879
w3 : -0.45455
w4 : -0.79579
w5 : 0.24767
w6 : 0.55227
w7 : 0.10088
w8 : 0.89826
Mean Square Error over 20 runs
Iterations



Published with MATLAB® R2015a