**EL 7133 (DSP II)**

**Spring 2016**

**Homework Assignment - 08**

Name: **Amitesh Kumar Sah**

NYU ID: **N19714360**

**Question 1:**

% Use Matlab to make the 7/9 symmetric biorthogonal Daubechies filters. Use the

% same product filter P(z) as for the design of orthonormal wavelet filters (i.e., a half-band Type 1 FIR filter of length 2N-1 with N zeros at z = -1). Use N = 8 to get the product filter P(z) needed here.

% Implement a perfect reconstruction filter bank using the filter you create. Verify the filter bank has the perfect reconstruction propertyclc;

clear all;

close all;

Qz=[0.0024 -0.0195 0.0640 -0.1016 0.0640 -0.0195 0.0024];

hp=poly([-1 -1 -1 -1 -1 -1 -1 -1]);

p=conv(Qz,hp);

r=roots(p);

r0=zeros(6,1);

j=1;

for i=1:length(r)

if 0.9< abs(r(i,1)) && abs(r(i,1))<1

r0(j,1)=r(i,1);

j=j+1;

end

if imag(r(i,1))==0

r0(j,1)=r(i,1);

j=j+1;

end

end

h0=-poly(r0);

r1=setdiff(r,r0);

g0=poly(r1);

figure,

subplot(3,2,1),zplane(p),title('Product filter');

subplot(3,2,2),zplane(h0),title('HO Low pass filter');

subplot(3,2,3),zplane(g0),title('G0 Low Pass filter');

subplot(3,2,4),stem(h0),title('H0 Impulse Response');

subplot(3,2,5),stem(g0),title('G0 Impulse Response');

%% For high pass filter

rh=-r;

ph=poly(rh);

r2=zeros(6,1);

j=1;

for i=1:length(rh)

if 0.9< abs(rh(i,1)) && abs(rh(i,1))<1

r2(j,1)=rh(i,1);

j=j+1;

end

if imag(rh(i,1))==0

r2(j,1)=rh(i,1);

j=j+1;

end

end

g1=poly(r2);

r3=setdiff(rh,r2);

h1=poly(r3);

figure,

subplot(3,2,1),zplane(ph),title('Product filter');

subplot(3,2,2),zplane(h1),title('H1 High pass filter');

subplot(3,2,3),zplane(g1),title('G1 High Pass filter');

subplot(3,2,4),stem(h1),title('H1 Impulse Response');

subplot(3,2,5),stem(g1),title('G1 Impulse Response');

%% Verifying perfect reconstruction property

x=[1 2 3 4];

y1=conv(x,h0);

y1=downsample(y1,2);

y1=upsample(y1,2);

y1=conv(y1,g0);

y2=conv(x,h1);

y2=downsample(y2,2);

y2=upsample(y2,2);

y2=conv(y2,g1);

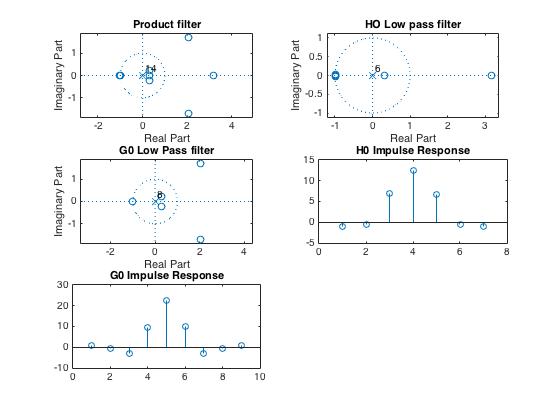
y=(y1+y2)/414;

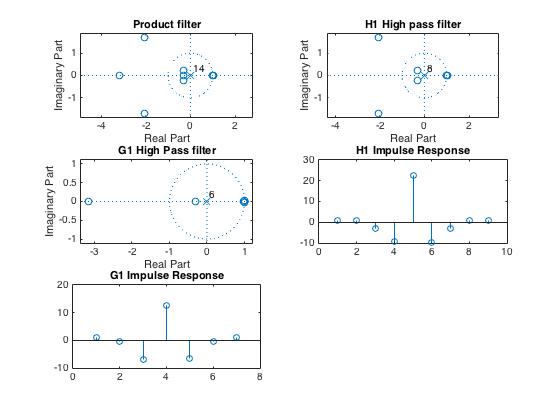
figure,

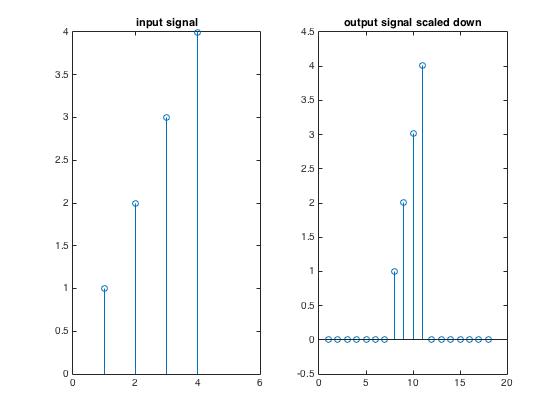
subplot(1,2,1),stem(x);title('input signal');xlim([0 6]);

subplot(1,2,2),stem(y);title('output signal scaled down');

Output:







Daubechies 7/9 filter

h0 =

-1.0000 -0.4859 6.8719 12.5762 6.6330 -0.5451 -0.9598

>> h1

h1 =

1.0000 0.6109 -3.1645 -9.4157 22.2948 -9.9336 -3.1558 0.7220 1.0419

>> g0

g0 =

1.0000 -0.6109 -3.1645 9.4157 22.2948 9.9336 -3.1558 -0.7220 1.0419

>> g1

g1 =

1.0000 -0.4859 -6.8719 12.5762 -6.6330 -0.5451 0.9598

Result

7/9 symmetric biorthogonal Daubechies filters is designed in Matlab. When I gave an input signal, I got the dame output signal, but with shift and scaled. This verifies the filter bank has the perfect reconstruction property.

Question 2

% Read DFT

% Construct the transform matrix for a 'real' 8-point DFT in Matlab. The matrix

% should be real-valued and orthonormal. Verify that the matrix is orthonormal (i.e., verify

% that the inverse matrix is the transpose matrix). Display the basis vectors (i.e., the rows

% of the forward transform or the columns of the inverse transform). The basis vectors of

% the 'real' DFT should be the real and imaginary parts of the basis vectors of the DFT

% with appropriate scaling constants for normalization.

% Real valued DFT is obtained by mirroring N length sequence to form 2N-1 sequence.

% DFT=X1(k)=e^(-j\*2\*pi\*-0.5\*k/2N)\*sum(x(n)\*cos(pi/N \*k\*(n+0.5)) from 0 to N-1.

% Transform matrix is D(k,n)=sqrt(2/N)\*cos(pi/N \*k\*(n-0.5)) . First row is sqrt(1/N)

clc;clear all;close all

N=8;

D=zeros(N,N);

for k=1:N-1

for n=1:N

D(k+1,n)=sqrt(2/N)\*cos(pi/N \*k\*(n-0.5));

end

end

D(1,:)=sqrt(1/N);

if inv(D)==transpose(D)

disp('Orthonormal Matrix')

end

%% Basis Vector

b1=D(1,:)

disp('Basis Vector 1');

b2=D(2,:)

disp('Basis Vector 2');

b3=D(3,:)

disp('Basis Vector 3');

b4=D(4,:)

disp('Basis Vector 4');

b5=D(5,:)

disp('Basis Vector 5');

b6=D(6,:)

disp('Basis Vector 6');

b7=D(7,:)

disp('Basis Vector 7');

b8=D(8,:)

disp('Basis Vector 8');

Result:

Transform Matrix for a real 8-point DFT

D =

0.3536 0.3536 0.3536 0.3536 0.3536 0.3536 0.3536 0.3536

0.4904 0.4157 0.2778 0.0975 -0.0975 -0.2778 -0.4157 -0.4904

0.4619 0.1913 -0.1913 -0.4619 -0.4619 -0.1913 0.1913 0.4619

0.4157 -0.0975 -0.4904 -0.2778 0.2778 0.4904 0.0975 -0.4157

0.3536 -0.3536 -0.3536 0.3536 0.3536 -0.3536 -0.3536 0.3536

0.2778 -0.4904 0.0975 0.4157 -0.4157 -0.0975 0.4904 -0.2778

0.1913 -0.4619 0.4619 -0.1913 -0.1913 0.4619 -0.4619 0.1913

0.0975 -0.2778 0.4157 -0.4904 0.4904 -0.4157 0.2778 -0.0975

>> inv(D)

ans =

0.3536 0.4904 0.4619 0.4157 0.3536 0.2778 0.1913 0.0975

0.3536 0.4157 0.1913 -0.0975 -0.3536 -0.4904 -0.4619 -0.2778

0.3536 0.2778 -0.1913 -0.4904 -0.3536 0.0975 0.4619 0.4157

0.3536 0.0975 -0.4619 -0.2778 0.3536 0.4157 -0.1913 -0.4904

0.3536 -0.0975 -0.4619 0.2778 0.3536 -0.4157 -0.1913 0.4904

0.3536 -0.2778 -0.1913 0.4904 -0.3536 -0.0975 0.4619 -0.4157

0.3536 -0.4157 0.1913 0.0975 -0.3536 0.4904 -0.4619 0.2778

0.3536 -0.4904 0.4619 -0.4157 0.3536 -0.2778 0.1913 -0.0975

b1 =

0.3536 0.3536 0.3536 0.3536 0.3536 0.3536 0.3536 0.3536

Basis Vector 1

b2 =

0.4904 0.4157 0.2778 0.0975 -0.0975 -0.2778 -0.4157 -0.4904

Basis Vector 2

b3 =

0.4619 0.1913 -0.1913 -0.4619 -0.4619 -0.1913 0.1913 0.4619

Basis Vector 3

b4 =

0.4157 -0.0975 -0.4904 -0.2778 0.2778 0.4904 0.0975 -0.4157

Basis Vector 4

b5 =

0.3536 -0.3536 -0.3536 0.3536 0.3536 -0.3536 -0.3536 0.3536

Basis Vector 5

b6 =

0.2778 -0.4904 0.0975 0.4157 -0.4157 -0.0975 0.4904 -0.2778

Basis Vector 6

b7 =

0.1913 -0.4619 0.4619 -0.1913 -0.1913 0.4619 -0.4619 0.1913

Basis Vector 7

b8 =

0.0975 -0.2778 0.4157 -0.4904 0.4904 -0.4157 0.2778 -0.0975

Basis Vector 8

Transpose of D is equal to inverse of D , hence it is Orthonormal Matrix. I have also shown all the 8 basis vector.

Question 3

% Principle component analysis (PCA)

% Use PCA to calculate an optimal orthogonal (non-separable) transform for 4x4

% image blocks. To do this, take all 4x4 image blocks from an image, create a data matrix,

% then create a covariance matrix, and then perform eigenvector decomposition. The

% covariance matrix should be of size 16x16 and the eigenvectors should be of length 16.

% Each eigenvector is a vectorized set of 16 pixels in block of size 4x4. For reference, see

% the Matlab demo from class PCA\_from\_image\_data.m which calculated an optimal

% transform for 8x1 image vectors.

% Show the two-dimensional basis functions of 4x4 blocks. For example, my result is

% shown in the file two-dimensional PCA.pdf

% Use your PCA to reconstruct an image from only four coefficients in each 4x4 block

% (i.e., set 12 of the 16 coefficients to zero).

% Reading: 'A Tutorial on Principal Component Analysis' by Jonathon Shlens

% http://arxiv.org/abs/1404.1100

clc;

clear all; close all

a=imread('lena\_gray.bmp');

a = double(a);

[r,c]=size(a);

%% Creating 4\*4 blocks

p=1;q=1;

for i=1:4:r

for j=1:4:c

d{p,q,:}=a(i:4,j:4);

q=q+1;

end

p=p+1;

end

%% Center the data (remove mean from each component)

p=1;q=1;

for i=1:4:r

for j=1:4:c

res=a(i:i+4-1,j:j+4-1)-mean(mean(a(i:i+4-1,j:j+4-1)));

x3{p,q,:}=reshape(res,16,1);

q=q+1;

end

p=p+1;

end

%% Compute covariance matrix

p=1;q=1;

for i=1:4:r

for j=1:4:c

z=x3{p,q,:};

N = size(z,2);

R{p,q,:} = (1/N) \* z \* (z');

q=q+1;

end

p=p+1;

end

size(R{1,1,:})

%% Compute covariance matrix

p=1;q=1;

for i=1:4:r

for j=1:4:c

R1=R{p,q,:};

[V, D] = eig(R1);

% eigenvalues

D1{p,q,:}=diag(D);

V1{p,q,:}=V;

q=q+1;

end

p=p+1;

end

%% Display PCA basis vectors

% Notice how similar the PCA basis vectors are to the DCT basis vectors!

% The first vector is a constant vector (approximately);

% the 2nd vector is like a half-cycle of a cosine waveform;

% the 3rd vector is like a whole cycle of a cosine waveform; etc.

%%

figure(4)

clf

p=1;q=1;

for k = 1:16

subplot(8,2,k)

plot(0:15, V1{p,q,:}, '.-', 'markersize', 12)

box off

xlim([0 15])

ylim([-1 1])

q=q+1;

end

orient landscape

print -dpdf PCA\_from\_image\_data

Result

Here, I just displayed the 1st 16 PCA vector.

