Revised 1 February 2017

Project #4: Implementation of Wavelet Transforms

In this lab you will use MATLAB to study the following topics:

- CDF(2, 2) wavelet transform (matrix form)
- Daub4 wavelet transform (matrix form)
- Fast Daub4 wavelet transform
- Multiresolution analysis using the CDF(2, 2) wavelet transform

Preliminaries

Reading from Textbook: Before beginning your MATLAB work, read Sections Section 3.4 and 3.5 of the textbook.

Function Files: You will need the MATLAB function m-files `downsamp.m` and `shift.m` from Project #2 and `threshold.m` from Project #3. Be sure that these files are in the directory that you are using for your MATLAB sessions.

Uvi_Wave: Some of the MATLAB m-files that are used in Question #4 are from the public-domain UVI_WAVE toolbox. You will need to download and unzip this package of files from the Math 357 web page or the web page of textbook:

http://www.worldscientific.com/worldscibooks/10.1142/9835#t=suppl

Put the UVI_WAVE folder and all its subfolders into a subdirectory of your MATLAB directory. Set the `path` so that MATLAB can find this folder and all its subfolders.

Diary File: You will need to record the results of your MATLAB session to generate your lab report. Create a directory (folder) in your computer workspace to save your MATLAB work in. Then use the `Current Directory` field in the desktop toolbar to change the directory to this folder. Now type

```
diary lab4.txt
```

followed by the `Enter` key. Now each computation you make in MATLAB will be saved in your directory in a text file named `lab4.txt`. You can then edit this file using your favorite text editor (such as Emacs, Notepad, or Word). When you have finished your MATLAB session you can turn off the recording by typing `diary off` at the MATLAB prompt (*don't do this now*).

Lab Write-up: Now that your diary file is open, type the comment line

```
% Math 357 Matlab Project #4 -- Implementation of Wavelet Transforms
%
```

at the MATLAB prompt. Type

```
format compact
```

so that your diary file will not have unnecessary spaces.

Random Seed: Initialize the random number generator by typing

```
rand('seed', abcd)
```

where *abcd* are the last four digits of your Student ID number. This will ensure that you generate your own particular random vectors and matrices.

BE SURE TO INCLUDE THIS LINE IN YOUR LAB WRITE-UP

Labels and Comments: Put labels to mark the beginning of your work on each part of each question, so that your edited lab write-up has the format

```
% Question 1 (a)  ...
⋮
% Question 1 (b) ...
```

and so on. Be sure to answer all the questions in the lab assignment. Insert comments in your diary file as you work through the assignment.

The lab report that you hand in must be your own work. The following problems use randomly generated matrices and vectors, so the matrices, vectors, and graphs in your lab report will not be the same as those of other students doing the lab. Sharing of lab report files is not allowed in this course.

Question 1. CDF$(2,2)$ Wavelet Transform

You will use MATLAB to explore the results in Sections 3.4.1 and 3.5 of the textbook.

(a) CDF$(2,2)$ analysis matrix: Implement the CDF$(2,2)$ one-scale wavelet transform in matrix form for periodic signals $\mathbf{s}$ of even length $N \geq 4$, where $\mathbf{s}$ is a column vector, as follows. The $N \times N$ analysis matrix is obtained by the lifting method as the product

$$\mathbf{T}_{\mathrm{a}} = D\,U\,P\,\boxed{\text{split}}\ . \tag{1}$$

Here $\boxed{\text{split}}$ is the downsampling matrix that separates the signal $\mathbf{s}$ into subsignals $\mathbf{s}_{\text{even}}$ and $\mathbf{s}_{\text{odd}}$ of length $N/2$. After downsampling, the first lifting operation is a prediction step that is given by the matrix P in equation (3.12) in the textbook. Next, there is an update step that is given by the matrix U in equation (3.16). Finally, there is a normalization step that is performed by the diagonal matrix D in equation (3.17).

Write the following function m-file to generate the matrix $\mathbf{T}_{\mathrm{a}}$. This m-file uses the MATLAB function m-files `shift.m` and `downsamp.m` from Project #2.

```
% Generates N x N analysis matrix for
% one-scale CDF(2, 2) transform
% N >= 4 must be even
function Ta = cdfamat(N)
% generate special matrices
I = eye(N/2); S = shift(N/2); Z = zeros(N/2);
split2 = downsamp(N/2);
% prediction step
P = [I Z; -(1/2)*(I+S^(-1)) I];
% update step
U = [I (1/4)*(I + S); Z I];
% normalization step
D = [sqrt(2)*I Z; Z I/sqrt(2)];
% product of all steps
Ta = D*U*P*split2;
```

Save this file as `cdfamat.m`. Test it by generating the 8×8 CDF$(2,2)$ one-scale analysis matrix $\mathbf{T}_{\mathrm{a}}$, a column vector $\mathbf{x}$ whose entries increase linearly, and the transform $\mathbf{y} = \mathbf{T}_{\mathrm{a}}\mathbf{x}$:

```
Ta = cdfamat(8), x = [1:2:15]', y = Ta*x
```

Your matrix `Ta` should be the analysis matrix in Example 3.12 in the textbook divided by $4\sqrt{2}$. The first four entries in $\mathbf{y}$ are the *trend* $\mathbf{s}$ and the last four entries are the *detail* $\mathbf{d}$. Use the explicit form of the matrix $\mathbf{T}_{\mathrm{a}}$ that you just generated to answer the following questions.

1. How many entries of $\mathbf{x}$ are used to calculate a single entry in $\mathbf{s}$? (Don't count entries in $\mathbf{x}$ that are multiplied by a zero from a row of $\mathbf{T}_a$.)

2. How many entries of $\mathbf{x}$ are used to calculate a single entry in $\mathbf{d}$? (Don't count entries in $\mathbf{x}$ that are multiplied by a zero from a row of $\mathbf{T}_a$.)

(b) CDF(2, 2) synthesis matrix: The synthesis matrix $\mathbf{T}_s$ is the inverse of the analysis matrix. From equation (3.19) in the textbook

$$\mathbf{T}_s = \boxed{\text{merge}}\, P^{-1}\, U^{-1}\, D^{-1}.$$

The matrix $\boxed{\text{merge}}$ is the inverse (= transpose) of the orthogonal permutation matrix $\boxed{\text{split}}$. Each of the other matrix inverses are also easy to calculate (this is one of the advantages of the lifting method).

Verify by a hand calculation attached to your lab report that

$$P^{-1} = \begin{bmatrix} I & 0 \\ \frac{1}{2}(I + S^{-1}) & I \end{bmatrix}, \quad U^{-1} = \begin{bmatrix} I & -\frac{1}{4}(I+S) \\ 0 & I \end{bmatrix}, \quad D^{-1} = \begin{bmatrix} \frac{1}{\sqrt{2}} I & 0 \\ 0 & \sqrt{2} I \end{bmatrix}.$$

Write the following function m-file to generate the matrix $\mathbf{T}_s$ (this m-file uses the MATLAB function m-files `shift.m` and `downsamp.m` from Project #2):

```
% Generate N x N synthesis matrix for one-scale CDF(2, 2) transform
% N >= 4 must be even
function Ts = cdfsmat(N)
% generate special matrices
I = eye(N/2); S = shift(N/2); Z = zeros(N/2); merge2 = downsamp(N/2)';
% prediction step
P = [I Z; (1/2)*(I+S^(-1)) I];
% update step
U = [I -(1/4)*(I + S); Z I];
% normalization step
D = [I/sqrt(2) Z; Z sqrt(2)*I];
% product of all steps
Ts = merge2*P*U*D;
```

Save this file as `cdfsmat.m`. Test it by generating the 8×8 CDF$(2, 2)$ one-scale synthesis matrix and checking that it is the inverse of the analysis matrix:

```
Ts = cdfsmat(8), norm(Ts*Ta - eye(8), 'fro')
```

The matrix `Ts` should be the synthesis matrix in Example 3.12 of the textbook, multiplied by $4\sqrt{2}$. The norm value should be (essentially) zero.

(c) CDF(2, 2) basis vectors: In part **(b)** you generated the one-scale CDF$(2, 2)$ synthesis matrix $\mathbf{T}_s$ of size 8×8. The columns of $\mathbf{T}_s$ give the CDF$(2, 2)$ basis for $\mathbf{R}^8$ (see Example 3.12 of the textbook). Column 1 is the *scaling vector*. Observe that columns 2–4 of $\mathbf{T}_s$ are obtained from column 1 by multiplying with S^2, S^4, and S^6, where S is the shift operator. The trend vector in a one-scale CDF$(2, 2)$ wavelet synthesis is a linear combination of these columns. Plot the columns by MATLAB:

```
figure, plot(Ts(:,1), 'r-'), axis([0  9  -.5  1.5]), hold on
plot(Ts(:,2), 'g-')
plot(Ts(:,3), 'b-')
plot(Ts(:,4), 'c-')
```

(see the left half of Figure 3.9 in the textbook). Insert arrows to identify the plots as column 1, column 2, and so on; note that `'r-'` plots red lines, `'g-'` green, `'b-'` blue, and `'c-'` cyan. In the MATLAB figure insert the title Fig. 1: CDF(2, 2) Trend Basis Vectors. Save and print the figure.

Column 5 of $\mathbf{T}_s$ is the *wavelet vector*, and columns 6–8 of $\mathbf{T}_s$ are obtained from column 5 by multiplying with S^2, S^4, and S^6, where S is the shift operator. The detail vector in a one-scale CDF$(2, 2)$ wavelet synthesis is a linear combination of these columns. Plot the columns by MATLAB:

```
figure, plot(Ts(:,5), 'r-'), axis([0  9  -.5  1.5]), hold on
plot(Ts(:,6), 'g-')
plot(Ts(:,7), 'b-')
plot(Ts(:,8), 'c-')
```

(see the right half of Fig. 3.9 in the textbook). Insert arrows to identify the plots as column 5, column 6, and so on. In the MATLAB figure insert the title Fig. 2: CDF(2, 2) Detail Basis Vectors. Save and print the figure.

Question 2. Daub4 Wavelet Transform

You will use MATLAB to explore the results in Sections 3.4.2 and 3.5 of the textbook.

(a) Daub4 Analysis Matrix: Implement the Daub4 one-scale wavelet transform in matrix form for periodic signals $\mathbf{s}$ of even length $N \geq 4$, viewing $\mathbf{s}$ as a column vector. The $N \times N$ analysis matrix is obtained by the lifting method as the product

$$\mathbf{T}_{\mathrm{a}} = D\,U_2\,P\,U_1\,\boxed{\text{split}} \tag{2}$$

(see the flow chart in Figure 3.7 of the textbook). Here $\boxed{\text{split}}$ is the downsampling matrix that separates the signal $\mathbf{s}$ into subsignals $\mathbf{s}_{\mathrm{even}}$ and $\mathbf{s}_{\mathrm{odd}}$ of length $N/2$. After downsampling, the first lifting operation is an update step given by the matrix U_1 in equation (3.21). This is followed by a prediction step given by the matrix P in equation (3.24). Next, there is a second update step that is given by the matrix U_2 in equation (3.26). Finally, there is a normalization step that is performed by the diagonal matrix D in equation (3.27).

Write the following function m-file to generate the matrix $\mathbf{T}_{\mathrm{a}}$ this m-file uses the MATLAB function m-files `shift.m` and `downsamp.m` from Project #2):

```
% Generates N x N analysis matrix for one-scale Daub4 transform
% N >= 4 must be even
function Ta = daub4mat(N)
% generate special matrices
I = eye(N/2); S = shift(N/2); Z = zeros(N/2); split2 = downsamp(N/2);
% first update step
U1 = [I sqrt(3)*I; Z I];
% prediction step
P = [I Z; -(1/4)*(sqrt(3)*I+(sqrt(3)-2)*S) I];
% second update step
U2 = [I -S^(-1); Z I];
% normalization step
D = [(sqrt(3)-1)*I/sqrt(2) Z; Z (sqrt(3)+1)*I/sqrt(2)];
% product of all steps
Ta = D*U2*P*U1*split2;
```

Save this function m-file as `daub4mat.m`. Test it by generating the 8×8 Daub4 one-scale analysis matrix

```
Ta = daub4mat(8)
```

The matrix $\mathbf{T}_{\mathrm{a}}$ is *orthogonal*; this is a special property of the Daub4 transform that is not true for the CDF$(2, 2)$ transform. Check this property by calculating

```
norm(Ta*Ta' - eye(8), 'fro')
```

(the distance between $\mathbf{T_a}\mathbf{T_a^{\mathrm{T}}}$ and the identity matrix). This norm value should be (essentially) zero. Suppose $\mathbf{x} \in \mathbf{R}^8$ and $\mathbf{y} = \mathbf{T}_{\mathrm{a}}\mathbf{x}$. The first four entries in $\mathbf{y}$ are the *trend* vector $\mathbf{s}$, and the last four entries are the *detail* vector $\mathbf{d}$. Use the explicit form of the matrix $\mathbf{T}_{\mathrm{a}}$ that you just generated to answer the following.

1. How many entries of $\mathbf{x}$ are used to calculate a single entry in $\mathbf{s}$? (Don't count entries in $\mathbf{x}$ that are multiplied by a zero from a row of $\mathbf{T}_{\mathrm{a}}$.)

2. How many entries of $\mathbf{x}$ are used to calculate a single entry in $\mathbf{d}$? (Don't count entries in $\mathbf{x}$ that are multiplied by a zero from a row of $\mathbf{T}_a$.)

(b) Two-scale Daub4 wavelet transform: Let $\mathbf{s}_3$ be a signal of length 2^3. To perform a two-scale Daub4 wavelet transform of $\mathbf{s}_3$, we use the pyramid algorithm. First we transform $\mathbf{s}_3$ into a *trend* $\mathbf{s}_2$ and a *detail* $\mathbf{d}_2$, each of length 2^2, using the 8×8 Daub4 analysis matrix. Then we transform $\mathbf{s}_2$ into a trend $\mathbf{s}_1$ and a detail $\mathbf{d}_1$, each of length 2, using the 4×4 Daub4 analysis matrix. We must stop at this point because the Daub4 transform needs input signals of length at least 4.

Generate the two-scale wavelet analysis matrix $\mathbf{W}_a^{(2)}$ for the Daub4 transform, which we denote as `Wa` for MATLAB calculations, as follows (be sure to use the semicolons at the end of lines so that only the matrix `Wa` appears on screen).

```
T3 = daub4mat(8);
I = eye(4); Z = zeros(4,4); T2 = [daub4mat(4)  Z; Z  I];
Wa = T2*T3
```

Check that the matrix `Wa` is orthogonal by calculating `norm(Wa*Wa' - eye(8), 'fro')`; the answer should be (essentially) zero.

Since the analysis matrix is orthogonal, you obtain the *two-scale* Daub4 *synthesis matrix* by

```
Ws = Wa'
```

(c) Two-scale Daub4 basis vectors: Let `Ws` be the 8×8 two-scale Daub4 synthesis matrix from **(b)**. The columns of $\mathbf{W}_s$ give the *two-scale Daub4 basis* for $\mathbf{R}^8$ (see Section 3.5.4 of the textbook). Since $\mathbf{W}_s$ is an orthogonal matrix, this is an orthonormal basis. Column 1 is the *scaling vector*. Observe that column 2 of `Ws` is obtained from column 1 by multiplying by S^4, where S is the 8×8 shift matrix. The trend vector in a two-scale Daub4 wavelet synthesis is a linear combination of these columns. Plot the columns by MATLAB:

```
figure, plot(Ws(:,1), 'r-'), axis([0  9  -1  1]), hold on
plot(Ws(:,2), 'g-')
```

Insert arrows into the figure to label the plots column 1 and column 2. Notice the shift by 4 between the two graphs. Insert the title Fig. 3: Daub4 Two-Scale Trend Basis Vectors. Save and print the figure.

Column 3 of $\mathbf{W}_s$ is the *coarse wavelet vector*. Observe that column 4 of $\mathbf{W}_s$ is obtained from column 3 by multiplying by S^4. The coarse detail vector in a two-scale Daub4 wavelet synthesis is a linear combination of these columns. Plot the columns by MATLAB:

```
figure, plot(Ws(:,3), 'r-'), axis([0 9 -1 1]), hold on
plot(Ws(:,4), 'g-')
```

Insert arrows into the figure to label the plots column 3 and column 4. Notice the shift by 4 between the two graphs. Insert the title Fig. 4: Daub4 Two-Scale Coarse Detail Basis Vectors. Save and print the graph.

Columns 5 of $\mathbf{W}_s$ is the *fine wavelet vector*. Columns 6–8 of $\mathbf{W}_s$ are obtained from column 5 by multiplying by S^2, S^4, and S^6. The fine detail vector in a two-scale Daub4 wavelet synthesis is a linear combination of these columns. Plot the columns by MATLAB:

```
figure, plot(Ws(:, 5), 'r-'), axis([0 9 -1 1]), hold on
plot(Ws(:, 6), 'g-')
plot(Ws(:, 7), 'b-')
plot(Ws(:, 8), 'c-')
```

Insert arrows into the figure to label the plots (column 5, column 6, and so on). Notice the shift by 2 between each graph. Insert the title Fig. 5: Daub4 Two-Scale Fine Detail Basis Vectors. Save and print the graph.

Question 3. Fast Multiscale Daub4 Transform

The matrix formulation of the Daub4 transform and inverse transform in Question 2 is helpful in understanding the theory of this transform and how it is similar to the discrete Fourier transform (we use the

Daub4 basis instead of the Fourier basis; both bases are orthogonal). For numerical calculation, however, the matrix formulation is impractical, since for a signal of length N direct matrix multiplication requires the order of N^2 arithmetic operations. Just as in the case of the Fourier transform, there is a fast implementation of the Daub4 transform through lifting. This algorithm only requires the order of N arithmetic operations.

(a) Implementing one-scale analysis transform: Create the following function m-file:

```
% one-scale Daubechies 4 wavelet transform
% input signal S is a row vector of even length N >=4
function T = daub4wt(S)
N = length(S);
% allocate space in memory
s1 = zeros(1, N/2); d1 = zeros(1, N/2);
% first update for trend
s1 = S(1:2:N-1) + sqrt(3)*S(2:2:N);
% right shift first trend with wrap-around
s1shift = [s1(N/2)  s1(1:N/2-1)];
% first prediction for detail
d1 = S(2:2:N) - sqrt(3)/4*s1 - (sqrt(3)-2)/4*s1shift;
% left shift first detail with wrap-around
d1shift = [d1(2:N/2)  d1(1)];
% second update for trend
s2 = s1 - d1shift;
% normalization of trend
s = (sqrt(3)-1)/sqrt(2)*s2;
% normalization of detail
d = (sqrt(3)+1)/sqrt(2)*d1;
T = [s  d];
```

Save this file as `daub4wt.m`. Notice that we are now writing signals as *row vectors* rather than column vectors. To test this file, let `S = [1  0  0  0]`, calculate `T = daub4wt(S)` and compare `T` with the first column of `daub4mat(4)`.

(b) Implementing one-scale synthesis transform: Create the following function m-file that reverses the steps of the function in part **(a)**.

```
% one-scale inverse Daubechies 4 wavelet transform
% input is a row vector T = [s  d] of even length N >=4
function S = daub4iwt(T)
N = length(T);
% Separate input into trend and detail
s = T(1:N/2); d = T(N/2+1:N);
% allocate space in memory
S = zeros(1, N);
% inverse normalization of trend and detail
s2 = (sqrt(3)+1)/sqrt(2)*s; d1 = (sqrt(3)-1)/sqrt(2)*d;
% inverse of second update for trend; shift first detail left
s1 = s2 + [d1(2:N/2)  d1(1)];
% inverse of first prediction for detail; shift first trend right
S(2:2:N) = d1 + sqrt(3)/4*s1 + (sqrt(3)-2)/4*[s1(N/2)  s1(1:N/2-1)];
% inverse of first update for trend
S(1:2:N-1) = s1 - sqrt(3)*S(2:2:N);
```

Save this file as `daub4iwt.m`. To test this file, let `T = [1 0 0 0]`, calculate `S = daub4iwt(T)` and compare `S` with the first row of `daub4mat(4)`.

(c) Multiscale Daub4 Scaling and Wavelet Vectors: Write the following function m-file:

```
% N-2 scale Daub4 scaling function of length 2^N
function S = daub4scale(N)
T = zeros(1, 2^N); T(1) = 1;
for k = 1:(N-2)
T = daub4iwt(T);
end
S = T;
```

Save this file as `daub4scale.m`. Now write the following function m-file:

```
% N-3 scale Daub4 wavelet function of length 2^N
function S = daub4wave(N)
T = zeros(1, 2^N); T(5) = 1;
for k = 1:(N-3)
T = daub4iwt(T);
end
S = T;
```

Save this file as `daub4wave.m`. Generate and plot a ten-scale Daub4 scaling function and a nine-scale wavelet function (both of length $2^{12} = 4096$):

```
S = daub4scale(12); W = daub4wave(12);
figure, plot(S, 'r-'), hold on, plot(W, 'g-')
```

Compare with Fig. 3.13 in the textbook, which shows the four-scale scaling and wavelet vectors of length 32. The jagged appearance is due to the continuous but non-differentiable scale and wavelet functions that are created when the number of scales goes to infinity. In this respect Daub4 scaling and wavelet functions are very different from the cosine and sine functions of continuous Fourier analysis. Label the graphs as scaling function and wavelet. Insert the title Fig. 6: Daub4 Nine-Scale Wavelet and Ten-Scale Scaling Function. Save and print the graph.

Question 4. Signal Processing with the CDF$(2, 2)$ Transform

You will apply a three-scale CDF$(2, 2)$ transform to analyze a signal, using the filter-bank approach (instead of matrix multiplication) to calculate the wavelet transforms (see Sections 4.5 and 4.8 of the textbook). For a test signal take the same sine wave with static noise and pops as in Project #3 Question #3. Since the detail for a one-scale CDF$(2, 2)$ transform is zero on linear functions (except near end points–see Figure 3.10 of the textbook), this transform should yield better results than the Haar transform on signals that are relatively smooth.

(a) Analysis of Synthetic Signals: Sample the signal $s(t) = \sin(4\pi t)$ at $2^9 = 512$ equidistant points in $0 \le t \le 1$ to obtain a discrete signal $\mathbf{s}_9$:

```
s9 = sin([1:512]*4*pi/512);
```

Now add some random noise and random pops to the signal $\mathbf{s}_9$. Generate a two-component vector `pop` whose entries are integers between 100 and 400:

```
pop = round(100 + 300*rand(2,1))
```

The following code adds 1 to the two components of $\mathbf{s}_9$ whose indices are given by the numbers in `pop`, adds some random noise throughout, and plots the resulting signal:

```
s9(pop(1)) = s9(pop(1)) + 1;
s9(pop(2)) = s9(pop(2)) + 1;
s9 = s9 + 0.1*randn(1, 512);
figure, plot(s9), axis([0  550  -1.5  1.5])
```

The location of the two pops should be clear in the plot. Insert the title Fig. 7: Signal with Pops and Static in the MATLAB figure. Save and print the figure.

To calculate a three-scale CDF$(2,2)$ wavelet transform of the signal, we use the *filter bank* implementation in the *Uvi_Wave* toolbox (where the CDF family of transforms are named `wspline`):

```
[h,g,rh,rg] = wspline(2,2)
```

The four *filter vectors* $\mathbf{h}, \mathbf{g}, \mathbf{rh}, \mathbf{rg}$ are obtained from the one-scale CDF$(2,2)$ analysis and synthesis matrices; to see use the m-files `cdfamat.m` and `cdfsmat.m` from Project #3 to generate these matrices:

```
Ta = cdfamat(6), Ts = cdfsmat(6)
```

Compare the filters and the matrices to answer the following questions (see Example 4.24 in the textbook):

1. How are the rows of `Ta` obtained from the vectors `h` and `g`?
2. How are the columns of `Ts` obtained from the transposes of the row vectors `rh` and `rg`?

Now obtain the three-scale CDF$(2,2)$ transform **st** of $\mathbf{s}_9$ and plot the result by

```
st = wt(s9,h,g,3);
figure, isplit(st,3,'','r.')
```

The vector `st` from the UVI_WAVE function `wt` is the CDF$(2,2)$ transform of `s9`. It is a concatenation of the four vectors making up the transform. The UVI_WAVE function `isplit` separates `st` into its four parts (as you did by hand for the Haar transform in Project #3 Question #3(a)). It plots the 3-scale trend coefficients of $\mathbf{s}_6$ and detail coefficients of $\mathbf{d}_6$ in the first and second graphs (these are vectors of length $64 = 2^{9-3}$). The third graph gives the 2-scale detail coefficients of $\mathbf{d}_7$ (a vector of length $128 = 2^{9-2}$). The fourth graph gives the 1-scale detail coefficients of $\mathbf{d}_8$ (a vector of length $256 = 2^{9-1}$). Put title Fig. 8: Three-Scale CDF$(2,2)$ Wavelet Coefficients at the top of the figure, and then put titles s6, d6, d7, d8 to the left of each of the plot windows. Save and print this figure.

The *multiresolution analysis* of the signal is obtained by taking the inverse CDF$(2,2)$ transforms of the trend and detail coefficient vectors. Do this using the UVI_WAVE function `multires`:

```
y = multires(s9,h,rh,g,rg,3);
figure, split(y)
```

Here `y` is a 4×512 matrix whose rows contain $\mathbf{d}_8, \mathbf{d}_7, \mathbf{d}_6$, and $\mathbf{s}_6$; the `split` command displays a graph for each row. Put the title Fig. 9: Multiresolution CDF$(2,2)$ Analysis of Signal at the top of the figure, and then put titles s6, d6, d7, d8 to the left of each of the plot windows. Save and print this figure.

(b) Filtering and Compression: Just as in the case of the Haar transform in Project #3 Question **3**, the pops are not evident in the trend but are obvious in the details. A crude way of filtering the signal to remove the pops and some of the static noise is to use only the trend $\mathbf{s}_6$ and detail $\mathbf{d}_6$ (rows 3 and 4 of the matrix `y`):

```
sfilter = y(3,:)  + y(4,:);
figure, subplot(311), plot(s9, 'r-'), hold on
subplot(312), plot(sfilter, 'g-')
```

To preserve the pops in the signal but remove the static noise we will need to use all the detail vectors. From the MATLAB figure in (a) you can see that the only large coefficients come from the two pops. We will compress and denoise the signal by setting to zero all coefficients in the detail vectors that are less than 0.2 in absolute value (do not change the transform coefficients of the trend vector $\mathbf{s}_6$). To do this efficiently, apply the `threshold` m-file from Project #3 Question **3(c)** to the detail entries (in positions 65–512 of the transform `st`):

```
cst = st;
cst(65:512) = threshold(st(65:512), 0.2);
```

Then calculate the 3-scale inverse CDF$(2,2)$ transform of the compressed vector `cst` and plot it in the same window as the original signal and the filtered signal:

```
cs = iwt(cst,rh,rg,3);
subplot(313), plot(cs, 'b-')
```

Label the plots as original signal, filtered signal, and filtered signal with pops. Put the title Fig. 10: Filtering by Removing Wavelet Details and Noise. Save and print the figure.

Final Editing of Lab Write-up: After you have worked through all the parts of the lab assignment, you will need to edit your diary file. Remove all errors and other material that is not directly related to the questions. Your write-up should only contain the keyboard input and the MATLAB output (including Fig. 1-10), together with the answers to the questions that you have written.

Preview the document before printing and remove unnecessary page breaks and blank space. Put your name and four-digit ID number on each page. (If you have difficulty doing this using your text editor, you can write this information by hand after printing the report.)

Do not include the codes for the m-files in your lab writeup, but be sure to save these m-files on your disc for future use.