Revised 1 February 2017

Project # 3: Haar Wavelet Transform

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

- Haar wavelet basis, Haar analysis matrix, and Haar synthesis matrix.
- Fast Haar transform implemented by lifting.
- Applications of the Haar transform

Preliminaries

Reading from Textbook: Before beginning your MATLAB work, read Sections 3.2 and 3.3 of the textbook.

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 lab3.txt
```

followed by the `Enter` key. Now each computation you make in MATLAB will be saved in your directory in a text file named `lab3.txt`. You can then edit this file using your favorite plain text editor (such as Emacs, Notepad, WordPad, 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 #3 -- Haar Wavelet Transform
%
```

at the MATLAB prompt. Type

```
format compact
```

so that your diary file will not have unnecessary spaces. 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.

Random Seed: When you start your MATLAB session, 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

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. Haar Basis and Haar Transform Matrices

(a) Haar Multiresolution Basis: This basis for $\mathbf{R}^8$ is built from the *scaling vector* h_0 (all components 1), and the *wavelet vector* h (see Section 3.3.1 of the textbook). Write the following function m-file to generate the wavelet vectors at various levels:

```
% haar wavelet of length 2^n and level k
function h = hwavelet(n,k)
% create vector of length 2^n
h = zeros(2^n, 1);
h(1:2^(n-k),1) = 1;
h(2^(n-k)+1:2^(n-k+1),1) = -1;
```

Save this file as `hwavelet.m`. Then use it to create the Haar multiresolution basis for $\mathbf{R}^8$: First generate the *scaling vector* h_0 and the basic wavelet vectors h_1, h_2, and h_4 at levels 1, 2, and 3:

```
h0 = ones(2^3, 1)
h1 = hwavelet(3,1)
h2 = hwavelet(3,2)
h4 = hwavelet(3,3)
```

The other wavelet vectors at level 2 and 3 are obtained by shifting the vectors h_2 and h_4. Use the `shift.m` function from MATLAB Project #2 to generate the 8×8 shift matrix S, and then apply this matrix to h_2 and h_4.

```
S = shift(8)
h3 = S^4 * h2
h5 = S^2 * h4,
h6 = S^4*h4,
h7 = S^6*h4
```

(b) Haar Transform Matrices: Now make the Haar basis vectors for $\mathbf{R}^8$ that you generated in *(a)* into the columns of the $2^3 \times 2^3$ three-scale *Haar synthesis matrix* $\mathbf{W}_\mathbf{s}^{(3)}$:

```
Ws = [h0  h1  h2  h3  h4  h5  h6  h7]
```

Calculate `Ws'*Ws`. What does this tell you about the orthogonality properties and squared lengths of the Haar basis vectors?

The Matlab command `diag(v)` creates a matrix with the entries of the column vector `v` on the diagonal and all other entries zero. Use this command to create a diagonal matrix `D` such that `Wa = D^(-1)*Ws'` is the *Haar analysis matrix* $\mathbf{W}_\mathbf{a}^{(3)}$ in equation (3.8) of the textbook. (HINT: Remember that *left* multiplication by a diagonal matrix multiplies each *row* by the corresponding diagonal entry.) Use MATLAB to verify that `Wa` is the inverse matrix to `Ws`.

(c) Data Compression: Generate a random digital signal `u` of length 8 and plot it as a solid red line (we consider `u` as a sample of an analog signal and use linear interpolation between the data points in the graph):

```
u = round(50*rand(8,1))
plot(u, 'r-'), axis([1  8  0  60]), hold on
```

Calculate the *Haar transform* `v = Wa*u`. Then create a *compressed* vector `vc5` by replacing any entries in `v` whose absolute value is ≤ 5 by 0. For example, if `v(2) = -3.5`, then set `vc5(2) = 0`). This is the *data compression step*. This is a *nonlinear* transformation, just like quantization.

- Calculate the *compression ratio* (the number of nonzero entries in the vector `v` divided by the number of nonzero entries in the compressed vector `vc5`).

Calculate the inverse transform `uc5` of the compressed vector and plot the graph of `uc5` on the same axes as the original signal `u` (using a dashed green line):

```
uc5 = Ws*vc5
plot(uc5, 'g--')
```

You should get a graph similar to Fig. 3.5 in the textbook (of course, your random signal is not the same as the signal in Fig. 3.5, so your graph will differ in details). Notice that the graph of `uc5` is fairly close to the graph of the uncompressed signal `u`.

- Calculate `100*norm(u - uc5)^2/norm(u)^2` to measure the relative percentage difference in energy between the compressed and uncompressed signals.

Finally, perform a more drastic compression on the signal `u`. Create another compressed Haar transform vector `vc10` by replacing all the entries in the Haar transform `v` whose absolute value is ≤ 10 by 0. (*Caution:* You must modify the *transform vector* `v`, not the original signal `u`.)

- Calculate the new compression ratio for `vc10`.

Calculate the inverse transform `uc10` of `vc10` and plot the graph of `uc10` on the same axes as the original signal `u` (using a dotted blue line):

```
uc10 = Ws*vc10
plot(uc10, 'b:')
```

The new graph will not be close to the original graph as the previous compression (unless your random transform vector `v` had no entries with absolute value ≤ 5). However, this graph follows the general trends of the original graph (this is the advantage of the wavelet method of compression).

- Calculate `100*norm(u - uc10)^2/norm(u)^2` to measure the relative percentage difference in energy between the compressed and uncompressed signals.

Insert arrows and labels on the graphs, indicating the original signal, the compression thresholds (5 and 10) of the two compressed signals, and the relative differences in energies of the two compressed signals. Insert the title Fig. 1: Haar Wavelet Compression of Length 8 Signal. Print the graph and include it with your report.

Question 2. Fast Multiscale Haar Transform

The matrix formulation of the Haar transform and inverse Haar transform in Question #1 is helpful in understanding the theory of this transform and how it is similar to the discrete Fourier transform (we use the Haar 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 Haar transform through lifting. This algorithm only requires the order of N arithmetic operations.

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

```
% one-step discrete Haar wavelet transform
function T = dwthaar(Signal)
N = length(Signal); s = zeros(1, N/2); d = s;
for n=1:N/2
s(n) = 1/2*(Signal(2*n-1) + Signal(2*n));
d(n) = Signal(2*n-1) - s(n);
end
T = [s, d];
```

Save this file as `dwthaar.m`. Notice that we are now writing signals as *row vectors* rather than column vectors. Test the file on the signal

```
Signal = [56   40   8   24   48   48   40   16]
```

that is used in Table 3.1 in Section 3.2 of the textbook. Calculate

```
y = dwthaar(Signal)
```

The row vector `y` should be the same as the *second row* in Table 3.1.

(b) Inverting a one-scale Haar transform: Create the following function m-file:

```
% one-step discrete inverse Haar wavelet transform
function R = iwthaar(T)
N = length(T); R = zeros(1, N);
for n=1:N/2
R(2*n - 1) = T(n) + T(N/2 + n);
R(2*n) = T(n) - T(N/2 + n);
end
```

Save this file as `iwthaar.m`. Test it on the vector `y` by calculating

```
z = iwthaar(y)
```

You should obtain the row vector `Signal`.

(c) Complete Haar Wavelet Decomposition: To carry out a J-step Haar wavelet decomposition on a signal of length $N = 2^K$ (where $J \leq K$), we must apply the one-scale Haar transform J times to obtain a $J \times N$ matrix. The bottom row of the matrix will be the J-step Haar wavelet decomposition. Create the following function m-file to do this:

```
% Discrete Haar wavelet decomposition to level J
function T = haarwave(Signal, J)
N = size(Signal,2); T = zeros(J, N); L = N;
T(1,:)  = dwthaar(Signal);
for j=2:J
L = L/2;
T(j, 1:L) = dwthaar(T(j-1, 1:L));
T(j, L+1:N) = T(j-1, L+1:N);
end
```

Save this file as `haarwave.m`. Then test it by calculating

```
Table = haarwave(Signal, 3)
```

You should get a 3×8 matrix `T` whose bottom row is given in Table 3.5 of the textbook.

Remark. Notice that the first row of `T` is the one-scale Haar wavelet transform of the input signal `y`. The `for` loop implements the lifting step to obtain row `j` of `T` from row `j-1`. It applies the one-scale Haar wavelet transform to the initial segment of the row vector that is one-half the length of the segment used for the previous row.

(d) Inverting the level J Haar Wavelet Decomposition: Create the following function m-file:

```
% Inverse of level J Discrete Haar wavelet decomposition
function T = ihaarwave(y, J)
N = size(y,2); T = zeros(J, N); L = N/2^(J-1);
T(J,1:L) = iwthaar(y(1:L));
T(J, L+1:N) = y(L+1:N);
for k = J-1:-1:1
L = 2*L;
```

```
T(k, 1:L) = iwthaar( T(k+1, 1:L) );
T(k, L+1:N) = T(k+1, L+1:N);
end
```

Save this m-file as `ihaarwave.m`. Test it by entering

```
y = [35  -3  16  10  8  -8  0  12]
T = ihaarwave(y,3)
```

The first row of the 3×8 matrix `T` should be the original signal in Table 3.1.

Remark. Note that entries `1:L` of row `J` of `T` are the one-scale inverse Haar transform of `y`, where `L = N/2^(J-1)`. The remaining entries of row `J` are the same as the corresponding entries of `y`. The `for` loop repeats this inversion algorithm at each level from row `J-1` down to row 1 of `T`, doubling the length `L` of the inverse transform vector at each step.

Question 3. Signal Processing with the Haar Transform

You will carry out some signal processing as in Section 3.3.2 of the textbook.

(a) Analyzing a synthetic signal: Consider the analog signal $s(t) = \sin(4\pi t)$. Sample this signal at $2^9 = 512$ equidistant points in $0 \le t \le 1$ to obtain a discrete signal called $\mathbf{s}_9$:

```
s9 = sin([1:512]*4*pi/512);
figure, plot(s9), axis([0  550  -1.2  1.2])
```

Insert the title Fig. 2: Sine Wave Signal, 512 samples. Save and print the figure.

Now calculate a three-scale Haar wavelet transform of $\mathbf{s}_9$ and plot the transform `hws9` as a bar graph:

```
T = haarwave(s9, 3);
hws9 = T(3,:);
figure, bar(hws9, 0.1)
```

Insert the title Fig. 3: Three-stage Haar transform of Sine Wave Signal, 512 samples. Save and print the graph.

Remark. If you had created a 512×512 Haar analysis matrix $W_a^{(9)}$ following the method of Question **#1(c)**, then you could have obtained the column vector `(hws9)'` as the product of the 512×512 matrix $W_a^{(9)}$ and the vector $\mathbf{s}_9$. This is very inefficient, however, compared to the calculation using the fast Haar transform, which uses the 3×512 matrix `haarwave(s9, 3)`.

To interpret the graph of the Haar transform of the signal, consider the row vector `hws9` as a function of the row index j (for $1 \le j \le 512$). It is made up of a four parts:

Level-3 trend:	$\mathbf{s}_6$	for $1 \le j \le 64$	(length 2^6)
Level-3 detail:	$\mathbf{d}_6$	for $65 \le j \le 128$	(length 2^6)
Level-2 detail:	$\mathbf{d}_7$	for $129 \le j \le 256$	(length 2^7)
Level-1 detail:	$\mathbf{d}_8$	for $257 \le j \le 512$	(length 2^8)

Create row vectors in MATLAB for each of these parts of the transform and plot the trend $\mathbf{s}_6$ as a bar graph:

```
s6 = hws9(1:64); d6 = hws9(65:128);
d7 = hws9(129:256); d8 = hws9(257:512);
figure, bar(s6, 0.1)
```

Notice that the level-3 trend $\mathbf{s}_6$ is a coarse version of the graph of the original signal (with a sampling rate of $64 = 512/2^3$). Insert the title Fig. 4: Level-3 Trend of Signal. Save and print the figure.

(b) Approximating a signal: Create an *approximate version* `fhws9` of the Haar transform using only the level-3 trend $\mathbf{s}_6$ and level-3 detail $\mathbf{d}_6$ of the signal (zero out the level-1 and level-2 details). Then take the inverse Haar transform `fs9` of `fhws9` and plot it:

```
fhws9 = hws9; fhws9(129:512) = 0;
fs9 = ihaarwave(fhws9, 3);
figure, plot(fs9(1,:)), axis([0  550  -1.2  1.2])
```

The graph is an approximation to the original signal. The first row `fs9(1,:)` of the 3×512 matrix `fs9` is the projection of `s9` onto the subspace spanned by the first 128 Haar wavelet basis vectors (out of the total of 512). Although these are only one-fourth of the total basis vectors, they capture most of the information in a smooth signal such as a low-frequency sine wave. Calculate the relative approximation error

```
error = norm(s9 - fs9(1,:))/norm(s9)
```

Insert the title Fig. 5: Approximation of Signal Using Level-3 Trend and Detail. Save and print the figure.

(c) Compressing a noisy signal: Now add some *pops* to the signal $\mathbf{s}_9$ to indicate some event such as a loud noise whose timing you want to determine. Do this by first generating a random two-component vector `pop` whose entries are integers between 100 and 400.

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

Now create a signal with pops by adding 1 to the two components of $\mathbf{s}_9$ whose indices are given by the numbers in `pop`:

```
ps9 = s9;
ps9(pop(1)) = s9(pop(1)) + 1;
ps9(pop(2)) = s9(pop(2)) + 1;
```

Finally, add some normally-distributed *random static noise* and plot the noisy version of the original signal:

```
nps9 = ps9 + 0.1*randn(1, 512);
figure, plot(nps9), axis([0  550  -1.5  1.5])
```

The location of the two pops should be clear in the graph. Insert the title Fig. 6: Signal with Pops and Static. Save and print the figure.

Use the Haar transform to compress the signal while still retaining the information about time location of the pops. This is something that is not possible with the discrete Fourier transform; recall the train whistle analysis from Project #2 (Example 2.10 in the textbook).

Calculate a three-scale Haar wavelet transform of the noisy signal with pops $\mathbf{nps}_9$:

```
T = haarwave(nps9, 3);
hwnps9 = T(3,:);
```

As before, create row vectors in MATLAB for each of the four parts of the transform:

```
s6 = hwnps9(1:64); d6 = hwnps9(65:128);
d7 = hwnps9(129:256); d8 = hwnps9(257:512);
```

Plot bar graphs of the trend vector $\mathbf{s}_6$ and the detail vectors $\mathbf{d}_6$, $\mathbf{d}_7$, and $\mathbf{d}_8$ in four windows of the same figure:

```
figure, subplot(2,2,1), bar(s6, 0.1)
subplot(2,2,2), bar(d6, 0.1)
subplot(2,2,3), bar(d7, 0.1)
subplot(2,2,4), bar(d8, 0.1)
```

Notice that the pops are mostly hidden in the level-3 trend $\mathbf{s}_6$, and may or may not show in the level-3 detail $\mathbf{d}_6$ because of the random noise. However, they are the very clear in the details $\mathbf{d}_7$ and $\mathbf{d}_8$. Put title Fig. 7 Trend and Details of Signal with Noise and Pops at the top of the figure, and then put titles s6, d6, d7, d8 to the right of each of the plot windows (s6 is the upper left plot, d6 is the upper right, d7 is the lower left, and d8 is the lower right). Save and print this graph.

In the Haar transform most of the coefficients in the detail vectors $\mathbf{d}_7$ and $\mathbf{d}_8$ are less than 0.1 in absolute value, and are largely due to the random noise. The only significant coefficients come from the two pops.

Compress and remove noise from the signal by setting to zero all entries in $\mathbf{d}_7$ and $\mathbf{d}_8$ that are less than 0.1 in absolute value (do not change the trend vector $\mathbf{s}_6$ or the detail vector $\mathbf{d}_6$). You did this by hand in Question **#1(c)**; however, now we have need to modify a vector with $512 - 128 = 384$ entries. To do this easily, create the following function m-file:

```
% threshold truncation function
% Zeros out all components of row vector x smaller than the level parameter
function y = threshold(x, level)
y = x.*(abs(x) >= level);
```

Save this file as `threshold.m`. Notice the period before * in the formula for `y`; this makes the product an entry-by-entry multiplication. The factor in parentheses after .* is a *logic vector*: its entries are 1 when the inequality is true for the corresponding entry of the vector `abs(x)` of absolute values of `x`, and 0 otherwise. Test this m-file by typing

```
Signal
threshold(Signal, 10)
```

(where `Signal` is the row vector from Question **#2(a)**). The answer should be a vector in which each component of `Signal` that is less than 10 has been replaced by 0.

Execute the following code which creates a row vector of length 384 by concatenating the detail vectors $\mathbf{d}_7$ and $\mathbf{d}_8$, removes the small components using the `threshold` function, and then concatenates this compressed row vector with $\mathbf{s}_6$ and $\mathbf{d}_6$ to obtain the compressed Haar wavelet transform of $\mathbf{s}_9$:

```
detail = [d7  d8];
cdetail = threshold(detail, 0.1);
chws9 = [s6  d6  cdetail];
```

To calculate the compression ratio in this case, count the number of nonzero coefficients in the vector `cdetail` by

```
count = ones(1, 384).*(abs(cdetail)>0);
ratio = 512/(128 + sum(count))
```

Finally, take the 3-scale inverse transform of the compressed Haar wavelet transform and plot the compressed version $\mathbf{cs}_9$ of the noisy signal $\mathbf{s}_9$:

```
T = ihaarwave(chws9, 3);
cs9 = T(1,:);
figure, plot(cs9), axis([0  550  -1.5  1.5])
```

Note that the graph of $\mathbf{cs}_9$ shows the main features of the signal with the two pops, namely the slow oscillation of the main signal and the location of the pops. Some of the static noise has been removed (other wavelet transforms are more efficient at removing such noise). Insert the title Fig. 8: Denoised and Compressed Signal with Pops and Static. Insert labels on the graph indicating the compression ratio and the locations of the pops. Save and print the graph.

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 Figures 1-8), 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 in your workspace for future use.