HOMEWORK ASSIGNMENT #8

Due Fri. Mar. 9, 2007 (in class)

1. Here we will write a Matlab program that takes the coefficients $\{h[n], n = 0 \dots N-1\}$ used in the scaling equation $\phi(t) = \sqrt{2}\sum_{n=0}^{N-1} h[n]\phi(2t-n)$ to calculate samples of the scaling function $\phi(t)$ on a fine sampling grid $\mathcal{T}$ using the "cascade" algorithm. Assume

$$\mathcal{T} = \{m2^{-9} : m \in \mathbb{Z}\} \cap [0, N-1]$$

which happens to correspond to ten iterations of the algorithm. After calculating $\phi(t)$ for $t \in \mathcal{T}$, the code will calculate the wavelet $\psi(t)$ for $t \in \mathcal{T}$ via the wavelet scaling equation $\psi(t) = \sqrt{2}\sum_{n=0}^{N-1} g[n]\phi(2t-n)$. Finally, the code will plot the CTFT magnitudes $|\Phi(\Omega)|$ and $|\Psi(\Omega)|$ over a specified range of frequencies. Here are some details:

- Your program should generate only two figures, each containing a number of subplots:
  1. The first figure shows the first five iterations of the cascade algorithm along with the final iteration. See Fig. 1 on the following page.
  2. The second figure plots $\phi(t)$ versus $t \in \mathcal{T}$ in the upper left corner, samples of $|\Phi(\Omega)|$ over the frequency range $0 \le \frac{\Omega}{2\pi} \le 6$Hz in the upper right corner, $\psi(t)$ versus $t \in \mathcal{T}$ in the bottom left corner, and samples of $|\Psi(\Omega)|$ over $0 \le \frac{\Omega}{2\pi} \le 6$Hz in the bottom right corner. See my Fig. 2.
- The CTFTs should be calculated with a suitably long FFT. This can be done via approximation of the CTFT integral by a Reimann sum (as you learned in your first calculus class). I found that an FFT length of 32768 generated a smooth CTFT over the region of interest.
- Matlab's Wavelet Toolbox comes with data files containing coefficents $\{h[n]\}$ for various popular wavelets. For some reason these coefficients are *not* scaled so that $\sum_n |h[n]|^2 = 1$ and $\sum_n h[n] = \sqrt{2}$. Use the following code fragment to load and scale the coefficients from the `.mat` made available on the course web page. (Replace the underlined word below with the appropriate coefficient filename.)

```
structure = load('db2.mat');
varnames = fieldnames(structure);
hh = getfield(structure,varnames{1});
h = hh(:)/sum(hh)*sqrt(2);
```

  You are not allowed to use any other functions from the Matlab Wavelet Toolbox.
- After you get your program working, generated the two figures described above for the coefficient sets `db1.mat`, `db2.mat`, and `db8.mat`. (These are the "Daubechies" wavelets, and `db1.mat` is also the Haar wavelet which is good for debugging your code.) Note that your code will have to operate for coefficient sets of various lengths N.

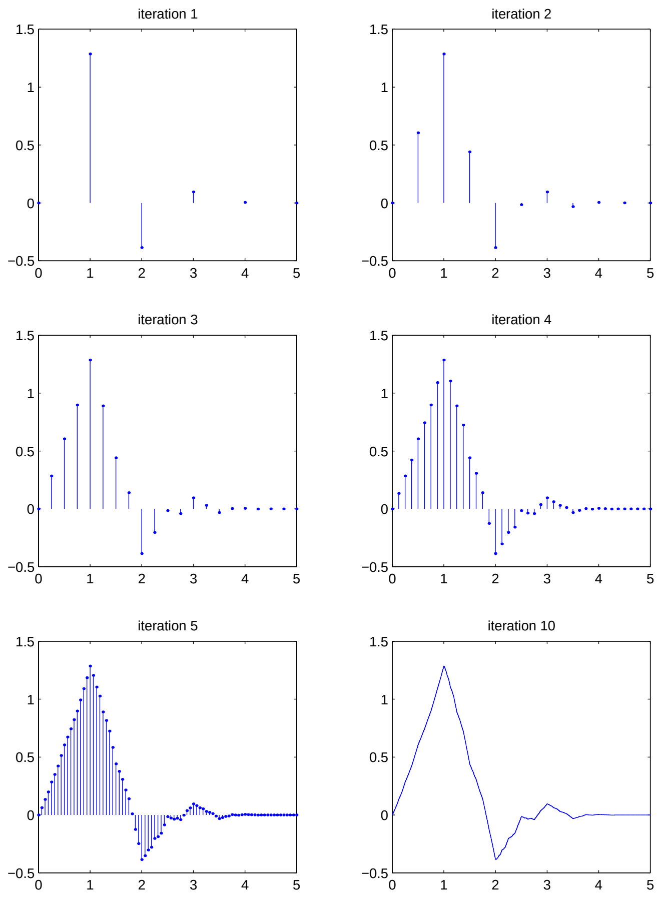

Figure 1: Example of the first figure to be generated by your program.

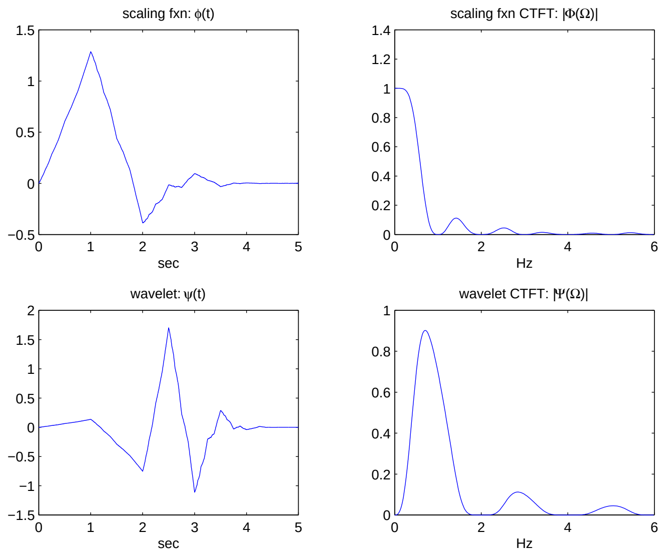

Figure 2: Example of the second figure to be generated by your program.

2. The goal of this problem is to get an appreciation for the Daubechies coefficients and what are known as the "regularity conditions". Consider the coefficients

$$h[n] = \begin{cases} 1/\sqrt{2} & n = 0, 3, \\ 0 & \text{else} \end{cases}$$

as used in the scaling equation. Do these coefficients lead to an orthonormal set $\{\phi(t-n), n \in \mathbb{Z}\}$? Attempt to recursively determine the corresponding scaling function using your code from the previous problem. As the number of iterations in the cascade algorithm approaches infinity, do we get a meaningful scaling function and wavelet?

3. The wavelet transform leads to an effective way to "denoise", i.e., detect and extract unwanted noise in a signal. The key assumption is that the wavelet transform of a noiseless signal has many (nearly) zero-valued wavelet coefficients and that noise makes these coefficients non-zero but small in magnitude. Therefore, small-valued wavelet coefficients contain more noise energy than signal energy, and thus by deleting them we can increase the signal to noise ratio. In this problem we will write Matlab code that performs the discrete wavelet transform of a noisy signal, thresholds the resulting wavelet coefficients, and reconstructs an estimate of the noiseless signal from the thresholded coefficients. (You are not allowed to use any commands from the Wavelet Toolbox.)

   (a) Implement a D-level wavelet analysis/synthesis filterbank pair based on circular (rather than linear) convolutions that works for arbitrary filter coefficients $\{h[n]\}$. Test your code with various Daubechies filters and various D on a randomly-generated test signal to verify that your system is perfectly reconstructing.

   The FFT presents a way to implement circular convolution, demonstrated below for convolution of the sequence $x[n]$ with the shorter sequence $h[n]$.

   ```
   a_new = ifft( fft(x).*fft(h,length(x)) );
   ```

   If one of the sequences is nocausal, you will need to circularly shift the output using `wshift`. I suggest organizing your transform coefficients using the Matlab `wavedec` convention (see `http://www.mathworks.com/access/helpdesk/help/toolbox/wavelet/ch02_u11.shtml#997029`)

   If debugging is needed, I suggest first trying the Haar coefficients with the input sequence `[1:8]`. The DWT matrix can be easily contructed for this case. When that works, repeat with the `db2.mat` coefficients...

   (b) Modify your code so that any transform coefficients whose magnitude is less than the value `thresh` are zeroed. Using the clean and noisy waveforms in the file `steps.mat`, experiment with various wavelets, various D, and various thresholds. What choices seem to give a denoised waveform closest to the clean original? Explain your findings.

   (c) Prepare a plot similar to Fig. 3 demonstrating your best design.

   (d) Repeat (b) and (c) for the signals in `chirp.mat`.

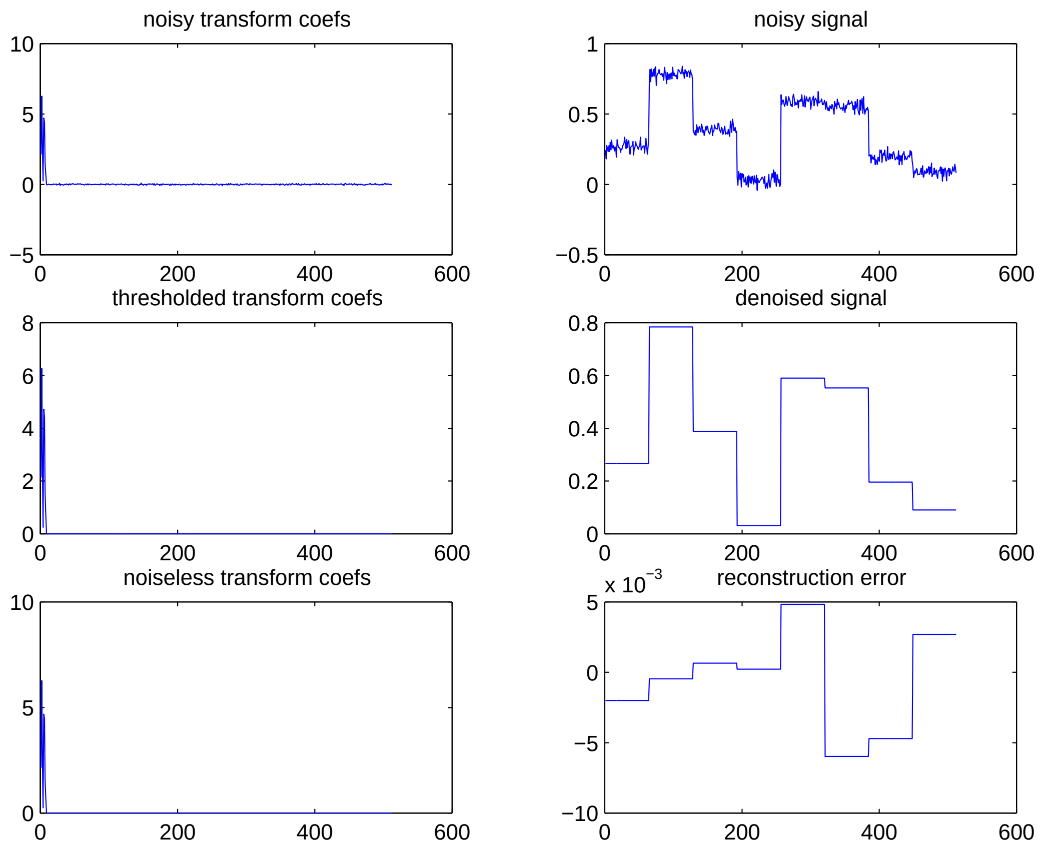

Figure 3: Example of wavelet denoising.