THE WAVELET TRANSFORM: A SHORT TUTORIAL AND SIMPLE APPLICATIONS IN SIGNAL DENOISING AND SPEECH RECOGNITION

Essaid Bouktache
Department of Electrical and Computer Engineering Technology
Purdue University Calumet
Hammond, IN, 46323

Abstract

This paper is a brief tutorial and an illustration of some applications of the wavelet transform that can be introduced in an undergraduate curriculum, preferably in a laboratory environment. It introduces the reader to some basic principles of both the continuous and the discrete wavelet transforms. These principles are illustrated and some previously established results are given. Two application examples that utilize the MATLAB Wavelet Toolbox are also presented that involve the Discrete Wavelet Transform in signal denoising and speech recognition.

Introduction

The wavelet transform is a powerful mathematical tool used for the analysis and synthesis of signals and images. Wavelet transforms are increasingly applied in fields as diverse as telecommunications, biology, medicine, and many others. For example, they can be used for the extraction of speech in hearing aids, for the early detection of breast cancer, for medical image compression, for signal classification and de-noising, and much more.

The wavelet transform has become the preferred alternative to the Fourier transform methods, such as the complex-valued FFT (Fast Fourier Transform) or the real-valued DCT (Discrete Cosine Transform) for two main reasons. First, although the Fourier transforms work well with signals that are stationary (whose statistical parameters do not vary with time) and well behaved in time, but they fail with signals that exhibit discontinuities and sharp edges. Unfortunately, real-word signals fall into this category. Second, the Fourier analysis is lacking time information along with its frequency information, as the example shown in Figure 1 illustrates.

In contrast, the wavelet transform overcomes some of the limitations of the Fourier transforms by its powerful abilities to analyze signals that are nonstationary in nature and nonperiodic. For example, it works well with images, where discontinuities are always present. In addition, because it uses position and scaling as parameters, the wavelet transform reveals not only frequency content but also the time locations of various occurrences within the signal, as shown in Figure 2, where the previous example is analyzed using the Continuous Wavelet Transform to show the differences between the two methods.

Both the Fourier and the wavelet transforms try to decompose a signal in terms of basis functions. The Fourier transform uses sinusoids at different frequencies as basis functions (sines and cosines at related frequencies). Each basis (sinusoid) is of infinite duration. In comparison, the wavelet transform uses *wavelets* that are of finite duration as basis functions. These wavelets differ from each other by varying the scale (expansion or compression) and time position (or shifting) of a single given wavelet prototype.

The following sections address some differences between the Fourier and the wavelet transforms, and include some basic equations that are known with regards to the Continuous Wavelet Transform (CWT) and the Discrete Wavelet Transform (DWT). Finally, two

application examples that utilize the MATLAB Wavelet Toolbox are included to illustrate some of the properties of the DWT: One in signal denoising and the other in speech recognition.

Fourier Transform vs. Wavelet Transform

As mentioned earlier, the main drawback of the FFT is its lack of time information in its frequency representation of a given signal. For example, as can be seen in Figure 1, a signal that consists of a sine wave at 100 Hz followed by another at 500 Hz will have the same FFT as a signal that has these frequencies reversed in their occurrences in time. Hence, the time information is lost during the time to frequency domain transformation.

On the other hand, the continuous wavelet transform (CWT), when applied to the same signals, reveals both time (space) and frequency (scale) content, as seen in Figure 2.

Wavelets vs. Sinewaves

Conventionally, a signal is analyzed or synthesized by the Fourier Transform into sinusoids of different frequencies, as shown in Figure 3.

Wavelets, on the other hand, are of limited duration, and can be irregular and asymmetric, as shown in Figure 4.

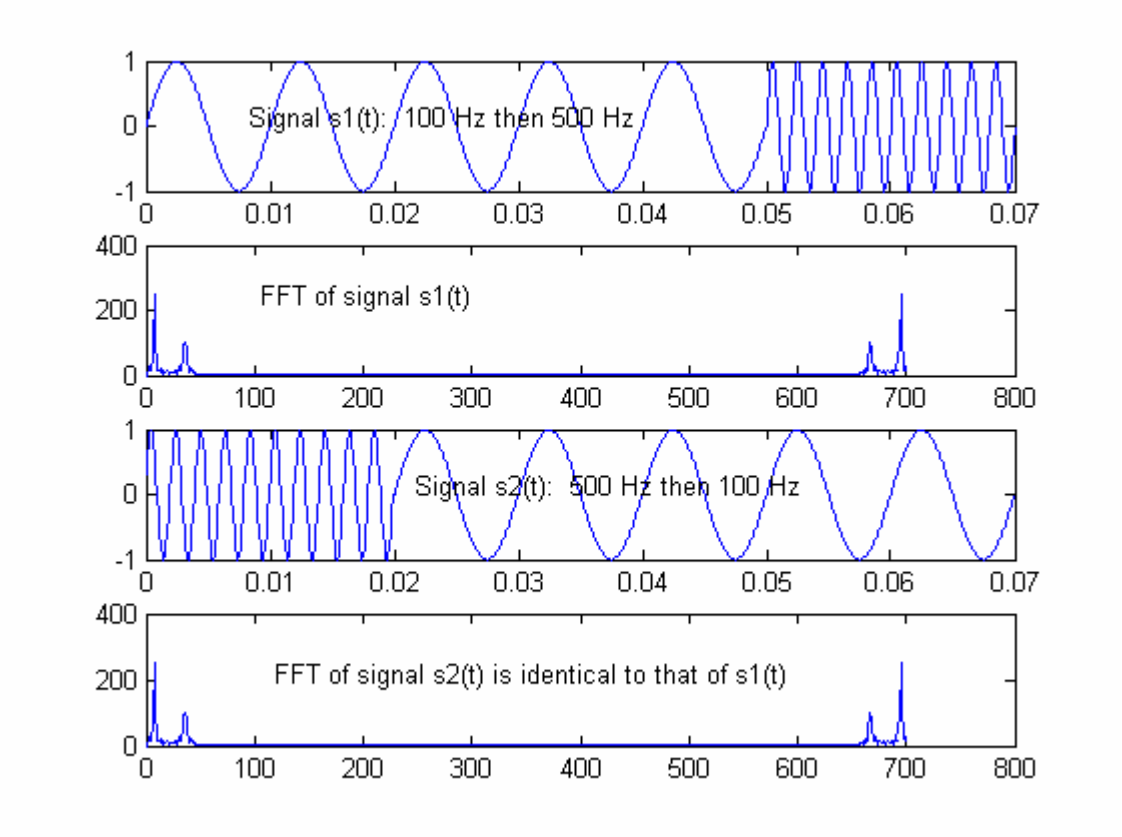


Figure 1. The Fast Fourier Transform and its lack of time information.

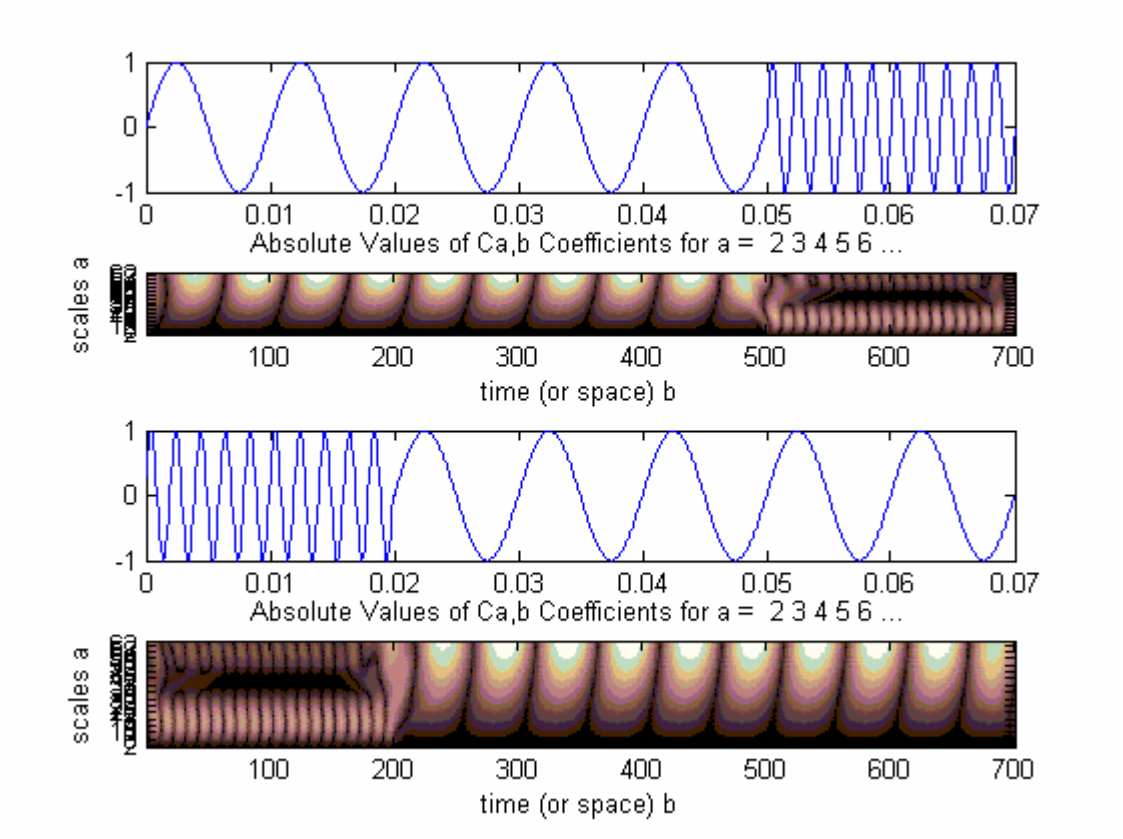


Figure 2. The CWT reveals the exact time of signal change.

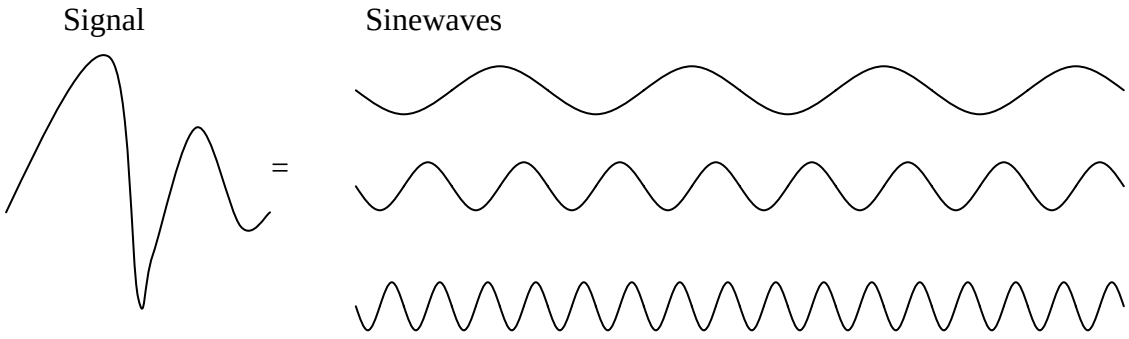
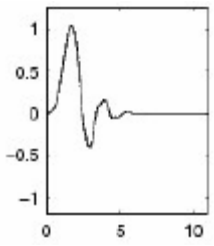
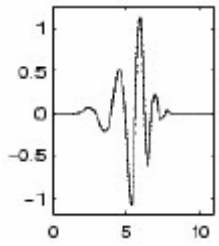


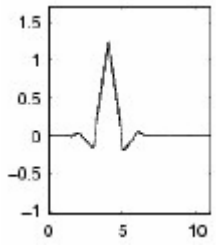
Figure 3. Constituent sinewaves of different frequencies in signal analysis.



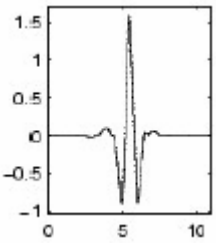
a) db6 Scaling Function $\phi(t)$



b) db6 Wavelet Function $\psi(t)$



c) coif2 Scaling Function $\phi(t)$



d) coif2 Wavelet Function $\psi(t)$

Figure 4. Examples of wavelets: a) and b) Daubechies-6, c) and d) Coiflet 2.

Using wavelets, a signal is analyzed into constituent wavelets of different scales and positions as Figure 5 illustrates.

Definitions

1. The Continuous Wavelet Transform (CWT)

The definition of the Continuous Wavelet Transform of a signal f(t) is given by the following integral, where ***ψ(t)*** represents the wavelet prototype in use, ***τ*** and ***s*** its position and scale, respectively, and ***C(τ, s)*** the wavelet coefficients in terms of position and scale. The same prototype ***ψ(t)***, called the basis wavelet, is used to create the other wavelets involved in the integral that result when both scale and time position of the wavelet prototype vary:

$$C(\tau, s) = \frac{1}{\sqrt{s}} \int_{-\infty}^{\infty} f(t)\psi(\frac{t-\tau}{s})dt \qquad \text{(Equ. 1)}$$

Equation 1 should be evaluated for all scales and positions of the wavelet with respect to the signal f(t) to be analyzed, as roughly illustrated in Figure 5. For a fixed scale ***s*** (which influences the shape of the wavelet) and for a fixed position ***τ*** of the wavelet, integrate Equation 1 over all time. Repeat for all scales

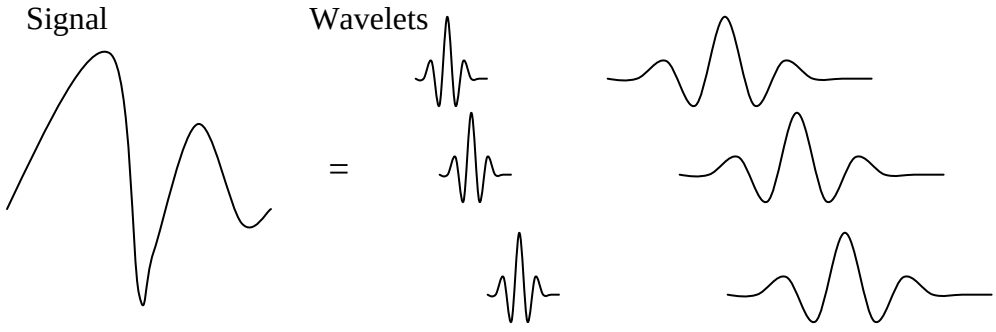


Figure 5. Constituent wavelets of different scales and positions in signal analysis.

and positions. The resulting values of the integral (wavelet coefficients) will give an indication of how closely a given section of the signal correlates with a specific shape and position of the wavelet.

Intuitively, the above operation involves a variable time window over which the signal and the wavelet are compared. When the wavelet is stretched (i.e., when the scale ***s*** increases), we integrate over a larger window, and when the wavelet is compressed, we integrate over a shorter window. As a result, when the wavelet is chosen correctly for a given signal, there will be coefficients with high similarity and others with low similarity. Plotting these coefficients over time will give an indication of different nuances and their occurrences at the corresponding times.

In short, the CWT is able to quantify the resemblance between the signal and the wavelet for a given time position and scale of the wavelet. The scale is the counterpart of frequency. Hence, a wavelet transform plot will show a time-scale representation of a given signal, the equivalent of the time-frequency plane used in the Short Time Fourier Transform (STFT), for example.

2. The Discrete Wavelet Transform (DWT)

With the Discrete Wavelet Transform, scales and time shifts are discrete and are expressed in negative powers of 2. The scale is $\mathbf{s = 2^{-j}}$. Hence, for each halving of **s**, the wavelet is squeezed (compressed) to half its original length, and for each doubling of **s**, it is stretched to double its length. Being discrete, the DWT produces far fewer data points than the CWT, and is therefore more efficient than the CWT in that respect.

As for the time shift $\boldsymbol{\tau}$ (or time translation) it is linearly related to scale **s** as follows:

$$\boldsymbol{\tau = k\, s = k\, 2^{-j}}.$$

Substituting $\boldsymbol{\tau}$ and **s** in the continuous wavelet term of Equation 1, yields:

$$\psi\left(\frac{t-\tau}{s}\right) = \psi(2^j t - k)$$

where **j** indexes the scaling of the wavelet prototype and **k** indexes its time shifts.

The linear decomposition of a signal can be performed using two expansions that complement each other: A linear expansion for its high frequency contents using the wavelet function $\psi(2^j t - k)$ and another for its low frequency contents using a wavelet called the scaling function $\varphi(2^j t - k)$.

It can be shown that a given signal f(t) can be approximated by the following decomposition in terms of the scaling and the wavelet functions by utilizing coefficients $a_j[k]$ and $d_j[k]$ at scale $s=2^{-j}$ such that

$$f(t) \cong f_{j+1}(t) = \sum_{k=-\infty}^{\infty} a_j[k] 2^{\frac{j}{2}} \varphi(2^j t - k) + \sum_{k=-\infty}^{\infty} d_j[k] 2^{\frac{j}{2}} \psi(2^j t - k) \qquad \text{(Equ. 2)}$$

where index ***j*** in $f_{j+1}(t)$ must be large enough to make the scale $s=2^{-j}$ small enough in order to capture all details of the initial signal.

The functions $\boldsymbol{\varphi}$ and $\boldsymbol{\psi}$ are orthonormal functions, and the factor $2^{\frac{j}{2}}$ preserves the unity norm of the basis functions at different scales.

The DWT analysis equations that relate the DWT coefficients $a_j[k]$ and $d_j[k]$ at a scale **j** to the next higher scale **j+1** are widely available and derived in the literature [2], [3], [11], [12]. They are repeated here in Equation 3 and Equation 4 below:

$$a_j[k] = \sum_m h_L[m-2k] a_{j+1}[m] \qquad \text{(Equ. 3)}$$

And

$$d_j[k] = \sum_m h_H[m-2k] a_{j+1}[m] \qquad \text{(Equ. 4)}$$

where $h_L[m]$ and $h_H[m]$ represent unit impulse functions of some equivalent digital filters that are related to the wavelet prototype in use. The above analysis operation, as well as its synthesis operation, is also known as Mallat's algorithm.[11] [12]

In terms of computation efficiency, Equations 3 and 4 require only N multiplications and N additions, as compared to an N-point FFT which requires $N\log_2 N$ of each, N being the original number of discrete time samples to be analyzed.

The DWT is similar to a Fourier series expansion of a signal but is much more powerful and flexible due to a better choice of basis functions. For example, it would take a large number of harmonics in a Fourier series of a periodic signal to represent a sharp discontinuity. Wavelet basis functions, in comparison, have sharp corners themselves, so a smaller number of basis functions can represent the same signal more efficiently. The other important difference is that the Fourier series maps a one-dimensional function of a continuous variable into a one-dimensional sequence of coefficients, whereas the wavelet transform maps the same function into a two-dimensional array of coefficients. It is this two-dimensional expansion that allows the analysis of a signal to reveal both time and frequency information.

a) The Discrete Wavelet Decomposition Tree Using Filter Banks

For many signals, the low-frequency content of their spectrum provides the identity of the signal, while the high-frequency provides nuances. Wavelet analysis captures these two aspects as *Approximation* coefficients and *Detail* coefficients, respectively. This operation is described by the filter bank form of the Discrete Wavelet Transform in the following sections.

The Approximation and Detail coefficients can be extracted as independent signals by passing the discrete samples of the original signal simultaneously through a pair of filters, a low-pass filter and a high-pass filter, as depicted in Figure 6. The low-pass filter yields the first-level Approximation coefficients $a_j[k]$, and the high-pass filter yields the first-level Detail coefficients $d_j[k]$.

How are these coefficients related to the signal to be analyzed? It turns out that if the scale is high enough, the scaling functions involved in the decomposition of the signal act like "delta functions" in the inner product so that the coefficients can be replaced by the discrete samples of the signal, provided that the samples are taken at a rate that is above the Nyquist rate. These samples become then a good approximation of the scaling coefficients at that scale and no wavelet coefficients are needed at that scale. For band-limited, practical signals, there is an upper scale above which the wavelet coefficients $d_j[k]$ are negligible.

This first-level decomposition appears in Figure 6 where $a_1[k]$ and $d_1[k]$ represent the level 1 Approximation and Detail coefficients, respectively. The high-pass and the low-pass filter are characterized by their unit impulse responses h_H and h_L, respectively. The down arrow indicates down-sampling by 2 in order for the combined number of data points in both paths (low-pass and high-pass) to be the same before and after splitting the original signal into these two paths. (Without down-sampling, we will have twice as many data points in the branches combined). Choosing the right filters and down-sampling by discarding every other point, for example, will result in no loss of information.

The level 1 Approximation coefficients $a_1[k]$ that are computed in Figure 6 can be further decomposed using the same two filters to yield another level of Approximation and Detail coefficients. For example, a two-level decomposition filter bank is shown in Figure 7. Since after each level the number of data points (or coefficients) is divided by 2, the initial number of data points is divided by $2^2 = 4$ in this two-level tree. For an L-level tree, the

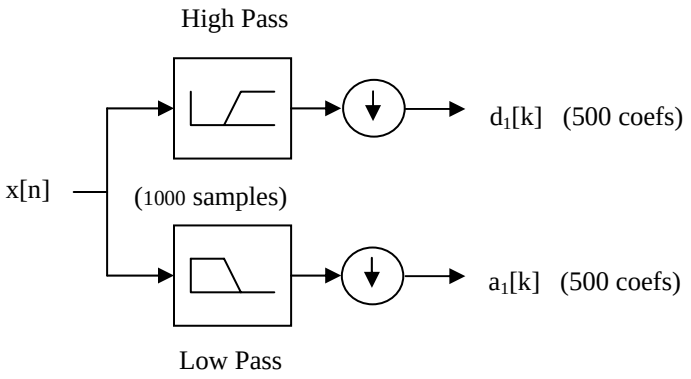


Figure 6. A one-level DWT decomposition tree.

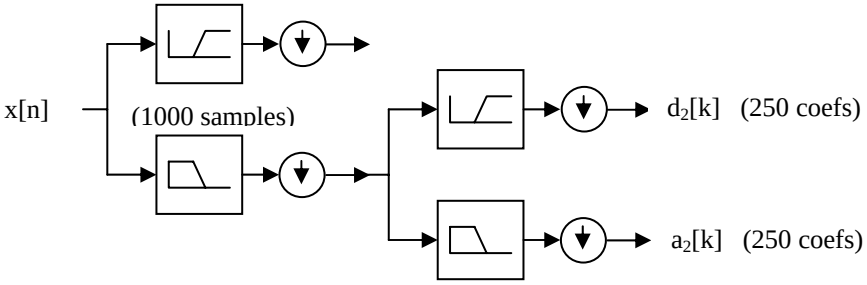


Figure 7. A two-level DWT decomposition tree.

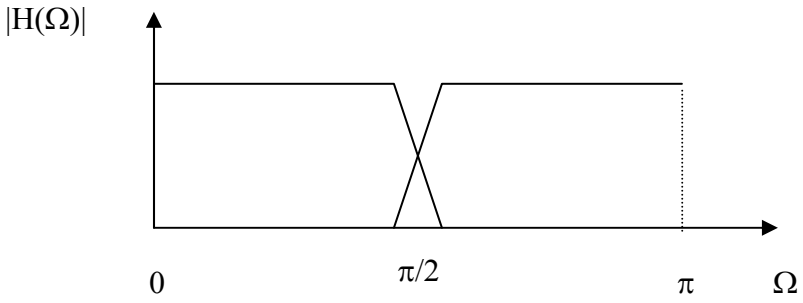


Figure 8. A Level 1 tree divides the spectrum into two equal frequency bands.

original number of available samples is divided by 2^L, a feature that can be exploited in database size reduction, signal compression, speedy computations and other applications.

b) Frequency Consideration in Filter Banks

The DWT is closely related to the *subband* coding schemes that are used in speech and image compression, since the frequency response of each branch of the decomposition tree is divided into smaller bands. For example, the frequency spectrum of the Level 1 decomposition tree of Figure 6 is shown in Figure 8, where the frequency spectrum is divided into two equal frequency bands, one for the low-pass filter and the other for the high-pass filter. Figure 9 shows the bands for the two-level decomposition tree of Figure 7, where the low-pass band is further divided into two equal bands, one for another low-pass and the other for another high-pass frequency band.

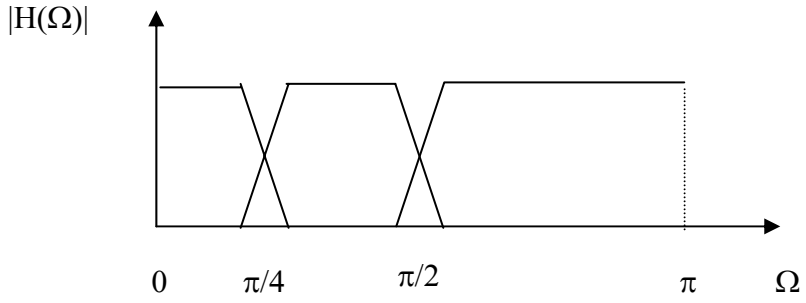


Figure 9. Frequency bands for the two-level DWT decomposition tree of Figure 7.

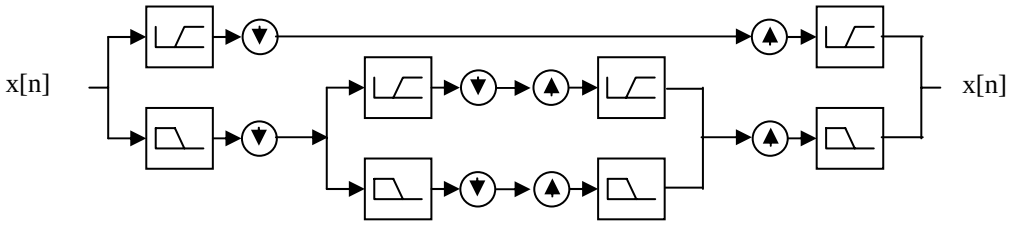


Figure 10. A two-level DWT decomposition and reconstruction tree.

This decomposition of the frequency axis exhibits a logarithmic resolution, by inspection of Figure 9: Narrower bandwidth for lower frequencies and wider bandwidth for higher frequencies. One can recognize this as the "constant Q" filtering, which is appropriate to some but not all applications. However, another technique that solves this problem is the use of "wavelet packets," which allow a variable resolution at high frequencies, but at a cost of more computations.[2]

c) The Discrete Wavelet Reconstruction Tree

The Discrete Wavelet Transform (DWT) can be inverted to recover the original signal, as illustrated in Figure 10. This reconstruction utilizes the Detail coefficients found in level 1 and both the Approximation and Detail coefficients computed in the last level of the decomposition tree.

Example 1: Signal Denoising Using the DWT

The following example illustrates how noise can be separated from a signal using a two-level filter bank. The MATLAB statements that are shown below use the MATLAB Wavelet Toolbox to extract noise from a noisy sine wave using the *db2* (Daubechies-2) wavelet. Figure 11 shows the signal plus noise at the top, and the corresponding Approximation and Detail coefficients after each level. The denoised signal is shown as ***a2*** (Second-Level Approximation Coefficients) with 250 samples.

%Generate a sine wave plus random noise:

```
x = sin(5*linspace(0,pi,1000)) + 0.5*rand
(1,1000);
```

%Generate and plot the level 1 coefficients a1 and d1 using the *db2* wavelet:

```
[a1,d1] = dwt(x,'db2');
subplot(511)
plot(x)
subplot(512)
```

```
plot(a1)
subplot(513)
plot(d1)
```

%Generate and plot the level 2 coefficients a2 and d2:

```
[a2,d2] = dwt(a1,'db2');
subplot(514)
plot(a2)
subplot(515)
plot(d2)
```

The following observations can be made by inspection of Figure 11:

- The high frequencies present in the signal (noise) are captured in the Detail coefficients
- The low frequency information (sine wave) is captured in the Approximation coefficients
- From one level to the next higher one, the number of samples is divided by 2

Example 2: Speech Recognition Using the DWT

To take advantage of the down-sampling feature of the discrete wavelet transform (DWT), several simulation runs were performed to recognize speech signals after they have been compressed by the DWT. The first step of the simulation consists of building a database by storing spoken words using Windows Sound Recorder, which generates a "*.WAV" file (Figure 12). This "wav" file is then read from within MATLAB and becomes the original, unprocessed signal. Next, the MATLAB built-in Discrete Wavelet Transform is applied as many times as necessary to yield the corresponding decomposition tree, that is, three times for a level 3, five times for a level 5, etc. Although the Detail Coefficients also contain valuable information in speech recognition, only the Approximation coefficients are stored for comparison purposes during this simulation.

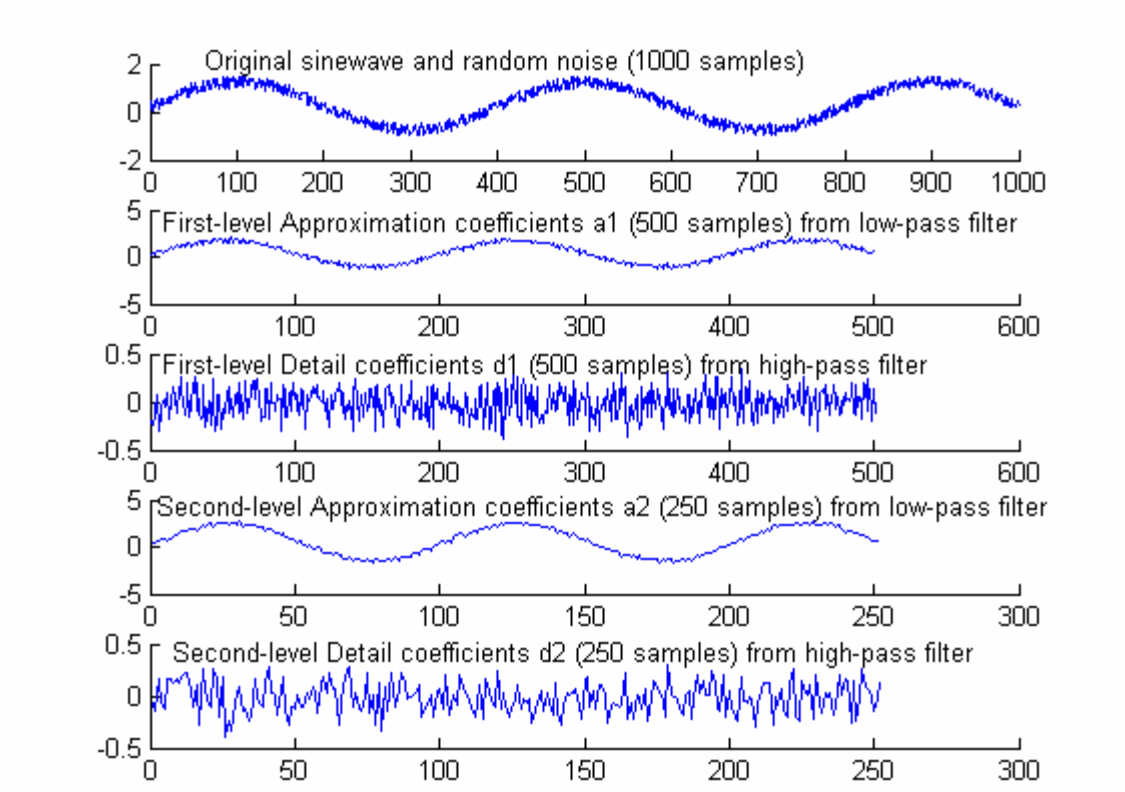


Figure 11. Signal denoising using the *db2* wavelet.

The number of samples stored as a result of this decomposition (after the final level) is therefore divided by 8 (2^3) for a level 3, divided by 32 (2^5) for a level 5, etc., since after each stage, the number of samples is divided by 2 as a result of the down-sampling that takes place after each stage.

In order to apply the recognition phase to each word, the same word is spoken again, stored, read by MATLAB and processed through the DWT as was done to the first one. The first stored word and the second can then be compared using a much smaller number of data points, hence resulting in a shorter processing time during the recognition phase. The comparison of both words can be done using the built-in Fast Fourier Transform (FFT) in MATLAB. The above operations are simulated in MATLAB Programs 3 and 4 shown in the Appendix.

Figure 13 shows the FFT that acts on 1/8 the original number of samples when the three vowels "e," "i," and "o" are acquired with Windows Sound Recorder (8000 Hz sampling frequency and 8-bit mono) and compressed with a 3-level DWT (see Program 3 in Appendix). The vowels were first acquired, processed and stored, resulting in signals (file names) **e1**, **i1**, and **o1** the first time around. The second time around the same vowels are again acquired, processed and stored as **e2**, **i2**, and **o2**. An FFT is taken of the Approximation Coefficients that are generated after the third level for both versions of the spoken vowels. As can be seen from the figure, all three spectra for "e," "i," and "o" are distinguishable and the spectra of the same vowels match within acceptable limits.

Note that for this level 3 decomposition and with a sampling frequency of 8000 Hz, and hence a folding frequency of 4000 Hz, we are only left with a low pass region from 0 to 500 Hz (4000/8) to work with for the recognition phase instead of the whole spectrum. (Extrapolating the plot of Figure 9 for a level 3, this corresponds to 0 to $\pi/8$.) This simulation shows that there is still enough low frequency information to compare one vowel to another.

Pushing this simulation to a level 5 still gave acceptable results. However, a level 6 simulation fails the recognition phase.

Figure 14 shows the FFT results of a level 5 DWT when applied to words "table," "chair," and "wall" (see Program 4 in Appendix). This time the sampling frequency is set at 22050 Hz with 16 bits mono in the Properties of Windows Sound Recorder. With a level 5 decomposition tree, the original number of acquired samples is divided by 2^5, or 32. Hence, given that the recording time was approximately 1 second for each word, only 1/32 of the 22050 samples are used to compare one word to another. In the frequency domain, this translates to the low pass region from 0 to 11025/32 = 345 Hz, since we are using only the Approximation coefficients for the comparison and we only rely on the portion of the spectrum between 0 and fs/2 in the FFT computation.

As can be seen from these plots, similar conclusions can be drawn regarding the capabilities of this method from distinguishing one word from another.

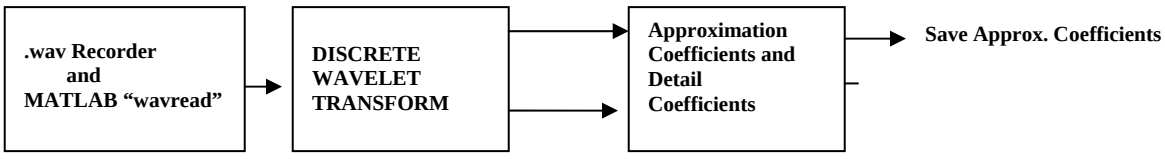


Figure 12. Acquiring words with sound recorder and processing them with the DWT.

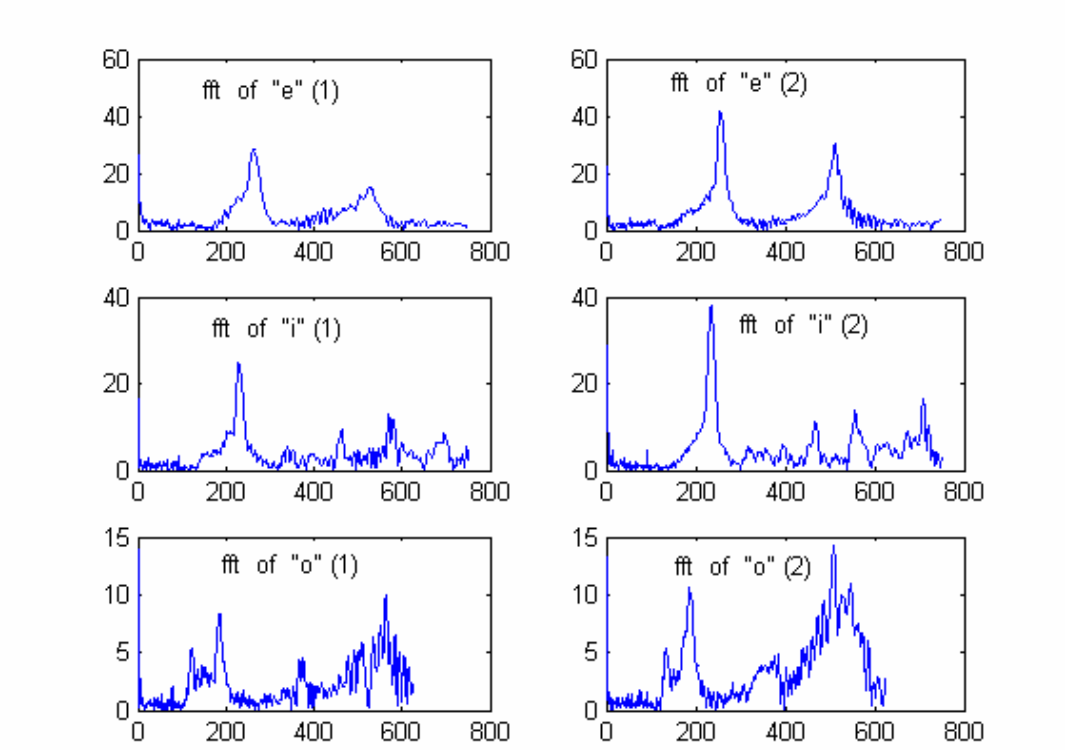


Figure 13. FFT of approximation coefficients of a 3-level DWT for vowels “e,” “i,” and “o.” 8 KHz sampling frequency. Index (1) represents the first time around and (2) the second time around for the same vowel.

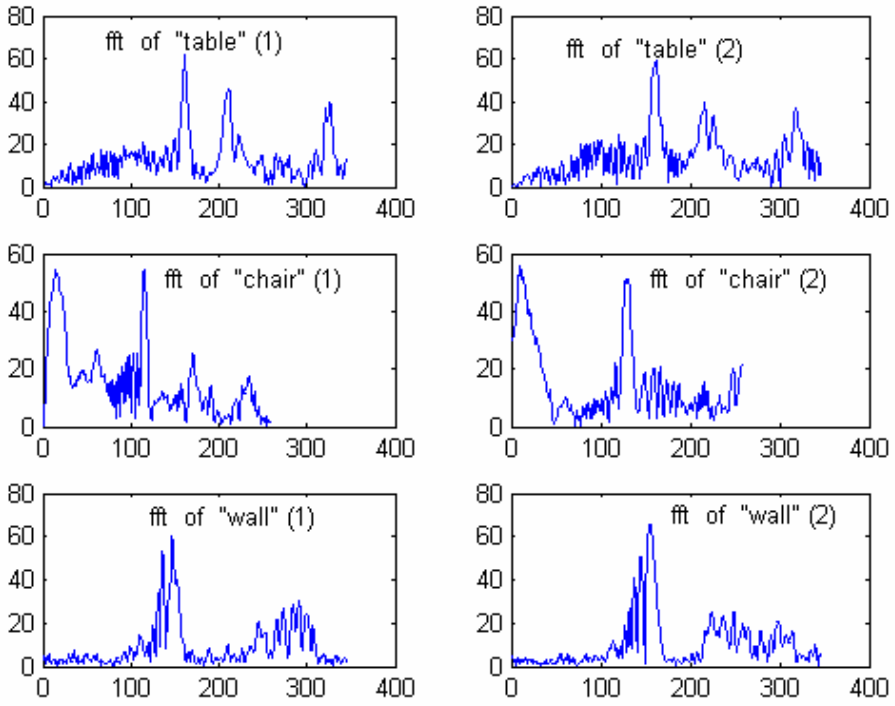


Figure 14. FFT of approximation coefficients of a 5-level DWT for words “table,” “chair,” and “wall.” The sampling frequency is 22050 Hz.

Repeating the simulation with a level 6 (still with a sampling frequency of 22050 Hz) also gave acceptable results. Note that this leaves a modest low pass region of 11025/64 = 172 Hz to compare the words. However, a level 7 DWT fails the test.

MATLAB Wavelet Toolbox

A good way to start with wavelets in MATLAB is to type ***wavedemo*** at its command prompt. This will lead to three different options: *Command line mode*, *GUI mode*, and *Short 1-D scenario*. One can also type ***wavemenu*** to go directly to the GUI mode.

Conclusion

This paper presented some brief highlights of the Wavelet Transform. The Discrete Wavelet Transform is a special case of the Continuous Wavelet Transform where both position and scale vary in a power of 2. The Discrete Wavelet Transform can be better understood with the filter bank representation, mostly if we are dealing with an undergraduate audience. We have seen that in this representation the low-pass filter gives the approximation coefficients of a signal, and the high-pass gives the detail coefficients. The approximation coefficients from the first level can be decomposed further using the same process, leading to a Wavelet Decomposition Tree with several levels. Since after each level the number of samples is down-sampled by two, this decomposition tree gives a much smaller number of data points to work with.

When this remarkable process is applied to a speech recognition example, we are left with a fraction of the computation efforts needed in the comparison of two speech signals.

Even though the Discrete Cosine Transform (DCT) is also widely used in signal and image compression, the WT has always been shown to perform better[6] and with less processing time.

Finally, the topic of Wavelet Transform is not an easy one to introduce at the undergraduate level. But illustrative examples and other simple applications similar to the ones presented here could serve as good incentives to every one to probe further into this subject. A first step into this process is Reference.[2]

References

1. The MATHWORKS, Inc., Applications Using Wavelets with MATLAB and the Wavelet Toolbox, 1996.

2. Burrus, C. Sidney, Gopinath, Ramesh A., and Guo, Haitao, Wavelets and Wavelet Transforms, A Primer, Prentice Hall, 1998.

3. Van de Vegte, Joyce, Fundamentals of Digital Signal Processing, Prentice Hall, 2002.

4. Strang, Gilbert, Wavelets, American Scientist, Vol. 82, pp 250-255, May-June, 1994.

5. Torrence, Christopher, and Compo, P. Gilbert, A Practical Guide to Wavelet Analysis, http://paos.colorado.edu/research/wavelets.

6. Bruce, Andrew, Donoho, David, and Gao, Hong-Ye, Wavelet Analysis, IEEE Spectrum, October 1996.

7. Akay, Metin, Wavelet Applications in Medicine, IEEE Spectrum, May 1997.

8. Sweeney, Nina, Wavelet Transforms Represent Signals in Terms of Both Time and Scale, Personal Engineering, August 1996.

9. Rioul, Olivier, and Vetterli, Martin, Wavelets and Signal Processing, IEEE Spectrum Magazine, October 1991.

10. The MATHWORKS, Inc., The Wavelet Toolbox for Use with MATLAB, 1996.

11. Mallat, S.G., Multiresolution Approximation and Wavelet Orthonormal Bases of L^2, Transactions of the American Mathematical Society, 315:69-87, 1989.

12. Mallat, S.G., A Theory for Multiresolution Signal Decomposition: The Wavelet Representation, IEEE Transactions on Pattern Recognition and Machine Intelligence, 11(7):674-693, July 1989.

Biographical Information

Essaid Bouktache is an Associate Professor in the Department of Electrical and Computer Engineering Technology at Purdue University Calumet, Hammond, Indiana. He received his M.S. and Ph.D. in Electrical Engineering from The Ohio State University in 1980 and 1985, respectively. His research interests and involvement include digital signal processing and real-time applications, adaptive algorithms, digital communications, and computer networks.

Appendix: MATLAB Programs

The following are the MATLAB programs that are used to obtain Figures 1, 2, 13, and 14. MATLAB 7.0 is the version used.

Program 1: FFT of 100 and 500 Hz sine waves (for Figure 1 plot).

```
f1 = 100;
f2 = 500;
T1 = 1/f1;
T2 = 1/f2;
T = 1e-4;
t1 = 5*T1;
t2 = 10*T2;
n1 = 0:T:t1;
x1 = sin(2*pi*f1*n1);
n2 = (t1+T):T:(t2+t1);
x2 = sin(2*pi*f2*n2);
n = [n1 n2];
x = [x1 x2];
subplot(411)
plot(n,x)
y = abs(fft(x));
subplot(412)
plot(y)
x = [x2 x1];
subplot(413)
plot(n,x)
y = abs(fft(x));
subplot(414)
plot(y)
```

Program 2: CWT of 100 and 500 Hz sine waves (for Figure 2 plot).

```
f1 = 100;
f2 = 500;
T1 = 1/f1;
T2 = 1/f2;
T = 1e-4;
```

```
t1 = 5*T1;
t2 = 10*T2;
n1 = 0:T:t1;
x1 = sin(2*pi*f1*n1);
n2 = (t1+T):T:(t2+t1);
x2 = sin(2*pi*f2*n2);
n = [n1 n2];
x = [x1 x2];
subplot(411)
plot(n,x)
subplot(412)
y = cwt(x,2:64,'db2','plot');
x = [x2 x1];
subplot(413)
plot(n,x)
subplot(414)
y = cwt(x,2:64,'db2','plot');
```

Program 3: Wavread, level 3 DWT, and FFT (for Figure 13 plot).

```
%Wavread and level 3 DWT for vowels "e" "i" and "o":
e11=wavread('e1');                %First time around.
[a,d]=dwt(e11,'db2');
[a,d]=dwt(a,'db2');
[e1,de1]=dwt(a,'db2');

i11=wavread('i1');
[a,d]=dwt(i11,'db2');
[a,d]=dwt(a,'db2');
[i1,di1]=dwt(a,'db2');

o11=wavread('o1');
[a,d]=dwt(o11,'db2');
[a,d]=dwt(a,'db2');
[o1,do1]=dwt(a,'db2');

e22=wavread('e2');                %Second time around.
[a,d]=dwt(e22,'db2');
[a,d]=dwt(a,'db2');
[e2,de2]=dwt(a,'db2');

i22=wavread('i2');
[a,d]=dwt(i22,'db2');
[a,d]=dwt(a,'db2');
[i2,di2]=dwt(a,'db2');

o22=wavread('o2');
[a,d]=dwt(o22,'db2');
[a,d]=dwt(a,'db2');
[o2,do2]=dwt(a,'db2');

%Compute FFT of Approximation Coefficients:
e1=abs(fft(e1));
i1=abs(fft(i1));
o1=abs(fft(o1));
```

```
e2=abs(fft(e2));
i2=abs(fft(i2));
o2=abs(fft(o2));

%Plot the spectra for comparison:
subplot(321);
e1=e1(1: size(e1)/2);
plot(e1)
axis([0, 800,0, 60]);
subplot(322);
e2=e2(1: size(e2)/2);
plot(e2)
axis([0, 800,0, 60]);
subplot(323);
i1=i1(1: size(i1)/2);
plot(i1)
axis([0, 800,0, 40]);
subplot(324);
i2=i2(1: size(i2)/2);
plot(i2)
axis([0, 800,0, 40]);
subplot(325);
o1=o1(1: size(o1)/2);
plot(o1)
axis([0, 800,0, 15]);
subplot(326);
o2=o2(1: size(o2)/2);
plot(o2)
axis([0, 800,0, 15]);
```

Program 4: Wavread, level 5 DWT, and FFT (for Figure 14 plot).

```
%Wavread and level 5 DWT for words "table" "chair" and "wall":
e11=wavread('table1');   %First time around.
[a,d]=dwt(e11,'db2');
[a,d]=dwt(a,'db2');
[a,d]=dwt(a,'db2');
[a,d]=dwt(a,'db2');
[e1,de1]=dwt(a,'db2');

i11=wavread('chair1');
[a,d]=dwt(i11,'db2');
[a,d]=dwt(a,'db2');
[a,d]=dwt(a,'db2');
[a,d]=dwt(a,'db2');
[i1,di1]=dwt(a,'db2');

o11=wavread('wall1');
[a,d]=dwt(o11,'db2');
[a,d]=dwt(a,'db2');
[a,d]=dwt(a,'db2');
[a,d]=dwt(a,'db2');
[o1,do1]=dwt(a,'db2');

e22=wavread('table2');   %Second time around.
```

```
[a,d]=dwt(e22,'db2');
[a,d]=dwt(a,'db2');
[a,d]=dwt(a,'db2');
[a,d]=dwt(a,'db2');
[e2,de2]=dwt(a,'db2');

i22=wavread('chair2');
[a,d]=dwt(i22,'db2');
[a,d]=dwt(a,'db2');
[a,d]=dwt(a,'db2');
[a,d]=dwt(a,'db2');
[i2,di2]=dwt(a,'db2');

o22=wavread('wall2');
[a,d]=dwt(o22,'db2');
[a,d]=dwt(a,'db2');
[a,d]=dwt(a,'db2');
[a,d]=dwt(a,'db2');
[o2,do2]=dwt(a,'db2');
```

%Compute FFT of Approximation Coefficients:

```
e1=abs(fft(e1));
i1=abs(fft(i1));
o1=abs(fft(o1));
e2=abs(fft(e2));
i2=abs(fft(i2));
o2=abs(fft(o2));
```

%Plot the spectra for comparison:

```
subplot(321);
e1=e1(1: size(e1)/2);
plot(e1)
axis([0, 400,0, 80]);
subplot(322);
e2=e2(1: size(e2)/2);
plot(e2)
axis([0, 400,0, 80]);
subplot(323);
i1=i1(1: size(i1)/2);
plot(i1)
axis([0, 400,0, 60]);
subplot(324);
i2=i2(1: size(i2)/2);
plot(i2)
axis([0, 400,0, 60]);
subplot(325);
o1=o1(1: size(o1)/2);
plot(o1)
axis([0, 400,0, 80]);
subplot(326);
o2=o2(1: size(o2)/2);
plot(o2)
axis([0, 400,0, 80]);
```