Signal Analysis

David Ozog

May 11, 2007

Abstract

Signal processing is the analysis, interpretation, and manipulation of any time varying quantity [1]. Signals of interest include sound files, images, radar, and biological signals. Potentials for application in this area are vast, and they include compression, noise reduction, signal classification, and detection of obscure patterns. Perhaps the most popular tool for signal processing is Fourier analysis, which decomposes a function into a sum of sinusoidal basis functions. For signals whose frequencies change in time, Fourier analysis has disadvantages which can be overcome by using a windowing process called the Short Term Fourier Transform. The windowing process can be improved further using wavelet analysis. This paper will describe each of these processes in detail, and will apply a wavelet analysis to Pasco weather data. This application will attempt to localize temperature fluctuations and how they have changed since 1970.

1 Introduction

Two hundred years after its discovery by Joseph Fourier in the early 19th century, the Fourier transform is still being applied to new mathematical problems. Fourier analysis is a major component of noise reduction, signal compression, spectroscopy, acoustic analysis, biomedical applications - and the list goes on. This paper will explain the fundamentals of Fourier theory, solidifying the concepts with a few examples. Then, we will broaden the approach into the realms of Short Term Fourier Transforms and wavelet analysis, which will increase potential for applications.

The Fourier series of a function f is its decomposition into an infinite sum of sine and cosine functions. In symbols, for a function $f(x)$ that satisfies

certain conditions (to be discussed in Section 2), its Fourier series is written as

$$f(x) = a_0 + \sum_{k=1}^{\infty} a_k \sin(kx) + \sum_{k=1}^{\infty} b_k \cos(kx). \tag{1}$$

It might not be surprising that the Fourier series of f reveals the frequency content of the function. So if f is a function of time, then we can think of the Fourier series as a transformation from time space to frequency space. Such a transformation reveals many properties of the signal that we might not have seen otherwise. This information can be quite useful in scientific applications.

A variation of Fourier analysis is the Short Term Fourier Transform, or STFT. This method retains time information of the signal, which enables us to localize *when* particular frequencies occur. The STFT still has analytical disadvantages, but these can be avoided with the method of wavelet analysis, which will become the main focus of this paper. Again, several examples and applications will be presented, and if all goes well, I will apply wavelet analysis to local temperature data, studying the affects of global warming on seasonal temperature fluctuations.

2 The Fourier Series

The Fourier series of a function f is its decomposition into an infinite sum of sines and cosines. In other words, the Fourier series is a "rewriting" of f in the following basis of function space:

$$\{\sin(mx), \cos(nx), 1\} \text{ for } m \text{ and } n = 1, 2, 3, \ldots \tag{2}$$

If f is piecewise continuous (and has no more than a *finite* number of discontinuities), periodic (with period $T_2 - T_1$), and square integrable over the period length, that is,

$$\int_{T_1}^{T_2} |f(x)|^2 < \infty,$$

then it is *always* possible [5] to expand f into the form of equation (1).

The most important aspect of equation (1) is the calculation of the coefficients a_k and b_k. Fortunately, the basis from sequence (2) is orthogonal, which drastically simplifies the calculations. To see this, we first define the **inner product** for this vector space [3]. For any two functions p and q that are continuous on a closed bounded interval $a \leq x \leq b$, their inner product

on that interval is

$$< p, q > \;=\; \int_a^b p(x)q(x)dx.$$

Also notice that, for positive integers n and k,

$$\int_0^{2\pi} \sin(kx)dx \;=\; 0 \tag{3}$$

$$\int_0^{2\pi} \sin(kx)\sin(nx)dx \;=\; \pi \qquad \text{when } n = k \tag{4}$$

$$\int_0^{2\pi} \sin(kx)\cos(kx)dx \;=\; 0. \tag{5}$$

Therefore, to solve for the coefficients a_k, multiply both sides of equation (1) by $\sin(nx)$,

$$f(x)\sin(nx) = a_0\sin(nx) + \sum_{k=1}^{\infty} a_k \sin(kx)\sin(nx) + \sum_{k=1}^{\infty} b_k \cos(kx)\sin(nx) \tag{6}$$

and integrate over the interval $\{0, 2\pi\}$.

$$\int_0^{2\pi} f(x)\sin(nx)dx = \int_0^{2\pi} a_0\sin(nx)dx + \int_0^{2\pi} \sum_{k=1}^{\infty} a_k \sin(kx)\sin(nx)dx + \int_0^{2\pi} \sum_{k=1}^{\infty} b_k \cos(kx)\sin(nx)dx. \tag{7}$$

Properties (3) and (5) above imply that the first and third integrals on the right side are equal to zero, so we can simply remove them from the equation. Also, after *choosing* $n = k$, we can exploit property (4), and solve for the a_k coefficients,

$$a_k\pi \;=\; \int_0^{2\pi} f(x)\sin(kx) \tag{8}$$

$$a_k \;=\; \frac{1}{\pi}\int_0^{2\pi} f(x)\sin(kx).$$

So we see that calculating the a_k coefficients is as easy as taking the inner product of the function f with $\sin(kx)$. It should be easy to see that if we had multiplied both sides of equation (1) by $\cos(nx)$ instead of $\sin(nx)$, then we would have a similar solution for the coefficients b_k. Therefore, all of the

coefficients from equation (1) can be computed by using the following inner products (the constant in front is neglected):

$$\begin{aligned} a_0 &= < f, 1 > \\ a_k &= < f, \sin(kx) > \\ b_k &= < f, \cos(kx) > . \end{aligned} \tag{9}$$

3 Implementing the Fourier Transform

In the real world of signal analysis, we almost always deal with *discrete* signals. The signal could be anything, like an audio recording, an encephalography measurement, a plot of temperature versus time, or whatever. In most if not all of these cases, the signal, call it s, is not a continuous function; instead, it is a sequence of data points that are measured with some constant *sampling rate*. The sampling rate is simply the frequency with which our measuring device records data. Since our signal is not continuous, we cannot calculate the Fourier coefficients a_k and b_k as easily as we did in section 2, because taking the inner product of two functions *requires* that they be continuous on the interval $[0, 2\pi]$. Therefore, other methods must be used.

The task is greatly simplified by using MATLAB software. MATLAB contains a Fast Fourier Transform (FFT) that performs the calculations very quickly. We import the signal s into MATLAB as a vector containing N points. But in most cases our signal is measured on some arbitrary interval, $[a, b]$, so it needs to be shifted to the interval $[0, 2\pi]$ before we apply the FFT to it. To begin, we shift the the interval $[a, b]$ to $[0, B]$ (where $B = b - a$), so that the signal begins at $x = 0$. Now, we scale each point in $[0, B]$ to another point in $[0, 2\pi]$ with the following mapping:

$$x_n \in [0, B] \rightarrow t_n = \frac{2\pi}{B} x_n \in [0, 2\pi].$$

Now, MATLAB computes the Fourier coefficients using an efficient method that constructs a matrix of complex exponentials, then takes the real components of an inner product to compute the a_k coefficients and takes the imaginary components to compute the b_k coefficients [1]. An important consequence of taking the FFT is that real-valued functions, which are what we will always deal with in this paper, have an exact symmetry about $N/2$.

[1] This process will not be described in detail here, but it is explained thoroughly in [4].

Figure 1: The asterisks are the points y_i. As a visual aide, they are connected by the continuous version of the function.

So, looking at half of the output coefficients will give us all the information that we need.

Since our input signal, s, is a one-dimensional signal with N points, the output of MATLAB's FFT is also a one-dimensional vector with N points. This vector, call it F, contains the coefficients a_0, a_k, and b_k from equation (1). However, every element of F is a complex number, F_k, that contains both the a and the b constants for a particular k value. The real part of F_k is the a_k value and the imaginary part is the b_k value. But when looking at the general frequency content of s, it is most convenient to plot the magnitude of each F_k,

$$|F_k| = \sqrt{a_k^2 + b_k^2}.$$

A plot of $|F_k|$ for every k is called a **power spectrum**. An example of using the power spectrum in MATLAB to filter noise from a signal is presented below.

3.1 Example

In many scientific applications, we are forced to deal with noisy signals. This is the result of random fluctuations in the measuring device, uncertainty in the measurement itself, and other factors. Nonetheless, we often need a quick and consistent way to remove the noise, so that the signal is easier to analyze.

Figure 2: The power spectrum of y_i. Notice the symmetry about $i = 32$. We only need to look at the first two peaks because our function, y_i, is real-valued.

This example will show how the power spectrum can be used to filter noise from a signal. First, take a look at this simple function that has 64 points in the xy plane (see figure 1.):

$$y_i = \sin\left(\frac{\pi i}{8}\right) + 2\cos\left(\frac{7\pi i}{8}\right), \text{ with } i = 0, 1, 2, \ldots, 63.$$

We take the FFT in MATLAB by simply typing $\texttt{F} = \texttt{fft}(\texttt{y}_\texttt{i})$. The resulting vector, F contains all of the a_k and b_k coefficients discussed in the previous section. A plot of the absolute value of all $|F_k|$ (the power spectrum) is shown in figure 2. We see that there are two distinct peaks in the power spectrum at positions $|F_5|$ and $F_29|$ (we ignore the other two, because they are the symmetric coefficients for real-valued functions). These two numbers correspond to the "dominant" frequencies of the function. In fact, this is all of the information that we need to reconstruct the signal. In MATLAB, we simply compute the **inverse FFT** and plot the result. This reconstruction would be the exact same function that is shown in figure 1.

Now, let's consider the same signal with random noise added to it. We construct the noise with MATLAB by adding a positive random number to y_i that is no greater than $(0.2)y_i$. A plot of this noisy function is shown in figure 3. We would like to remove the noise from the signal, but the power spectrum of this function contains the frequencies associated with the noise (figure 3). Taking the inverse FFT would still give us the noisy function.

Figure 3: The noisy signal (left) and its power spectrum (right).

Fortunately, it is very easy in MATLAB to set all of the "unimportant" coefficients in the power spectrum to zero. This is done with a few simple lines of code - if the coefficient is less than some threshold (say, 10), then set that coefficient to zero. After doing this, we end up with the previous power spectrum that contained only the two dominant frequencies. Taking the inverse FFT of this power spectrum gives us a clean, de-noised signal.

In real-world applications, we can de-noise a signal as far as we want (down to nothing!). However, the data is usually most useful if it is de-noised by just the right amount, which takes some trial and error to find. The importance of this fact is definitely seen with EEG analysis, and we will also see it in other applications.

4 A Problem with the Fourier Transform

We have seen that Fourier analysis is performed on short intervals that are shifted and scaled to the interval $[0, 2\pi]$. While this method is helpful in understanding the short time behavior of the signal of interest, we are sometimes interested in frequency behavior on longer time scales. Therefore, an alternate method called **windowing** is necessary. I will present an in depth definition of that method in the next section.

The problem with Fourier analysis, as we saw it in the last section, is a lack of **time localization**, which means that the power spectrum does not give us information about *when* certain frequencies occur in the signal. The spectrum only tells us that they do indeed occur *somewhere*. This phenomenon is not a problem with all functions, only those in which the signal switches at some point in time. To see this idea clearly, we take a

look at a simple example.

4.1 Example

Consider the piecewise-defined function,

$$f(t) = \begin{cases} \sin(t), & \text{if} \quad 0 \leq t < 2\pi \\ \sin(5t), & \text{if} \quad t \geq 2\pi \end{cases} \tag{10}$$

as shown in figure 4.

In words, f is a $\sin(t)$ function that is "turned off" at $t = 2\pi$ and switched to $\sin(5t)$ for all $t \geq 2\pi$. A function with such behavior can easily be expressed as a Heaviside function, $H(t-c)$:

$$H(t-c) = \begin{cases} 0, & \text{if } t < c \\ 1, & \text{if } t \geq c \end{cases}$$

where $c = 2\pi$ in this case. The discontinuous Heaviside function is useful for signals that display "on/off" behavior, such as a step-function. It is plotted in figure 5 with $c = 0$.

Rewriting $f(t)$ in terms of the Heaviside function yields

$$f(t) = \sin(t)(H(t-0) - H(t-2\pi)) + \sin(5t)H(t-2\pi).$$

Using Maple software, we can quickly calculate the Fourier coefficients using equations (9) and plot the continuous frequency power spectrum, which is shown in figure 6.

Unsurprisingly, the plot shows that $\omega = 1$ and $\omega = 5$ are the most prevalent frequencies in $f(t)$. The problem is that there is no information regarding *when* each frequency was prevalent in the signal. Figure 6 tells us nothing about the frequency "switch" that occurs at $t = 2\pi$. The goal of Fourier windowing is to track when the frequencies occur by shifting a *window* across the signal. We will explore this technique in the next section.

Figure 4: Here, $f(t)$ is defined as $\sin(t)$ on $[0, 2\pi)$ and $\sin(5t)$ for $t \geq 2\pi$.

Figure 5: The Heaviside Function

5 Windowing with the STFT

Using windows with Fourier analysis is a technique that reveals *when* certain frequencies are active in a signal. This technique is very useful when time-localization of frequencies is important.

When windowing a signal, we use another function that selects a specific portion of the signal, and we apply the Fourier transform as usual. The window is then shifted to different portions of the signal, and we apply more Fourier transforms until the entire signal is analyzed. This method is called the Short Term Fourier Transform (STFT), and it allows for time-localization of frequencies in a signal.

Windows come in many shapes and sizes, but they all share an important characteristic: they all go to zero as you leave the center of the window (see figure 7). This characteristic allows us to select a region of the signal to analyze with a Fourier transform. Multiplying the signal by the window function eliminates regions that are far away from the center of the window, and retains regions inside the window. The STFT is a transform of the product of the window and the signal.

Figure 6: A plot of the frequency spectrum of f. Notice that $\omega = 1$ and $\omega = 5$ are the most prevalent frequencies, and we cannot tell when the "switch" occurs.

Figure 7: Three window types - Heaviside, triangle, and Gaussian [5].

The window is shifted across the entire signal. We denote this shifting with the variable τ, and the window is a function of $(t - \tau)$. The STFT is a transform of the product of the window function and the signal for every value of τ. The equation becomes

$$\mathcal{F}(\tau, \omega) = \int_{-\infty}^{\infty} w(t - \tau) f(t) \mathrm{e}^{-i\omega t} dt \tag{11}$$

where $w(t - \tau)$ is the window function. This is the continuous Fourier transform, and its derivation is described thoroughly in [5]. The utility of the STFT is shown below by revisiting the function from example 4.1.

5.1 Example

Refer back to the sinusoidal function f from example 4.1, and recall that we were unable to localize the frequency switch that occurs at $t = 2\pi$. We will now apply the STFT to localize this switch using a Gaussian window. In this case, we will use a Gaussian window defined as

$$\omega(t - \tau) = \mathrm{e}^{\frac{-(t-\tau)^2}{\alpha}} \tag{12}$$

where α is a parameter dictating the width of the Gaussian curve. Plugging this and our equation for f, (10), into equation (11) gives us our spectrogram, which is shown in figure 8.

Notice the 3-dimensional shape of the curve. The higher points on the curve correspond to active frequencies. The spectrogram has two definite peaks meaning the signal f contains two prevalent frequencies. We can gain more information from the spectrogram by taking a bird's eye view. Recall that as τ increases, the window shifts to the right across f. When $\tau \approx 0$, there is a conspicuous switch in frequencies from $\omega = 1$ to $\omega = 5$. We have localized a frequency change!

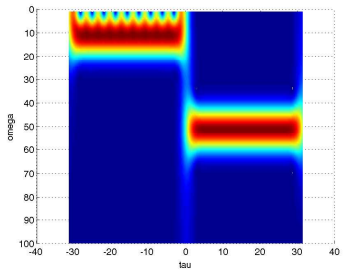

Figure 8: A spectrogram of f.

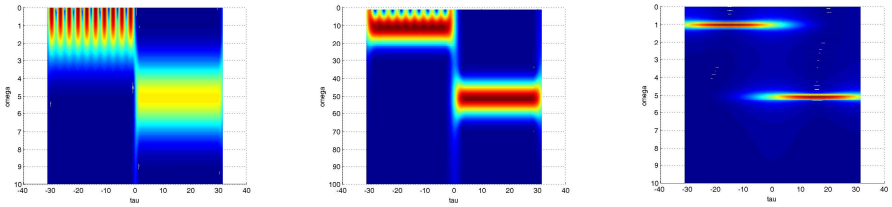

Figure 9: Spectrograms for 3 different α values. The left spectrogram used a narrow window and the right used a wide window.

5.2 The Disadvantage with the STFT

A difficulty with the STFT is deciding on an appropriate window width. For example, the spectrogram comes out differently for different values of the width parameter α from equation (12), as seen in figure 9. For narrow windows, the peaks in the spectrogram are less defined, but the switch at $t = 0$ is quite clear. In other words there is poor frequency resolution and good time resolution. On the other hand, wide windows display poor time resolution and good frequency resolution. There appears to be a tradeoff between time and frequency resolution that is governed by the window width. The only way to determine a window width that will give decent time and frequency resolution is by trial and error. The next development is the use of **wavelets** which eliminates the need for window widths by computing transform over *all* width scales. The properties of wavelets will be explored in the next section.

6 The Wavelet Advantage

At the end of the last section, we discovered that the STFT has a tradeoff between time and frequency localization that is governed by the width of the window that is used (see figure 9). This section will discuss the origin of this tradeoff, and it will introduce the wavelet method which eliminates the concern of window widths.

6.1 Time-Frequency Tradeoff

Figure 9 suggests that narrow windows have better time localization than wide windows. That is, a narrow window's spectrogram clearly shows the *time* at which frequency changes occur in the signal. On the other hand, narrow windows have poor frequency localization compared to wide windows. This means that wide windows capture the frequencies of a signal with greater certainty than narrow windows, as we can see by the thin frequency bands in the rightmost spectrogram of figure 9.

The origin of this tradeoff is simple: if the signal contains large frequencies, then more cycles can fit inside a given window, but if the frequencies are small, then few will fit inside. It is even possible for the window width to be smaller than 1/2 the wave period, meaning that the window will capture *no* information regarding the frequency of the signal. Therefore, a window that is very wide will capture the most frequency information.

Furthermore, a wide window contains a large chunk of time relative to a narrow window. Therefore, if a switch in frequency were to occur within that chunk of time, we would be uncertain as to *where* the switch occurs. A narrow window on the other hand localizes the switch with greater precision.

One solution to the localization tradeoff problem is to find the appropriate window width with simple trial and error. Unfortunately, this is an inefficient process, and some signals contain such a large range of frequencies that an appropriate window may not exist! Another solution is to use wavelet analysis instead of the STFT. Using wavelets in a sense explores *all* possible window widths.

6.2 Wavelet Analysis

The window function for this new form of analysis takes the form of a wavelet. A wavelet is similar to a sine or cosine function, except that it has a limited duration and periodicity. One example of a wavelet is the sine function multiplied by a Gaussian. This Morlet wavelet (figure 10) decays

to zero away from the middle, just like a typical STFT window.

With the STFT, we translate a window across the signal, but with wavelet analysis, we translate a wavelet of a certain **scale** across the signal. The scale of a wavelet is a reference to how compressed it is, as shown in figure 11. By shifting wavelets of different scales across the signal, we gain information about both the times and frequencies of the signal.

Let's compare the process of the STFT with that of the continuous wavelet transform. Recall that the STFT coefficients are found by taking the real and imaginary parts of the Fourier transform of $f(t)$ multiplied by a window function, $w(t-\tau)$:

$$\mathcal{F}(\tau,\omega) = \int_{-\infty}^{\infty} w(t-\tau)f(t)\mathrm{e}^{-i\omega t}dt$$

In other words, as we shift the window across the signal, we find the 2-dimensional Fourier transform of the product of the window and the signal for every value of τ. A collation of the transforms produces a 3-D spectrogram (see section 5).

On the other hand, the **continuous wavelet transform** is defined as the inner product of the entire signal, $f(t)$, with the shifted and scaled versions of the wavelet function Ψ.

$$C(\text{scale},\text{position}) = \int_{-\infty}^{\infty} f(t)\Psi(\text{scale},\text{position})dt \tag{13}$$

The resulting coefficients C are called the **wavelet coefficients**, and they are dependent on the wavelet's scale and position. A plot of the coefficients is called a **wavelet coefficient plot**.

We see that the process of wavelet analysis is quite similar to that of the STFT. The wavelet process begins by shifting the location of the wavelet

Figure 10: The Morlet wavelet is a sine function multiplied by a Gaussian.

Figure 11: Different scales of wavelets. The more compressed wavelets are said to have a *low scale* and the more stretched wavelets have a *higher scale*.

across the signal and computing the coefficients. But unlike the STFT, we start *again* from the beginning of the signal with a wavelet of a different scale, and the coefficients are calculated for this new wavelet. This process is repeated for all scales.

6.3 The Coefficient Plot

With the wavelet coefficients, we create a coefficient plot (figure 12). Notice the coloring scheme: lighter shades refer to larger coefficients and darker shades refer to smaller coefficients. A relatively large coefficient implies a greater correlation between the wavelet of that particular scale and position to the signal. For example, figure 12 might suggest that wavelets of scale 20-30 best resemble the signal, especially at time 4100. However, the coefficients corresponding to these larger scales are interspersedly dark, meaning there is a *poor* correlation between the wavelet and the signal at certain times. This suggests that there are frequency changes in the signal at these times, and the lower scales are more active.

7 De-noising with the Discrete Wavelet Transform

De-noising signals is an important and popular application of the wavelet transform. Through de-noising, a function is smoothed out and represented with less data. Most types of 1 or 2-dimensional signals can be de-noised. Also, since de-noising reduces the amount of information used to represent a data-set, it is commonly used for image compression.

In MATLAB, a signal (or more generally, a sequence) is quickly and easily de-noised using the graphical user interface (GUI) in the wavelet toolbox. While the GUI makes this task fairly straightforward, the underlying

Figure 12: A wavelet coefficient plot.

computations are not fully explained in either the GUI or the toolbox documentation. MATLAB functions for the discrete wavelet transform (DWT) output approximation and detail coefficients, and it is important to understand the meaning behind these coefficients. This section will explain how the approximation and detail coefficients are computed, and it will present a simple example which de-noises a 1-dimensional signal.

7.1 The Continuous Wavelet Transform

The notion of the continuous wavelet transform (CWT) was introduced in equation (13). In words, the continuous wavelet coefficients are computed by taking an integral of the product of the wavelet and the signal. Unfortunately, equation (13) is a simplified definition. In actuality, if the chosen wavelet fulfils the admissibility condition, that is,

$$c_\psi = 2\pi \int_{-\infty}^{\infty} \frac{|\psi(\omega)|^2}{|\omega|} d\omega < \infty,$$

then the CWT is given by

$$C(\text{scale}, \text{position}) = L_\psi f(a,t) = \frac{1}{\sqrt{c_\psi}} \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} \psi\left(\frac{u-t}{a}\right) f(u) du \qquad (14)$$

where we have switched to a different notation [8]. The wavelet coefficients are now $L_\psi f(a,t)$ where a, the scale, is assumed to be nonzero and $t \in \mathbb{R}$ describes the position of the wavelet. The reason for the appearance of the constant factors in front of the integral has to do with the **energy** of the functions. The energy, which is the integral of the magnitude squared, of the reconstruction of $f(t)$ will only be the same if the constant factors are there. Now, let's take a look at the discrete wavelet transform.

Figure 13: In the top row is the sequence, f , and the function $f(t)$. The bottom row contains the sequence f^1 and the function $f^1(t)$.

7.2 The Discrete Wavelet Transform

When using the CWT, the the analyzed function must be continuous. However, in most real-world applications data-measuring devices have a limiting sampling rate. The data is a discontinuous sequence rather than a function, so how do we apply the CWT to a signal? We use what is called the **discrete wavelet transform**.

Our signal is a sequence f reads

$$f = \{f_k\}_{k=0}^{N-1} = \{f(kT_S)\}_{k=0}^{N-1}$$

where N is the number of points and T_S is the spacing between the points. The DWT has two outputs, the approximation coefficients and the detail coefficients. The next section will present a simple example that shows the origin of these coefficients.

7.3 An Example of a Simple Case

The following example of a DWT is based on a similar one from [8].

Suppose that our sequence f is

$$f = \{f_k\}_{k=0}^{7} = \{8, 2, 6, 8, 9, 5, 4, 2\}$$

A stem plot of this signal is shown in the upper left of figure 13.

The DWT constructs a piecewise continuous version of this signal by multiplying each element of f by a **scaling function**:

$$\phi_H(t) = \begin{cases} 1 & \text{if } 0 \leq t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

A piecewise continuous function, $f(t)$, is constructed by taking the product of the elements of f_k with ϕ_H:

$$f(t) = \sum_{k=0}^{7} f_k \phi_H(t-k)$$

The piecewise continuous version is shown in the upper right part of figure 13. Of course, the scaling function is not unique since there are infinitely many ways to construct a piecewise continuous function from a sequence. In fact, every scaling function belongs to a corresponding wavelet that will be used for the reconstruction. In this case, ϕ_H belongs to the Haar wavelet, ψ_H defined as:

$$\psi_H(t) = \begin{cases} 1 & \text{if } 0 \leq t < 1/2 \\ -1 & \text{if } 1/2 \leq t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

When a DWT is applied, the number of points used to represent the signal is halved. This is done by taking the arithmetic mean of neigh boring elements in f_k as shown below. We call this new "coarser" representation of f the sequence f^1:

$$f^1 = \{f_k^1\}_{k=0}^{3} = \left\{ \frac{f_0 + f_1}{2}, \frac{f_2 + f_3}{2}, \frac{f_4 + f_5}{2}, \frac{f_6 + f_7}{2} \right\} = \{5, 7, 7, 3\}$$

Just as before, we can construct a piecewise continuous function, $f^1(t)$ in time by multiplying by the scaling function $\phi(t)$. Furthermore, since the averaging process halved the number of points in the sequence, we want to construct the continuous-time signal on intervals of length 2, so that the function will span the same time interval as $f(t)$. Therefore, we change the time argument for ϕ_H to $t/2 - k$:

$$f^1(t) = \sum_{k=0}^{3} f_k^1 \phi_H \left(\frac{t}{2} - k \right)$$

Both the coarser sequence and function are shown in the bottom row of figure 13.

The process thus far has been a simple method of representing a signal using half the number of points. Now we will calculate the detail function. Notice that there is an amount of error in f^1. We can quantify this error and call it $d^1(t)$:

$$d^1(t) = f(t) - f^1(t)$$

This detail function is plotted in figure 14. We can define the corresponding sequence:

$$\begin{aligned} d^1 = \{d_k^1\}_{k=0}^3 &= \left\{ f_0 - f_0^1, f_2 + f_1^1, f_4 + f_2^1, f_6 + f_3^1, \right\} \\ &= \{8-5, 6-7, 9-7, 4-3\} \\ &= \{3, -1, 2, 1\} \end{aligned} \tag{17}$$

We would like to construct a piecewise continuous $d^1(t)$ from this sequence, and upon inspection of figure 14, we see that each step in the detail function of a certain height is immediately followed by a step of the same magnitude, but opposite in sign. Therefore, to construct $d^1(t)$ from d^1 in equation. 17, we will use the Haar wavelet from equation. 16:

$$d^1(t) = \sum_{k=0}^{3} d_k^1 \psi_H \left(\frac{t}{2} - k \right)$$

7.4 Conclusions

The elements of the sequence d^1 are called detail coefficients in MATLAB. Furthermore, this entire process can be repeated on sequence f^1, and the result is an even coarser signal function, f^2. The number above the f and the d refers to the level of resolution for the DWT. The detail function, $d^n(t)$, represents the signal contributions which are needed to recover the original signal $f(t)$ from the coarser version $f^n(t)$. The beauty of the detail coefficients is that they can be used to directly compute the wavelet coefficients

Figure 14: The left picture is the original containing noise. After de-noising with the wavelet toolbox (using a Haar wavelet) the picture is smoothed out.

from equation 14. This is done using the following formula:

$$L_{\psi_H} f(2, 2k) = \sqrt{\frac{2}{c_{\phi_H}}} d_k^1 (k = 0, \cdots, 3)$$

whose derivation is found in [8].

8 Weather Analysis

We will now apply wavelet analysis to a set of temperature data from Pasco, Washington. The dataset contains maximum temperature measurements taken daily since 1948. A portion of the signal is shown in figure 15. Our hope is to use wavelet analysis to detect temperature fluctuations over the past several decades.[2] Certain wavelet scales will result in larger coefficients, implying a consistent weather fluctuation that corresponds to that scale. For example, we know that every year there is a seasonal cycle, with temperature peaks occurring in the summer and temperature valleys occurring in the winter. Therefore, we would expect that the wavelet scale corresponding to that frequency would have a relatively large coefficient. After analysis, we would also like to know which weather patterns fluctuation the most and which fluctuate the least.

This raises another issue that we must always remember: wavelet scales are *not* the same as frequencies. The scale is a number that corresponds

[2]We are trying to emulate the data presentation and results from another paper [9]. Refer to this paper for more information.

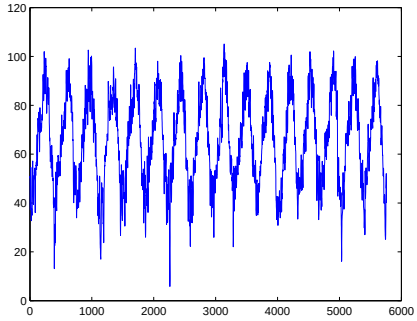

Figure 15: Daily maximum temperatures in Pasco, Washington since 1970.

to the stretch of the wavelet, not its frequency. However, there is an easy way to convert between scale and "pseudo-frequency", assuming we know the center frequency of the wavelet:

$$f_a = \frac{f_c}{s \cdot \Delta} \tag{18}$$

where f_a is the pseudo-frequency, f_c is the center frequency of the wavelet, s is the scale, and Δ is the constant spacing between points. The reason we call this the pseudo-frequency is because f_c might not be the frequency of the wavelet everywhere.

In order to analyze the weather data, we will need a few definitions. First, the **strength** of a coefficient is denoted as $S = ||C(t_0, s)||^2$ where C is the coefficient at a particular time, t_0 and scale, s. The **total power** is the sum of all timescale powers, that is,

$$S_T = \sum_i S_i$$

for some time, t_0. Each coefficient has a **relative power** (S/S_T), which is a percentage that quantifies the influence of a coefficient to the total power. By plotting the average relative power for each timescale, we see which coefficients are the most important. For example, we expect the yearly timescale to be the maximum of the average relative power plot.

After applying a continuous wavelet transform to the data, we end up with the coefficient plot shown in Figure 16. Notice that there is an salient strip of large coefficients around the 365 scale. This is expected, since that

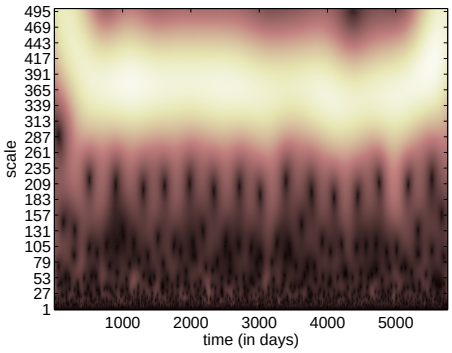

Figure 16: The wavelet coefficient plot of the weather data.

scale corresponds to one year. [3] Unfortunately, the prevalence of this yearly strip overshadows any other weather fluctuations that might be occuring.

In order to discern more obscure fluctuations in the weather patterns, we filter out the wavelet coefficients that correspond to the yearly cycle and reconstruct the signal without those coefficients. The filtered reconstruction would not contain the peaks and valleys of the original signal. By taking a wavelet transform of that this data, we might see some interesting patterns in the average relative powers plot.

A level 6 DWT decomposition using a Debaucie wavelet was used to filter out the yearly cycle from the data. The filtering is performed in MATLAB

[3] We are lucky with this dataset since there is one point per day. This means that the scales easily convert to days. With most datasets though, we need to use equation (18) to convert between scale and period.

Figure 17: Average relative powers for each scale with bars showing the standard deviation for each scale.

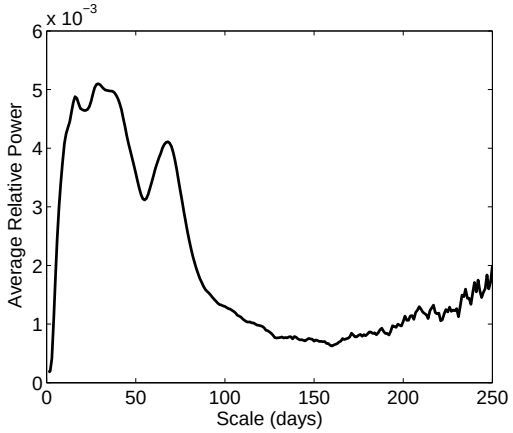

Figure 18: The standard deviation for each scale.

by simply setting the approximation coefficients of the wavelet coefficient matrix to zero. Then, only the detail coefficients are used to reconstruct the signal.

The red line in Figure 17 is an average relative power curve for the filtered weather data. First notice that the scales corresponding to yearly fluctuations (365) are relatively low, because those scales were filtered from the dataset. The rest of the plot shows some interesting fluctuations that would have gone unnoticed without the filtering. The maximum occurs at a scale of 83, which corresponds to a 2.76 month fluctuation.

Each point on the curve of Figure 17 is an average; therefore, we can compute the standard deviation of each fluctuation. The deviation for a given scale is represented by the blue errorbars in the figure. Interestingly, the points with the largest standard deviation seem to fall around scales of 80, with the maximum occuring at 83. Figure 18 shows the standard deviation value for each scale. Notice there is a greater amount of variation over the small scales of around $20 - 40$ and near scale 72. This implies that the coefficients corresponding to these scales tend to fluctuate, and in turn, the weather flucuations occur less consistently. Even if the coefficients have a high relative power, they may not be seen in weather consistently. This could explain why humans cannot easily sense temperature fluctuations of this scale: they are relatively inconsistent fluctuations. Wavelets also allow us to see which fluctuations are active at particular times. An example is shown over 5 years in Figure 19, where a close look supports the result that there are large fluctuations for scales that are around 70-80.

Figure 19: The wavelet coefficient plot with the yearly fluctuations removed.

References

[1] "signal processing"
http://en.wikipedia.org/wiki/Signal_processing

[2] Hundley, Douglas:
"Windows to Wavelets." October 11, 2006. Can be downloaded on-line at http://marcus.whitman.edu/ hundledr

[3] Brown, James Ward and Churchill, Ruel V.
"Fourier Series and Boundary Value Problems" *sixth edition.* 2001

[4] Hundley, Douglas:
Notes - "Chapter 13: Fourier Analysis". Can be downloaded on-line at http://marcus.whitman.edu/ hundledr

[5] Weisstein, Eric W:
"Fourier Series." From MathWorld–A Wolfram Web Resource.
http://mathworld.wolfram.com/FourierSeries.html

[6] Misiti, Michel and Yves; Oppenheim, Georges; Poggi, Jean-Michel:
"Wavelet Toolbox: For Use with MATLAB." The MathWorks.

[7] Torrence, Christopher; Compo, Gilbert P.: "A Practical Guide to Wavelet Analysis". http://atoc.colorado.edu/research/wavelets/

[8] H.G. Stark "Wavelets and Signal Processing: An Application-based Introduction." Springer 2005.

[9] Yan et al.:"Recent trends in weather and seasonal cycles: An analysis of daily data from Europe and China" *Journal of Geophysical research*, Vol. 106, No. D6, pages 5123-5138, March 27,2001.

[10] Kirby, Michael.
"Geometric Data Analysis: An empirical approach to dimensionality reduction and the study of patterns". John Wiley and Sons. 2001.