CHAPTER

1

Preview of Wavelets, Wavelet Filters, and Wavelet Transforms

As mentioned in the Preface, wavelets are used extensively in many varied technical fields. They are usually presented in mathematical formulae, but can actually be understood in terms of simple comparisons or correlations with the signal being analyzed.

In this chapter we introduce you to wavelets and to the wavelet filters that allow us to actually use them in Digital Signal Processing (DSP). Before exploring wavelet transforms as comparisons with wavelets, we first look at some simple everyday "transforms" and show how they too are comparisons. We next show how the familiar discrete Fourier transform (DFT) can also be thought of as comparisons with sinusoids. (In practice we use the speedy fast Fourier transform (FFT) algorithm to implement DFTs. To avoid confusion with the discrete wavelet transforms soon to be explored, we will use the term fast Fourier transform or FFT to represent the discrete Fourier transform.[*]*)*

Time signals that are simple waves of constant frequencies can be processed in a straightforward manner with ordinary FFT methods. Real-world signals, however, often have frequency content that can change over time or have pulses, anomalies, or other "events" at certain specific times. They can be intermittent, transient, or noisy. This type of signal can tell us where something is located on the planet, the health of a human heart, the position and velocity of a "blip" on a RADAR screen, stock market behavior, or the location of underground oil deposits. For these signals, we will usually do better with wavelets.

Jargon Alert: Signals (or noise) that stay at a constant frequency are called "stationary signals" in wavelet terminology. Signals that can change over time are called "non-stationary".

One final thought before beginning: Wavelets deal simultaneously with both time and frequency and require some effort to master. However their powerful capabilities in achieving this feat make them well worth the effort. This conceptual method makes

[*] MATLAB also uses the term "FFT" rather than "DFT" to compute the discrete Fourier transform.

learning them possible without advanced math skills and gives you a gut-level comprehension in the bargain.

The goal of this preview chapter is to introduce you to some new concepts, show you some basic diagrams, familiarize you with the jargon, and give you a preliminary feel for what's going on. Please don't be discouraged if everything is not obvious at first glance.

The next few short chapters will walk you step-by-step through the main concepts and the later chapters should answer most of the remaining questions. You should then be prepared to correctly and confidently use wavelets and to better understand the more advanced math-based texts and papers after you have seen wavelets "in action".

1.1 What is a Wavelet?

A *wavelet* is a waveform of limited duration that has an average value of zero. Unlike sinusoids that theoretically extend from minus to plus infinity, wavelets have a beginning and an end. Figure 1.1–1 shows a representation of a continuous sinusoid and a so-called "continuous" wavelet (a Daubechies 20 wavelet is depicted here).

Sinusoids are smooth and predictable and are good at describing constant-frequency (stationary) signals. Wavelets are irregular, of limited duration, and often non-symmetrical. They are better at describing anomalies, pulses, and other events that start and stop within the signal.

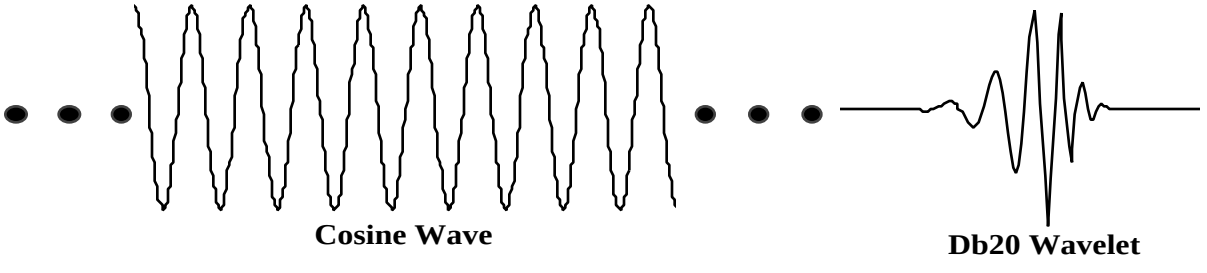


Figure 1.1–1 A portion of an infinitely long sinusoid (a cosine wave is shown here) and a finite length wavelet. Notice the sinusoid has an easily discernible frequency while the wavelet has a *pseudo frequency* in that the frequency varies slightly over the length of the wavelet.

Figure 1.1–2 shows how wavelets can be stretched or "*scaled*" to the same frequency as the anomaly, pulse, or other event. Notice that as the wavelet is stretched it has a lower frequency. Wavelets can also be *shifted* in time to line up with the event. Knowing *how much* the wavelet was stretched and shifted to line up (correlate) with the event gives us information as to the time and frequency of the event.

Jargon Alert:* In DSP *scaling* usually means changing the amplitude of a signal or waveform. In *wavelet* terminology, however, the term *scaling* means stretching or shrinking the wavelet in time. Thus the term *scaling* usually has reference to the *frequency* (or more precisely *pseudo frequency*) of the wavelet. The term *dilation* is also used to describe either stretching *or* shrinking the wavelet in time (despite the dictionary definition).[*]**

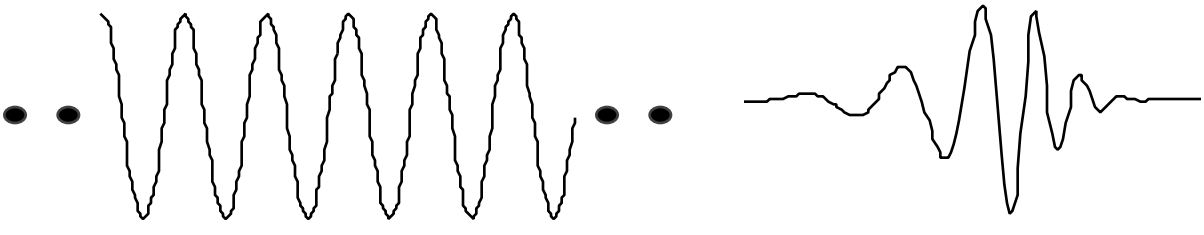


Figure 1.1–2 The infinitely long sinusoid is stretched (or *scaled* in wavelet terminology) and is now a lower frequency. The Db20 wavelet is also stretched (scaled) and its *pseudo frequency* (average frequency) is also lower.

1.2 What is a Wavelet Filter and how is it different from a Wavelet?

Wavelets are a child of the digital age. Some wavelets are defined by a mathematical expression and are drawn as continuous and infinite. These are called *crude wavelets.* However to use them with our digital signal, they must first be converted to *wavelet filters* having a finite number of discrete points. In other words, we evaluate the *crude wavelet equation* at the desired points in time (usually equispaced) to create the filter values at those times.

[*] By this definition "dilated pupils" can mean eyes constricted to pinhole openings. "When I use a word it means just what I choose it to mean—neither more nor less."—Humpty Dumpty in *"Through the Looking Glass"* by Lewis Carroll.

***Jargon Alert*: "Crude" wavelets are generated from an explicit mathematical equation.**

Figure 1.2–1 shows the whimsically-named Mexican Hat *crude* wavelet that looks like the side view of a sombrero. The mathematical expression for this particular wavelet as a function of time (t) is given by

$$mexh(t) = \left\{2/(\sqrt{3}\pi^{-1/4})\right\}(1-t^2)e^{-t^2/2}$$

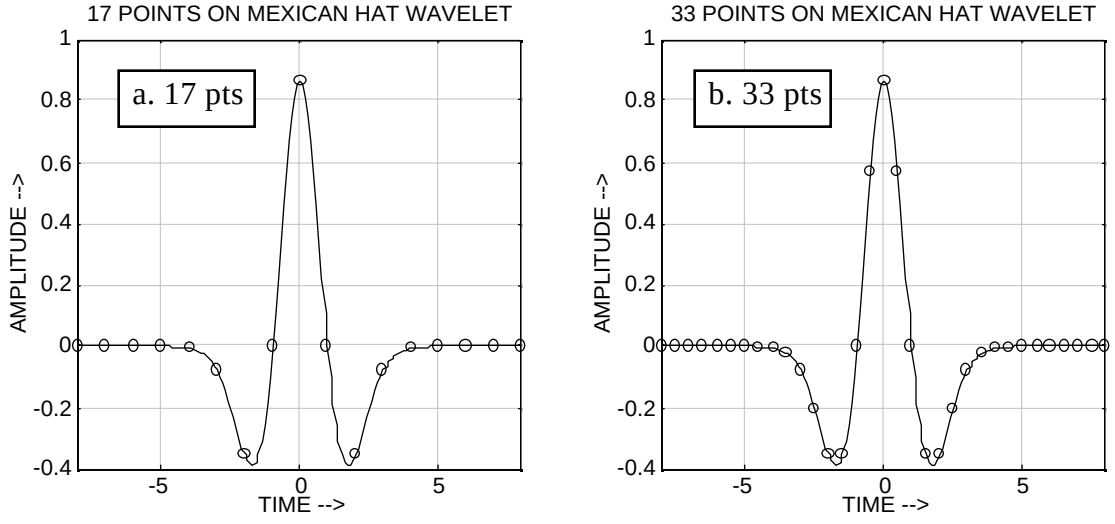


Figure 1.2–1 "Crude" Mexican Hat Wavelet with 17 points (a) then 33 wavelet filter points superimposed on the continuous *(crude)* representation (a then b). Although the defining equation describes an infinite, continuous waveform, by using equispaced discrete points we have created discrete, finite-length filters ready for use with digital computers.

Other wavelets *start out* as filters having as little as 2 points. Then an approximation or *estimation* of a continuous wavelet (for depictions) is *built* by interpolating and extrapolating more points. For these wavelets, there really is no true *continuous* form, only an estimation built from the original filter points. Figure 1.2–2 shows the 4 original filter points (plus 2 zeros at the same spacing) of the Daubechies 4 (Db4) wavelet superimposed on a 768 point estimation of a *"continuous wavelet"* built from these points.*

* We will demonstrate later how we go from 6 points to 12, 24, etc. to 768. MATLAB uses 768 points as a suitable approximation (estimation) of a "continuous" Db4 wavelet.

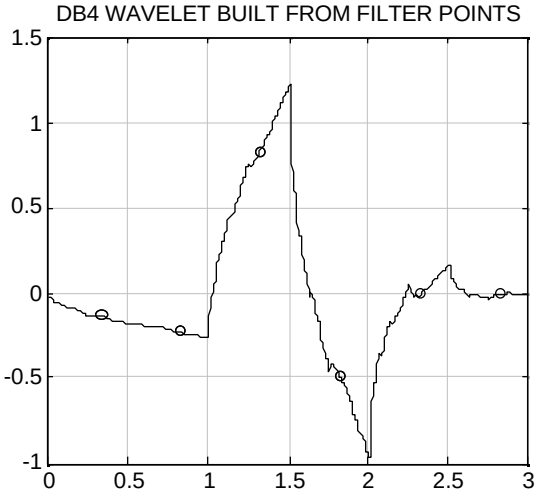


Figure 1.2–2 768 point estimation of a "continuous" Daubechies 4 wavelet built from 6 equispaced filter points (the 4 original filter points and 2 end zeros) superimposed on the graph.

Some wavelets have symmetry (valuable in human vision perception) such as the Biorthogonal wavelet pairs. Shannon or "Sinc" wavelets can find events with specific frequencies. (These are similar to the "sin(x)/x" sinc function filters found in traditional DSP.) Haar wavelets (the shortest) are good for edge detection and reconstructing binary pulses. Coiflets wavelets are good for data with self-similarities (fractals) such as financial trends. Some of the wavelet families are shown below in Figure 1.2–3.

You can even create your own wavelets, if needed. However there is "an embarrassment of riches" in the many wavelets that are already out there and ready to go. With their ability to stretch and shift, wavelets are extremely adaptable. You can usually get by very nicely with choosing a less-than-perfect wavelet. The only "wrong" choice is to avoid wavelets entirely due to an abundant selection.

As you can see (Fig. 1.2–3), wavelets come in various shapes and sizes. By stretching and shifting ("*dilating* and *translating*") the wavelet we can "match" it to the hidden event and thus discover its frequency and location in time. In addition, a particular wavelet shape may match the event unusually well (when stretched and shifted appropriately). This then tells us also about the *shape* of the *event* (It probably looks like the wavelet to obtain such a good match or *correlation*.) For example, the Haar wavelet would match an abrupt discontinuity while the Db20 would match a *chirp* signal (see the first and fourth wavelets in Fig. 1.2–3).

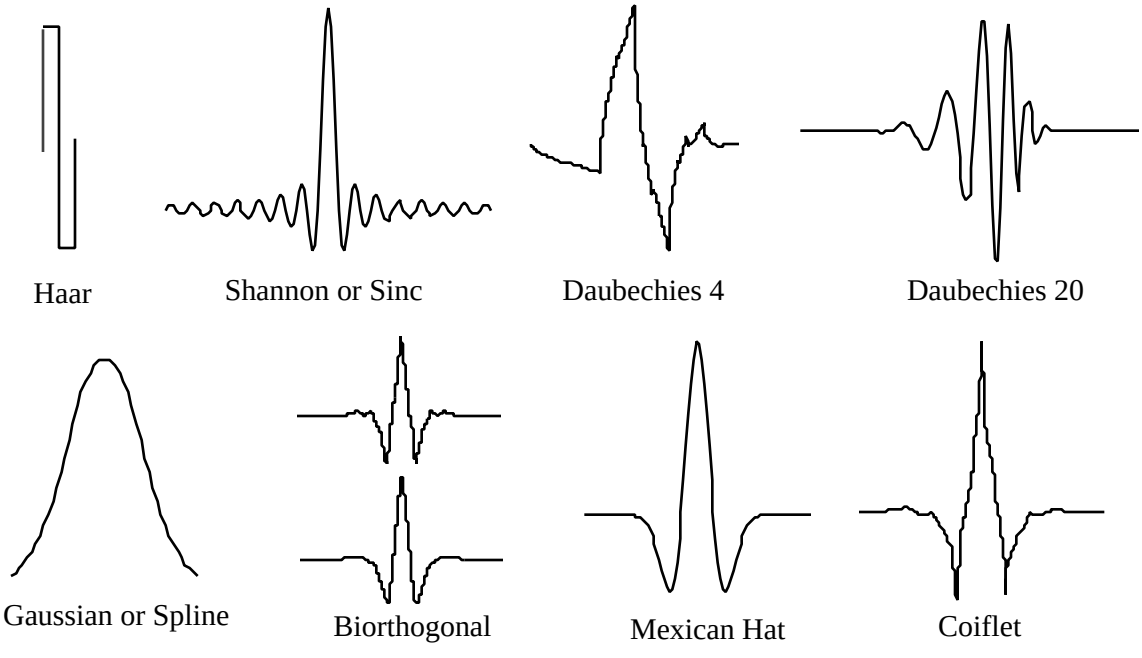


Figure 1.2–3 Examples of types of wavelets. Note 2 wavelets for the Biorthogonal. The Shannon, Gaussian, and Mexican Hat are "crude" wavelets that are defined by an explicit mathematical expression (and whose wavelet filters are obtained from evaluating that expression at specific points in time). The rest are estimations of a "continuous" wavelet built up from the original filter points.

***Jargon Alert*: Shifting or sliding is often referred to as *"translating"* in wavelet terminology.**

1.3 The value of Transforms and Examples of Everyday Use

Perhaps the easiest way to understand wavelet transforms is to first look at some transforms and other concepts we are already familiar with.

The purpose of any transform is to make our job easier, not just to see if we can do it. Suppose, for example, you were asked to *quickly* take the year 1999 and double it. Rather than do direct multiplication you would probably do a home-made "*millennial transform*" in your head something like 1999 = 2000 – 1. Then after transforming you would multiply by 2 to obtain 4000 – 2.

You would then take an "*inverse millennial transform*" of 4000 – 2 = 3998 for the correct answer. You would have described the years in terms of millennia (2000 – 1, 4000 – 2). In other words you *compared* years with millennia.

Another even more common example is when you ask a dieter how the program is working out. They will usually tell you their weight loss, but not their current weight. This in spite of the fact they have been doing daily forward and inverse transforms between the bathroom scale reading and the "brag value* that they share with the world. Here they would be describing their progress in terms of weight loss (instead of bulk weight).

A more advanced example is of course the fast Fourier transform or FFT which allows us to see signals in the "frequency domain". Fig. 1.3–1 shows us the constituent sinusoids of different frequencies (spectrum) that make up the signal. In other words, we are *correlating* (comparing) the signal with these various sinusoids and describing the signal in terms of its frequencies).

***Jargon Alert*: The use of the FFT is now so commonplace that it's results are referred to as the *"frequency domain"* of a signal. (*"Time domain"* is simply the original amplitude vs. time plot of a signal.)**

Thus we can say that in the FFT we are comparing and describing the signal in terms of sinusoids of different frequencies or "*stretched*" sinusoids (to use wavelet terminology). In the wavelet transforms we will be comparing and describing the signal in terms of *stretched* and *shifted* wavelets.

The FFT also allows us to manipulate the transformed data and then do an inverse FFT (*IFFT*) for custom filtering such as eliminating constant frequency noise. For signals with embedded events (the most interesting kind!) the FFT tells us the frequency of the event but not the time that it occurred.

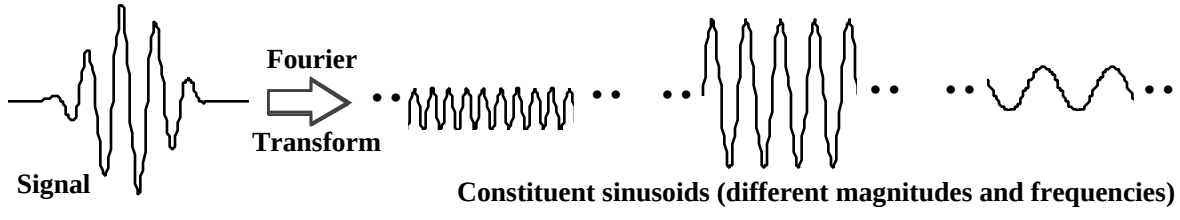


Figure 1.3–1 The signal can be transformed into a number of sinusoids of various sizes and frequencies. When added together (inverse), these sinusoids reconstruct the original signal.

* Apologies to Sir Lawrence Bragg, a Physics and Signal Processing pioneer.

1.4 Short-Time Transforms, Sheet Music, and a first look at Wavelet Transforms

A possible solution to providing both time *and* frequency information about an embedded event might be to divide the total time interval into several shorter time intervals and then take the FFT for each interval. This *time-windowing* method would narrow down the time to that of the interval where the event was found. This *short-time Fourier transform* (STFT) method has been around since 1946 and is still in wide use today.

While the STFT gives us a compromise of sorts between time and frequency information, the accuracy is limited by the size and shape of the window. For example, using many time intervals would give good time resolution but the very short time of each window would not give us good frequency resolution, especially for lower frequency signals.

Longer time intervals for each window would allow us better frequency resolution, but with these fewer, longer windows we would suffer in the time resolution (i.e. with very few windows we would have very few times to associate with the event). Longer time intervals are also not needed for high frequency signals.

Wavelet transforms allow us variable-size windows. We can use long time intervals for more precise low-frequency information and shorter intervals (giving us more precise time information) for the higher frequencies.

We are actually already familiar with this concept. Ordinary sheet music is an everyday example of displaying both time and frequency information—and it happens to be set up very similar to a wavelet transform display.

Besides demonstrating the concept of longer time for lower frequencies and shorter times for higher frequencies, sheet music even has a logarithmic vertical frequency scale (each octave is twice the frequency of the octave below it). Musicians know that low notes take longer to form in a musical instrument. (Engineers know it takes a longer *time* to examine a low *frequency* signal.) This is why the Piccolo solo from John Philip Sousa's "Stars & Stripes Forever" (Fig. 1.4–1) can't be played on a Tuba.*

* There are in fact recordings of tuba players that can press the big valves fast enough to play all the notes in the piccolo solo, but some of the notes themselves do not have enough time to form in the horn and are not heard. DSP engineers would refer to this as *insufficient integration time.*

Figure 1.4–1 Portion of the piccolo solo from John Philip Sousa's "Stars and Stripes Forever"

Figure 1.4–2 shows octaves of the note "C" on sheet music (left). A wavelet display is very much like the configuration to the right. If we use a base 2 log scale so we have increasing *frequency* in powers of 2 as the y axis we can see a remarkable comparison between the sheet music and the wavelet display. (We will see more of this *octave* or *power of 2* behavior a little later in the *discrete wavelet transforms*.)

If you think about it, sheet music has another dimension besides discrete time and frequency. The volume or *magnitude* of each note is indicated by discrete indicators such as *ff* for *fortissimo* or very loud.* Sheet music even describes the time increments. For example, Tempo 60 indicates 60 beats per minute or 1 beat per second.

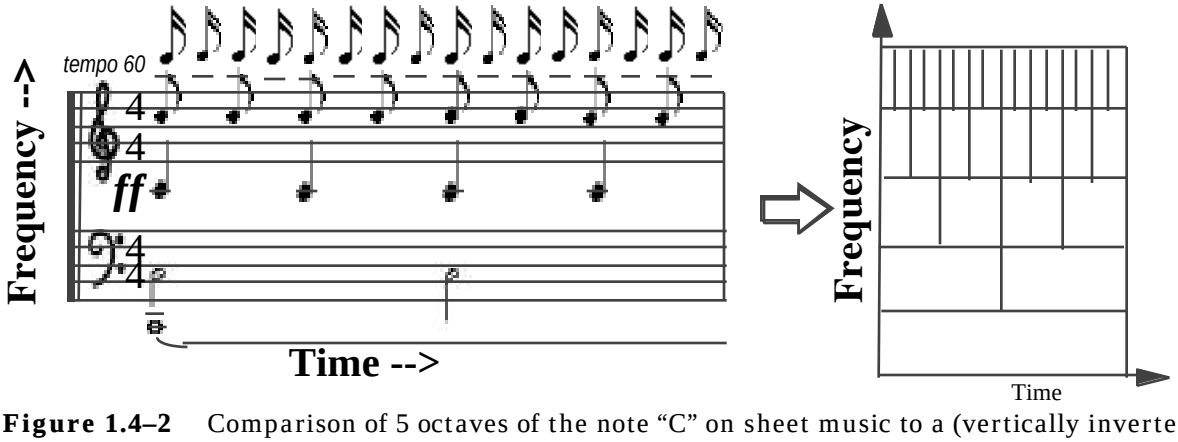


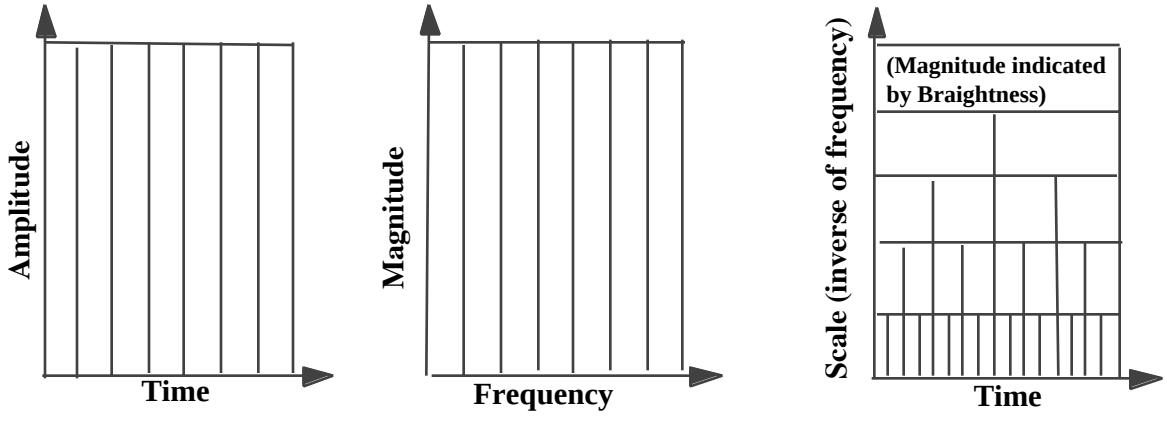
Figure 1.4–2 Comparison of 5 octaves of the note "C" on sheet music to a (vertically inverted) wavelet transform time/frequency diagram. Note the powers of 2 in both time (linear scale) and frequency (log scale) and that higher frequencies require less time for adequate resolution. Note how magnitude is indicated by *fortissimo* (*ff*) on sheet music. Magnitude on the time/frequency diagram at right is indicated by the brightness or color. Note that both representations indicate discrete frequencies, discrete times, and discrete magnitudes.

* *Fortissimo* is Italian for very loud, *pianissimo* or *pp* indicates very soft. The other intensity notations (*f, mf, mp, p, etc*.) indicate discrete levels of loudness. The modern piano or *pianoforte* got its name from its ability to play notes either *piano* (soft) or *forte* (loud).

In most (but not all) texts, the wavelet display at the right of Figure 1.4–2 is drawn inverted ("flipped" vertically). In other words the high frequencies are on the bottom and the lower frequencies are on top. The y axis on most wavelet displays shows increasing *scale* (stretching of the wavelet) rather than increasing frequency.

Figure 1.4–3 shows a "sneak preview" of a wavelet transform display (c) and how it compares to an ordinary time-domain (a) and frequency-domain (b) display.

Imagine if a musical composer had to "de-compose"* by changing a single note. If he used a musical form similar to (a) he would have to change *all* the notes at a particular time (or "beat"). If he used a form similar to (b) he would have to change *all* the notes of a particular frequency (or "pitch"). Using a form similar to (c), as is sheet music, he can change only one note. This is how denoising or compression is accomplished using wavelet technology. An unwanted or unneeded portion of data (a computational "wrong note" if you will) can be easily identified and then changed or deleted without appreciable degradation of the signal or image.



a. 2-DimensionalTime Domain Display

b. 2-D Frequency Domain Display (FFT)

c. 3-D Wavelet Display. Brightness = Magnitude

Figure 1.4–3 Time domain, frequency domain, and "wavelet domain" display. Note that the wavelet display (c) incorporates both time *and* frequency. Note the similarity to sheet music except that this display (c) is inverted with increasing stretching or *scale* (inverse of frequency) as the vertical or y-axis.

* "*Decomposing*" is a term actually used in the *discrete wavelet transforms* we will soon discuss. Beethoven is even now decomposing. (You knew that was coming, right?)

1.5 Example of the Fast Fourier Transform (FFT) with an Embedded Pulse Signal

In this example we start with a point-by-point comparison of a time-domain pulse signal (A) with a high frequency *sinusoid* of constant frequency (B) as shown in Figure 1.5–1. We obtain a single "goodness" value from this comparison (a *correlation value*) which indicates how much of that particular sinusoid is found in our original pulse signal.

We can observe that the pulse has 5 cycles in 1/4 of a second. This means that it has a frequency of 20 cycles in one second or *20 Hz*.

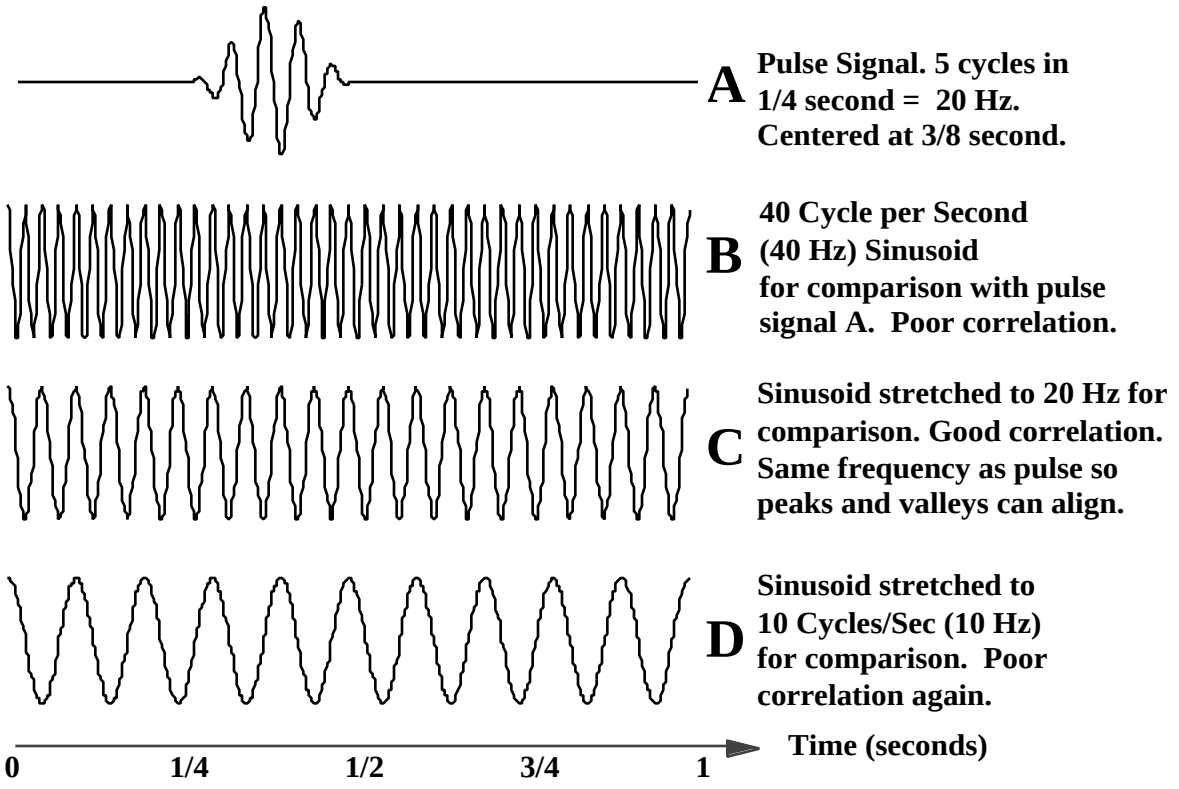


Figure 1.5–1 FFT-type comparison of Pulse Signal with several stretched sinusoids. The pulse (A) has 5 discernible "peaks" (local maxima) and 5 discernible "valleys" (local minima). These peaks and valleys will line up best with those of sinusoid C. (We discuss shifts in time or *phase shifts* to better align the pulse and sinusoid in a later chapter).

The first comparison sinusoid (B) has twice the frequency of the pulse or 40 Hz. Even in the time interval where the signal is non-zero (the pulse) it doesn't seem intuitively like the comparison would be very good. (A small mathematical correlation value bears this out.)

By lowering the frequency of (B) from 40 to 20 Hz (waveform C) we are effectively "stretching in time" (*scaling)* the sinusoid (B) by 2 so it now has only 20 cycles in 1 second. We compare (C) point-by-point again over the 1 second interval with the pulse (A). This time the correlation of the pulse (A) with the comparison sinusoid (C) is very good. The peaks and valleys of (C) and of the pulse portion of (A) align in time (or can be easily phase-shifted to align) and thus we obtain a large correlation value.

This same diagram (Fig. 1.5–1) shows us one more comparison of the pulse with (A). This is our original sinusoid (B) stretched by 4 so it has only 10 cycles in the 1 second interval. The correlation is poor once again because the peaks and valleys of (A) and (D) no longer line up. We could continue stretching until the sinusoid becomes a straight line having zero frequency or "DC" (named for the zero frequency of Direct Current) but all these comparisons will be increasingly poor.

An actual FFT compares many "stretched" sinusoids ("*analysis signals*") to the original pulse rather than just the 3 shown in Figure 1.5–1. The best correlation is found when the comparison sinusoid frequency exactly matches that of the pulse signal. Figure 1.5–2 shows the first part of an actual FFT of our pulse signal (A).

The locations of our sample comparison sinusoids (B, C, and D) are indicated by the large dots. (The spectrum of our pulse signal is shown by the solid curve.) Again, the FFT tells us correctly that the pulse has primarily a frequency of 20 Hz, but does *not* tell us where the pulse is located in time.*

---

* The generalized mathematical equation for the DFT (implemented by the FFT algorithm) is a shortcut for indicating the real cosines and imaginary sines ($\mathbf{X_k}$) that make up the signal ($\mathbf{x_n}$). (We showed *cosines only* in the above example to simplify and visually portray the process.)

$$X_k = \sum x_n \cos(2\pi nk / N) - j \sum x_n \sin(2\pi nk / N)$$

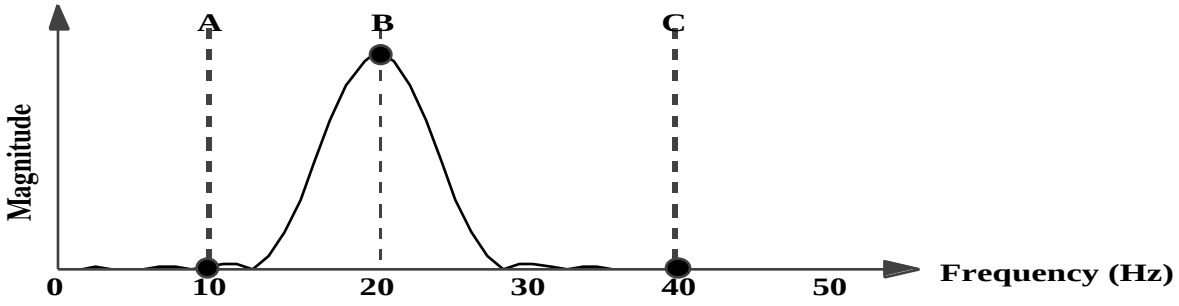


Figure 1.5–2 Actual FFT plot of the above pulse signal with the three comparison sinusoids.

1.6 Examples using the Continuous Wavelet Transform

Wavelet transforms are exciting because they too are comparisons, but instead of correlating with various stretched, constant frequency sinusoid *waves* they use smaller or shorter waveforms ("*wave–lets*") that can start and stop. In other words, the *fast Fourier transform* relates the signal to *sinusoids* while the *wavelet transforms* relate signals to *wavelets*. In the real world of digital computers, *wavelet transforms* relate our discrete, finite (digital) *signal* to the discrete, finite, *wavelet filters*.

Fig. 1.6–1 shows us some of the constituent wavelets that have been shifted and stretched (from the mother wavelet) that make up the signal. In other words, we are *correlating* (comparing) the signal with these various shifted, stretched wavelets. An actual wavelet transform compares many stretched and shifted wavelets ("*analysis wavelets*") to the original pulse rather than just these few shown in Figure 1.6–1.

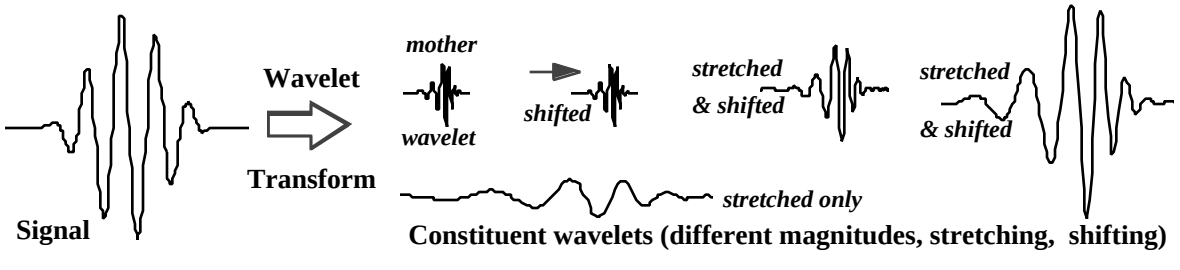


Figure 1.6–1 The signal can be transformed into a number of wavelets of various stretching, shifting, and magnitude. When added together these wavelets reconstruct the original signal.

Figure 1.6–2 demonstrates the stretching and shifting process for the continuous wavelet transform. Instead of sinusoids for our comparisons, we will use wavelets. Waveform (B) shows a Daubechies 20 (Db20) wavelet about 1/8 second long that starts at the beginning (t = 0) and effectively ends well before 1/4 second. The zero values are extended to the full 1 second. The point-by-point comparison* with our pulse signal (A) will be very poor and we will obtain a very small correlation value.

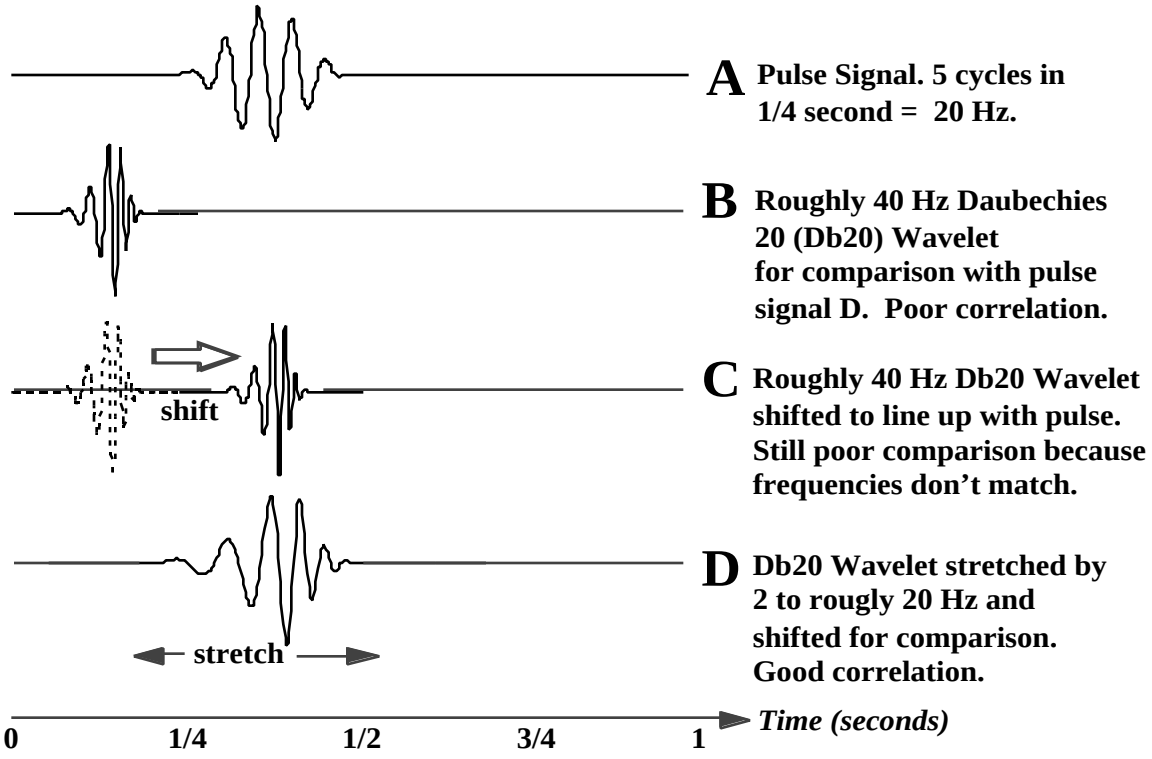


Figure 1.6–2 CWT-type comparison of pulse signal with several stretched and shifted wavelets. If the energy of the wavelet and the signal are both unity, these the comparisons are *correlation coefficients.* Note: Knowing how much (B) was stretched and shifted to match (A) tells us the location and approximate frequency of the pulse. Also, a good match with this particular wavelet tells us that the pulse looks a lot like the wavelet (sinusoidal in this case).

* The pulse and the wavelets are drawn here as continuous functions. In DSP we would have a finite number of data points for the signal and we would be comparing these point-by-point with the finite number of values of the Db20 wavelet filter.

In the previous FFT discussion we proceeded directly to stretching. In the wavelet transforms here we first shift the unstretched basic or *mother* wavelet slightly to the right and perform *another* comparison of the signal with this new waveform to get *another* correlation value. We continue to shift and when the Db20 wavelet is in the position shown in (C) we get a little better comparison than with (B), but still very poor because (C) and (A) are different frequencies.

Jargon Alert*: The unstretched wavelet is often referred to as the *"mother wavelet"*. The Db20 wavelet filter we are using here starts out as 20 points long (hence the name) but can be stretched to many more points. (A counterpart lowpass filter used in the upcoming *discrete wavelet transform* is often called a *"father wavelet"*. Honest!)[*]**

After we have continued shifting the wavelet all the way to the end of the 1 second time interval, we start over with a slightly stretched wavelet at the beginning and repeatedly shift to the right to obtain *another full set* of these correlation values.

Waveform (D) shows the Db20 wavelet stretched to where the frequency (pseudo-frequency—ref. Fig. 1.1–1) is roughly the same as the pulse (A) and shifted to the right until the peaks and valleys line up fairly well. At these particular amounts of shifting and stretching we should obtain a very good comparison and a large correlation value. Further *shifting* to the right, however, even at this same stretching will yield increasingly poor correlations. Further *stretching* doesn't help at all because even when lined up, the pulse and the over-stretched wavelet won't be the same frequency.

In the CWT we have one correlation value for every shift of every stretched wavelet.[†] To show the correlation values (quality of the "match") for all these stretches and shifts, we use a 3-D display. Figure 1.6–3 shows a Continuous Wavelet Transform (CWT) display for the pulse signal (A) in our example.

---

[*] Mathematically speaking, we replace the infinitely oscillating sinusoid basis functions in the FFT with a set of locally oscillating basis functions which are stretched and shifted versions of the fundamental, real-valued bandpass *mother wavelet.* When correctly combined with stretched and shifted versions of the fundamental, real-valued lowpass *father wavelet* they form an orthonormal basis expansion for signals.

[†] The generalized equation for the CWT (shown below) is a shortcut that shows that the correlation coefficients depend on both the stretching and the shifting of the wavelet, ψ, to match the signal ($\mathbf{x_n}$ here) as we have just seen. The equation shows that when the "dilated and translated" wavelet matches the signal the summation will produce a large correlation value.

$$C(stretching, shifting) = \sum x_n \psi\,(stretching, shifting)$$

The bright spots indicate where the peaks and valleys of the stretched and shifted wavelet align best with the peaks and valleys of the embedded pulse (dark when no alignment, dimmer where only *some* peaks and valleys line up, but brightest where *all* the peaks and valleys align). In this simple example, stretching the wavelet by a factor of 2 from 40 to 20 Hz (stretching the filter from the original 20 points to 40 points) and shifting it 3/8 second in time gave the best correlation and agrees with what we knew *a priori* or "up front" about the pulse (pulse centered at 3/8 second, pulse frequency 20 Hz).

We chose the Db20 wavelet because it looks a little like the pulse signal. If we didn't know *a priori* what the event looked like we could try several wavelets (easily switched in software) to see which produced a CWT display with the brightest spots (indicating best correlation). This would tell us something about the shape of the event.

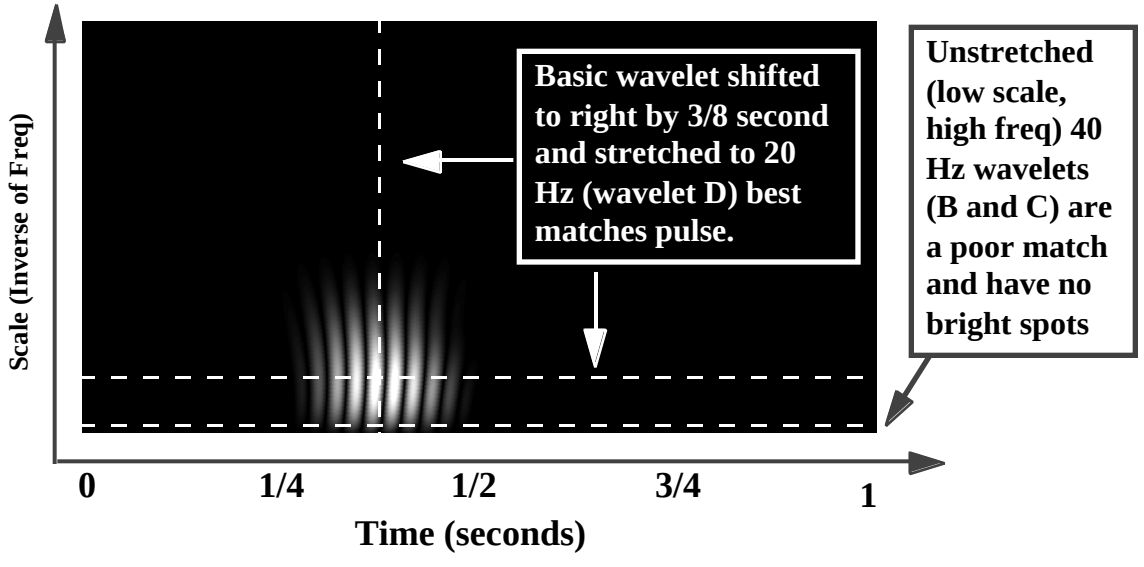


Figure 1.6–3: Actual CWT display of the above example indicating the time and frequency of the Pulse Signal. Shifting or *translation* of the wavelet (filter) in time is the x or horizontal axis, stretching or *dilation* of the wavelet (the inverse of its pseudo frequency) is the y or vertical axis, and the "goodness" of the correlation of the wavelet (at each x-y point) with the signal (pulse) is indicated by brightness. The fainter bands indicate where some of the peaks and valleys line up while the center of the brightest band (in the cross-hairs) shows the best "match" or correlation.

For the simple tutorial example above we could have just visually discerned the location and frequency of the pulse (A). The next example is a little more

representative of wavelets in the real world where location and frequency are not visible to the naked eye. Wavelets can be used to analyze *local* events as we will now see.

We construct a 300 point slowly varying sine wave signal and add a tiny "glitch" or discontinuity (in slope) at time = 180 as shown in Figure 1.6–4 (a). We would not notice the glitch unless we were looking at the closeup (b). Using a conventional fast Fourier transform (FFT) on the signal shows its frequency components (Fig. 1.6–5). The low frequency of the sine wave is easy to notice, but the small glitch cannot be seen.

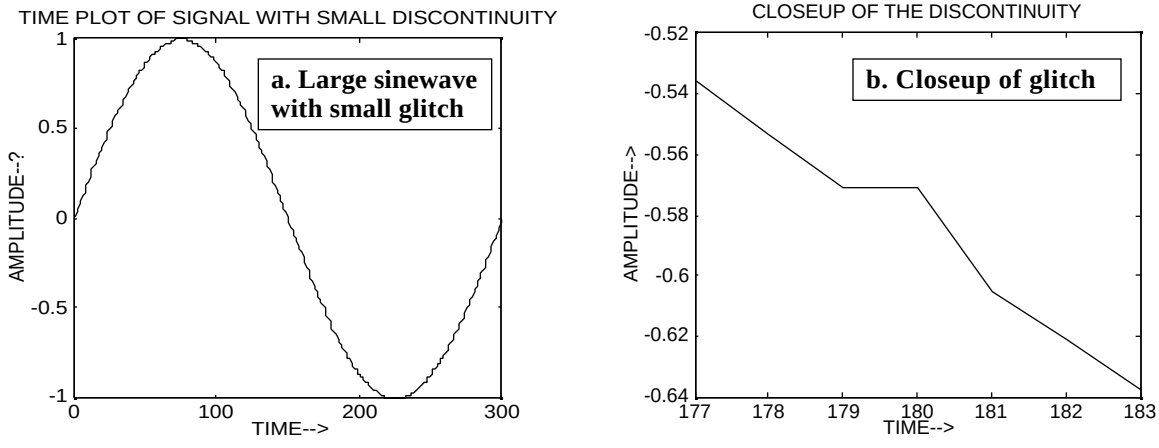


Figure 1.6–4 Very small discontinuity at time = 180 (a) cannot be seen without a closeup (b).

While neither the (full-length) time domain or frequency domain display tell us much about the glitch, the CWT wavelet display (Fig. 1.6–6) clearly shows a vertical line at time = 180 and at low scales. (The wavelet has very little stretching at low scales, indicating that the glitch was very short.) The CWT also compares well to the large oscillating sine wave which hides the glitch. At these higher scales the wavelet has been stretched (to a lower frequency) and thus "finds" the peak and the valley of the sine wave to be at time = 75 and 225 (see Fig. 1.6–4). For this short discontinuity we used a short 4-point Db4 wavelet (as shown) for best comparison.

Note that if we were to vertically invert this display with the lower frequencies shown on the bottom, we would see many similarities to the sheet music notation described earlier.

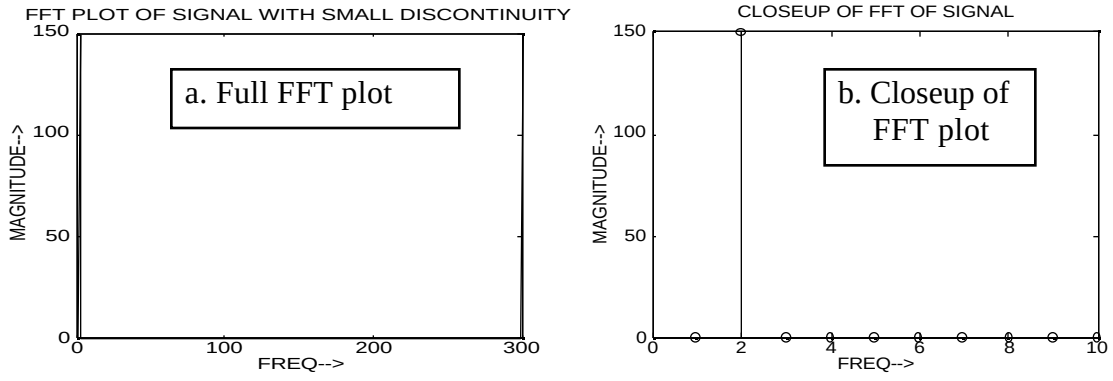


Figure 1.6–5 FFT magnitude plot (a) clearly indicates the presence of a large low-frequency sinusoid. A closeup of the FFT (b) further defines the sine wave in frequency, but does not help to find the glitch. Note: Even if the glitch were large enough to show a noticeable frequency component in the FFT, this would still not indicate the *time* of the glitch-event.

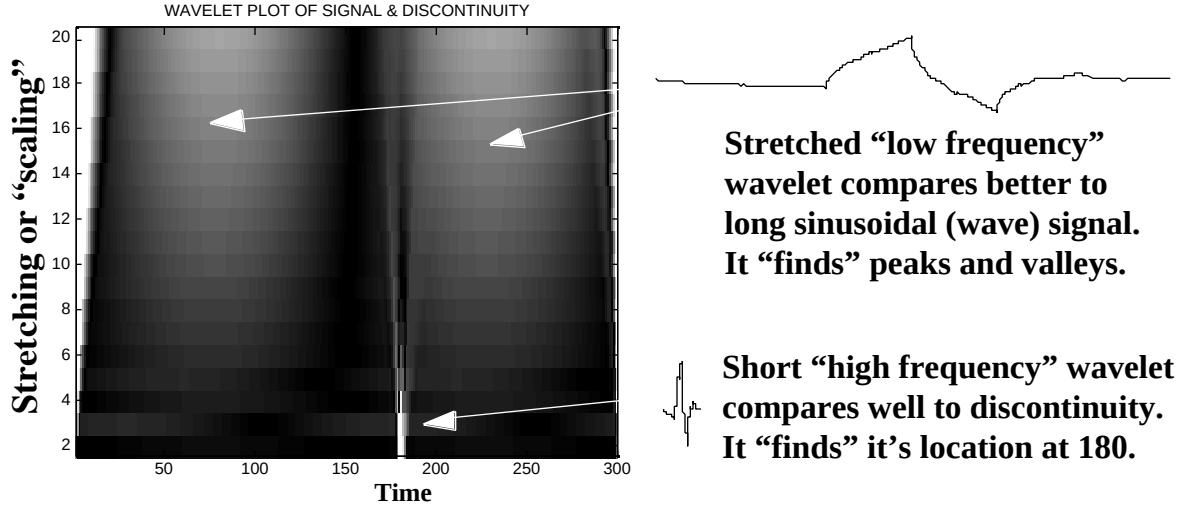


Figure 1.6–6 CWT display of result of correlation of signal with various scales (stretching) of the Daubechies 4 (Db4) wavelet. The short mother wavelet (filter) at scale = 2 is only 4 points long (the "continuous" estimation of the Db4 is drawn). This short filter compares well with the short glitch at time 180. The stretched wavelet (filter) at scale = 20 (top) is about 50 points long and compares better to the large 300 point sinusoid of the main signal than to the glitch.[*]).

[*] We show only to scale = 20 here. A CWT display with much larger scale values would show the best correlation with the sinusoid to be at about scale = 150. The Db4 wavelet filter is stretched to 300 points at scale = 150, and best fits the 300 points of the sine-wave signal. However the glitch at time = 180 would not be so easily discernible on such a large scale display ant thus it is not shown. We can also adequately locate the peak and valley of the sine wave at 75 and 225 using just this abbreviated-scale CWT plot—compare Fig 1.6–3 (a).

1.7 A First Glance at the Undecimated Discrete Wavelet Transform (UDWT)

Besides acting as a "microscope" to find hidden events in our data as we have just seen in the *continuous* wavelet transform (CWT) display, *Discrete wavelet transform*s (DWTs) can also separate the data into various frequency components, as does the FFT. We already know that the FFT is used extensively to remove unwanted noise that is prevalent throughout an entire signal such as a 60 Hz hum.

Unlike the FFT, however, the *discrete wavelet transform* allows us to remove frequency components at *specific times* in the data. This allows us a powerful capability to throw out the "bad" and keep the "good" part of the data for denoising or compression. Discrete wavelet transforms also incorporate easily computed *inverse* transforms (IDWTs) that allow us to reconstruct the signal after we have identified and removed noise or superfluous data.

A fair question before proceeding is "What is *continuous* about the *continuous wavelet transform* in our world of digital computers that works with discrete data? Aren't *all* these transforms "discrete"? What then differentiates these *discrete* wavelet transforms from the so-called *continuous* ones?"

The answer is that although all wavelet transforms in DSP *are* technically *discrete*[*], the so-called *continuous* wavelet transform (CWT) differs in how it stretches and shifts. The term "continuous" in a CWT indicates all possible integer factors of shifting and stretching (e.g. by 2, 3, 4, 5, etc.) rather than a mathematically continuous function. By contrast, we will see that *discrete* wavelet transforms stretch and shift by powers of 2. Another difference is that the *continuous* wavelet transform uses only the one wavelet filter while the *discrete* wavelet transform uses 3 additional filters as we will soon discover. We will now look at the 2 best known and most utilized of the DWTs—the *conventional discrete wavelet transform* (DWT) and the *undecimated discrete wavelet transform* (UDWT).

***Jargon Alert*: Stretching or shifting by powers of 2 is often referred to as *"dyadic"*. For example, *dyadic dilation* means stretching (or shrinking) by factors of 2 (e.g. 2, 4, 8, 16 etc).**

[*] As is the *discrete Fourier transform* (implemented by the FFT algorithm).

The *undecimated discrete wavelet transform* (we'll explain why it's called "*undecimated*" in a moment) is not as well known as the c*onventional* discrete wavelet transform. However it is simpler to understand than the conventional DWT, compares better with the *continuous wavelet transform* we have just studied and is similar enough to the DWT to provide a clear learning "bridge". UDWTs also don't have the "aliasing" problems we will soon encounter and discuss in the conventional DWT.

Figure 1.7–1, (a) shows the simplest UDWT. The first thing you will notice on this signal flow diagram is that it has 4 filters. This is called a *filter bank* for this reason. (We will run into this type of figure a lot during the book, but it's not necessary to completely understand it at this point in the preview.) These 4 filters are closely related (complimentary) as we will see later.

The left half of the UDWT is called the *decomposition* or *analysis* portion and comprises the *forward* transform. The right half is called the *reconstruction* or *synthesis* portion and comprises the *inverse* transform.* The vertical bar separating the 2 halves represents the area where we can add more complexity (and capability), but we proceed by ignoring the bar for now.

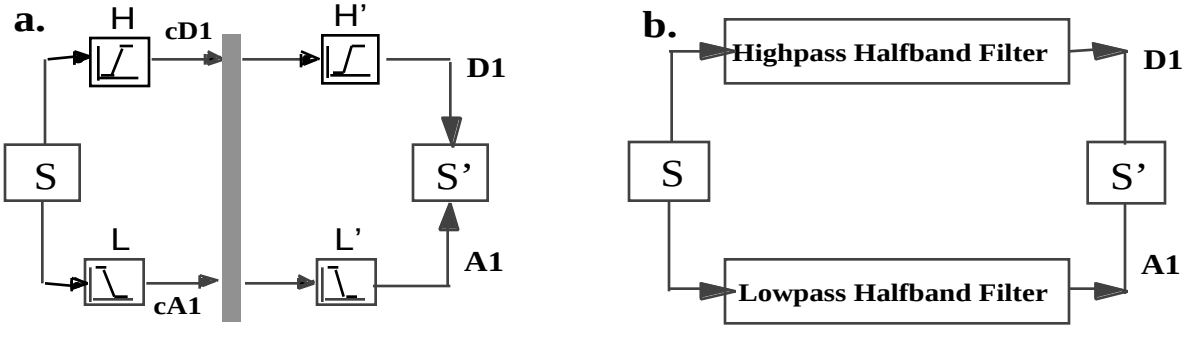


Figure 1.7–1 Single-level undecimated discrete wavelet transform (UDWT) filter bank shown at left (a). The *forward transform* or *analysis* part is the half to the left of the vertical bar and is usually referred to as the *decomposition portion*. The *inverse transform* or *synthesis* part is the half to the right of the vertical bar and is called the *reconstruction* portion. The bar itself is where additional levels of decomposition and reconstruction can be inserted, producing higher level UDWTs. If the data is left unchanged (no activity in the vertical bar) the functional equivalent of this single-level UDWT is shown in the right diagram (b).

* The terms UDWT and IUDWT are occasionally used as labels for the forward and inverse transforms. Usually, however, the term UDWT refers to both halves. The discrete Fourier transform (DFT) and the functionally-equivalent *fast Fourier transform* (FFT) also use the terms *analysis* and *synthesis* to describe their left and right (forward and inverse) halves (FFT and IFFT).

***Jargon Alert*: "*Decomposition*" in wavelet terminology means splitting the signal into 2 parts using a highpass and a lowpass filter. Each of the 2 parts themselves can be *decomposed* further (split into more parts) using more filters. "*Reconstruction*" means using filters to combine the parts. "*Perfect reconstruction*" means that that the signal at the end is the same as the original signal (except for a possible delay and a constant of multiplication).**

On the upper path of Fig. 1.7–1, (a) the signal, **S**, is first filtered by **H** (*highpass decomposition filter*) to produce the coefficients **cD1**. At this point we can do further decomposition (analysis) for compression or denoising, but for now we will proceed directly to reconstruct (synthesize) the signal. **cD1** is next filtered by **H'** (*highpass reconstruction filter*) to produce the *Details* (**D1**). The same signal is also filtered on the lower path by the *lowpass decomposition filter* **L** to produce the coefficients **cA1** and then by the *lowpass reconstruction filter* **L'** to produce the *Approximation* (**A1**).

***Jargon Alert*: "*Approximation*" in Wavelets is the smoothed signal after all the lowpass filtering. "*Details*" are the residual noise after all the highpass filtering. cA1 designates the *Approximation Coefficients* and cD1 designates the *Details Coefficients.* These coefficients can be broken down (decomposed) into further coefficients in higher level systems (depicted by the vertical bar in the center of the diagrams).**

H and **H'** together produce a highpass *halfband filter* while **L** and **L**' produce a lowpass *halfband filter* as seen in Fig. 1.7–1, (b). These 4 *wavelet filters are* non-ideal filters and there is some overlap as depicted below in the frequency allocation diagram (Figure 1.7–2).

Jargon Alert*: "*Halfband filters*" split the frequency into a lowpass and a highpass "half" (A1 and D1 here), usually with some overlap. We refer to all 4 filters as *wavelet filters,* but some texts refer to the 2 lowpass filters as *scaling function filters* and the 2 highpass as *wavelet filters.* Some call only H' the *wavelet filter.* Again, *Caveat Emptor.

Fig. 1.7–1 showed a *single-level* UDWT. Fig. 1.7–3 shows a *2-level* UDWT. (Note the additional decomposition and reconstruction as **cA1** is split into **cD2** and **cA2**.) Multi-level UDWTs allow us to stretch the filters, similar to what we did in the CWT, except that it is done dyadically (i.e. by factors of 2). The stretching is done by *upsampling by 2* (e.g. "$\mathbf{H_{up}}$") and then filtering (a common method of interpolation in DSP). With this further decomposing and reconstruction we can split the signal into more frequency bands (Fig. 1.7–4).

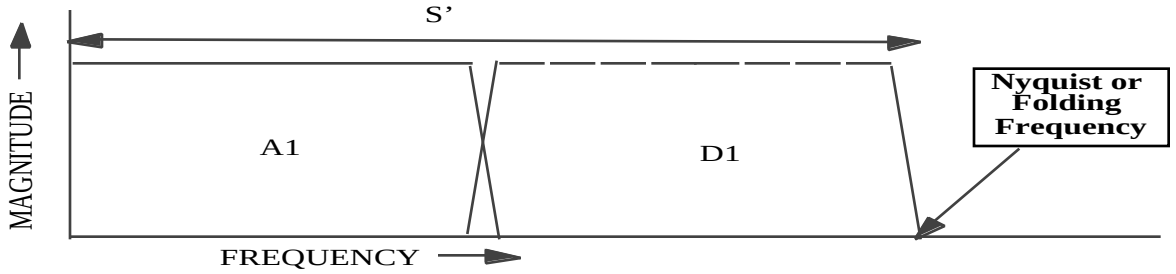


Figure 1.7–2 Frequency allocation after a single-level UDWT*. The diagram is illustrative only and the actual shape depends on the wavelet filters. Note overlap from non-ideal filtering. When the Details and Approximations are added together they *reconstruct* **S'** (**D1 + A1 = S'**) which is identical to the original signal, **S**, except for a delay and usually a constant of multiplication. For a very simple denoising, we could just discard these high frequencies in **D1** (for whatever time period in the signal we choose) and **A1** by itself would be a rudimentary "denoised" signal.

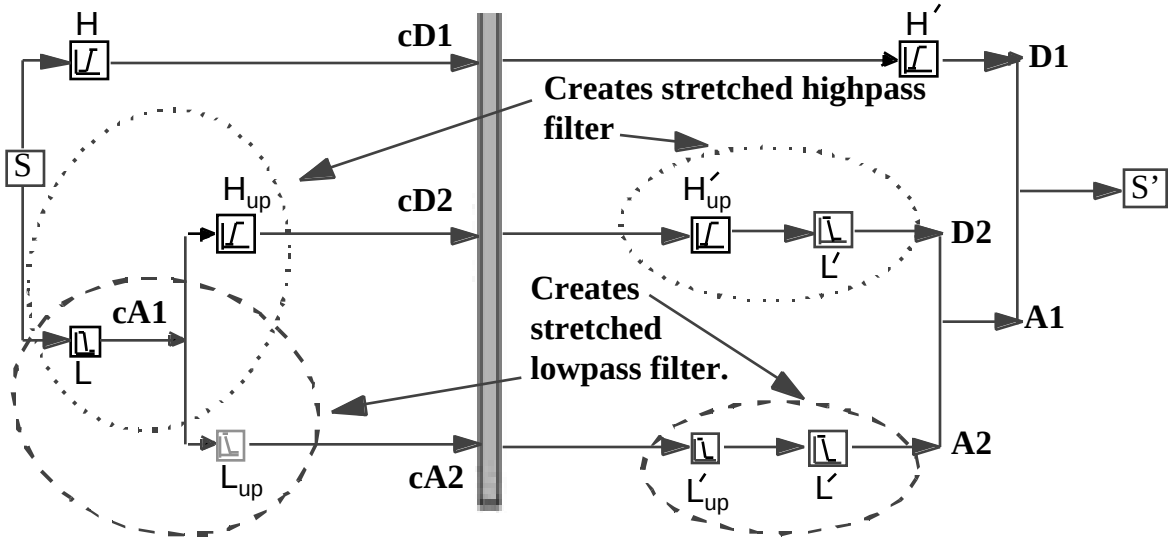


Figure 1.7–3 A 2-level UDWT. The signal, **S**, is split into **cD1** and **cA1**. We then split **cA1** into **cD2** and **cA2**. The final signal, **S'**, is now reconstructed by combining **A1** and **D1**. Since **A1** is obtained by combining **D2** and **A2**, **S' = A1 + D1 = A2 + D2 + D1**. We could do some denoising or compression at this point. If there was nothing of interest in **D2**, for example, we could zero it out and would have **S' = A2 + 0 + D1 = A2 + D1**. Notice that if we set the coefficients **cD2** to zero that this would also cause **D2** to be zero (filtering of zeros still produces zeros). We will discuss multi-level UDWTs in detail later.

* The Nyquist or "folding" frequency is the highest possible frequency without aliasing (discussed shortly).

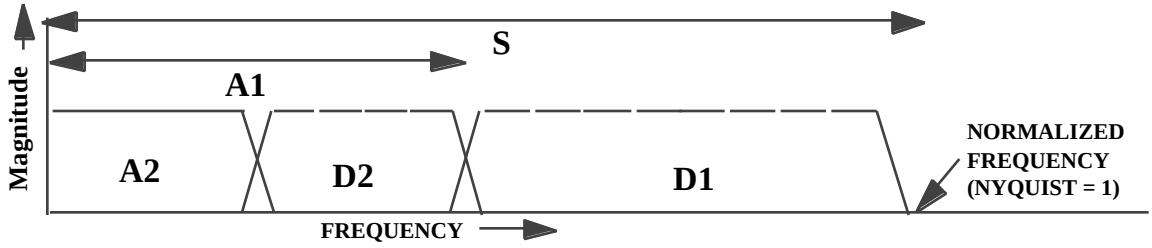


Figure 1.7–4 Frequency allocation after a 2-level UDWT. Note that the A1 is now split into 2 sub-bands. This allows us better flexibility in denoising or compression.

Jargon Alert: "Upsampling by 2" **means placing zeros between the existing data points. For example, A time-sequence of the numbers [6, 5, 4, 3] would become with upsampling by 2**

[6, 0, 5, 0, 4, 0, 3].

or in some cases

[0, 6, 0, 5, 0, 4, 0, 3, 0]

with a leading and/or a trailing zero (more on interpolation later).

The UDWT (sometimes referred to as the *redundant* DWT or RDWT) with it's method of inserting zeros as part of the stretching of the filters is thus also called the "*A' Trous*" method which is French for "*with holes*" (zeros).

A 4-level UDWT with more stretched filters splits the signal into 5 frequency sub-bands (Figure 1.7–5).

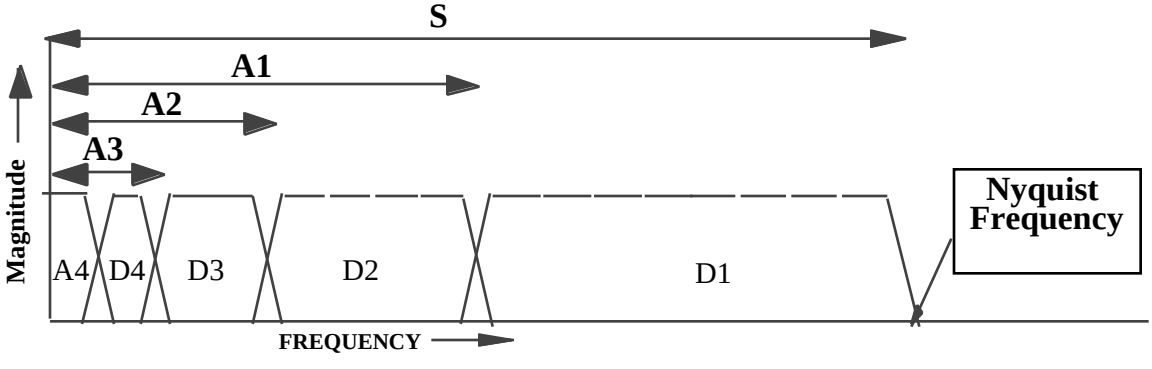


Figure 1.7–5 Frequency allocation in a 4-level UDWT. Note that **S** is split into **A1** and **D1**, **A1** is split into **A2** and **D2**, **A2** is split into **A3** and **D3**, and finally **A3** is split into **A4** and **D4**.

The 4-level UDWT signal flow diagram is not shown in this preview because of its size and complexity, but it functions very similar to the 2-level UDWT except that the filters are stretched not only by 2, but also by 4 and 8 to give us these additional sub-bands. Don't worry if you don't understand these multi-level systems in this preview. We'll talk more about them later. They are presented here mainly to show you what they look like and how they have stretched filters similar to the CWT. Hang in there.

1.8 A First Glance at the conventional Discrete Wavelet Transform (DWT)

We stretched the wavelet continuously (by integer steps) in the CWT and dyadically (by factors of 2) in the UDWT. In the conventional DWT we shrink the *signal* instead (dyadically) and compare it to the unchanged wavelet filters. We do this through "*downsampling by 2*".

***Jargon Alert*: *"Downsampling by 2"* means discarding every other signal sample. For example a sequence of numbers (signal) [5 4 3 2] becomes [5 3] (or [4 2] depending on where you start). This is also referred to in wavelet terminology as *"decimation by 2"* (in spite of the dictionary definition for the prefix "Deci").**

A single-level conventional DWT is shown in Figure 1.8–1 with the *decomposition* or *analysis* portion on the left and the *reconstruction* or *synthesis* portion on the right half*. Downsampling and upsampling by 2 is indicated by the arrows in the circles. For example, if we downsampled then immediately upsampled [5 4 3 2] we would first have [5 3] and then [5 0 3 0].

Further decomposition and reconstruction (a higher level DWT) is done in the vertical bar separating the 2 halves. The single-level *DWT* shown here is the same as the single level *UDWT* except that in discarding every other point, we have to deal with *aliasing*. We must also be concerned with *shift invariance* (do we throw away the odd or the even values?—it matters!).†

* As with the UDWT, the term DWT usually refers to both the left half or *forward* DWT and the right half or *inverse* DWT (occasionally called the IDWT).

† The UDWT has neither of these problems. We will discuss aliasing and shift invariance more in later chapters.

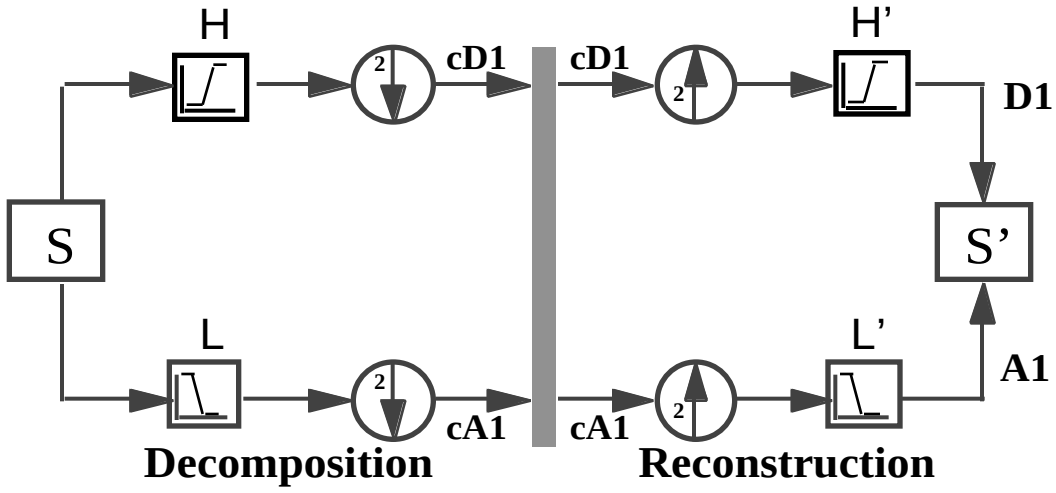


Figure 1.8–1 Single-level conventional DWT. Similar to the single-level UDWT with the same filters (**H**, **H'**, **L**, and **L'**) but with upsampling and downsampling. With no activity in the vertical bar, the coefficients **cD1** and **cA1** will be unchanged between the end of decomposition and the start of reconstruction. Notice that with downsampling the coefficients **cD1** and **cA1** are about half the size as those in the Undecimated DWT (UDWT).

Jargon Alert: "Aliasing" **means 2 or more signals have the same sample values. One pathological example of aliasing caused by downsampling by 2 would be a high frequency oscillating time signal:**

```
[1 -1  1 -1  1 -1  1 -1  1 -1  1 ...
```

If we downsample by 2 we have left over

```
[1      1      1      1      1      1 ...
```

which is a DC (zero frequency) signal. This is obviously *not* the high frequency signal we started with but an "alias" instead.

With the potential for aliasing problems because of downsampling we would not expect to be able to perfectly reconstruct the signal as we did in the UDWT. One of the remarkable qualities of DWTs is that with the right wavelet filters (**H**, **H'**, **L** and **L'**) we *can* perfectly reconstruct, even with aliasing! The stringent requirements on the wavelet filters to be able to cancel out aliasing is part of why they often look so strange (as we saw in Figure 1.2–3).

***Jargon Alert*: Filters in these filter banks that are able to cancel out the effects of aliasing (if used correctly) are called "*Perfect Reconstruction Quadrature Mirror Filters*" or PRQMFs".**

As with the UDWT, we can denoise our signal by discarding portions of the frequency spectrum—as long as we are careful not to discard vital parts of the alias cancellation capability. Correct and careful downsampling also aids with compression of the signal. With downsampling, **cD1** and **cA1** are only about half the size as in the UDWT. So compared to the conventional DWT, the UDWT is "redundant". This is why it's often called a *redundant* DWT.

Multi-level conventional DWTs produce the same frequency sub-bands as the multi-level UDWTs we saw earlier (if the aliasing is correctly dealt with). Figure 1.8–2 shows a 2-level conventional DWT. The frequency allocation is the same as the 2-level UDWT (see Fig. 1.7–2). Notice that we use the same 4 wavelet filters (**H**, **H'**, **L**, and **L'**) repeatedly in a conventional DWT.

It's usually the high-frequencies that comprise the noise in a signal, thus we decompose the lower frequencies in these multi-level transforms. Figure 1.8–2 shows **cA1** split into **cA2** and **cD2** but **cD1** is not split further. We can, of course split these Details further if we want to. This is done using a *wavelet packet transform* and we will look at these in later chapters.

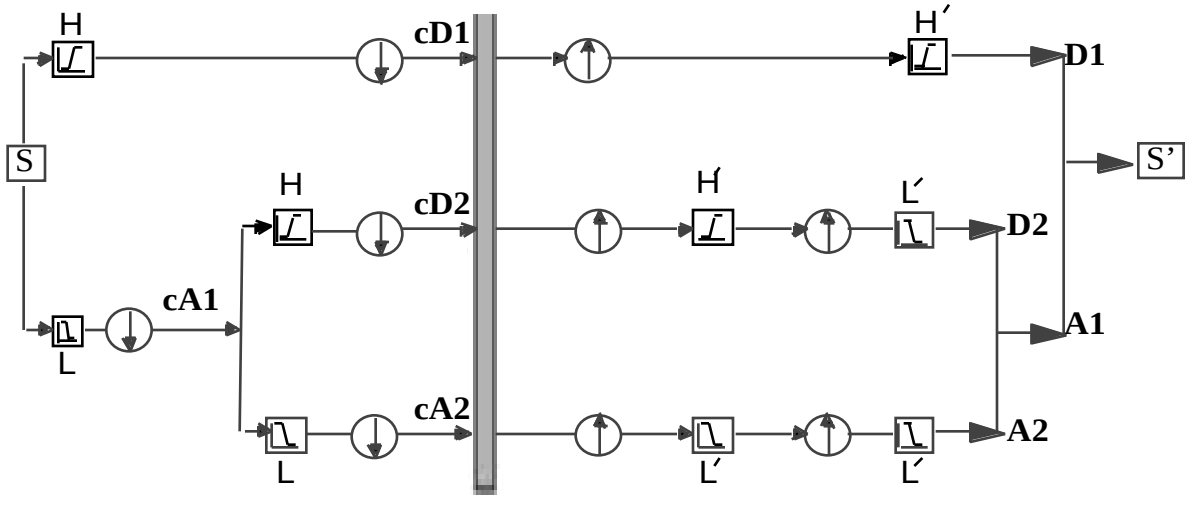


Figure 1.8–2 2-level conventional DWT. Instead of *stretching the filters* as in the UDWT (and CWT), we "shrink" the signal through downsampling and use the same 4 filters (**H**, **H'**, **L**, and **L'**) throughout. Note that with downsampling **cD2** and **cA2** are about 1/4 the size as those in the UDWT.

1.9 Examples of use of the conventional DWT

As mentioned, an important advantage of a wavelet transform is that, unlike an FFT, we can *threshold* the wavelet coefficients for only part of the time.

***Jargon Alert:* To *"threshold"* (used as a verb here) means to disallow all numbers that are either greater or less (depending on the application) than a specified value or threshold (used now as a noun).**

We will use a seven-level DWT for this next example. Instead of simply **A1** and **D1** as we saw in Figure 1.8–1, we would have further decomposition of **A1** into **A2** and **D2**, then **A2** into **A3** and **D3**, and so on until **A6** is decomposed into **A7** and **D7**. The frequency allocation for a conventional DWT (assuming no aliasing problems) is the same as that for the UDWT. For example, see figure 1.7–5 for the allocation by a 4-level DWT (or 4-level UDWT).

Suppose we had a binary signal that had a great deal of noise added which changed frequency as time progressed (e. g. "chirp" noise). Using a 7-level DWT the noise would appear at different times in the different frequency sub-bands (**D1**, **D2**, **D3**, **D4**, **D5**, **D6**, **D7** and **A7**). We could automatically threshold out the noise at the appropriate times in the frequency sub-bands and keep the "good" signal data.

A portion of the original noiseless binary signal is shown in Figure 1.9–1. The values alternate between plus and minus one (a Polar Non-Return to Zero or PNRZ signal). We next bury the signal in chirp noise that is 10,000 times as great (80 dB). Looking at the signal buried in noise (Figure 1.9–2) we see only the huge noise in the time domain (a). Using an FFT on the noisy signal we see only the frequencies of the noise (b).

However, using a conventional DWT with a time-dependant automatic threshold for the various frequency sub-bands, we are able to reconstruct the binary signal (see Fig. 1.9–3) from the "scraps" left over after the chirp segments were thresholded out at the appropriate times. (More details on how this was done will be given later).

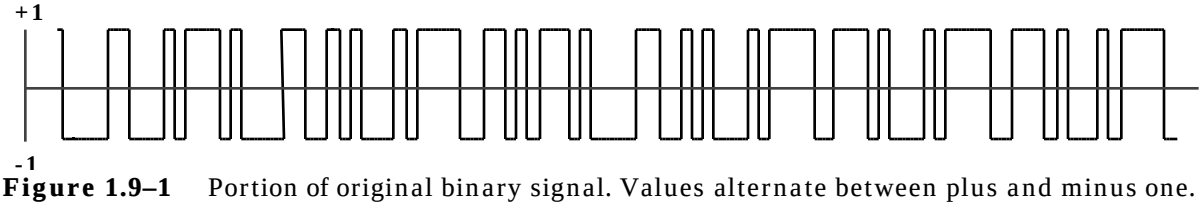


Figure 1.9–1 Portion of original binary signal. Values alternate between plus and minus one.

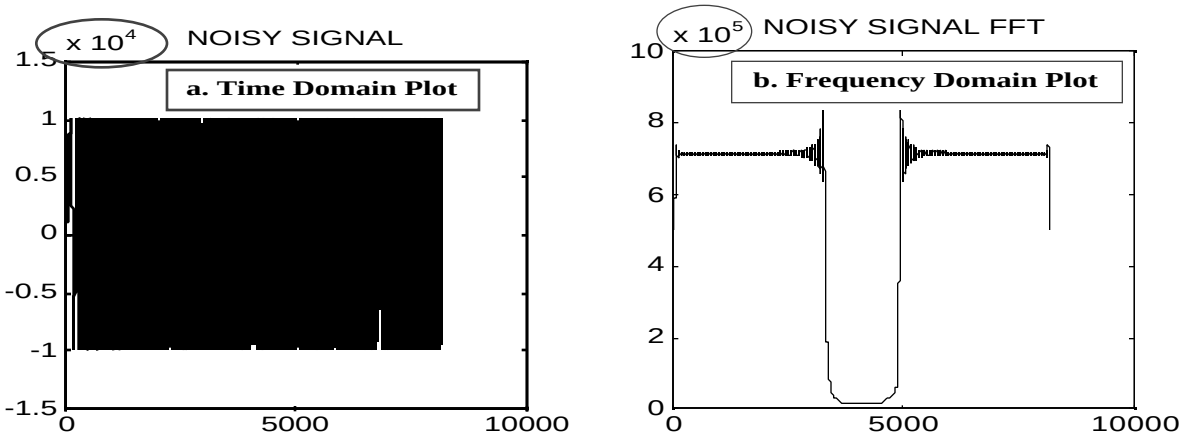


Figure 1.9–2 Signal buried in 10,000 times chirp noise is undetectable in either the amplitude vs. time plot (a) or the magnitude vs. frequency (FFT) plot (b).

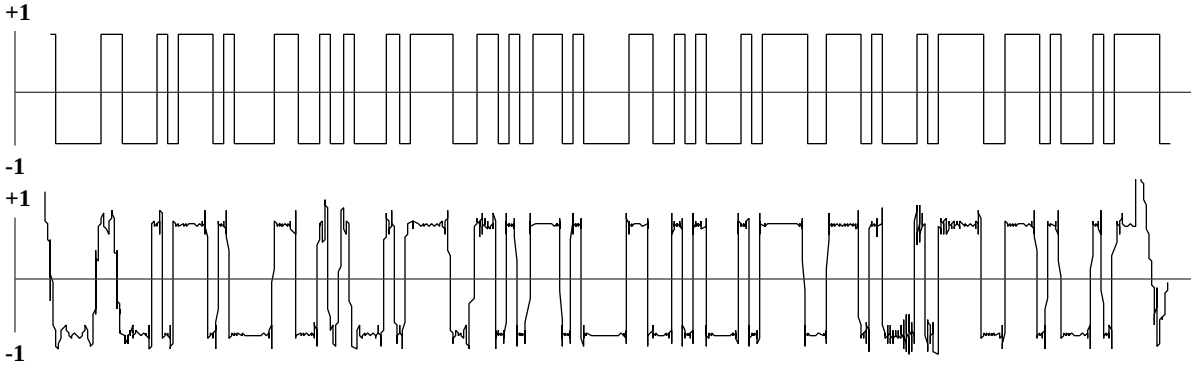


Figure 1.9–3 Successful use of discrete wavelet transform. Portion of denoised signal using time-specific thresholding with a 7-level conventional DWT is shown at bottom. Original binary signal. is redrawn at top for comparison. The final result is not a perfect reconstruction of the original, but close enough to discern the binary values.

Modern JPEG compression also uses wavelets. Figure 1.9–4 shows JPEG image compression. The image on the right was compressed by a ratio of 91:1 using a conventional DWT with a Biorthogonal 9/7* set of wavelets.

* As will be explained further in later chapters, the Biorthogonal 7/9 filters have 7 points in **H** and **L'** and 9 points in **H'** and **L**. This particular set of wavelet filters is referred to in MATLAB as "Bior4.4"

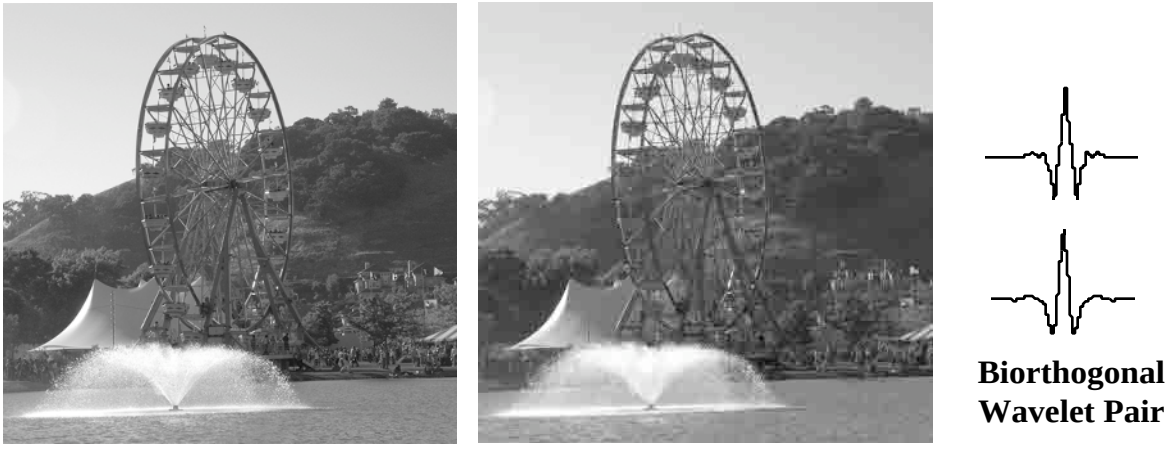


Figure 1.9–4 JPEG image compression of 91:1 achieved with a conventional DWT using a Biorthogonal 9/7 set of symmetrical wavelets.

1.10 Summary

In this preview chapter we introduced wavelets by drawing them as continuous functions, but told how they are actually implemented in a digital computer as discrete, short *wavelet filters*. We showed how some filters come from a mathematical expression for a continuous wavelet (crude wavelets) while other wavelets start out as filters with just a few points and then are built into a suitable estimation of a continuous wavelet. We then looked at various types of wavelets and their uses.

We next looked at transforms we use everyday and the (hopefully) familiar FFT and showed how they can be thought of as *comparisons* (correlations). We saw that the FFT has the shortcoming of not being able to determine the *time* of an embedded event. We discussed *short-time Fourier transforms* and then introduced the concept of wavelet transforms by comparing them to ordinary sheet music. We compared the fast Fourier transform (FFT) to the *continuous wavelet transform* (CWT) using an embedded pulse signal as an example. We next showcased the ability of a CWT to identify the time of occurrence of an embedded glitch, it's frequency, and it's general shape.

We moved on to the *undecimated discrete wavelet transform* (UDWT) and showed how it is similar in many ways to the CWT but uses all 4 wavelet filters rather than just one. We also noted that the stretching is done only by

factors of 2 (dyadically) in the UDWT rather than by every possible integer value as in the so-called "continuous" wavelet transform.

We continued building our understanding from FFT to CWT to UDWT by next moving on to the *conventional* DWT. The DWT is similar to the UDWT but introduces downsampling and thus potential aliasing problems. We mentioned special filters that (if used correctly) can cancel out the effects of aliasing! We showed two examples of uses of the DWT in signal denoising in a severe environment and in image compression (JPEG). In the next few short chapters we will do a step-by-step walk through of these various transforms.

We stress again that this preview is intended to give the reader a feel for how wavelets and wavelet transforms work. The next 12 chapters and the appendices will provide much more information and facilitate a real-world understanding and applications of these amazing tools.

Another option for understanding wavelets is to attend one of the open seminars by Mr. Fugal. The comments from attendees have been very favorable and are one reason why a book and website were developed. In fact, all the chapters in the book (including this one) are written using the completed seminar slides as the basis. Contact D. Lee Fugal at (toll-free) 877-845-6459 for information on the next open seminar. Private seminars are also available for your company or organization.

Be sure to visit our website at ***www.conceptualwavelets.com*** *for more information, downloads, updates, color slides, additional case studies, corrections, and FAQs.*

You can also selectively solidify your understanding by using the consulting services of Mr. Fugal to clarify or expand on specific sections. You are also welcome to contact him for comments and suggestions, a short specific question, or for general advice. He can be reached during business hours at the above number or at l.fugal@ieee.org.

There is much more to discover than can be presented in this short preview. The time spent, however, in learning, understanding and correctly using wavelets for these "non-stationary" signals with anomalies at specific times or changing frequencies (the fascinating, real-world kind!) will be repaid handsomely.
