CHAPTER

11

Case Studies of Wavelet Applications

Having seen the properties and some general applications of the various types of wavelets, we are now ready to gain a conceptual understanding of some applications to case studies. While we will not be demonstrating all the wavelets or types discussed, we should be able to gain some intuitive insights into wavelet use.

In addition to some new examples, we will re-visit some that have been introduced earlier in the book and can now be better understood with the further knowledge and insights we have since acquired.

11.1 White Noise in a Chirp Signal

We begin by adding white noise to a chirp signal. The result is shown in Figure 11.1–1 below at left. The FFT at right shows that the white noise appears at all frequencies (hence the name "white" as in *all colors*) and will be difficult to remove using conventional FFT methods.

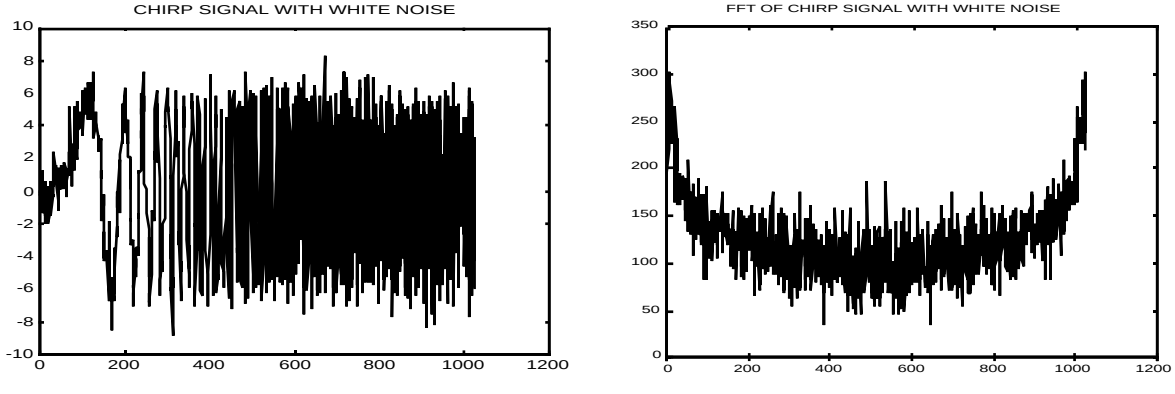

Figure 11.1–1 A chirp signal with white noise is shown in the time and frequency domains.

Using a conventional (non-wavelet) lowpass filter to keep only the low frequencies produces the time and frequency results shown in Figure 11.1–2. The low-frequency portion is fairly well denoised, but the high-frequency portion (most of this chirp signal) has been severely attenuated.

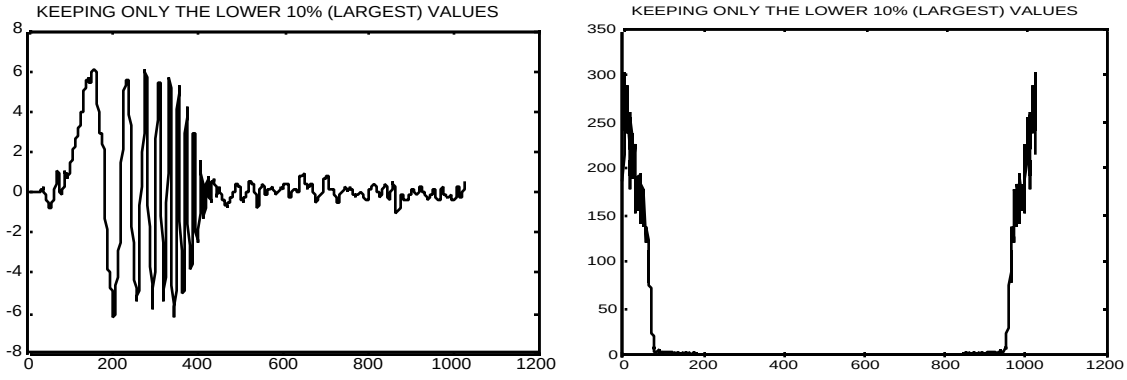

Figure 11.1–2 Conventional lowpass filtering accomplishes some denoising at the start of the signal (left graph) but destroys most of the signal as well as the noise. This is also seen in the FFT shown at right.

If we adjust the lowpass filter to be less severe we have the denoised signal as shown in Figure 11.1–3. More of the signal is preserved but the highest frequencies are still lost and the beginning of the signal is not de-noised very well.

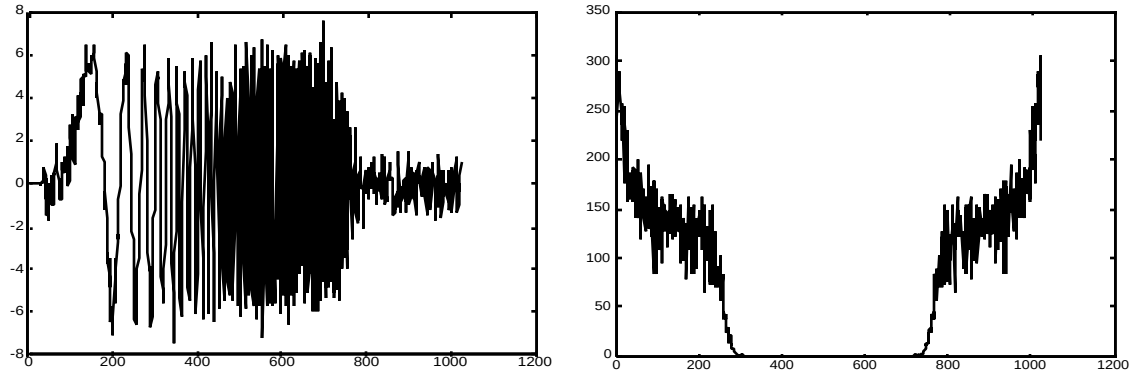

Figure 11.1–3 . Less severe conventional lowpass filtering keeps more of the signal (but not all) and the high frequency noise is still present, especially at the beginning.

We use a conventional DWT to attempt to denoise this signal. The decomposition is shown if Figure 11.1–4. We try first a Db6 filter because it looks like it might "match" portions of this asymmetric signal (ref. Fig. 6.4–2) and because it is short (6 points).

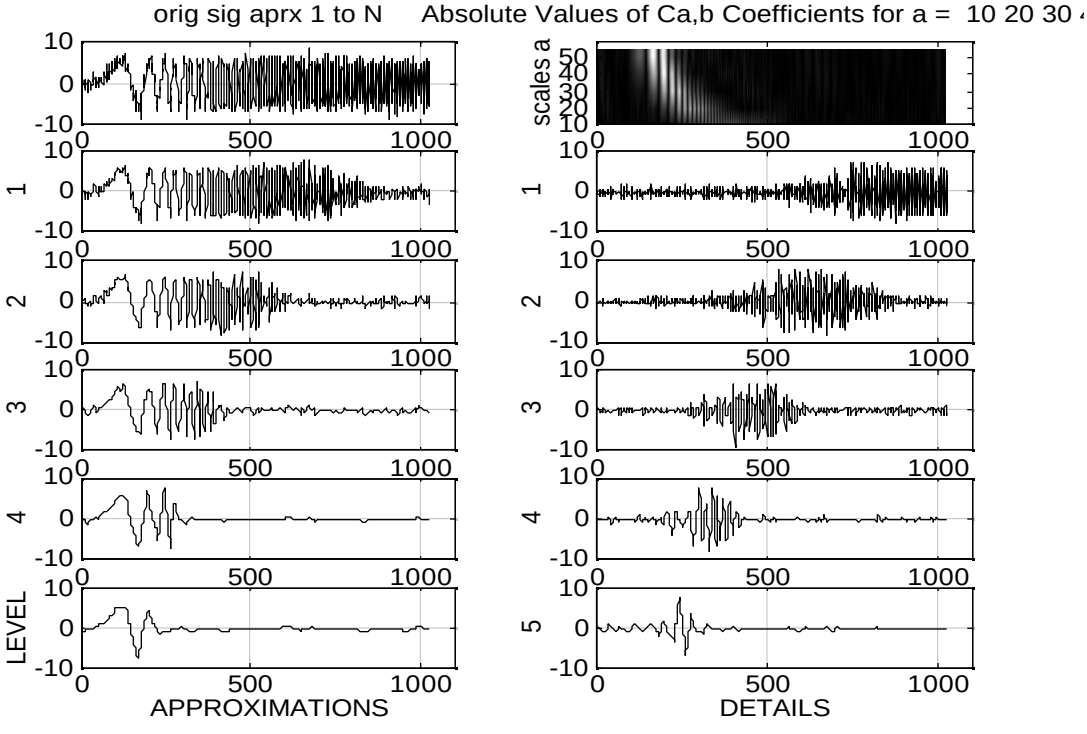

Figure 11.1–4 Display of a conventional DWT of signal using a Db6 wavelet.

The signal is 1024 points (2^10) long so we could downsample 10 times, however a 5-level DWT, as shown here, is sufficient. The original noisy chirp signal is shown in the upper left graph while a mini-CWT of the signal is shown at upper right (every 10th scale is sufficient in this case). The 5 levels of *Approximations* are at left while the 5 levels of *Details* are shown at right.

A frequency allocation diagram for a 5-level DWT (or UDWT) is shown in Figure 11.1–5. We now demonstrate the time/frequency manipulation capabilities of *wavelet technology*. We will keep only *some* of the data within a *certain time* and within a *certain frequency range.*

We notice in the above DWT display that different parts of the noisy chirp signal appear in levels **D1** through **D5**. For example, in **D1** (graphic directly beneath the mini-CWT) it appears that the "signal" portion begins at about **time = 650** and that everything before that can be attributed to noise. We thus set the first 650 **D1** values to zero to attempt to de-noise the signal.

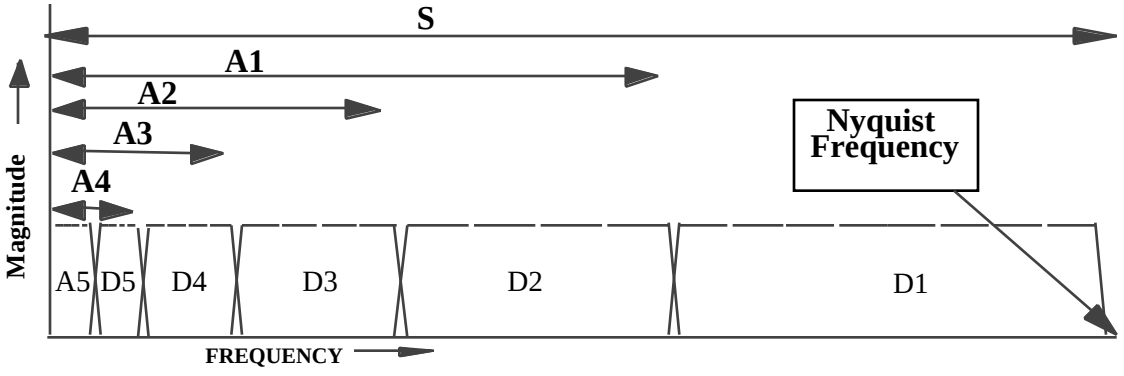

Figure 11.1–5 Frequency allocation for a 5-level DWT. Notice that **S = D1+A1 = D1+D2+A2 = • • • = D1+D2+D3+D4+D5+A5**. Note also that the filters are imperfect.

For **D2** it appears that the signal portion is located between about 400 to 800. Thus we place zeros everywhere else.* This process is shown below in Figure 11.1–6 for **D1** and **D2**.

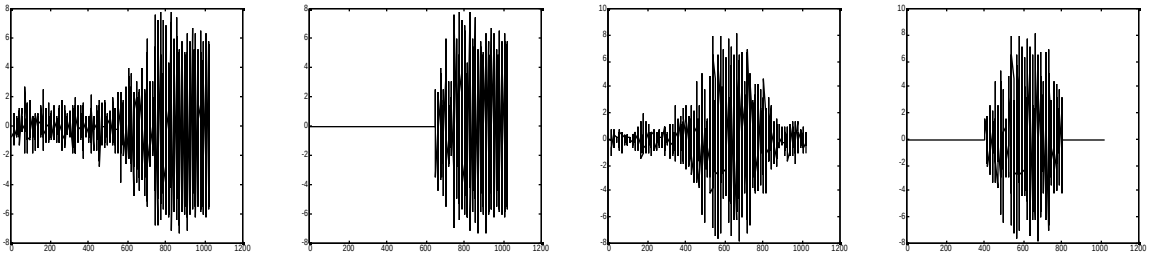

Figure 11.1–6 **D1** (1st graph) and **D2** (third graph) are adjusted to be zero except for the specified times (650 to 1024 for **D1** and 400 to 800 for **D2**).

We continue this process through **D5**, keeping only the signal portions. We are now ready to add the denoised details **D1** through **D5** together with **A5** (ref. Fig. 11.1–5) to reconstruct our signal. We are not seeking perfect reconstruction here but instead we wish to denoise the signal. Figure 11.1–7 shows the final result with this method of denoising.

* Although this process is being done somewhat "by hand" here, the MATLAB Wavelet Toolbox has interactive graphics called "*interval-dependent thresholding"* that allows the user to quickly discard unwanted data in specified periods of time.

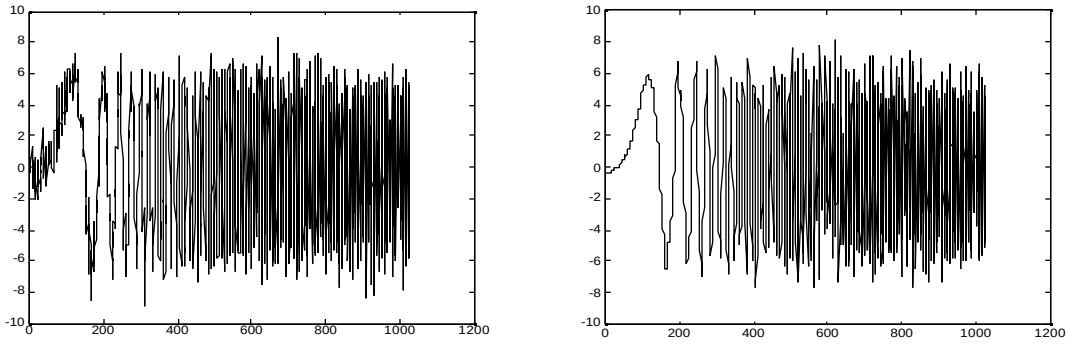

Figure 11.1–7 The original noisy signal is shown at left. Denoising using a 5-level DWT with Db6 wavelet filters gives the result at right. Compare with Figs. 11.1–2 and 11.1–3.

The results are impressive, especially when compared with conventional DSP methods. They can be possibly improved by trying other wavelets or by performing the time/frequency manipulations on additional levels.

We will talk further in an upcoming chapter about "throwing out the baby with the bathwater"—throwing away some alias cancellation capability as we discard the noisy parts of **D1** through **D6**. For now, we can state that in each of the levels the noise was far less than the signal and the effect of aliasing was minimal. We will compare this later with the results obtained using the Undecimated DWT which has no such aliasing problems.

11.2 Binary Signal Buried in Chirp Noise

This next example is similar to the first except we have a binary signal and the *noise* is in the form of a chirp.* The process is similar to that of the last section. Figure 11.2–1 shows a Binary Phase Shift Keying, Polar Non-Return to Zero (BPSK PNRZ) signal in both the time and frequency domains. This example was mentioned in the overview Chapter One. We now provide more details.

* A constant frequency jammer can be easily removed from data by conventional *notch filtering* using FFT methods. A *chirp jammer* is more difficult because the frequency keeps changing. Wavelets work well here.

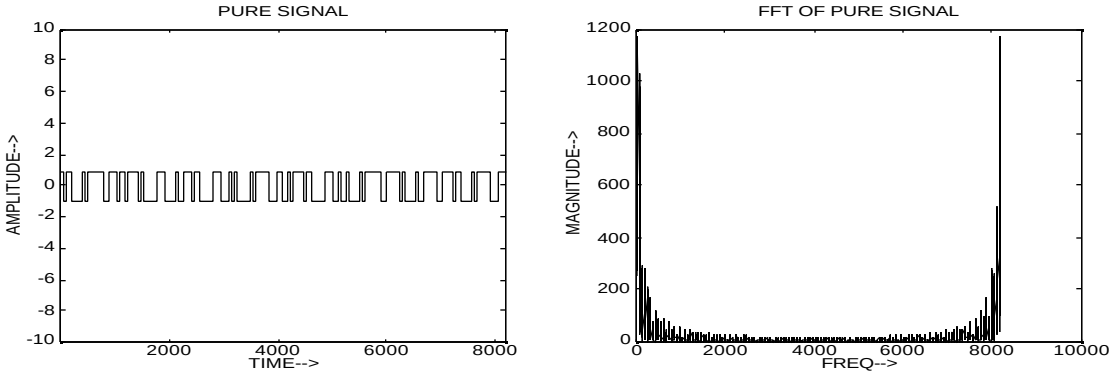

Figure 11.2–1 The original binary signal is shown in the time domain. Note the values alternate between –1 and +1. The FFT of this signal is shown at right.

The signal is next buried in 80 dB* of chirp noise as shown in Figure 11.2–2. The left plot shows the chirp noise values from –10,000 to +10,000. Although we can't see the small binary signal with this much noise added, we show a close-up (right graphic) of the noise from –100 to +100 with the original binary signal over-plotted for comparison (the signal overplotted on the full noise would look like a straight line). Looking at the signal in the frequency domain does not offer much hope of finding it either (ref. Fig. 1.9–2).

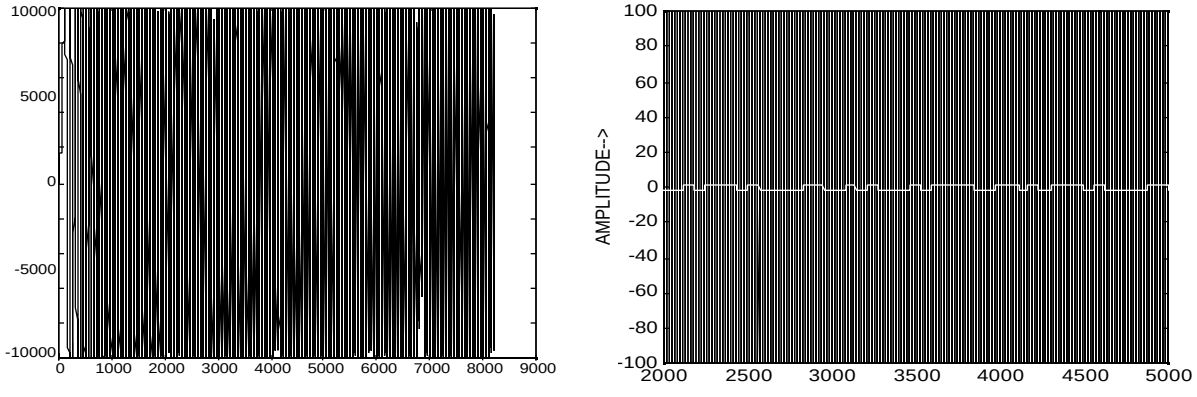

Figure 11.2–2 Binary signal from –1 to +1 with noise from –10,000 to +10,000 added is shown at left. A close-up is shown at right with the original binary signal overplotted (the signal would not be visible in 80 dB of noise). Note the dimensions.

* *Decibels*, not *Daubechies*—the decibel is named for Alexander Graham **B**ell and is abbreviated "dB" while a Daubechies wavelet is named for Ingrid **D**au**b**echies and is abbreviated "Db".

This is another instance where we use wavelets. We talked earlier about matching the wavelet to the signal. In this case we might consider a Haar wavelet because it looks like the binary signal. However, with this much noise we wouldn't be able to find it! Instead, we will match the wavelet to the noise. Because the signal is 8192 points long we will use a longer Db40 wavelet as shown in Figure 11.2–3. This looks a lot like the chirp signal and, as we mentioned for large Daubechies filters, is fairly smooth. Also, this longer wavelet should provide for fairly good frequency discrimination.

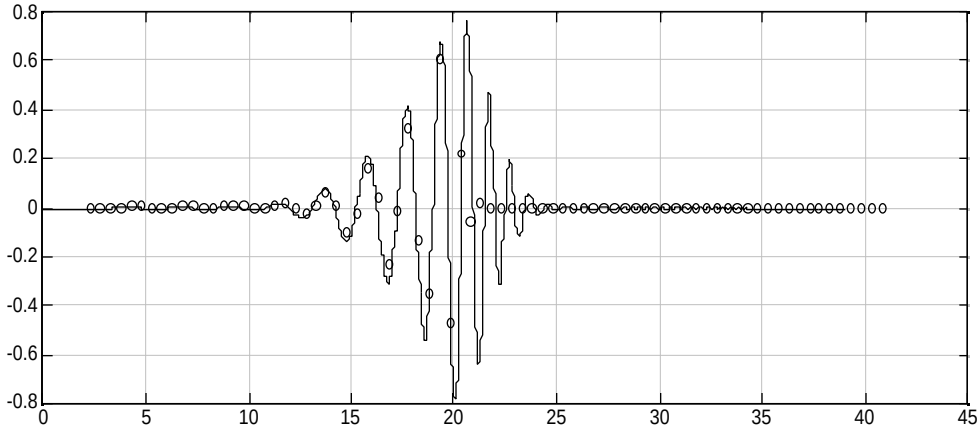

Figure 11.2–3 4954 point estimation of the "continuous" Db40 wavelet with the 40 points of the **H'** filter that created it superimposed (along with trailing zeros).

We will use a 7-level DWT here. The results are shown in Figure 11.2–4 below. We notice that the match of the wavelet to the noisy signal is excellent and that the highest frequency portion of the chirp noise is found in right-hand part of **D1** (ref. Fig. 11.1–5). Mid-frequency portions are found in the center of the **D2** through **D5** plots and the lowest frequencies are found at the left of the **D6** and **D7** plots.

We can also literally see how **A1** and **D2** combine to make the signal (left top graph), how **A2** and **D2** combine to make **A1** and so on.

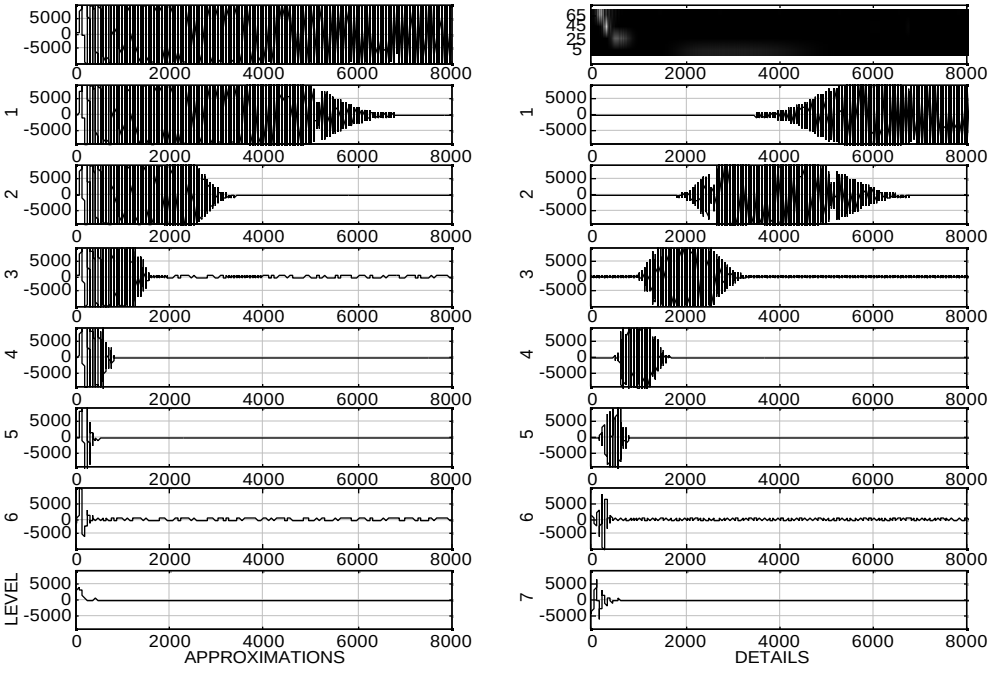

Figure 11.2–4 7-level DWT of binary signal with 80 dB of noise added. Wavelet filters used are Db40.

Looking at **D2** (Details, #2) on the above display we might assume we would be "safe" to delete the values from about 1500 to 7500 because the chirp portion seems to be isolated in this area. A close-up look at **D2** limiting the values to the range –2 to +2 (rather than –10000 to + 10000) tells a different story as shown in Figure 11.2–5 (left). We see a binary pattern in the first 1000 points and perhaps a few more points at the end. Rather than plotting close-ups and setting the chirp jammer portions to zero by hand, we can automate this process. We use a "reverse threshold' in which any values of the signal greater than, say, 2 (or less than –2) are set to zero.*

* We can set up this "reverse" threshold in software or we can use *median filtering* to keep the binary parts.

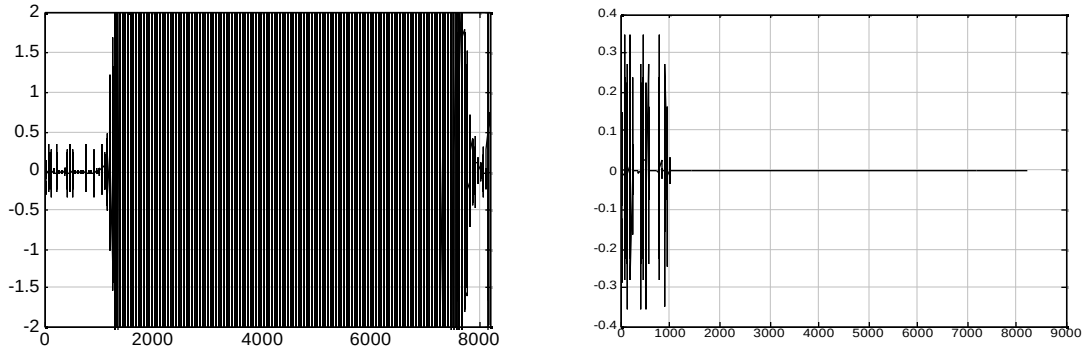

Figure 11.2–5 Close-up of **D2** showing remnants of the signal at left. After reverse thresholding using a median filter we keep only the values *less than* 2 and set to zero all the large values as shown in the right graph.

The right graph of Figure 11.2–5 above shows the result. We have only the "scraps" left over (in this frequency bandwidth of **D2**). However, looking again at the 7 Details on the above DWT display we can see that these "scraps" will be located at different times. **D1** with the chirp noise removed gives us remnants at the beginning while **D3** and **D4** give us remnants in the middle and **D6** and **D7** give us remnants at the end.

With the chirp portions thresholded to zero we now combine these de-noised Details as

Sig′ = A7 + D7′ + D6′+ D5′ + D4′ + D3′ + D2′ + D1′

where the prime indicates de-noised. A close-up of the denoised signal is shown in Figure 11.2–6 along with the original binary signal for comparison. The denoising is not perfect, but allows us to reconstruct a recognizable binary signal. This would not have been possible using conventional DSP methods.

This is a good example of how wavelets are useful to match either the signal or the noise and how the time/frequency nature of wavelet processing allows us flexibility we would not find in either the time or the frequency domains by themselves.

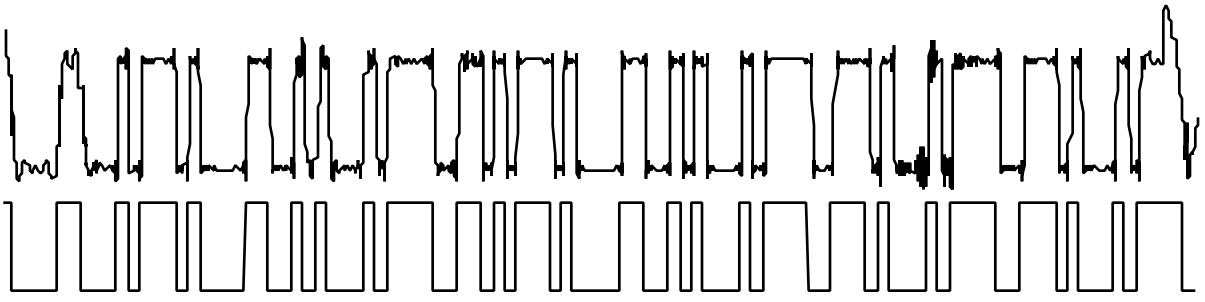

Figure 11.2–6 Portion of signal pulled from 80 dB of noise using time-specific thresholding with a 7-level conventional DWT is shown at top. Original binary signal. is redrawn at bottom for comparison.

11.3 Binary Signal with White Noise

This time a binary signal has intermittent white noise added as shown in Figure 11.3–1. The signal has 16 "chips" (binary value +1 or –1) per bit. For example, the binary sequence **[1 –1]** is represented by 16 values of +1 followed by sixteen values of –1. The "pure" signal here is 1024 chips (points) long and represents 64 bits. The first 8 bits are **[1 –1 1 –1 1 1 –1 –1]**.

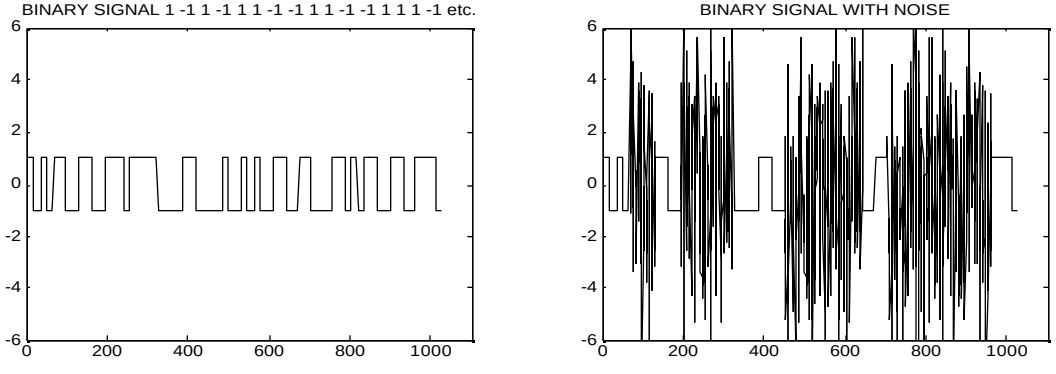

Figure 11.3–1 Binary signal with 16 "chips" per bit is shown at left. Intermittent pseudo-random noise is added as shown at right.

We can actually see part of the signal in the time domain, but not enough to decode it. Figure 11.3–2 shows the pure binary signal and the noisy signal in the frequency domain.

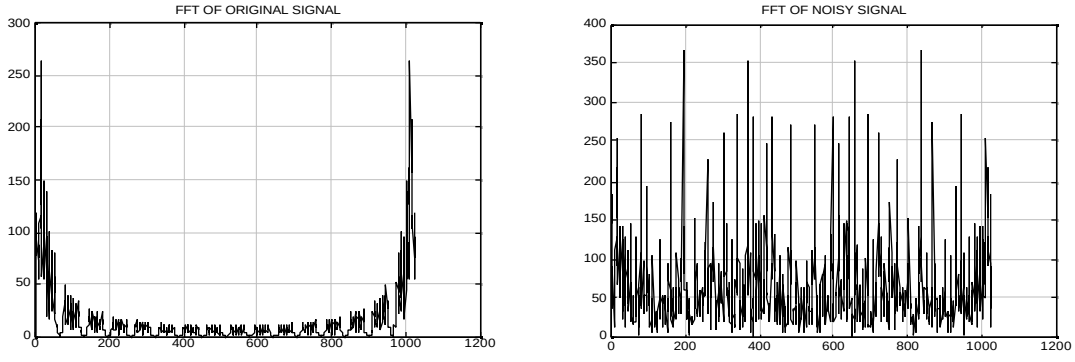

Figure 11.3–2 FFT of the original binary signal at left and the FFT of the signal with intermittent noise at right.

We will encounter problems using conventional DSP lowpass filtering techniques because of the amount of high frequencies that come from the "untouched" portions of the pure binary signal (ref. Fig. 11.3–2, left). In other words, both the signal and the noise have high frequency components.

A square wave, as most audio enthusiasts know, is made up of many frequencies.* Also, we have tried to make the binary signal realistic by including "wide" areas (e.g. sections where the bit pattern is "**–1 –1 –1 –1**" etc. instead of just "**1 –1 1 1 –1 1 –1**" etc.). Thus having sections in the signal with the (variable-length) square bits may actually make it harder to separate from the sporadic noise using conventional filtering techniques.

One conventional method might be to use a series of Short Time Fourier Transforms (STFTs) on the "clean" and noisy sections separately. However, wavelet technology incorporates this same time/frequency capability with a better *match* to the signal than the FFT sinusoids.

We have seen in the previous 2 examples methods of using the simultaneous time/frequency capabilities of wavelets by selectively denoising the Details at

* Early Rock musicians would intentionally crank up their tube-type amplifiers to distortion. Instead of "clean" sinusoids the tops and bottoms of the sine waves would be *clipped* flat and would look more like square waves and sound to the human ear like a combination of many high-frequency harmonics and overtones. The next generation of amplifiers provided a smoother attenuation and less harmonics—which caused the old tube amplifiers to be highly sought after by the Rock musicians! Amplifier manufacturers finally caught on and developed solid-state units that provide flat clipping and the harmonic distortions so loved by some young musicians (and tolerated at best by most senior DSP professors).

specified times and we could do that again here. However, we will show instead the power of a good match of wavelet to signal.

This time we will begin by choosing a wavelet that matches the binary signal—the Haar would be a good choice. Using a conventional DWT we could decompose up to 10 levels (**1024 = 2^10**) but 5 levels will adequately demonstrate the process. First we will perform the DWT on the pure noiseless binary signal using the Haar wavelet. The display is shown in Figure 11.3–3.

We notice that the Details in levels 1 through 4 are zero. This is true for *any time interval* of the binary signal. For example the "skinny" square waves at the beginning of the pure binary signal (ref. Fig. 11.3–1 at left) as well as the "fat" square waves closer to the middle (times roughly 200 to 500) all produce zero values for the Details in levels 1 through 4. This means that for *any* binary signal similar to this test case the information is captured in the higher levels (**D5**, **D6**, etc.) that we have learned represent the lower frequencies

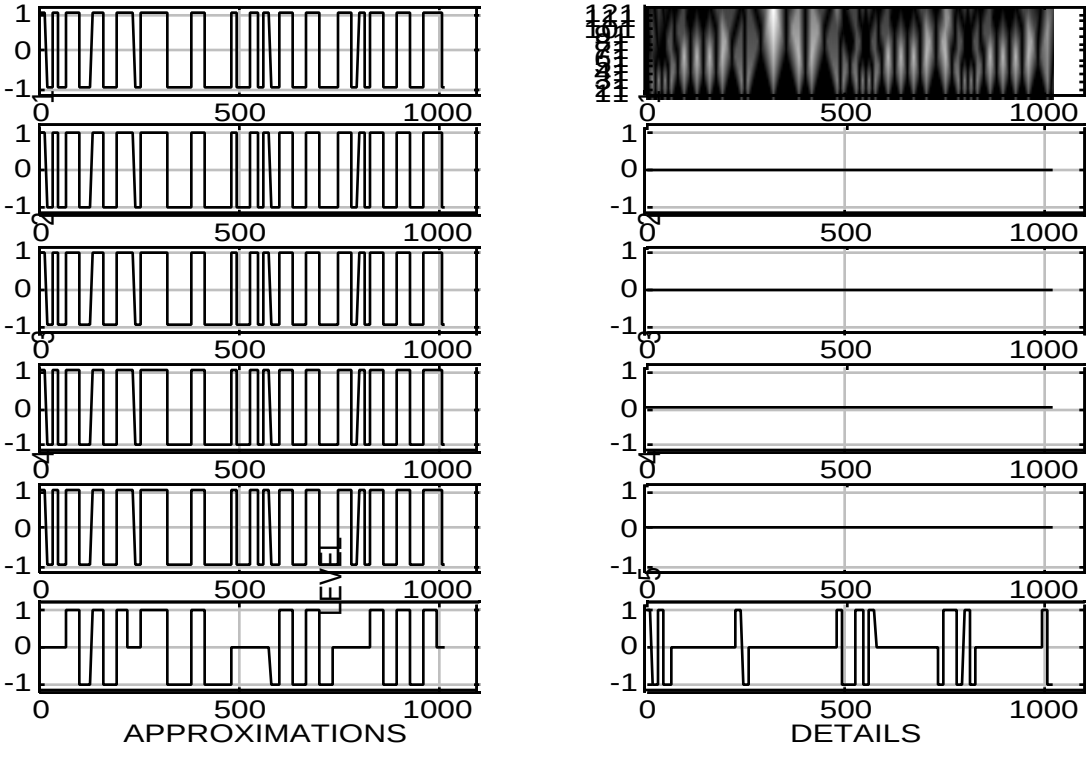

Figure 11.3–3 DWT of the original binary signal (top left) with CWT (top right).

Specifically, we know that the pure binary signal, S, can be represented by

S = D1+A1 = D2+D1+A2 = . . . = D4+D3+D2+D1+A4

But with the Details being zero on the first 4 levels we have

S = 0 + 0 + 0 + 0 + A4 = A4

We can see this in the above DWT display as we compare the signal (top left) to the Approximations in levels 1 through 4.

The DWT display of the binary signal with intermittent noise is shown in Figure 11.3–4.

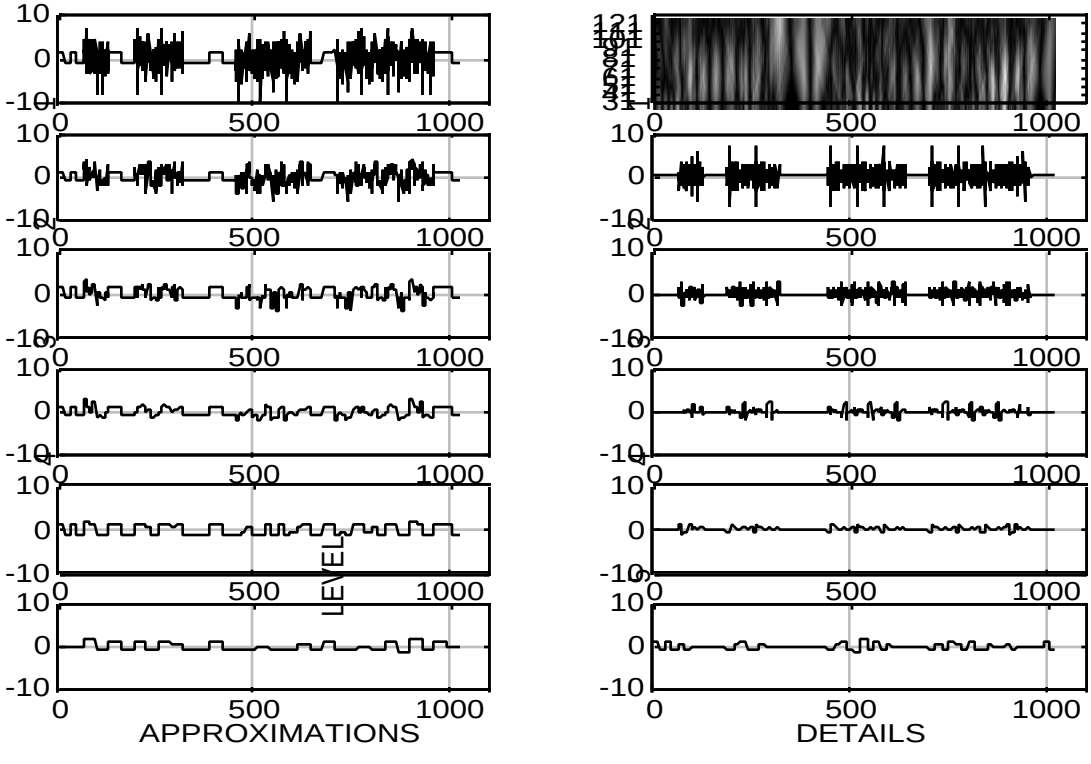

Figure 11.3–4 DWT of the original binary with intermittent noise added.

We notice that there is information in all 5 levels of Approximations and Details. However, we now know that for a binary signal similar to our test case that the information in levels 1 through 4 of the Details is all noise. This

means that the level 4 Approximations (**A4**) contains the signal with much of the noise removed.

Figure 11.3–5 shows at left the partially de-noised signal found in **A4**. The right graph shows the original signal for comparison. The original signal is also superimposed on the denoised signal (dotted lines).

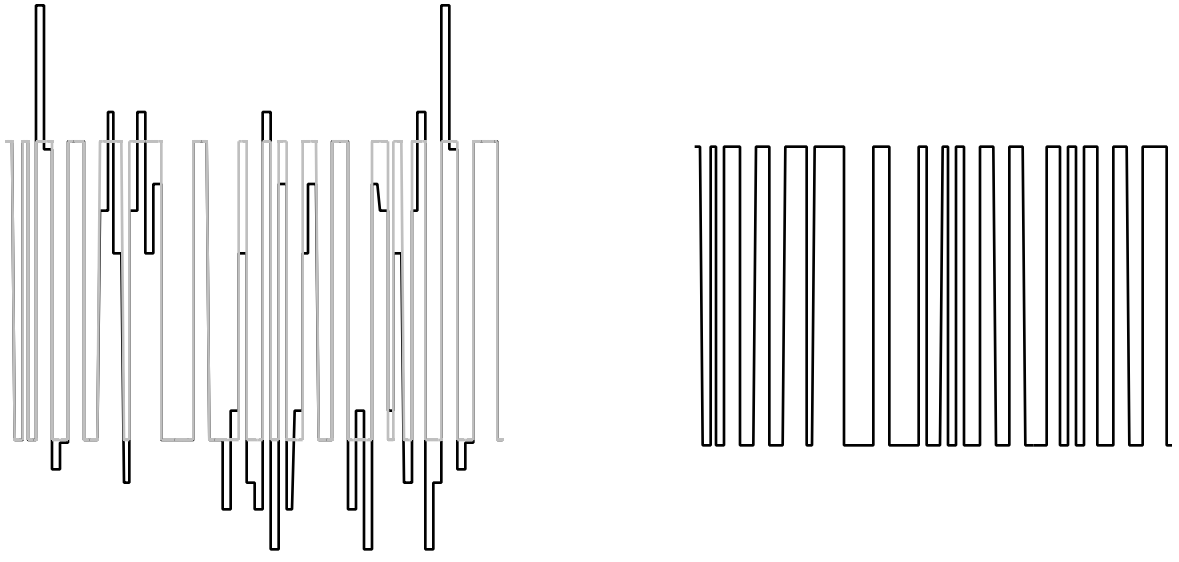

Figure 11.3–5 Denoised binary signal at left. For comparison, the original pure binary signal is superimposed on the left graph and presented by itself at right.

Although not perfect, we can tell which bits are positive and which are negative and thus +1 or –1. The bit pattern is thus preserved after denoising.

The CWTs we saw in the upper right corner of the DWT displays deserve a closer look. Figure 11.3–6 shows the CWTs for the pure binary signal and the signal with the intermittent noise added.

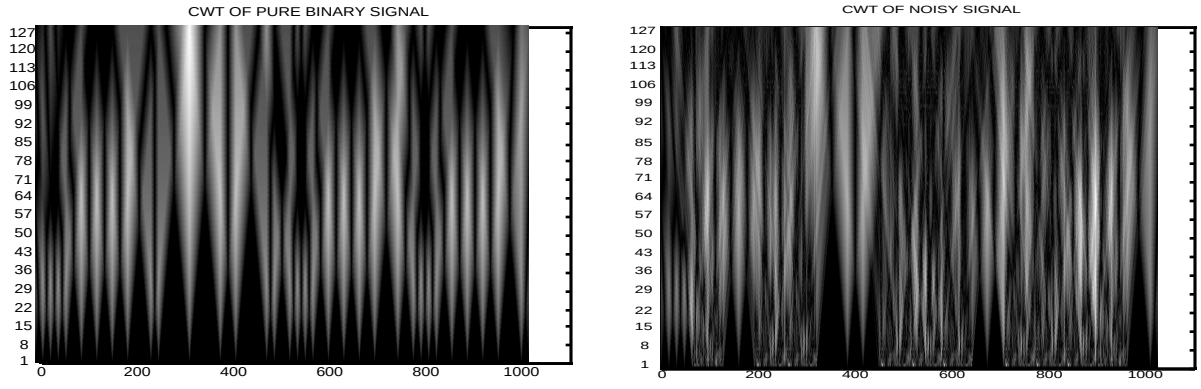

Figure 11.3–6 CWT displays using Haar wavelet of pure binary signal and of noisy signal.

The displays look somewhat like a row of pipes on a pipe organ. The smaller "pipes" indicate the "skinny" square waves as found at the left side of each of the CWT displays (times from about 1 to 40). The larger "pipes" indicate the "fatter" waveforms just to the left of the middle of each display (about **time = 400**). These displays show the magnitude of the values so **–1** will appear as bright as **+1**. We can, however, adjust the display so that negatives are dark and positives are bright so we can discern the bit patterns. Notice that we can "see past the noise" enough to discern bits in the noisy signal (right graph).

Figure 11.3–7 (left) shows the CWT of the Haar-denoised signal. Notice how well it agrees with the original binary signal (ref. Fig 11.3–6). We chose the Haar wavelet because it looked like the binary signal. It is interesting to see what would happen if we chose a wavelet that looked nothing at all like the signal (or the noise). The right side of Fig. 11.3–7 shows the results of using a Db20 "chirp" wavelet. The DWT (not shown) for the Db20 is also of no use.*

This is why conventional DSP falls short—especially with signals that do not look sinusoidal and do not match the sinusoids of the Fourier Transform.

* In practice, if we are unsure about the shape of the signal we would start with a more general-purpose wavelet such as the Db4. In this case, however, we can see portions of the binary signal and can tell up front that a Haar wavelet would be an excellent choice but that a 20-point chirp wavelet (as used in the previous example) would be a poor match to either the signal or the noise.

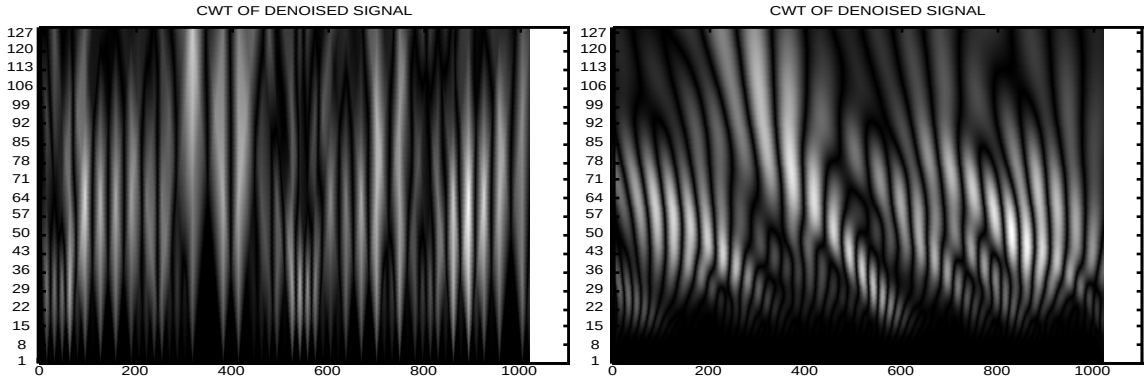

Figure 11.3–7 CWT of Haar-denoised signal using the Haar wavelet at left clearly shows the bits. At right, the CWT of the denoised signal using a Db20 wavelet is interesting-looking, but gives no practical visual information.

11.4 Image Compression/De-noising

Compression and/or de-noising using wavelets are in wide use in image processing. We saw an example of JPEG image compression earlier (ref. Fig. 1.9–4). While a full discussion of the use of wavelets in image processing is beyond the scope of this book, we can provide a brief overview.

Image processing means two-dimensional processing. Instead of the simultaneous *time/frequency* capabilities of wavelets we are usually talking about *space/frequency or distance/*frequency. In other words, a signal would have various amplitudes as we vary the time while a monochromatic image would have various brightness as we vary the position (space) on the image (e.g. "3 centimeters to the right and 4 centimeters down on the photo").

To get our bearings we will look first at a simple 256-point split sine signal and a single-level one-dimensional conventional DWT display as shown in Figure 11.4–1 (top left graph). We will use the set of 4 Haar wavelet filters (the **H'** filter is used by itself to produce the CWT at top right). Notice that the Details and Approximations at level 1 (**D1** and **A1**) combine to produce the original signal. Note also that both halves of **A1** have high amplitudes (like the original signal) while the left half of **D1** has much smaller values than the right half). This is of course because **D1** represents the higher frequencies while **A1** represents the lower frequencies as we have discussed (ref. Fig. 4.5–1).

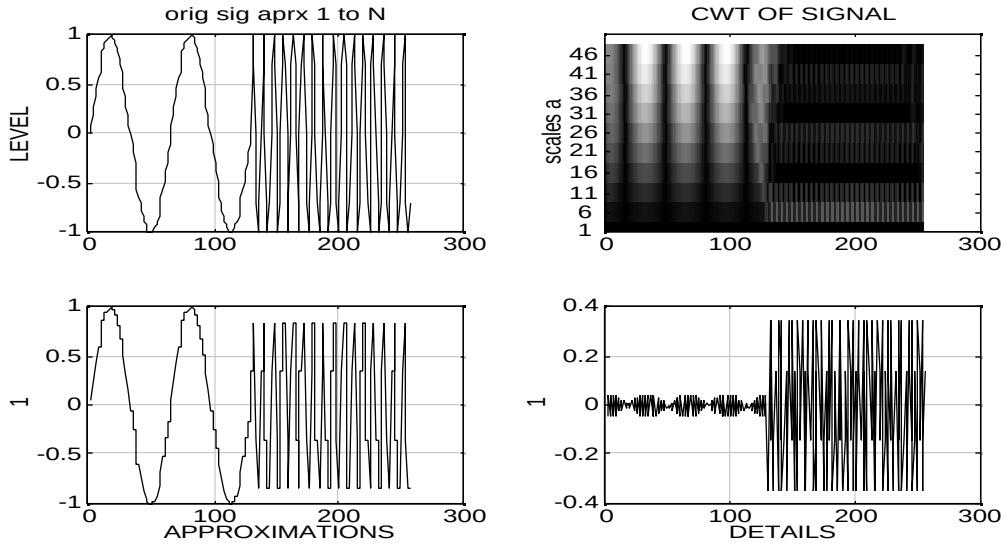

Figure 11.4–1 Single-level DWT of a split-sine signal using the Haar wavelet filters.

We next construct a 2-dimensional 256 by 256 image "test pattern" The first 128 *rows* will be identical copies of the above split sine signal. The image is shown below in Figure 11.4–2 at left. Comparing with the 1-D signal we can discern the tops of the 2 low-frequency sine waves (cycles) and the tops of the 16 high-frequency sine waves as bright spots. We use a shorter 128 point split sine signal with one low-frequency sine wave and 8 high-frequency sine waves (not shown) to fill the lower half of the test pattern (right graph) by constructing 256 identical *columns* each 128 points long. As with the longer split sine test signal we can see the bright spots corresponding to the top of the low-frequency sine wave (single cycle) and the 8 high-frequency sine waves.

We now proceed with a single level *two-dimensional* conventional DWT of this test-pattern image. Whereas the 1 level 1-D DWT would decompose the signal into **A1** and **D1**, the 2-D DWT converts the image into **A1** (the lower-frequency Approximation), **H1** (a vertical scan yielding **H**orizontal components), and **V1** (a horizontal scan yielding **V**ertical components).*

* Some software also uses a diagonal scan and/or additional methods to further decompose the data at each level.

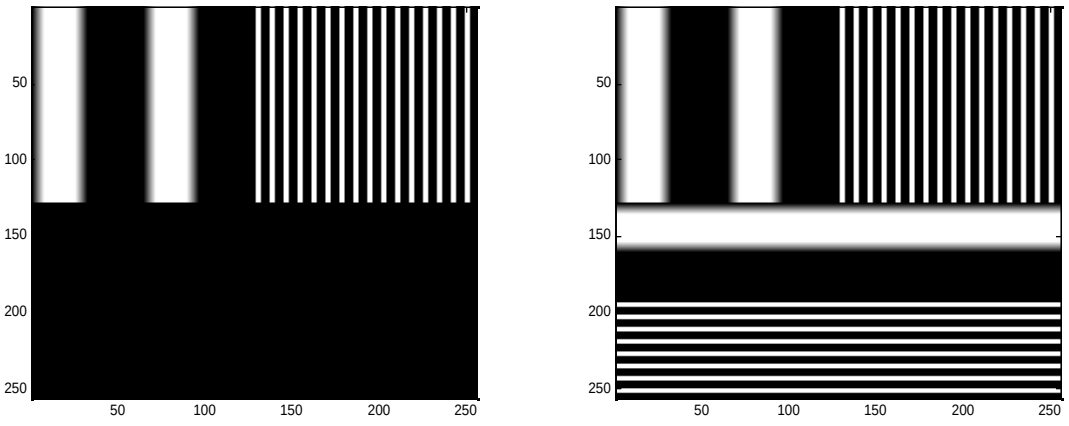

Figure 11.4–2 Test Pattern produced by 128 identical rows of the 256-point split-sine signal (left image) followed by the addition of 256 columns of a shorter 128 point split-sine signal (bottom half of complete test pattern image as shown at right).

Figure 11.4–3 shows the **A1**, **V1** and **H1** portions of the decomposed image. The Approximation, **A1**, looks much like the original image (as did the 1-D **A1** from Fig. 11.4–1 look much like the original signal). Just as the one-dimensional **D1** (right lower plot from Fig. 11.4–1) had very low values until the signal changed to high frequency, the upper half of **V1** (the center image of Fig. 11.4–3 below) is dark for the first half then we can see the higher frequency sine waves as vertical components from the horizontal scans.

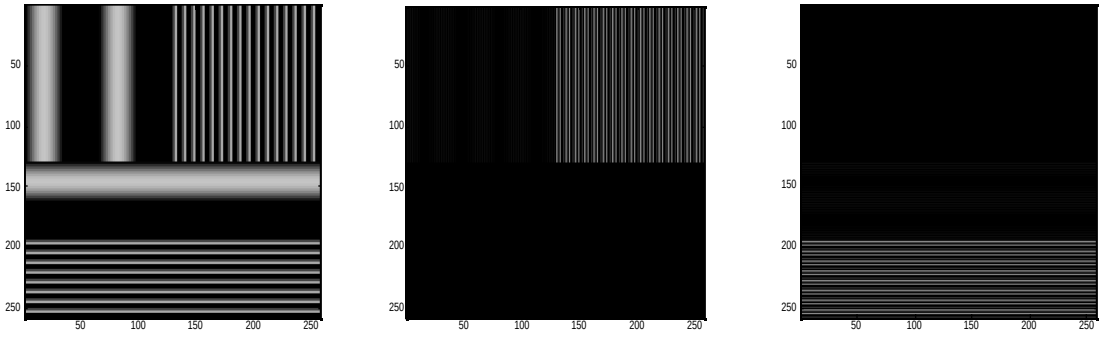

Figure 11.4–3 Single-level 2-D DWT of test pattern produces the lower frequency Approximation (**A1** at left), the **V**ertical components from the horizontal scan (**V1** in center) and the **H**orizontal components from the vertical scan (**H1** at right).

The lower half of **V1** is all dark because horizontal scans of the test pattern in that area will produce constant values (whether bright or dark, the values don't change horizontally) and thus zero frequency. The vertical scans in the rightmost graphic will also produce zero values (dark portions) until they encounter the high frequency portions of the shorter split-sine signal.

Similar to the 1-D cases, the image can be reconstructed by combining **A1**, **V1**, and **H1**. If we do this without changing the components we will have perfect reconstruction. However, our goal here is to remove some noise, compress the signal or both.

Figure 11.4–4 shows the familiar "*Barbara*" image (left). We have taken a 200 x 200 pixel close-up to show the facial quality. Then we have added some noise as shown at right. Notice particularly the "freckles" we have added around the forehead, cheeks, and nose areas.

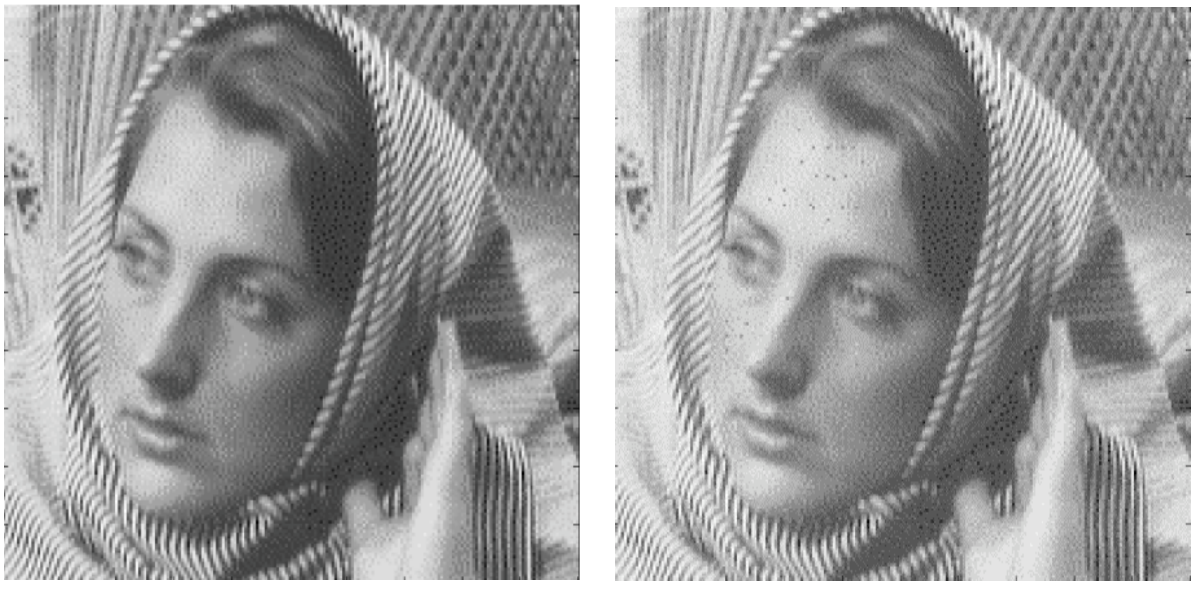

Figure 11.4–4 Classic "Barbara" image 200 x 200 close-up is shown at left. Noise is added along with facial skin imperfections ("freckles") to the "pure" image giving us the image at right.

We will now compress this image. Since compression often involves removing high frequency components we might expect a possible improvement in skin quality (i.e. freckles and other skin imperfections less pronounced).

Since this image is small (200 x 200) we will want to use a small wavelet (filter). The 2-point Haar comes to mind. The result of a 9 to 1 compression is shown in Figure 11.4–5 at left. Even after fine-tuning, the facial quality will still be very poor. However, notice how the fabric of her scarf is very pronounced. The Haar wavelet provides for good edge detection.

A better choice for her complexion would be a biorthogonal wavelet. We choose the 7/9 wavelet because it is perfectly symmetrical and still short (9 points maximum filter length). The results of a 9 to 1 compression using this biorthogonal wavelet is shown below at right. Notice her complexion has cleared up considerably and that she now has a "softer" look.*

Figure 11.4–5 9 to 1 compression using a set of 2-point Haar filters is shown at left. The same compression using a set of Biorthogonal 7/9 filters is shown at right.

We saw earlier how wavelets can be used to denoise a signal at specific time intervals. Similarly, with wavelet image processing we can denoise specific areas of an image. In the above example we could use heavier filtering on the freckles areas and lighter filtering on the rest of the image.

* In the early days of Hollywood, long before Digital Image Processing, some older screen actresses would insist upon a gauze "filter" stretched across the movie camera lens. This "soft" effect would hide wrinkles and age spots. One young actress also insisted on this "soft" effect—not to hide wrinkles but to hide her freckles!

11.5 Improved Performance using the UDWT

So far in this chapter we have successfully used the conventional DWT for de-noising and compression. As mentioned earlier, when we remove noise we also remove part of the alias cancellation capability of the conventional DWT. In the examples so far this has not been a problem but in this last example we highlight a pathological case where aliasing *is* problematic and show why the Undecimated DWT (UDWT) can provide better results.

Figure 11.5–1 (left graph) shows a generalized test signal that begins with Frequency Shift Keying (FSK) and ends with Frequency Modulation (FM). The FFT of this signal (right graph) shows the low and high frequency portions (peaks) from the FSK modulation and from the linear FM modulation (wide portions at low frequencies).

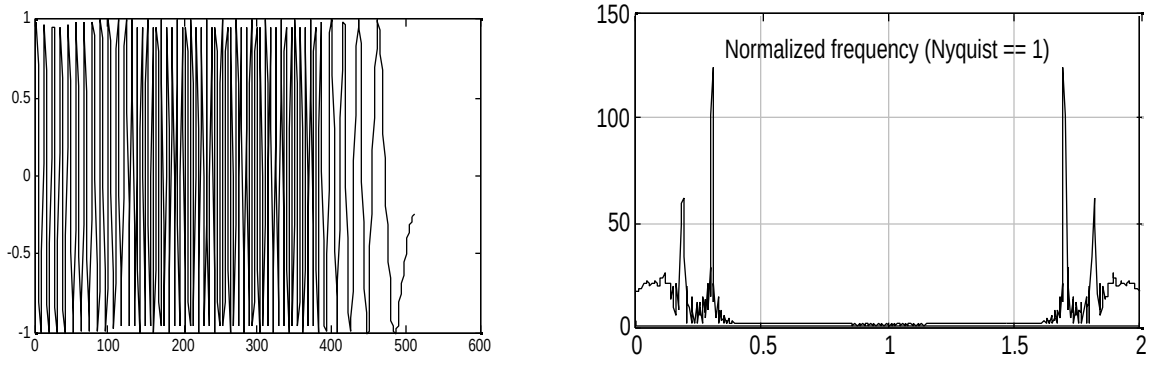

Figure 11.5–1 Composite demonstration signal with FSK modulation followed by FM modulation shown in the time domain at left and in the frequency domain at right.

We next add some high-frequency noise in the form of harmonics/intermodulation effects. In other words, as the modulated frequency of the original signal changes, so does the frequency of the noise. This is shown in Figure 11.5–2

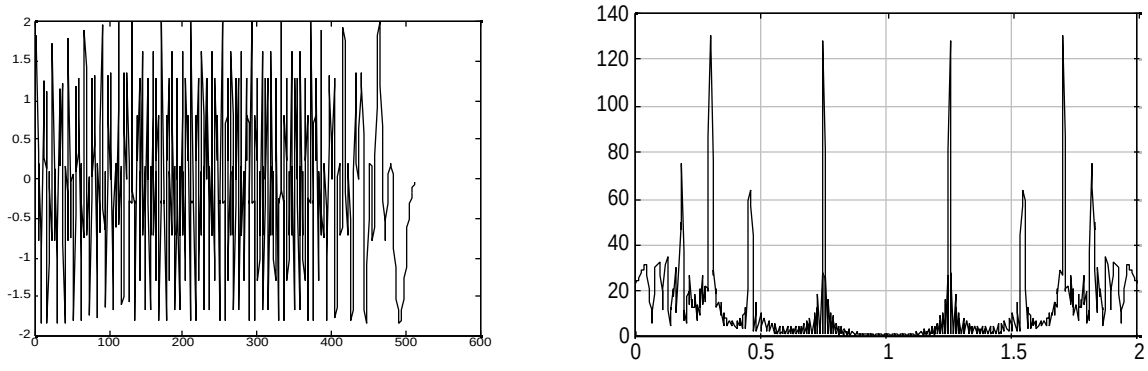

Figure 11.5–2 Noisy composite demonstration signal shown in the time domain at left and in the frequency domain at right.

As can be seen by comparing the FFT of the original signal to the FFT of the noisy signal (right graphs of Figs. 11.5–1 and 11.5–2) we cannot isolate the noise from the signal using conventional DSP filtering techniques. We turn again to wavelet processing.

We use a "general purpose" set of Db4 wavelet filters and perform a conventional DWT and a UDWT on the noisy signal as shown below in Figure 11.5–3. As we compare the DWT and the UDWT displays we notice differences, especially in the higher frequency sub-bands of **D1**, **D2**, and **D3**. Furthermore, the UDWT looks "cleaner" in isolating the sections of the signal. The frequency allocation is the same for both the DWT and UDWT (ref. Fig. 11.1–5) and the wavelet is the same (Db4) causing us to ask "Why the differences in results from using the 2 methods?"

The answer, as we will demonstrate, is non-canceled aliasing in the conventional DWT while the UDWT (which uses stretched filters instead of downsampling) has no such problems. We note in passing that the CWT (which also uses the stretched "**H**'" filter) is the same for the DWT and the UDWT customized displays (upper right graphs in both displays).

To better see what's going on, we look at the center part of the signal by itself. We would do something similar to this "isolation in time" as we exploit the time/frequency capabilities of wavelet processing to order to impose different thresholds on the various time segments of the signal (interval dependent thresholds) as shown earlier in this chapter (ref. Figs. 11.1–6 and 11.2–5).

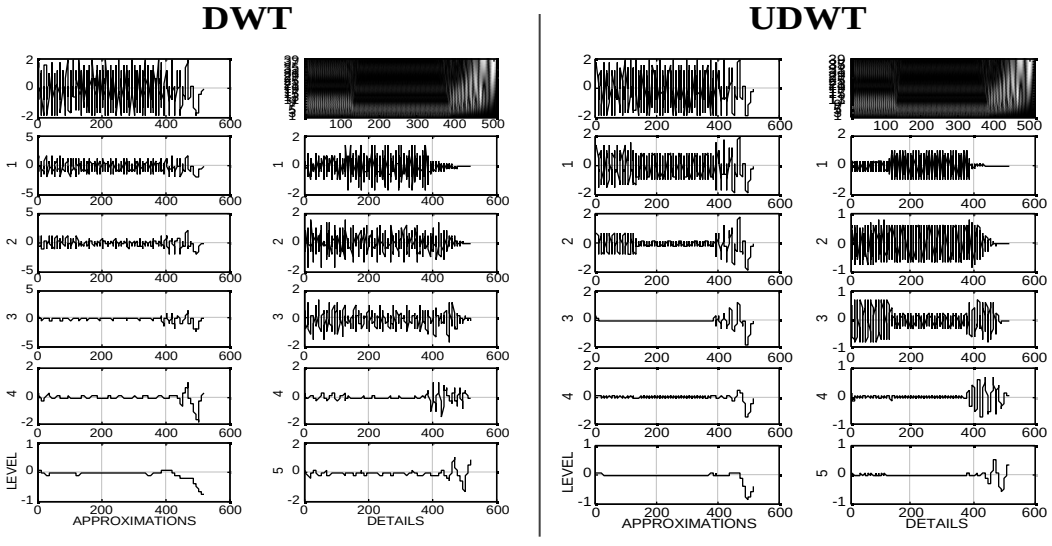

Figure 11.5–3 Db4 wavelet decomposition of the noisy composite FSK/FM signal using a convention al DWT at left and an Undecimated DWT at right.

Figure 11.5–4 shows the center 256 points of this 512 point noisy signal. It is composed of a 0.3 Nyquist sinusoid with a 0.75 Nyquist sinusoid as the noise. Before proceeding, we note that this center portion of this FSK/FM noisy signal is a *stationary signal* in that it does not change frequencies or amplitude (envelope) over time. If the entire 512 points of the signal looked like this we would be better off using a sinusoid as the "wavelet"—in other words to use conventional Fourier techniques instead of wavelet processing. However, since this "sum of sines" can occur, even for a very short time,* in a large number of ways and in a variety of signals we will proceed.

Another glance at the frequency graph from Fig. 11.5–4 below (also ref. Fig. 11.1–5) shows the frequency subband **D1** to be roughly from 0.5 to 1.0 Nyquist and the subband **A1** to be from roughly 0.0 to 0.5 Nyquist (with some overlap due to the imperfect filtering of the Db4). Because the signal at 0.3 Nyquist is well-isolated from the noise at 0.75 Nyquist it appears that the signal should be found in **A1** with a minimal amount in **D1** while the noise should be found almost entirely in **D1** with minimal amounts in **A1**. A single-level DWT or UDWT should show this.

* Although a Short Time Fourier Transform (STFT) could be used for stationary portions of a signal, if the time is too short we won't achieve meaningful results. The Wavelet Transforms are a better choice here.

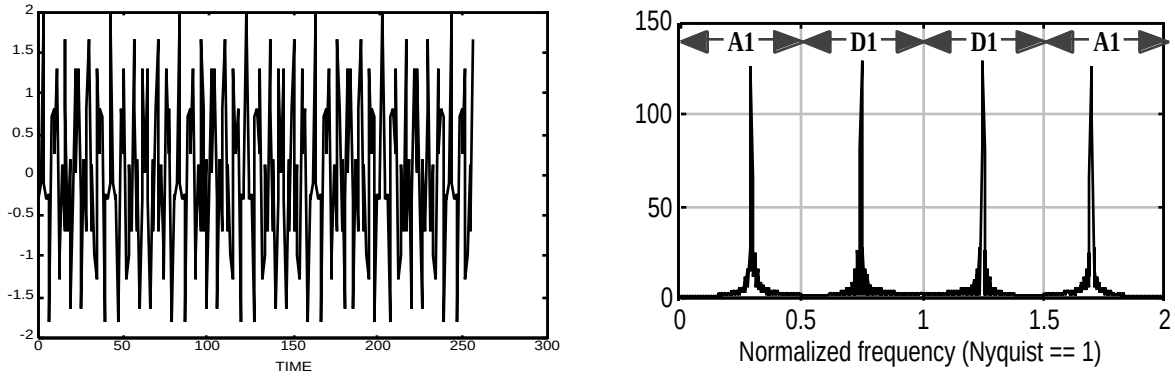

Figure 11.5–4 Center 256 points of noisy 256-point signal shown in both the time and frequency domains.

As a "sanity check",* we will first look at a single-level decomposition of the *noiseless* signal by itself using both the conventional (decimated) DWT and the Undecimated DWT (UDWT). Figure 11.5–5 shows the single-level DWT and then the UDWT displays for the *noiseless signal*. The signal is found almost entirely in **A1** for the UDWT with very little in **D1** (note different scales) as expected. But the conventional DWT has much more high-frequency components in **D1**. We will show that this is not due to noise, but instead actually due to the aliasing of the noiseless *signal*!

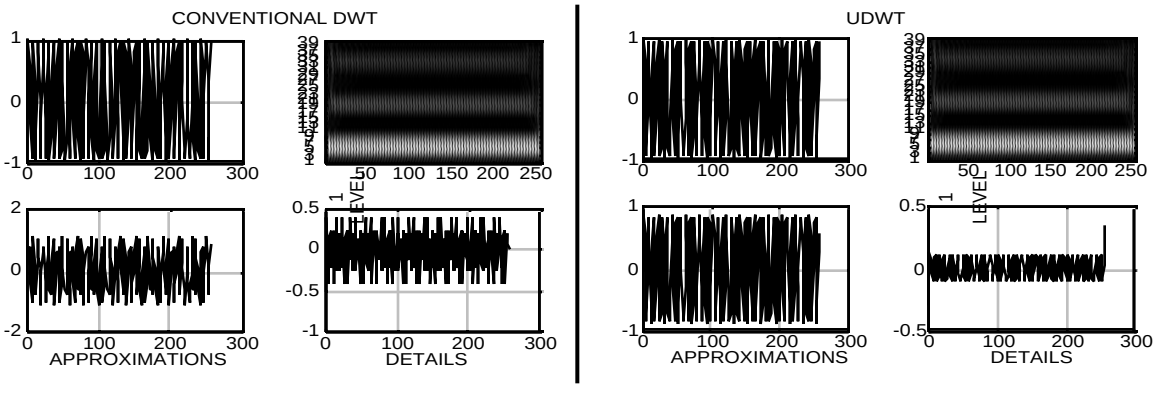

Figure 11.5–5 Results of a single-level decomposition of the pure signal with no noise using a Db4 filter set. Conventional DWT results are compared with UDWT results.

* Some would argue that it will take more than this simple check to establish the author's sanity—but it is a good idea when faced with puzzling results to look at the basics and build from there.

The block diagrams for the single-level conventional DWT and the UDWT are drawn again in figure 11.5–6. We can see the potential for aliasing from downsampling in the Conventional DWT here.

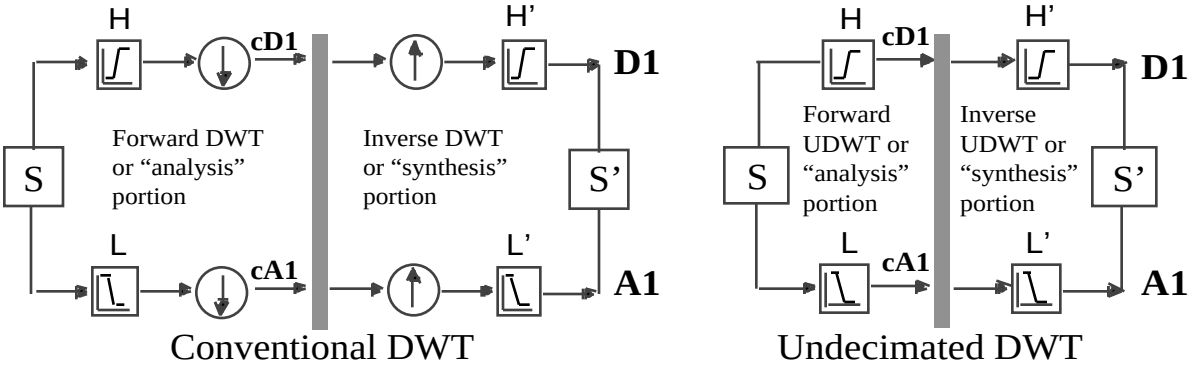

Figure 11.5–6 Single-level Conventional DWT and Undecimated DWT. The 2 are identical (for the single level case) except for the lack of downsampling (decimation) by 2 in the UDWT.

A look in the frequency domain will illustrate the problem. Figure 11.5–7 shows the *noiseless* signal after the single-level decomposition using a Db4 UDWT. As expected, most of the 0.3 Nyquist signal is found in **A1** (left graph). Because of imperfect filtering we have a small amount of the signal in **D1** (right graph). Note the difference in scales however with **A1** containing almost 9 times the signal content of **D1**.

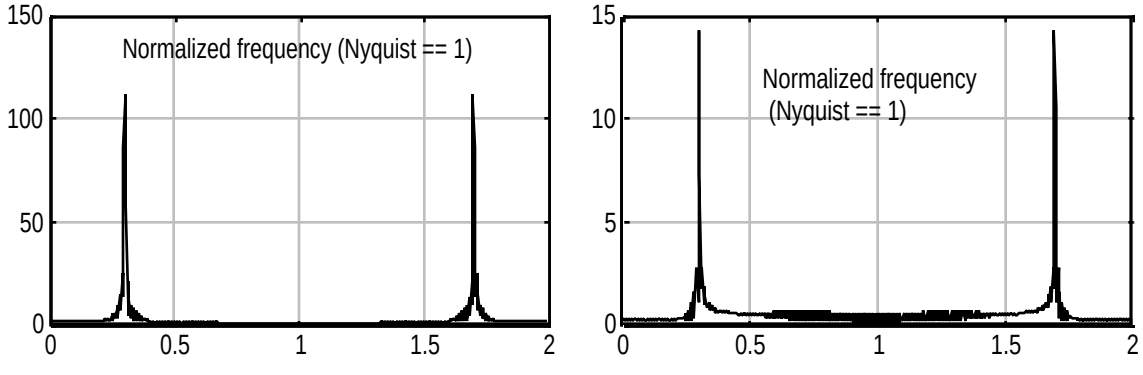

Figure 11.5–7 Frequency domain results for the Undecimated (non-downsampled) UDWT *noiseless* pure signal. **A1** (left graph) contains the signal at 0.3 Nyquist with no noise as expected. **D1**, as shown in the right graph, contains some small aliasing due to imperfect filtering.

We next look at frequency domain results using the same *noiseless signal* in the **A1** and **D1** subbands produced by the *conventional* DWT as shown in Figure 11.5–8. The results show aliasing present in both **A1** and **D1**. Recall from DSP that for a signal at 0.3 Nyquist aliasing from downsampling will "reflect" the signal across Nyquist. Thus we see the aliasing components at Nyquist minus 0.3 Nyquist or 0.7 Nyquist. This is not to be confused with our earlier noise at 0.75 Nyquist—*there is no noise* in this sanity check! In other words, this alias artifact will appear even in a noiseless signal.

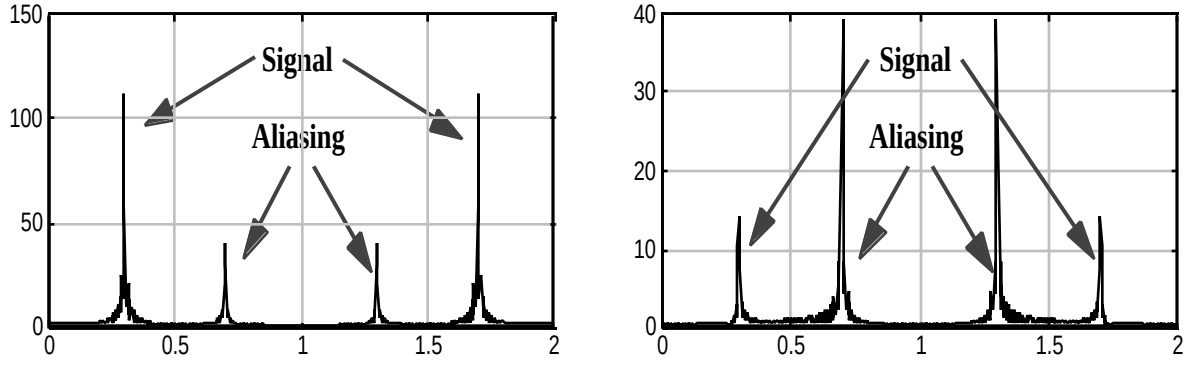

Figure 11.5–8 Frequency domain results for the *conventional* (downsampled) DWT of the *noiseless* pure signal. **A1** (left graph) has aliasing components at (**1.0 – 0.3**) **= 0.7** Nyquist. **D1** (right graph) also has magnitude 40 aliasing components at 0.7 Nyquist .

Notice from the above figure 11.5–8 that the signal at 0.3 Nyquist has the same components (magnitude 120 in **A1** and 14 in **D1**) as with the UDWT case (ref. Fig 11.5–7). Notice also the *aliased components* at 0,7 Nyquist (magnitude 40) are the same size! This is important because when we add **A1** to **D1** in the conventional DWT these components cancel. But when we throw away **D1** to get rid of some high frequency noise (added later) we are also throwing away the alias cancellation components and we are left not only with the 0.3 Nyquist signal but also a substantial 0.7 Nyquist alias artifact. Note: We have shown only the magnitudes of the (complex) signal here. We will discover in the next chapter how the aliased components in **A1** and **D1** are 180 degrees (π radians) out of phase and thus cancel.

Having demonstrated the superior performance of the UDWT on the *noiseless* signal, we now compare the DWT and UDWT on the signal (center portion of the FSK/FM) with *added noise*. With the signal at 0.3 Nyquist and the noise at 0,75 Nyquist (ref. Fig. 11.5–4) we will look at the results of keeping

A1 while discarding **D1**. A look at the frequency domain of the UDWT **A1** and **D1** in Figure 11.5–9 below shows this to be a viable option for this particular signal.

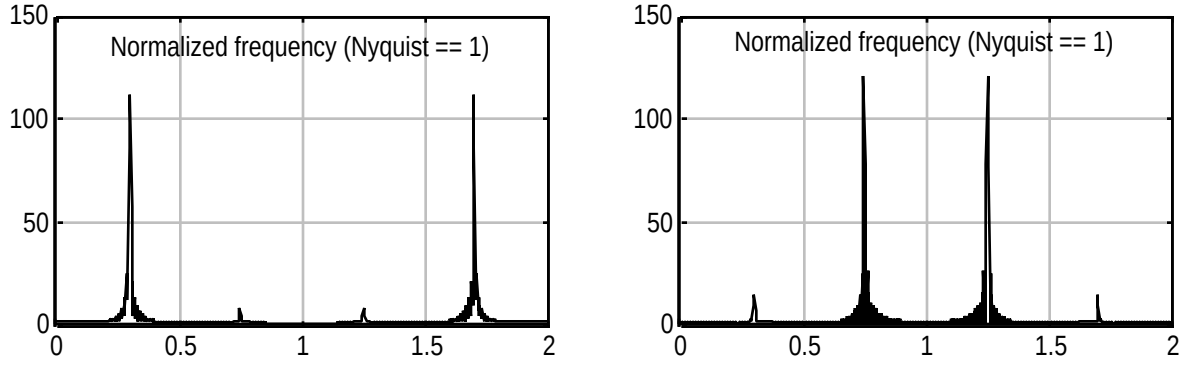

Figure 11.5–9 Frequency domain results for the UDWT noisy signal. **A1** (left graph) contains almost all the signal at 0.3 and very little of the noise at 0.75 Nyquist. **D1** (right graph) contains almost all the noise and very little of the signal.

However, a look at the *conventional* DWT **A1** and **D1** in the frequency domain indicates that this option of keeping **A1** as the "de-noised" signal is not viable because of aliasing problems caused by the downsampling. This is shown below in Figure 11.5–10

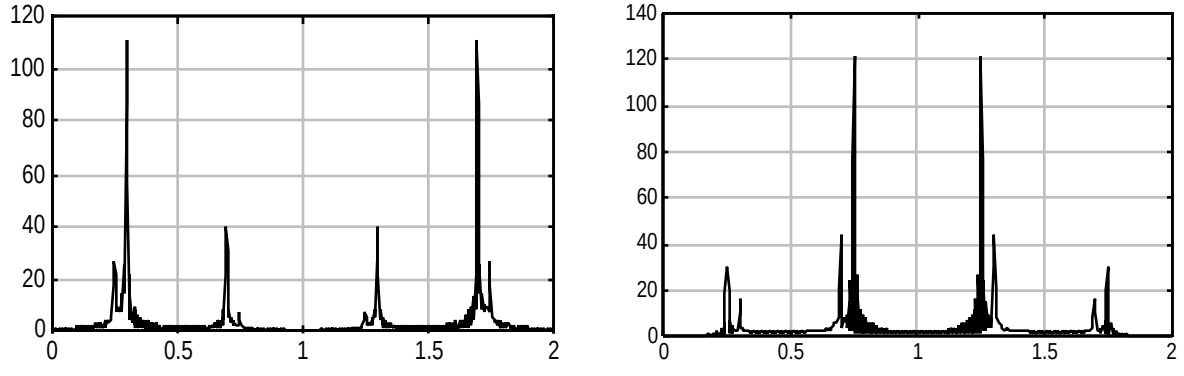

Figure 11.5–10 Frequency domain results for the conventional (downsampled) DWT noisy signal. **A1** (left graph) contains the signal at 0.3 Nyquist along with significant aliasing effects. **D1** shows the added noise at 0.75 Nyquist along with further aliasing effects.

We can now compare the results of denoising using the conventional DWT with those of the UDWT for this stationary portion of the noisy signal using a single-level Db4 wavelet transform (keeping **A1** and discarding **D1**). We saw **A1** in the frequency domain earlier in Figs. 11.5–7 and 11.5–8. We now take a look in the time domain. Figure 11.5–11 shows a close-up of the original noiseless signal and the same signal with noise added.

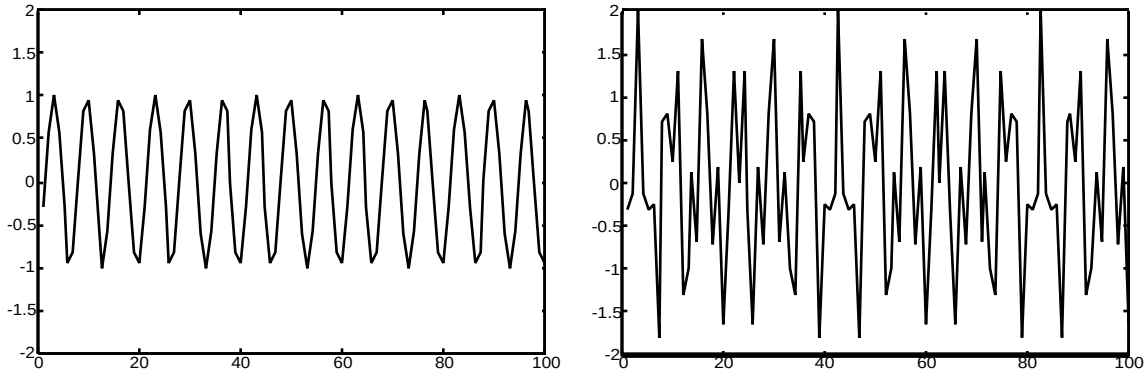

Figure 11.5–11 Close-up in time domain of original and noisy signals.

Figure 11.5–12 shows a close-up of the de-noised signal using the conventional DWT and the same signal denoised using the UDWT.* As can be seen, the UDWT does an almost perfect job of de-noising even using a single-level transform in this pathological (but very possible) case. The conventional DWT will need additional levels, a longer wavelet, and/or further processing.

We can of course use the UDWT to denoise the rest of the 512 point FSK/FM signal. Because the conventional (decimated) DWT is in such wide use, it would be a good idea to better understand and utilize it correctly! Thus, we will look in more depth at how the alias cancellation works within the conventional DWT in the next chapter.

* Another reminder that the Undecimated DWT is also referred to as Redundant, Stationary, Quasi-Continuous, Translation Invariant, Shift Invariant, and Algorithme à Trous in some texts, papers, and software.

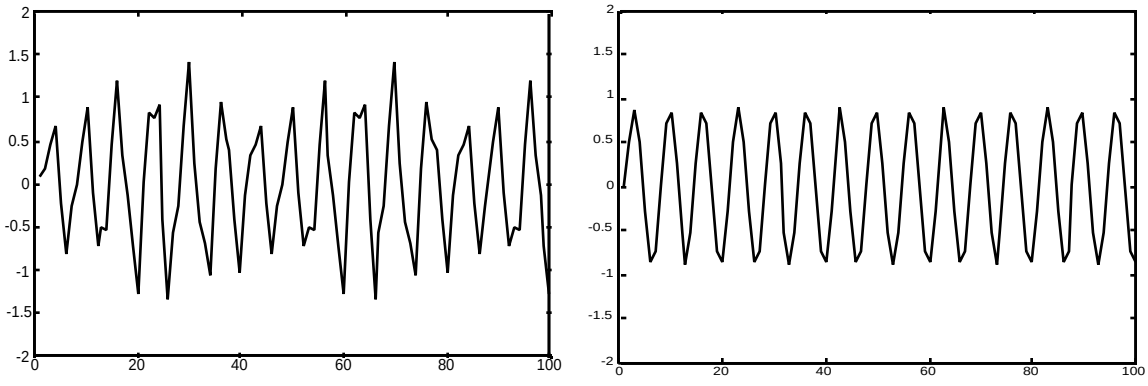

Figure 11.5–12 Close-up in time domain of conventional DWT de-noised signal (left) and UDWT denoised signal (right).

Andrew P. Bradley in his landmark paper "Shift-invariance in the Discrete Wavelet Transform" * reminds us "It should be noted that the aliasing introduced by the DWT cancels out (only) when the inverse DWT (IDWT) is performed using ***all*** of the wavelet coefficients, that is, when the original signal is reconstructed". He offers several suggestions (besides exclusive use of the UDWT) including (1) using longer filters with better frequency resolution and (2) creating a hybrid that uses the UDWT at some levels to prevent non-canceled aliasing and using the conventional DWT for improved speed and storage efficiency.

11.6 Summary

In this chapter we explored and demonstrated some of the capabilities of wavelet processing techniques using specific examples. We first added white noise to a chirp signal and then removed the (supposedly unknown) noise using a Db6 conventional DWT and exploiting the time/frequency capability of wavelet processing. We employed a somewhat similar process using a Db20 chirp wavelet to find a binary signal buried in 80 dB of noise from a chirp jammer.

Our next case was a 16 chip per bit binary signal with intermittent pseudo-random noise. The best match to the binary signal was a Haar wavelet. Performing a 7-level conventional DWT on the noiseless signal we saw there were no components of the noiseless signal in **D1** through **D4**. Thus any-

* Proc. VIIth Digit. Image Comp., Dec. 2003, pp. 29–38.

thing found in **D1** through **D4** would be noise and could be safely discarded. We were thus able to re-create the original binary signal enough to discern the binary values (+1 or –1). We also revisited the CWT for this example and demonstrated its capability.

We demonstrated image compression on first a test pattern and then using a subset of the "Barbara" image. We added some noise in the form of skin imperfections (freckles). By thresholding the levels (in two dimensions) we were able to compress the image by almost an order of magnitude and in the process selectively remove high frequency components. The final image had a "soft" look that removed the skin imperfections. The wavelet filters of choice were those of the Biorthogonal 7/9, chosen for their symmetry and short length. We used a Haar for comparison and obtained better edge detection but far worse skin quality.

In the last example, we featured a case where de-noising using a UDWT was superior to using a conventional DWT. In both the DWT and UDWT a Db4 set of filters and a single level decomposition was used. The problem was shown to be with the conventional DWT itself. In real-life Digital Signal Processing we can get aliasing when we downsample. The (conventional downsampled or decimated) DWT cancels aliasing, but when we throw away parts of the decomposition (**D1** for a single-level DWT) we also lose the alias cancellation. Alternatives include (1) using the *Undecimated* DWT exclusively; (2) using the UDWT for part of the decomposition; and (3) using a longer wavelet with better frequency resolution or (4) checking first to see if the values in the Details (for a given length of time or space) are close enough to zero that they can be safely suppressed for that interval.

In all these cases, conventional DSP methods with filtering and/or FFT methods would not work. A Short Time Fourier Transform might work on some of these examples but the dynamic nature of wavelets usually make them a better option. Also, in every case we tried to match the wavelet to either the signal or noise for best discrimination.

With these examples under our belt we can take a closer look at the alias-cancellation methods used in the conventional DWT and gain further conceptual understanding and, hopefully, increased wisdom in using the various wavelet transforms.