JOHN SADOWSKY

THE CONTINUOUS WAVELET TRANSFORM: A TOOL FOR SIGNAL INVESTIGATION AND UNDERSTANDING

In this article, the continuous wavelet transform is introduced as a signal processing tool for investigating time-varying frequency spectrum characteristics of nonstationary signals. The transform is discussed within the context of Fourier methods and the general problem of time–frequency representation and is compared with the more traditional Gabor method of windowed Fourier transforms. To take full advantage of the potential benefit of wavelet characterization, a computer algorithm for generating wavelet images from data is needed. Such an efficient algorithm, the *algorithme à trous*, was invented and published several years ago by an important group of engineering researchers. A detailed derivation and analysis of their algorithm is presented.

INTRODUCTION

Five to 10 years ago, the theory of wavelets caught the imagination of researchers in harmonic analysis, signal and image processing, and applied science as the new methodology, promising to usurp the throne occupied by Fourier analysis and solve the problems that have confounded classical analysts for hundreds of years. Five years ago I met with Dr. Howard Resnikoff, president of Aware, Inc., and an early advocate of the potential of wavelets. He declared that some of the most important problems in applied mathematics were being solved by a group of French mathematicians and scientists using new tools for analyzing signals in new types of basis functions. He advised us to learn about wavelets if we wished to remain on the cutting edge of signal and image processing. Dr. Resnikoff is a well-respected scholar in applied mathematics and signal processing, so his level of enthusiasm conveyed a sense that yet another major breakthrough in science was occurring.

Wavelet analysis is indeed an important development in mathematics, science, and engineering. Rather than replacing Fourier methods, however, it complements and extends these classical approaches, solving problems for which Fourier analysis is unsuited and failing in cases for which Fourier analysis is ideal. One cannot design wavelet bases and analyze the wavelet transform without significant use of Fourier methods.

Moreover, the roots of wavelet analysis reach back almost 100 years. They can be found in a variety of sciences, from quantum mechanics and Brownian motion to the mathematical works of Littlewood and Paley in the 1930s, Gabor in the 1940s, and Calderón in the late 1950s. Meyer[1] presents an excellent, though terse, overview of the history of wavelet analysis.

Wavelet analysis is an essential addition to the toolbox of researchers and developers in the field of signal processing. Important and exciting new systems planned at the Laboratory could benefit greatly from wavelet-based processors. Applications of wavelet technology at APL include essential defense needs (e.g., signal intelligence, smart weapons, command and control, and communications systems), space and oceanographic data analysis, biomedical systems (e.g., medical imaging systems, speech and visual processing systems, and biological sensor monitoring), and basic science research and development.

This article is the first in a series planned for the *Johns Hopkins APL Technical Digest* to introduce and explore this growing and fascinating field. This first article begins with the definition of wavelets, the wavelet transform, and bases of wavelets and then derives an algorithm for the continuous wavelet transform (CWT). The second article will examine data processed with the algorithm to investigate how the signal parameters and characteristics are manifest in the complex surface of a wavelet transform. Future articles will explore the atomic decomposition of signals and images with respect to wavelet bases and frames and the application of wavelet technologies to signal analysis, signal and image compression, and pattern recognition; computational algorithms for such applications; and the use of wavelet-based processing in APL systems and problems.

WHAT IS A WAVELET?

There are several approaches to understanding the nature of wavelet analysis. These different approaches—actually, different perspectives of the same overall concept—reflect different tastes, styles, and areas of application among researchers (see Refs. 1–4 for varied points of view). One possible approach is from the viewpont of time–frequency representations of nonstationary and wideband signals. Consider a musical composition, for example. Our emotional and intellectual response to music results from the time variation of the frequency spectrum of the music—different notes are played at different times. A Fourier transformation of a piece of music does not directly represent this time variation of spectrum.

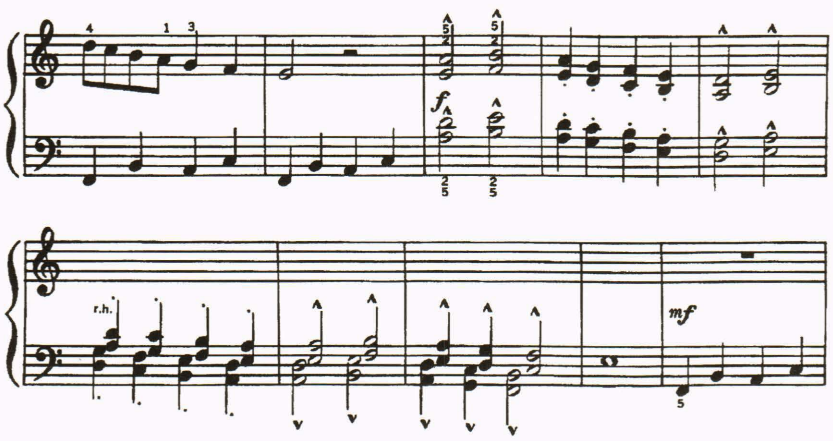


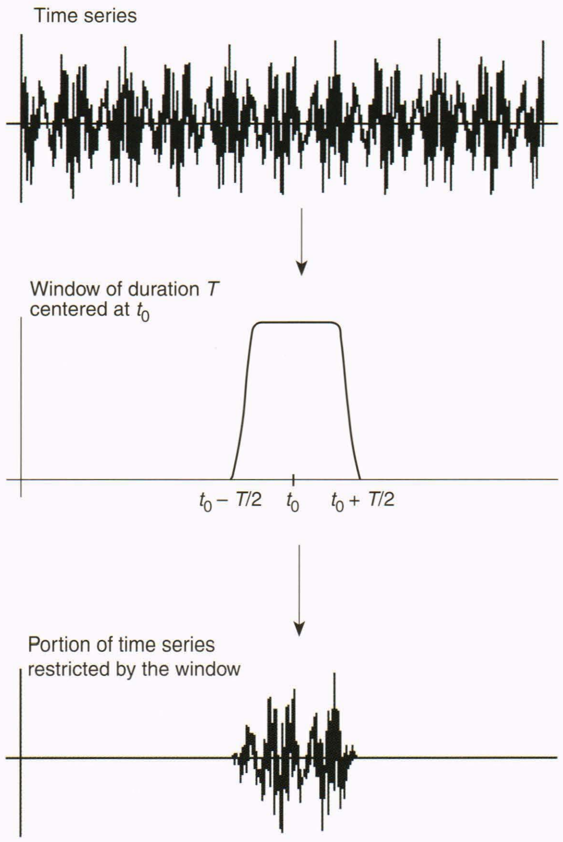
Figure 1. Example of an effective time–frequency representation, a musical score.

The frequencies present in the music are certainly visible in a Fourier transformation, but the time-dependence seems to be lost.

Of course, the time-dependence cannot actually be lost because an inverse Fourier transform of the spectrum will re-create the piece of music. (I am making the mathematician's assumption that the music is of infinite duration and that the spectrum is complete, from $-\infty$ to ∞ in both cases.) The time variation is contained in the phase of the Fourier spectrum, however, in a way that is inconsistent with the way that musicians create music. The Fourier spectrum contains frequencies that are not present in the piece of music but rather at phases such that coherent superposition of frequencies causes the proper amount of destructive interference to cancel notes not present at specific times. It is as if musicians were specifically playing notes not in the score, phased so as to cancel out some notes to produce proper durations of other notes.

Music, however, occurs because musicians play notes to be heard at specific times and for specific durations. These notes and times of performance are well repesented in the musical score (Fig. 1). Here, the horizontal axis represents time and the vertical staff indicates frequency (notes) to be played at specific times. Durations are indicated by the types of notes (quarter, half, whole, etc.), times of absence of frequencies for specific instruments are indicated by rests, and amplitudes of frequncies are indicated by notations for crescendos, diminuendos, and other dynamic markings (such as piano and forte). The score is a good representation of the music in time (horizontal) and frequency (vertical). A problem in time–frequency representation is to design a signal processing system that will produce the score from the music.

Wavelet theory offers a novel approach to this problem and, as such, can be seen as an extension and enhancement of the Gabor windowed Fourier transform method for time–frequency representation (see, for example, Meyer[1] and Daubechies[2]). In the Gabor method, we define a finite-duration window $w(t)$ as shown in Fig. 2. By translating this window in time and multiplying it by the signal $s(t)$, we obtain a chunk of the signal over a particular small piece of time. If the window is nonzero between $-T/2$ and $T/2$, for example, then the product $s(t)w(t-t_0)$ is an approximation to the signal in the time interval from $t_0 - T/2$ to $t_0 + T/2$. That is, the product selects a portion of the signal of duration T centered at



Time series
Window of duration T centered at t_0
$t_0 - T/2$
t_0
$t_0 + T/2$
Portion of time series restricted by the window

Figure 2. Restriction of the time duration of a time series with a Gabor window (Gaussian).

time t_0. The Gabor windowed Fourier transform is then defined to be the Fourier transform of this windowed signal:

$$G(\omega, t_0) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} s(t) w^*(t - t_0) e^{-i\omega t} dt \ ,$$

where * denotes the complex conjugate. Although the window often is real, we have written the Gabor transform in its more general form. Also, we will denote signals, that is, functions of the time variable t, with lowercase Latin and Greek letters and the Fourier transforms of signals with uppercase Latin and Greek letters. Thus, for example, the Fourier transform of the signal $f(t)$ is

$$F(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt \ .$$

The frequency variable in the Fourier transform is in units of radians per second, for consistency with the wavelets literature, and i denotes $\sqrt{-1}$.

This transform is a time–frequency representation as well as a function of time t_0 and frequency ω, and provides an approximation to the frequency content of the signal over a small time interval about time t_0. If the window is well concentrated in time and frequency, as is the case, for instance, with a Gaussian window, then this time–frequency representation is reasonable for some applications. It has limitations, however. For example, resolution in frequency is a function of the duration of the window; the longer the window duration, the finer the frequency resolution. Conversely, increasing the window duration will smear a rapidly changing time variation in the spectrum.

One can recast the definition of the Gabor transform as a filtering, or convolution, of the signal with the translated window, $w_\omega(t) = w(-t)e^{-i\omega t}$. This filtering kernel is the Gabor window translated in frequency via modulation by the factor $e^{-i\omega t}$. Frequency translation is affected by the number of wavelengths within the window (Fig. 3).

The wavelet transform is also built from a window function called the mother wavelet ψ. The mother wavelet usually satisfies some admissibility condition, such as a requirement that $\int_{-\infty}^{\infty} \psi(t)\,dt = 0$, and any finite energy function satisfying the admissibility condition can be used as a mother wavelet. This admissibility condition implies that the mother wavelet has some oscillations; it must be negative in places to compensate for the places in which it is positive, so that the wavelet integrates to 0. To be useful, the mother wavelet is usually nonzero only on a finite interval, or decreases rapidly (at least quadratically) for t approaching $-\infty$ and ∞.

If $\Psi(\omega)$ were to denote the Fourier transform of the mother wavelet, then one sees directly from the definition of the Fourier transform that $\Psi(0) = \int_{-\infty}^{\infty} \psi(t)\,dt$. Thus, from the admissibility condition, one has $\Psi(0) = 0$; that is, the mother wavelet, viewed as a filter, notches out the DC term of a signal. A standard example of a mother wavelet is $\psi(t) = (1 - t^2)e^{-t^2/2}$, the "Mexican Hat" function, illustrated in Fig. 4a. One can see that the wavelet is low-pass in the time domain. We compute the Fourier transform of the Mexican Hat function, $\Psi(\omega) = \omega^2 e^{-\omega^2/2}$, and illustrate this function also in Fig. 4b. As is seen, this function is band-pass in the frequency domain. These are characteristics of useful mother wavelets.

We can now define the CWT. Suppose that $\psi(t)$ is a mother wavelet, meeting the admissibility condition, and that $s(t)$ is a finite energy signal. The CWT of $s(t)$ is the function of two variables, $a > 0$ and b, defined by

$$W(a, b) \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} s(t) \psi^* \left(\frac{t - b}{a} \right) dt \ .$$

Like the Gabor transform, the CWT can be viewed as a filtering of the signal by a dilated version of the mother wavelet, defined and denoted by $\psi_a(t) = 1/\sqrt{a}\,\psi(-t/a)$. Unlike the filter in the Gabor case, however, the frequency translation is effected by dilation and contraction rather than by modulation. Figure 5 illustrates the dilated wavelets for different values of the so-called scaling variable a. As can be seen, for $a > 1$, the dilated wavelet expands in time; for $a < 1$, the dilated wavelet contracts in time. Regarding the wavelet as a sort of window in time, the window width adjusts, depending on scale, widening for large-scale (low-frequency) information and narrowing for small-scale (high-frequency) content.

The CWT is a time–frequency, or more correctly, a time-scale representation. To demonstrate this, we derive a "frequency domain" formulation of the CWT as follows. Substituting the inverse Fourier transforms $s(t) = 1/\sqrt{2\pi} \int_{-\infty}^{\infty} S(\omega)e^{i\omega t} d\omega$ and $\psi(t) = 1/\sqrt{2\pi} \int_{-\infty}^{\infty} \Psi(\nu)e^{i\nu t} d\nu$ into the

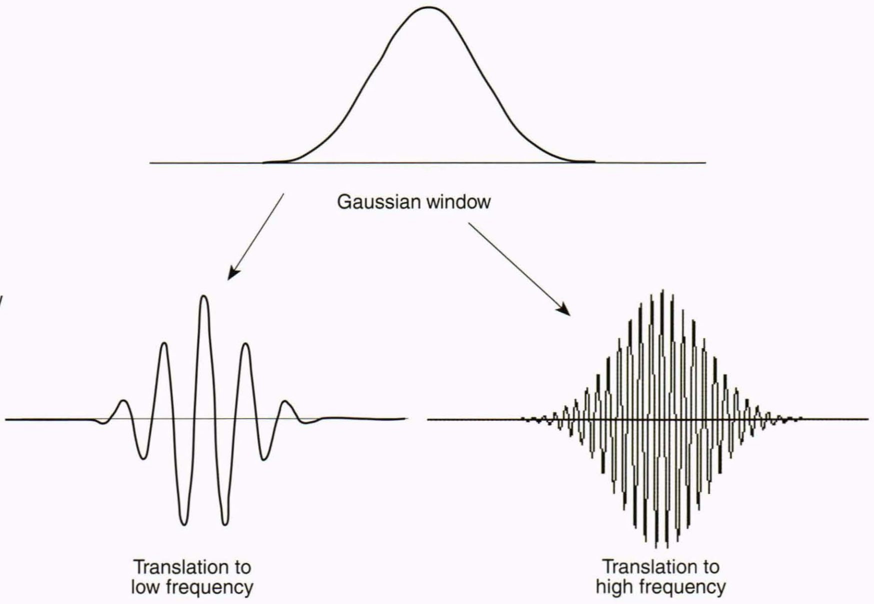


Figure 3. Translation of a window in frequency via modulation.

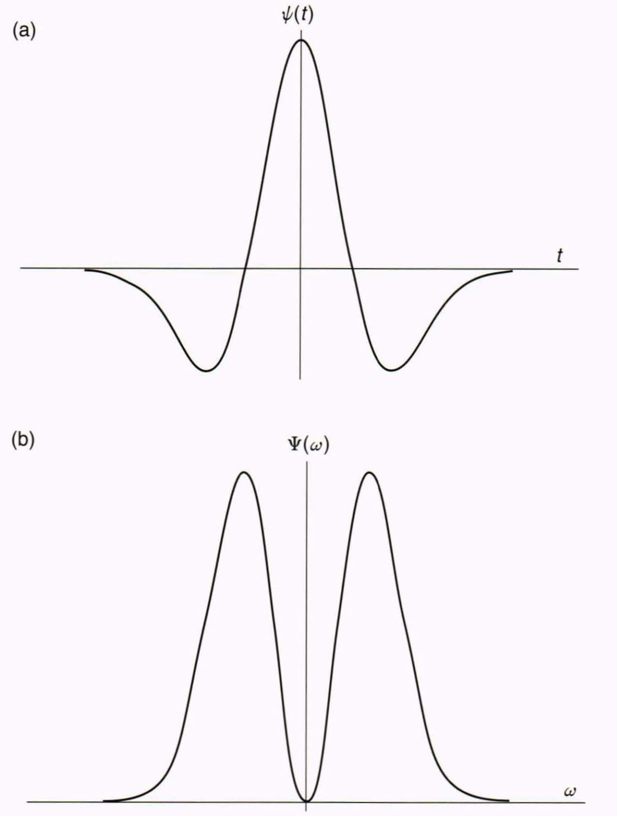


Figure 4. The Mexican Hat wavelet and its Fourier transform. (a) Mexican Hat function, $\psi(t) = (1 - t^2)e^{-t^2/2}$. (b) Fourier transform of the Mexican Hat function, $\Psi(\omega) = \omega^2 e^{-\omega^2/2}$.

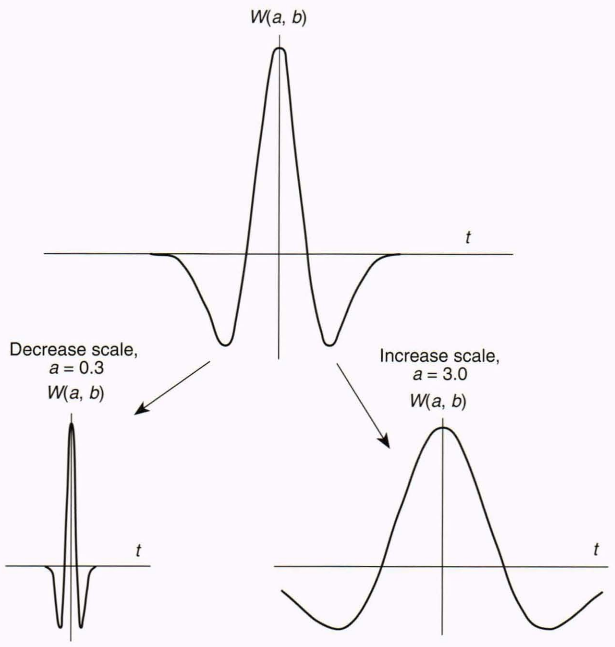


Figure 5. Frequency translation of the Mexican Hat function via scaling (dilation): time domain.

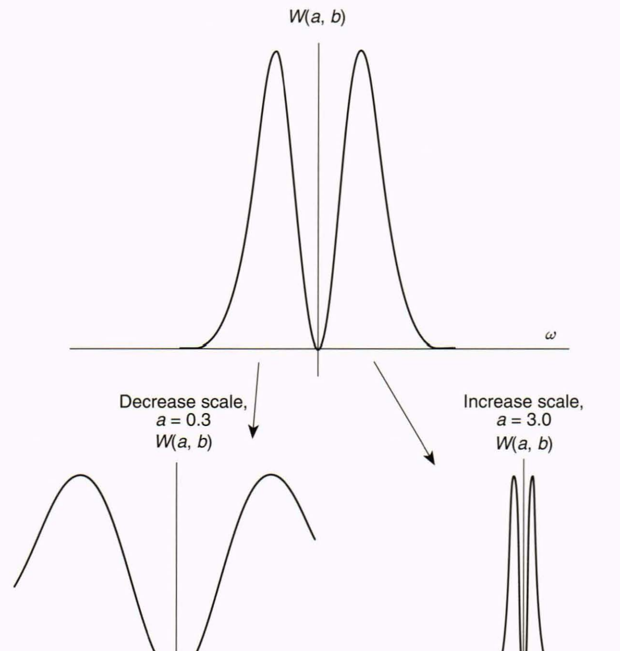


Figure 6. Frequency translation of the Mexican Hat function via scaling (dilation): frequency domain.

formula for $W(a, b)$, integrating the resulting Dirac δ-function, and simplifying, one obtains

$$W(a,b) = \sqrt{a}\int_{-\infty}^{\infty} S(\omega)\Psi^*(a\omega)e^{i\omega b}\, d\omega\,.$$

We see that the CWT can be viewed as a frequency-domain filtering of the signal by the dilated filter $\sqrt{a}\Psi(a\omega)$. Figure 6 illustrates this dilation filter (in the frequency domain) for the Mexican Hat wavelet. As can be seen, the large-scale case, $a >> 1$, has the filter compressed to small-bandwidth (fine-resolution), low-frequency parts of the spectrum; the small-scale case, $a << 1$, has the filter expanded to wide-bandwidth (coarser-resolution), high-frequency parts of the spectrum. Scale has the relationship to frequency that one would expect, and the frequency resolution of the CWT adjusts such that the ratio of frequency resolution to frequency is constant. Not only is this frequency-dependent resolution consistent with the performance of visual and auditory systems, but it also permits an efficient computation of a range of frequencies, without the need to set a fixed, high resolution and compute redundant information when lower resolution would suffice.

The preceding paragraph describes the zoom-in property of the CWT. A wavelet transform zooms in on the fine detail and zooms out on the coarser trends of a signal. This property distinguishes the CWT from the Gabor transform, for which the frequency resolution is preset and is independent of the detail or coarseness of the parts of the signal being analyzed. In addition, this frequency-dependent resolution is the basis for the application of wavelet analysis to multiresolution decomposition, that is, decomposition of signals and images that represent a range of different resolution scales. Multiresolution decomposition with wavelets will be a topic in a future article about the discrete wavelet transform.

The CWT produces a complex-valued surface, as it is a function of two variables, a and b. Following convention, the b axis (time) is drawn horizontally and the a axis (scale, or frequency) is drawn vertically. Because the variable a is positive, the a axis is actually a ray. We draw the a axis oriented downward, so that smaller values of a, corresponding to higher frequencies, are above larger values of a, corresponding to lower frequencies (Fig. 7). Some researchers prefer to represent the scale axis logarithmically, in which case the $\log(a)$ axis extends between $-\infty$ and ∞. In this case, the orientation of the axis remains downward for increasing values of a.

To understand the CWT surface, consider local influences on the surface from the signal and local parts of the signal that affect the surface.[5] In particular, we wish to identify the region of the CWT surface that is influenced by the value of the signal at a particular time t_0 and the particular time intervals of the signal that contribute to a specific point (a_0, b_0) in the CWT surface.

Assume that the mother wavelet has support which is an interval Λ; that is, $\psi(t)$ is zero for t not in Λ. From the time-domain definition of the CWT, one can see that, if we fix the variables a and b, then the integral is computed over the interval $a\Lambda + b$. Thus, for time t_0 to be of influence, one must have $t_0 - aq \le b \le t_0 - ap$, where the interval Λ has endpoints p and q. Where the interval is centered about the origin, for example, this describes a cone in the ab plane with vertex at the point t_0 on the b axis, as is shown in Fig. 7b.

Similarly, the point (a_0, b_0) in the CWT surface is influenced by signal values at times in the interval $a_0\Lambda + b_0$. This can be illustrated with an upward cone

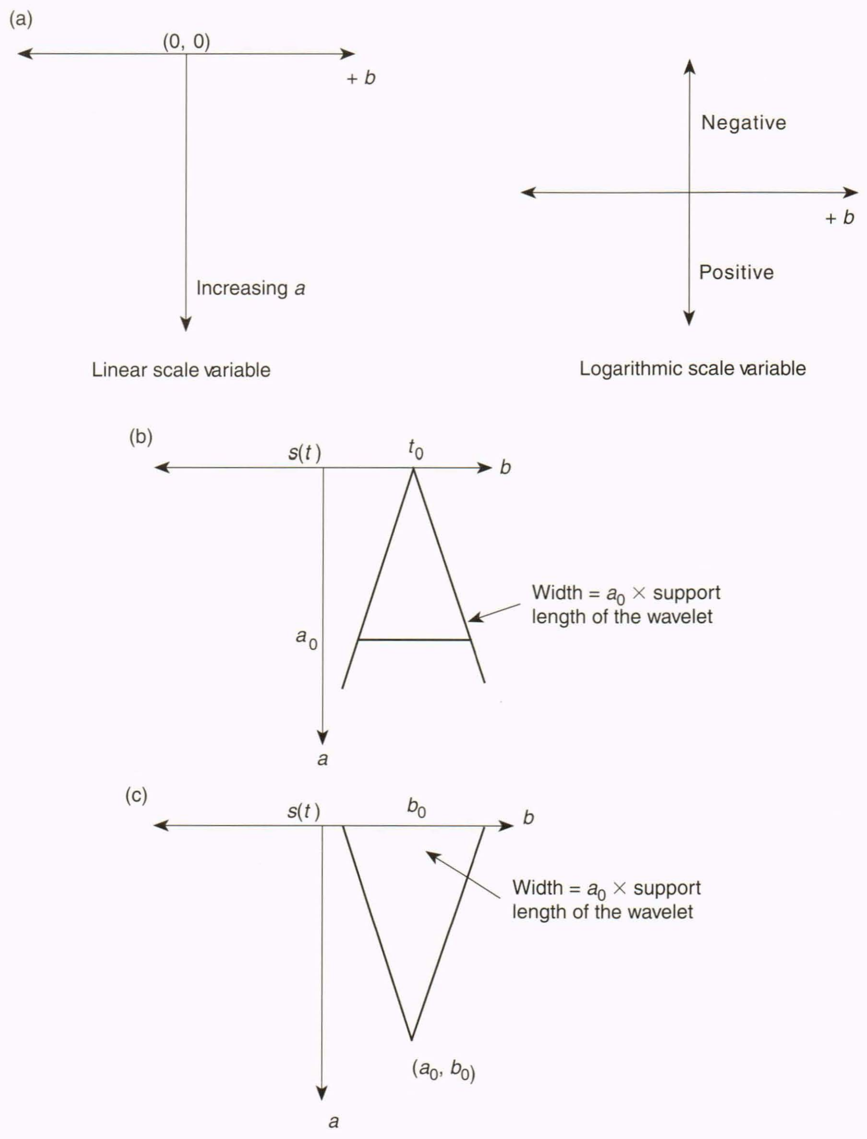

(a)
(0, 0)
+ b
Increasing a
Linear scale variable
Negative
+ b
Positive
Logarithmic scale variable
(b)
s(t)
t0
b
Width = a0 × support length of the wavelet
a0
a
(c)
s(t)
b0
b
Width = a0 × support length of the wavelet
(a0, b0)
a

Figure 7. Representation of the CWT surface. (a) The coordinate axes for image representations of the CWT. (b) Cone of influence of a time point of the signal on the surface. (c) Cone of influence of a point on the surface on an interval in time.

with vertex t_0 (Fig. 7b), where the interval on the b axis in the cone represents the time interval of the signal that affects the point in the CWT surface.

Grossmann et al.[5] suggest a method for representing the complex values of the surface without having to draw separate real and imaginary surfaces or amplitude and phase surfaces. In their method, complex amplitude is represented by color from a lookup table, and phase is represented by a density of black dots overdrawn on the colors, from no black dots (phase 0) to total saturation (phase 2π).

As a small modification to this approach, we reverse the roles of the dot density and color. Each pixel can be expanded to a 4×4 array of subpixels or dots. Normalized amplitude is represented as the number of subpixels that are colored in the surface, from zero (for an amplitude of 0 or sufficiently small) to all 16 (for an amplitude of 1 or sufficiently high). The particular subpixels to color are selected randomly. Thus, amplitude is reflected in the surface by sparseness to saturation of color. The phase is color-coded using a lookup table. Figure 8 is an example of a CWT surface generated from actual data from an intermittant process, which illustrates this concept.

The rest of this article reviews the mathematical basis for the CWT algorithm used to generate the surface in Fig. 8.

ALGORITHM FOR THE CONTINUOUS WAVELET TRANSFORM

In this section we derive an algorithm for computing the CWT. The approach follows closely the derivation and data flow diagrams of Holschneider et al.[6] with added detail and carefully computed filter delays and algorithm constructions. The algorithm explicitly recognizes that the CWT can be realized with several filters, all of which are dilations of a single filter. Their construction introduces the method of the *algorithme à trous* for efficiently implementing such a structure.

To motivate the need for such an algorithm, we consider a numerical approximation to the integral in the CWT:

$$W(a,b) = a^{-1/2}\int_{-\infty}^{\infty} s(t)\psi^*\left(\frac{t-b}{a}\right)dt\,. \qquad (1)$$

If the signal $s(t)$ is sampled with sampling interval T_s, then the CWT at the time step $b = kT_s$ is approximated by the sum

$$W(a,kT_s) \approx T_s a^{-1/2}\sum_n s(nT_s)\psi^*\left[\frac{(n-k)T_s}{a}\right]. \qquad (2)$$

The summation is essentially over the range of samples of the signal $s(t)$. The computational problem arises from the scaling factor a in the denominator of the argument of the sampled wavelet. Because of this denominator, the wavelet must be resampled at a sampling interval of T_s/a for the wavelet transform at scale a. If, for example, we wish to display the CWT over 10 octaves, the high-scale computational complexity (size of the summation) increases by a factor of $2^{10} = 1024$. The algorithm by Holschneider et al.[6] solves this problem for certain classes of wavelets by replacing the need to resample the wavelet with a recursive application of an interpolating filter.

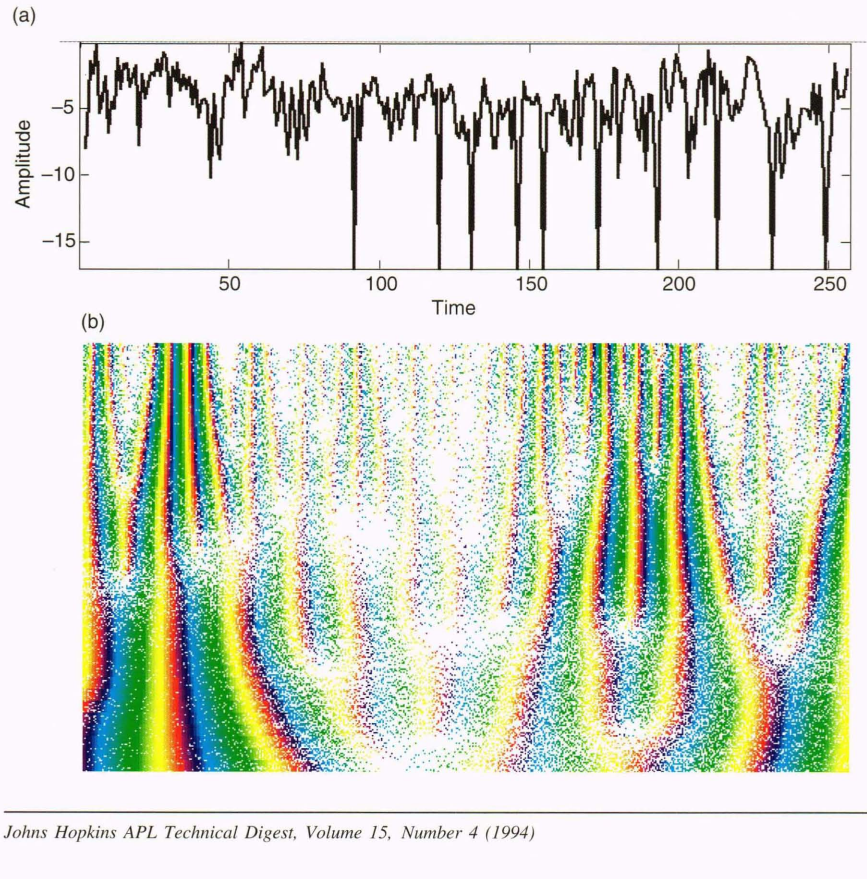


Figure 8. Example of a CWT surface generated from actual data from an intermittent process. (a) Section of the time series. (b) CWT of the time series. The horizontal axis is increasing time and the vertical axis is scale (frequency-related) from small (top of figure) to large (bottom of figure). Concentration of color reflects amplitude and color reflects phase. Interesting structure is found in the phase; however, three peaks appear in the image—two at the left and one in mid-image toward the right.

We must introduce some notation to explain exactly how to incorporate the interpolation filter. Our signal $s(t)$ is a function of a continuous variable. We can perform some operations on such functions. One operation is dilation (we use the term dilation whether we actually dilate or constrict the function). If a is a positive real number, then the dilation of $s(t)$ by a is also a function of the variable t and is defined and denoted by

$$(\mathfrak{D}_a s)(t) = a^{-1/2} s(t/a) .$$

The factor $a^{-1/2}$ is for conservation of energy under the operation.

The operation of filtering is actually a convolution, and so we introduce notation for the convolution of two signals. Given two functions, $s(t)$ and $h(t)$, the convolution of s by h is a function of the variable t defined and denoted by

$$(\mathfrak{K}_h s)(t) = \int_{-\infty}^{\infty} s(x) h^*(t-x)\, dx .$$

With these two concepts defined, we can easily see that the CWT is effectively a filtering of a signal by a dilated, inverted mother wavelet. Specifically, if we denote the time-reversed mother wavelet by $\tilde{\psi}(t) = \psi(-t)$, Eq. 1 becomes

$$W(a,b) = (\mathfrak{K}_{\tilde{\psi}_a} s)(b) , \quad (3)$$

where $\tilde{\psi}_a = \mathfrak{D}_a \tilde{\psi}$.

Defining dilation and convolution operators for sequences allows us to represent Eq. 2 as a discrete filtering operation as well. We will use boldface lowercase letters to denote doubly infinite sequences, such as $\mathbf{s} = \{\ldots, s_{-3}, s_{-2}, s_{-1}, s_0, s_1, s_2, s_3, \ldots\}$. We must carefully define the dilation operator because we cannot divide the sequence indices to obtain fractional indices. Instead, we change the sampling interval to stretch our sequences by interleaving zeroes between sample values. If p is a positive integer, then the dilation of sequence $\mathbf{s}$ by p is defined as the sequence $D_p\mathbf{s}$ whose nth element is

$$\left(D_p \mathbf{s}\right)_n = \begin{cases} p^{-1/2} s_{n/p}, & \text{if } n \text{ is a multiple of } p \\ 0, & \text{otherwise} \end{cases} .$$

When $p = 3$, for example, $D_3\mathbf{s} = \{\ldots, 0, 0, s_{-3}, 0, 0, s_{-2}, 0, 0, s_{-1}, 0, 0, s_0, 0, 0, s_1, 0, 0, s_2, 0, 0, s_3, 0, 0, \ldots\}$, with the element s_0 corresponding to the sample time $n = 0$.

Convolution is in direct analogy with the convolution definition of the continuous variable case. Suppose that $\mathbf{s} = \{\ldots, s_{-2}, s_{-1}, s_0, s_1, s_2, \ldots\}$ and $\mathbf{h} = \{\ldots, h_{-2}, h_{-1}, h_0, h_1, h_2, \ldots\}$ are two sequences. The convolution of $\mathbf{s}$ by $\mathbf{h}$ is the sequence denoted by $K_\mathbf{h}\mathbf{s}$ whose nth element is defined as

$$(K_\mathbf{h}\mathbf{s})_n = \sum_{m=-\infty}^{\infty} s_m h^*_{n-m} .$$

We will also need to use a delay operator Z, which delays the sequence by one sample position. Given $\mathbf{s}$ as defined above, one defines $Z\mathbf{s}$ such that the nth element is $(Z\mathbf{s})_n = s_{n-1}$.

Finally, to relate the signals of the continuous variable with the doubly infinite sequences, we introduce a sampling operator to map the former into the latter. For this we fix a sampling interval T_s and define the sampling operator P to take a signal $s(t)$ and map it to a sequence Ps whose nth element is $(Ps)_n = s(nT_s)$.

We can now rewrite Eq. 2 in a form similar to Eq. 3 as follows. If the function $\tilde{\psi}_a$ defined in Eq. 3 is sampled to produce the sequence denoted by $\mathbf{g_a} = P\tilde{\psi}_a$, and if the signal $s(t)$ is sampled to produce the sequence $\mathbf{s} = Ps$, then Eq. 2 becomes

$$W(a, kT_s) = T_s (K_{\mathbf{g_a}} \mathbf{s})_k . \quad (4)$$

Equation 4 is a filtering operation. The exponentially increasing computational complexity, as a function of number of octaves of scale, arises because the sampling operator is applied after the application of the continuous variable dilation operator. To solve this problem, Holschneider et al.[6] introduced another operator on sequences to approximate these operations.

For the moment, suppose that we are only interested in a set of scale values that have constant ratio 2. Suppose also that we are interested in a range of scale over N octaves, that is, scale values from 1 to 2^N. The scales can, therefore, be considered to be octaves, $a_0 = 1$, $a_1 = 2$, $a_2 = 4$, …, $a_N = 2^N$. We will extend the algorithm later in the article to permit scales between successive octaves. It is not too difficult to show that for such a scale value, say $a = a_k = 2^k$, that the operator $\mathfrak{D}_a = (\mathfrak{D}_2)^k$, that is, k successive applications of the operator $\mathfrak{D}_2$.

We need an operation on the sampled signal $\mathbf{s} = Ps$ whose effect is as if we have sampled the dilated signal $\mathfrak{D}_a s$. In particular, we would like to define an operator $\mathbb{O}$ on the sampled signals, such that $\mathbb{O}(Ps) = P(\mathfrak{D}_2 s)$, and, in general,

$$\mathbb{O}^n Ps = P(\mathfrak{D}_2)^n s . \quad (5)$$

The obvious choice is to use the dilation operator for sequences D_2; however, it can easily be shown that Eq. 5 in this case would only be satisfied by signals $s(t)$ that are zero on all of the dyadic numbers (rational numbers whose denominators are powers of 2). This follows because D_2 interleaves zeroes between successive sample values. If $s(t)$ were a continuous signal, then it would have to be zero everywhere; thus, one would need a different choice for $\mathbb{O}$ to have a more interesting class of signals for candidate wavelets.

D_2 is not acceptable because it interleaves zeroes. An operator that interleaves interpolated samples could potentially satisfy Eq. 5, at least to a reasonable approximation. An innovative idea in the work of Holschneider et al.[6] is to introduce an interpolation filter F and to define

$$\mathbb{O} = D_2 + ZD_2K_F . \quad (6)$$

The boxed insert (Examples of Interpolation Filters) presents some choices for the filter F. The introduction of the delay operator Z in Eq. 6 interleaves the interpolated values of the signal in place of the zeroes. It is still not possible to find suitable choices for F such that $\mathbb{O}$ will satisfy Eq. 5 except in a few trivial cases for candidate wavelet signals. Thus, another idea introduced by Holschneider et al. is to relax the requirement of Eq. 5 to the approximation case. First, one fixes a tolerance $\varepsilon > 0$ and a maximum number of octaves N. Then Eq. 5 becomes

$$\|\mathbb{O}^n Ps - P(\mathfrak{D}_2)^n s\| < \varepsilon, \text{ for } 0 \leq n \leq N \ . \qquad (7)$$

The notation $\| \ \|$ denotes the norm in the space of sequences and is defined such that, for $\mathbf{g} = \{\ldots, g_{-2}, g_{-1}, g_0, g_1, g_2, \ldots\}$, $\|\mathbf{g}\| = (\sum_{n=-\infty}^{\infty} |g_n|^2)^{1/2}$. For a particular choice of wavelet signal $s(t)$ and interpolating filter F, and for explicit range N and tolerance ε, the condition in Eq. 7 can be numerically verified. This verification is performed as part of the selection of algorithm parameters. An operator $\mathbb{O}$ satisfying Eq. 7 is called a pseudo-dilation operator.

Holschneider et al.[6] showed that, for a selection of interpolating filter and a pseudo-dilation operator $\mathbb{O}$, as defined in Eq. 6, the filtering operation for the CWT presented in Eq. 4 can be factored into simple filtering operations. Moreover, these simple filters are recursively related to each other, thus facilitating an *algorithme à trous* approach to their implementation. The specific factoring is presented in the lemma below, whose proof is presented in the boxed insert (Proof of Lemma). Recall that the CWT filtering is by the sequence $\mathbf{g_a} = \mathbf{g_{2^n}}$. Moreover, $\mathbf{g_a} = P\tilde{\psi}_a$, and $\tilde{\psi}_a$ is defined in Eq. 3. Thus, the CWT requires a filtering by the sequence $\mathbf{g_{2^n}} = P\mathfrak{D}_{2^n}\tilde{\psi} = P(\mathfrak{D}_2)^n\tilde{\psi} \approx \mathbb{O}^n(P\tilde{\psi})$. For simplicity, let $\mathbf{f}$ denote the sequence $P\tilde{\psi}$. Then the CWT requires filtering by the sequence $\mathbb{O}^n\mathbf{f}$, that is, the performance of the convolution, $K_{\mathbb{O}^n\mathbf{f}}$.

LEMMA: Suppose that $\mathbf{f}$ is a sequence and that F is an interpolation filter defining the pseudo-dilation operator $\mathbb{O} = D_2 + ZD_2K_F$. Then the convolution operator $K_{\mathbb{O}^n\mathbf{f}}$ can be factored as

$$K_{\mathbb{O}^n\mathbf{f}} = \alpha^n K_{\mathbf{f}_n} K_{F_1} K_{F_2} \ldots K_{F_n} \ , \qquad (8)$$

where $\alpha = 1/\sqrt{2}$, and

$$\mathbf{f}_n = (\alpha^{-1}D_2)^n\mathbf{f} \ , \qquad (9)$$

EXAMPLES OF INTERPOLATION FILTERS

Here we consider the effect of $\mathbb{O} = D_2 + ZD_2K_F$ for various interpolation filters F. Only the nonzero elements of the impulse response for F will be shown.

First, suppose that the impulse response to F is $\{f_0 = 1\}$ (and all other samples are therefore zero). Then the convolution of a signal $\mathbf{s}$ with F produces $(KF\mathbf{s})_n = s_n$. Thus,

$$(\mathbb{O}_{\mathbf{s}})_n = \begin{cases} \dfrac{1}{\sqrt{2}} s_{n/2}, & \text{if } n \text{ is even} \\ \dfrac{1}{\sqrt{2}} s_{(n-1)/2}, & \text{if } n \text{ is odd} \ . \end{cases}$$

Thus, this simple filter merely stretches the signal and interleaves a repeat of each sample so that $\{\ldots, s_{-3}, s_{-2}, s_{-1}, s_0, s_1, s_2, s_3, \ldots\}$ becomes $\{\ldots, s_{-3}, s_{-3}, s_{-2}, s_{-2}, s_{-1}, s_{-1}, s_0, s_0, s_1, s_1, s_2, s_2, s_3, s_3, \ldots\}$, appropriately scaled by $2^{-1/2}$.

As a second example, consider the interpolation filter F whose impulse response is $\{f_{-1} = f_0 = 1/2\}$. One can compute at once that the operation of $\mathbb{O}$ on the signal $\mathbf{s}$ is

$$(\mathbb{O}_{\mathbf{s}})_n = \begin{cases} \dfrac{1}{\sqrt{2}} s_{n/2}, & \text{if } n \text{ is even} \\ \dfrac{1}{\sqrt{2}} \dfrac{(s_{(n-1)/2} + s_{(n+1)/2})}{2}, & \text{if } n \text{ is odd} \ . \end{cases}$$

Ignoring the scale change by $2^{-1/2}$ for the moment, the sample at time 0 is s_0 and the sample at time 2 is s_1. The sample at time 1 is $(s_0 + s_1)/2$. Continuing for all times, one sees that this filter is a straightforward linear interpolator.

Similarly, we can construct polynomial interpolators by considering the effects of samples beyond the two adjacent samples. A cubic interpolator, for example, uses the filter F with impulse response $\{f_{-2} = -1/16, f_{-1} = 9/16, f_0 = 9/16, f_1 = -1/16\}$. The interpolated value, in this case, between the samples $s_{n/2}$ and $s_{n/2+1}$ is obtained by fitting a cubic polynomial to the samples $s_{n/2-1}$, $s_{n/2}$, $s_{n/2+1}$, and $s_{n/2+2}$, and evaluating it at $n/2 + 1/2$.

PROOF OF LEMMA

We present here a proof of the lemma in the article. We wish to prove that

$$K_{\mathbb{O}^n\mathbf{f}} = \alpha^n K_{\mathbf{f}_n} K_{F_1} K_{F_2} \ldots K_{F_n} \ .$$

Recall that z-transforms change such operations into multiplications, and there is a one-to-one correspondence between sequences and their z-transforms. If we denote a z-transform of a sequence $\mathbf{s}$ by $\mathbf{s}(z)$, then it therefore suffices to prove that

$$(\mathbb{O}^n\mathbf{f})(z) = a^n\mathbf{f}_n(z)F_1(z)F_2(z) \ldots F_n(z).$$

We prove this by induction on n.

It is helpful to review the effects of the various operators on the z-tranforms. The delay operator Z, for example, has the effect of multiplying the z-transform of the sequence by z^{-1}. The convolution operator K_G has the effect of multiplying the z-transform of the sequence by $G(z)$. The dilation operator D_2 has the effect of a change of variable to the z-transform $\alpha\mathbf{s}(z^2)$. Thus, if the operator $\mathbb{O}$ is applied to the sequence $\mathbf{s}$, then the resulting sequence has z-transform, $\alpha\mathbf{s}(z^2) + \alpha z^{-1}F(z^2)\mathbf{s}(z^2)$.

For $n = 1$, the computation is straightforward. From the preceding paragraph, one has

$$(\mathbb{O}\mathbf{f})(z) = \alpha\mathbf{f}(z^2) + \alpha z^{-1}F(z^2)\mathbf{f}(z^2).$$

The z-transform of $\mathbf{f}_1$ is computed, from its definition, to be $\mathbf{f}(z^2)$, and that of F_1 is computed to be $1 + z^{-1}\mathrm{F}(z^2)$, using the fact that the z-transform of δ is 1. Thus, the convolution by $\alpha\mathbf{f}_1F_1$ is a multiplication by

$$\alpha[\mathbf{f}(z^2)][1 + z^{-1}F(z^2)]$$

in the z-transform domain. But this is exactly $(\mathbb{O}\mathbf{f})(z)$.

Now assume, via induction, that $n \geq 2$ and that

$$(\mathbb{O}^{n-1}\mathbf{f})(z) = \alpha^{n-1}\mathbf{f}(z^{2n-1})[1 + z^{-1}F(z^2)][1 + z^{-2}F(z^4)] \ldots [1 + z^{-2n-2}F(z^{2n-1})] \ .$$

Applying $\mathbb{O}$ one more time evaluates the above at z^2 rather than z and multiplies it by $\alpha[1 + z^{-1}F(z^2)]$. Carrying out these operations shows that the identity holds for n as well, completing the proof.

$$F_1 = \delta + T(\alpha^{-1}D_2)F \,, \qquad (10)$$

and

$$F_{i+1} = (\alpha^{-1}D_2)F_i, \text{ for } i = 1, \ldots, n-1 \,. \qquad (11)$$

In the lemma, δ is the impulse sequence $\{\ldots, d_{-1}, d_0, d_1, \ldots\}$ for which $d_0 = 1$ and $d_i = 0$ for $i \neq 0$. Note also that the convolution operators in the factoring of Eq. 8 commute, and so the order of the filtering implied is not important.

The lemma presents a simple method for deriving the filtering convolution for the kth octave in terms of the filtering convolution for the $(k-1)$ octave, because from Eq. 9 one has $\mathbf{f}_k = (\alpha^{-1}D_2)\mathbf{f}_{k-1}$, and this fact and Eq. 11 show that the convolutions for octave k can be computed from the convolutions from octave $k-1$. The boxed insert (Computation of $N+1$ Octaves) shows precisely how these octaves are computed; a data flow diagram of the computation is presented in Fig. 9. Here, arcs showing the flowdown of the $(\alpha^{-1}D_2)$ factor in the definitions of the filters F_i and f_i are indicated. The *algorithme à trous* is a method for incorporating this flowdown as a sequence of multiplexing operations.

We must first consider the relationship between convolution with an arbitrary filter H and convolution with the dilated filter $(\alpha^{-1}D_2)H$. Suppose the impulse response for H has nonzero taps $h_{k_0}, h_{k_{0+1}}, \ldots, h_{k_1}$, with $k_0 \leq 0 \leq k_1$. Although the filter H is not necessarily causal, its support is consistent with the idea of an interpolating filter. For example, the nth output, given sequence s as input, is

$$\text{Output}_n = h^*_{k_0} s_{n-k_0} + h^*_{k_{0+1}} s_{n-k_0-1} + \ldots + h^*_{k_1} s_{n-k_1} \,.$$

The output at time n is a function of future inputs s_{n+1} through $s_{n+|k_0|}$, current input s_n, and past inputs s_{n-1} through s_{n-k_1}. It interpolates past and future samples. If the convolution is performed with $(\alpha^{-1}D_2)H$, a direct calculation verifies that the output at time n is

$$\text{Output}_n = \sum_{k=k_0}^{k_1} h^*_k s_{n-2k} \,. \qquad (12)$$

The subscript on the s indicates an introduction of two delays on the input between successive filter taps. Thus, Fig. 10a illustrates a tapped delay line implementation of Eq. 12. Here, the output at time n requires "future" inputs to time $n + 2|k_0|$. Similarly, Fig. 10b illustrates the iterative application of the dilation factor to produce the output when filtering with $(\alpha^{-1}D_2)^j H$.

An alternative formulation of the architecture shown in Fig. 10 can be constructed using the multiplexer process and an interleaver process illustrated in Fig. 11. The multiplexer receives as input a sampled sequence and passes this input on an alternating basis onto two output sequences. The interleaver accepts two input sequences and interleaves them so as to construct a single output sequence. The implicit timing must be such that a multiplexer followed by an interleaver acts as an identity processor. If, for example, the multiplexer is alternately placing outputs on output lines A, B, A, B, . . ., the cycles with output on line A must be synchronous with interleaver inputs from line A, and similarly for line B. This assumes a pairing of multiplexers and interleavers.

Figure 12 can easily be derived as an alternative implementation of the filters $(\alpha^{-1}D_2)H$ and $(\alpha^{-1}D_2)^jH$ using multiplexers and interleavers. The multiple input interleaver in Fig. 12b is synchronized so that the constructed signal is correctly timed. This synchronization is somewhat tricky. For a three-stage ($j = 3$) implementation, for example, the interleaving is constructed, in order, from legs 1, 5, 3, 7, 2, 6, 4, and 8. In the re-sorting of

COMPUTATION OF N+1 OCTAVES OF THE CWT

Suppose $\mathbf{s}$ is the sampled input signal and $\mathbf{f} = P\tilde{\psi}$ for the wavelet ψ. The octaves of the CWT are computed as follows:

Octave 0:	Output $K_{\mathbf{f}}\mathbf{s}$
Octave 1:	Define $F_1 = \delta + T(\alpha^{-1}D_2)F$
	Define $\mathbf{f}_1 = (\alpha^{-1}D_2)\mathbf{f}$
	Compute $X_1 = \alpha K_{F_1}\mathbf{s}$
	Output $K_{\mathbf{f}_1}X_1$
For $k = 2, \ldots, n$	
Octave k:	Define $F_k = (\alpha^{-1}D_2)F_{k-1}$
	Define $\mathbf{f}_k = (\alpha^{-1}D_2)\mathbf{f}_{k-1}$
	Compute $X_k = \alpha K_{F_k}X_{k-1}$
	Output $K_{\mathbf{f}_k}X_k$

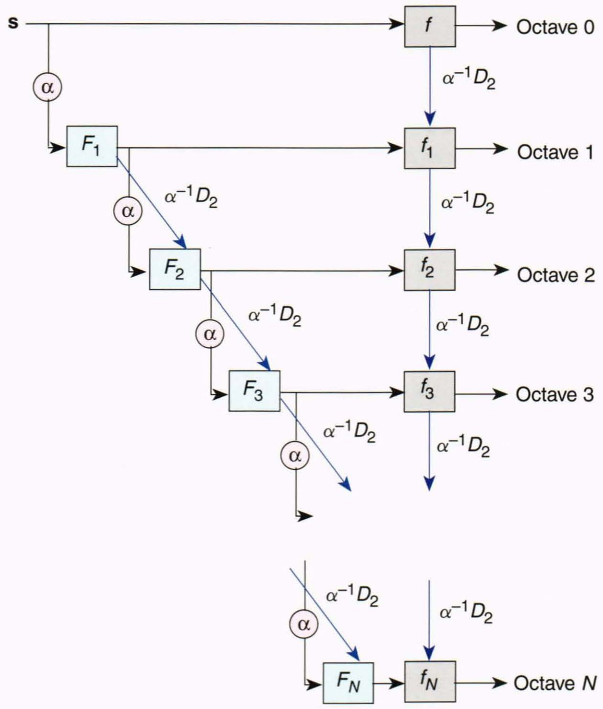


Figure 9. Data flow diagram of the CWT over $N + 1$ octaves. The blue arrows indicate modifications to construct a filter from a previous filter via the dilation operator. Rectangles represent filtering operations (convolutions) and circles represent amplification (multiplication).

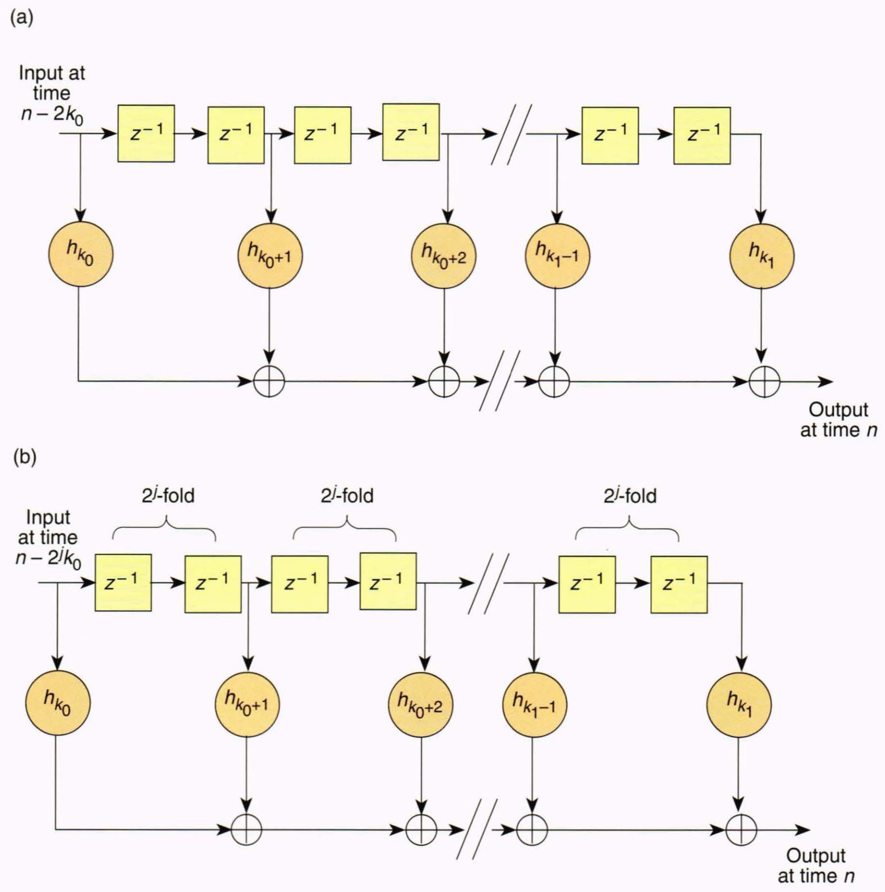


Figure 10. Tapped delay line implementation of the dilated H filter. The sample delay operator T is denoted by its z-transform z^{-1}. Weights are complex conjugates of the impulse response for H. (a) Effect of a single dilation operation on the filter H. (b) Effect of j iterations of the dilation operation on the filter H.

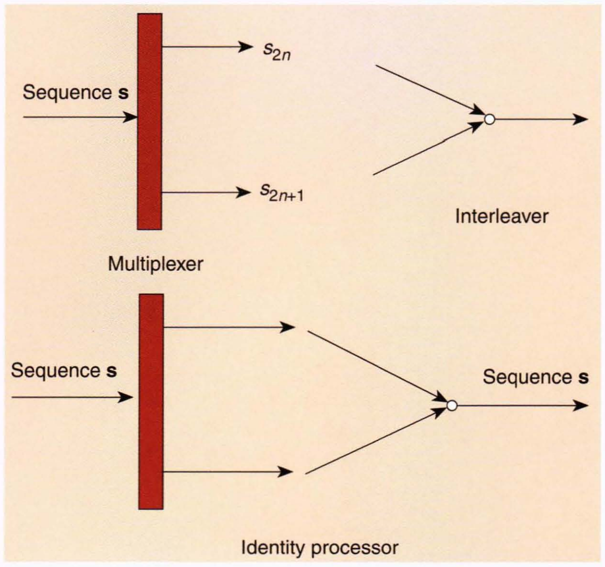


Figure 11. Definition of multiplexer and interleaver processors.

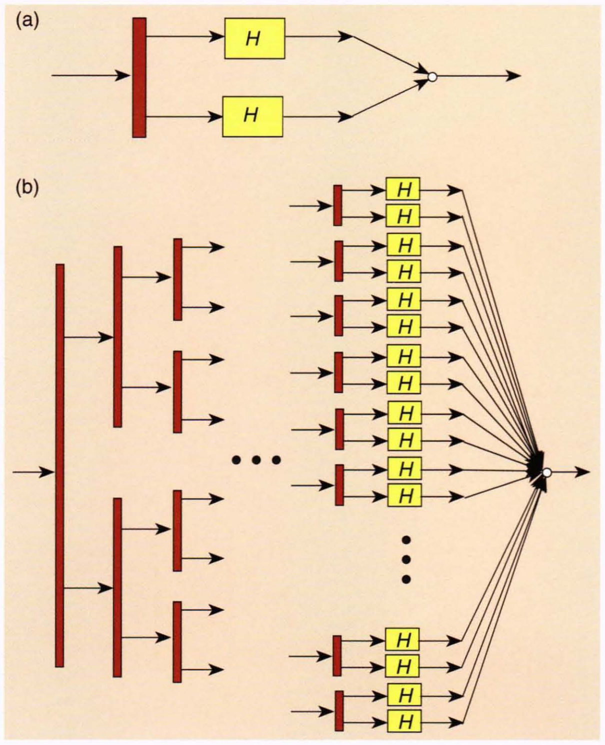


Figure 12. Implementation of a dilation of a filter using multiplexers and an interleaver. (a) Multiplexer/interleaver implementation of the dilated filter $(\alpha^{-1}D_2)H$. (b) Multiplexer/interleaver implementation of the dilated filter $(\alpha^{-1}D_2)^jH$.

the graph representation of the algorithm, described below, this ordering is automatically performed.

Now, to construct the CWT algorithm, we must replace the arbitrary filter H with specific filters. Equations 9 and 11 in the lemma show that the filters **f** and $F_1 = \delta + T(\alpha^{-1}D_2)F$ are iteratively dilated and so H in the preceding discussion will be replaced by these two filters. The effect will be an expansion of the vertical and diagonal flows of Fig. 9 using the multiplexer/interleaver diagrams of Figs. 10–12. This more complicated flow diagram can be sorted out into an elegant algorithm, as will be seen presently.

Consider next the specific computation of convolution with $F_1 = \delta + T\alpha^{-1}D_2F$. Assume that F has an impulse response whose nonzero terms are $w_{k_0}, w_{k_0+1}, \ldots, w_{k_1}$, with $k_0 \le 0 \le k_1$. It is a simple calculation to see that the impulse response to F_1 is

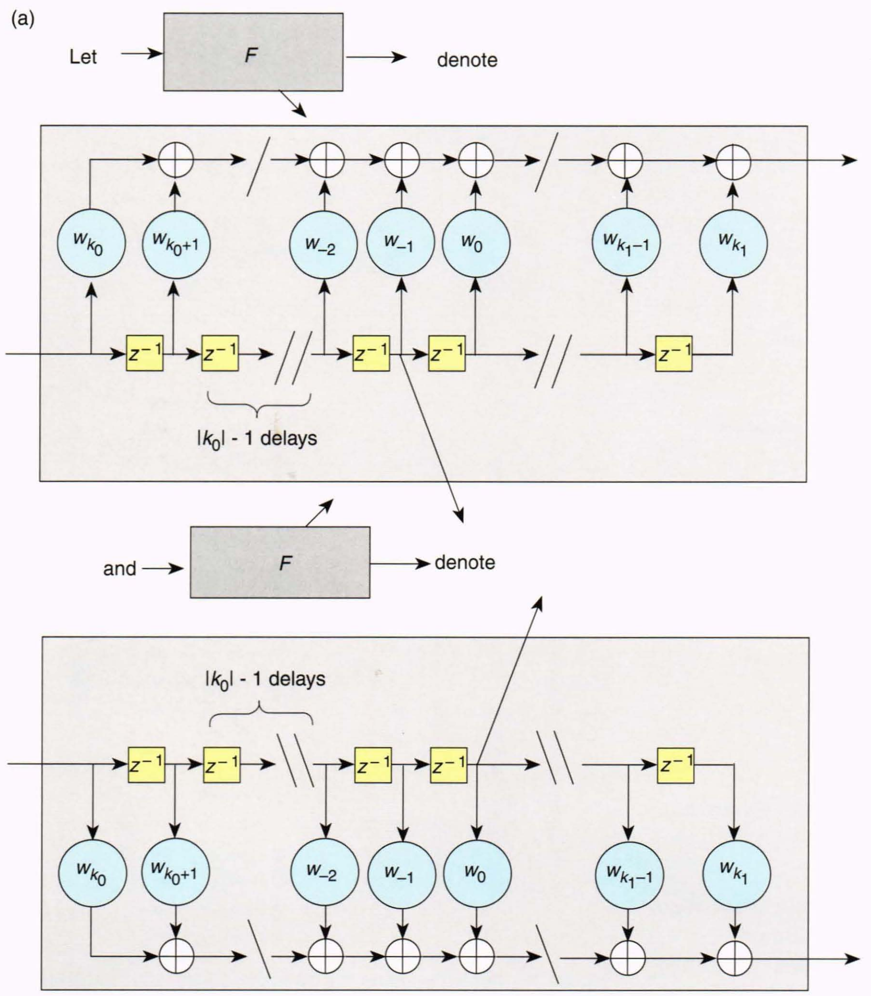
$$(F_1)_n = \begin{cases} 1, & \text{if } n = 0 \\ w_{(n-1)/2}, & \text{if } n \text{ is odd and } 2k_0 + 1 \le n \le 2k_0 + 1 \\ 0, & \text{otherwise}. \end{cases} \qquad (13)$$

Substituting Eq. 13 into the formula for the convolution, we compute that the output at time n of the filtering of sequence $\mathbf{s}$ by F_1 is

$$s_n + \sum_{k=k_0}^{k_1} w_k s_{n-2k-1}\,. \qquad (14)$$

The form of the summation in Eq. 14 is similar to the convolution in Eq. 12, except for the delay by one (time is $n - 1$ rather than n). It can therefore also be implemented with multiplexers and interleavers as before. Moreover, the additional term s_n is such that if n is odd, then the subscripts of the s term in the sum, $n - 2k - 1$, are all even and if n is even, then the subscripts are all odd. Thus, if we implement this form for F_1 using multiplexers and interleavers, then for each of the two legs out of the multiplexer in Fig. 12, there must be a tap in the opposite leg with which to sum. In other words, the top output of the multiplexer is summed with a specifically tapped register from the bottom leg, and the bottom output of the multiplexer is summed with a specifically tapped register from the top leg. Figure 13a presents a careful determination of where these two taps should be, and Fig. 13b presents an implementation flow graph of this convolution, using these tapped convolution processors.

Finally, we place the resulting processors into the original data flow architecture of Fig. 9. The graph is correct but rather complex. If we were to trace the paths from the input to the various octave outputs, recording the processors along the way, these paths would define



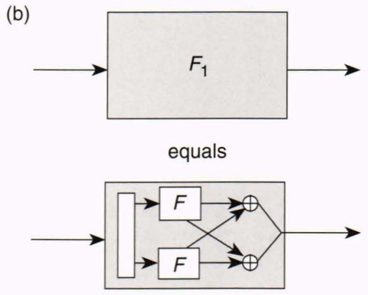


Figure 13. Data flow diagram of the F_1 filter. (a) Definitions of the upper and lower F processors. Note the different locations of the delay line taps in the two processors. (b) Diagram of the basic filter.

an invariant of the data flow graph for the algorithm—any other graph with a single input source and octave output sources that produces the same traces is equivalent to the algorithm. The primary method of re-sorting the algorithm graph assumes that, through the implicit timing of multiplexers and interleavers, a data flow constructed from a multiplexer followed by an interleaver followed by a second interleaver has the same effect as the first multiplexer alone. Using this method to reduce the complexity of the algorithm graph will appropriately incorporate the ordering of a multiple input interleaver.

In this way, we can construct a simpler data flow for the algorithm than that obtained in the way just described. In particular, this was performed by Holschneider et al.[6] and is illustrated in Fig. 14. If we define an elemental component (EC) to be a processor that performs as illustrated in Fig. 14a, then the data flow diagram of Fig. 14b, using ECs, produces the CWT with octave rows.

The algorithm described above produces rows of the CWT surface corresponding to scales at octaves, that is, scale values $a = 1, 2, 4, 8, \ldots$. The nice quality of the algorithm is that the wavelet need only be sampled at scale $a = 1$, and the interpolator and *algorithme à trous* take care of the octave jumps in scale. The remaining concern is how to handle the scales between successive octaves.

Because scale is a multiplicative rather than an additive parameter, a standard method for introducing levels between octaves is by the introduction of voices. Voices are defined to be the scale levels between successive octaves, uniformly distributed in a multiplicative sense. Thus, the ratio between two successive voices is constant. For example, if one wishes to have 10 voices per octave, then the ratio between successive voices is $2^{1/10}$. The distance between two levels 10 voices apart is an octave.

Suppose now that one wants to display a CWT surface covering N octaves with L voices per octave. There are NL rows in this surface, which we can number from row 0 to row $NL - 1$. If we were to consider row k, for example, then by dividing k by L, say, $k = qL + r$ (q = quotient and r = remainder), and we find that row k corresponds to voice r of octave q. The scale associated with row k is $a = 2^{k/L} = 2^{(qL+r)/L} = 2^q 2^{r/L}$. This, in turn, corresponds to octave q of the CWT that begins with a sampling of the mother wavelet

$$(2^{r/L})^{-1/2}\psi^*(-2^{-r/L}t)\,. \qquad (15)$$

Thus, by executing the algorithm L times, for voice level $r = 0, 1, \ldots, L-1$ of the range of octaves, using the sampling of the corresponding wavelet given by Eq. 15 for the appropriate voice, one can construct the CWT surface with the intermediate values filled in.

An algorithm with additive spacing of scales can also be constructed by modifying the lemma in the preceding derivation. Holshneider et al.[6] outline this approach.

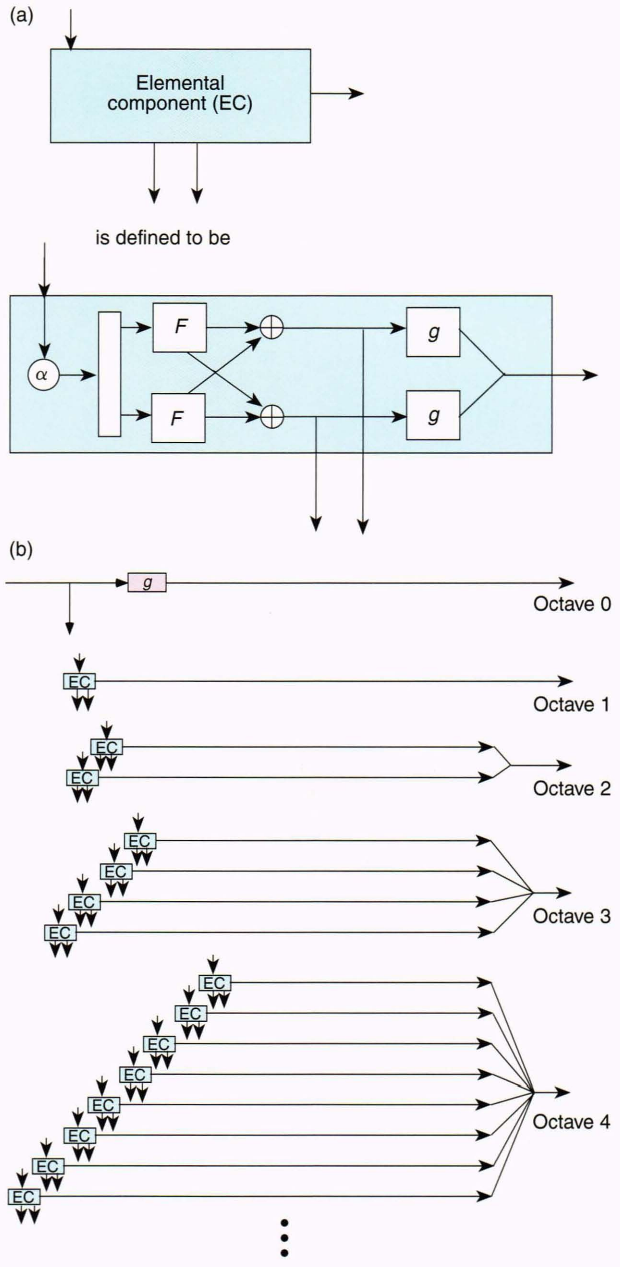


Figure 14. The graphical representation of the simplified CWT algorithm. (a) Diagram of the elemental component (EC). (b) Data flow diagram connecting ECs.[6]

FINAL COMMENT

Software, written in generic C, has been designed and is available for investigation of CWT representation of data. This is prototype source code for generating the CWT over a specified number of octaves with a specified number of voices per octave, and for generating a raw image file from the complex data in the forms discussed in this article. The author will send the source code via e-mail to interested researchers who request it from js@aplcomm.jhuapl.edu.

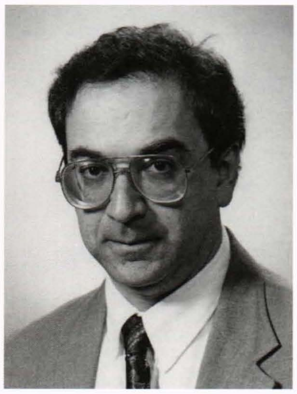
REFERENCES

[1] Meyer, Y., *Wavelets: Algorithms and Applications*, SIAM Press, Philadelphia (1993).
[2] Daubechies, I., *Ten Lectures on Wavelets*, SIAM Press, Philadelphia (1992).
[3] Meyer, Y., *Ondelettes* (Vol. 1 of the series, *Ondelettes et Operateurs*), Hermann, Paris (1990) (now translated as *Wavelets and Operators*, Cambridge University Press, New York [1993]).
[4] Chui, C. K., *An Introduction to Wavelets*, Wiley and Sons, New York (1992).

[5] Grossmann, A., Kronland-Martinet, R., and Morlet, J., "Reading and Understanding Continuous Wavelet Transforms," in *Wavelets: Time-Frequency Methods and Phase Space*, J. Combes, A. Grossmann, and Ph. Tchamitchian (eds.), Springer Verlag, New York, pp. 2–21 (1989).

[6] Holschneider, M., Kronland-Martinet, R., Morlet, J., and Tchamitchian, Ph., "A Real-Time Algorithm for Signal Analysis with the Help of the Wavelet Transform," in *Wavelets: Time-Frequency Methods and Phase Space*, J. Combes, A. Grossmann, and Ph. Tchamitchian (eds.), Springer Verlag, New York, pp. 286–297 (1989).

THE AUTHOR



JOHN SADOWSKY received a B.S. in mathematics from The Johns Hopkins University and a Ph.D. in mathematics with a speciality in number theory from the University of Maryland. Before joining APL, he worked as a supervisor and Principal Scientist for the Systems Engineering and Development Corporation and as a mathematician for the Social Security Administration and the Census Bureau. He joined APL's Research Center in 1989 and is currently the Assistant Supervisor of the Mathematics and Information Science Group. In addition, since 1981 he has served on the adjunct faculty for the Computer Science Program of the G.W.C. Whiting School of Engineering. Dr. Sadowsky's current interests include signal processing, applied algebra and number theory, computational complexity, the design of algorithms, and sensory engineering.