Wavelets: A New Tool for Signal Analysis

1-3 Fourier Analysis
1-4 Short-Time Fourier Analysis
1-5 Wavelet Analysis
1-7 What is Wavelet Analysis?
1-8 The Continuous Wavelet Transform
1-16 The Discrete Wavelet Transform
1-20 Wavelet Reconstruction
1-27 Wavelet Packet Analysis
1-29 History of Wavelets
1-30 An Introduction to the Wavelet Families
Everywhere around us are signals that need to be analyzed. Seismic tremors, human speech, engine vibrations, medical images, financial data, music, and many other types of signals have to be efficiently encoded, compressed, cleaned up, reconstructed, described, simplified, modeled, distinguished, or located.

Wavelet analysis is a new and promising set of tools and techniques for doing this.
Fourier Analysis

Signal analysts already have at their disposal an impressive arsenal of tools. Perhaps the most well-known of these is Fourier analysis, which breaks down a signal into constituent sinusoids of different frequencies. Another way to think of Fourier analysis is as a mathematical technique for transforming our view of the signal from a time-based one to a frequency-based one.

For many signals, Fourier analysis is extremely useful because the signal’s frequency content is of great importance. So why do we need other techniques, like wavelet analysis?

Fourier analysis has a serious drawback. In transforming to the frequency domain, time information is lost. When looking at a Fourier transform of a signal, it is impossible to tell when a particular event took place.

If a signal doesn't change much over time — that is, if it is what is called a stationary signal — this drawback isn't very important. However, most interesting signals contain numerous non-stationary or transitory characteristics: drift, trends, abrupt changes, and beginnings and ends of events. These characteristics are often the most important part of the signal, and Fourier analysis is not suited to detecting them.
Short-Time Fourier Analysis

In an effort to correct this deficiency, Dennis Gabor (1946) adapted the Fourier transform to analyze only a small section of the signal at a time — a technique called windowing the signal. Gabor's adaptation, called the Short-Time Fourier Transform (STFT), maps a signal into a two-dimensional function of time and frequency.

The STFT represents a sort of compromise between the time- and frequency-based views of a signal. It provides some information about both when and at what frequencies a signal event occurs. However, you can only obtain this information with limited precision, and that precision is determined by the size of the window.

While the STFT's compromise between time and frequency information can be useful, the drawback is that once you choose a particular size for the time window, that window is the same for all frequencies. Many signals require a more flexible approach — one where we can vary the window size to determine more accurately either time or frequency.
Wavelet Analysis

Wavelet analysis represents the next logical step: a windowing technique with variable-sized regions. Wavelet analysis allows the use of long time intervals where we want more precise low frequency information, and shorter regions where we want high frequency information.

Here’s what this looks like in contrast with the time-based, frequency-based, and STFT views of a signal:

You may have noticed that wavelet analysis does not use a time-frequency region, but rather a time-scale region. We’ll have more to say about the concept of scale later.

What Can Wavelet Analysis Do?

One major advantage afforded by wavelets is the ability to perform local analysis — that is, to analyze a localized area of a larger signal.

Consider a sinusoidal signal with a small discontinuity — one so tiny as to be barely visible. Such a signal easily could be generated in the real world, perhaps by a power fluctuation or a noisy switch.
Wavelets: A New Tool for Signal Analysis

A plot of the Fourier coefficients (as provided by the \texttt{fft} command) of this signal shows nothing particularly interesting: a flat spectrum with two peaks representing a single frequency. However, a plot of wavelet coefficients clearly shows the exact location in time of the discontinuity.

Wavelet analysis is capable of revealing aspects of data that other signal analysis techniques miss, aspects like trends, breakdown points, discontinuities in higher derivatives, and self-similarity. Further, because it affords a different view of data than those presented by traditional techniques, wavelet analysis can often compress or de-noise a signal without appreciable degradation.

Indeed, in their brief history within the signal processing field, wavelets have already proven themselves to be an indispensable addition to the analyst's collection of tools and continue to enjoy a burgeoning popularity today.
What is Wavelet Analysis?

Now that we know some situations when wavelet analysis is useful, it is worthwhile asking the questions “What is wavelet analysis?” and even more fundamentally, “What is a wavelet?”

A wavelet is a waveform of effectively limited duration that has an average value of zero.

Compare wavelets with sine waves, which are the basis of Fourier analysis. Sinusoids do not have limited duration — they extend from minus to plus infinity. And where sinusoids are smooth and predictable, wavelets tend to be irregular and asymmetric.

![Sine Wave](image1.png) ![Wavelet (db10)](image2.png)

Fourier analysis consists of breaking up a signal into sine waves of various frequencies. Similarly, wavelet analysis is the breaking up of a signal into shifted and scaled versions of the original (or mother) wavelet.

Just looking at pictures of wavelets and sine waves, you can see intuitively that signals with sharp changes might be better analyzed with an irregular wavelet than with a smooth sinusoid, just as some foods are better handled with a fork than a spoon.

It also makes sense that local features can be described better with wavelets, which have local extent.

**Number of Dimensions**

Thus far, we’ve discussed only one-dimensional data, which encompasses most ordinary signals. However, wavelet analysis can be applied to two-dimensional data — images; and, in principle, to higher-dimensional data.

This toolbox uses only one- and two-dimensional analysis techniques.
The Continuous Wavelet Transform

Mathematically, the process of Fourier analysis is represented by the Fourier transform:

\[ F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt, \]

which is the sum over all time of the signal \( f(t) \) multiplied by a complex exponential. (Recall that a complex exponential can be broken down into real and imaginary sinusoidal components.)

The results of the transform are the Fourier coefficients \( F(\omega) \), which when multiplied by a sinusoid of appropriate frequency \( \omega \), yield the constituent sinusoidal components of the original signal. Graphically, the process looks like:

Similarly, the continuous wavelet transform (CWT) is defined as the sum over all time of the signal multiplied by scaled, shifted versions of the wavelet function \( \psi \):

\[ C(\text{scale}, \text{position}) = \int_{-\infty}^{\infty} f(t) \psi(\text{scale}, \text{position}, t) dt \]

The result of the CWT are many wavelet coefficients \( C \), which are a function of scale and position.
Multiplying each coefficient by the appropriately scaled and shifted wavelet yields the constituent wavelets of the original signal:

![Wavelet Transform Diagram]

**Scaling**

We've already alluded to the fact that wavelet analysis produces a time-scale view of a signal, and now we're talking about scaling and shifting wavelets. What exactly do we mean by scale in this context?

Scaling a wavelet simply means stretching (or compressing) it.

To go beyond colloquial descriptions such as “stretching,” we introduce the scale factor, often denoted by the letter \( a \). If we're talking about sinusoids, for example, the effect of the scale factor is very easy to see:

\[
\begin{align*}
  f(t) &= \sin(t) \quad ; \quad a = 1 \\
  f(t) &= \sin(2t) \quad ; \quad a = \frac{1}{2} \\
  f(t) &= \sin(4t) \quad ; \quad a = \frac{1}{4}
\end{align*}
\]
Wavelets: A New Tool for Signal Analysis

The scale factor works exactly the same with wavelets. The smaller the scale factor, the more “compressed” the wavelet.

\[
\begin{align*}
    f(t) &= \psi(t) \quad ; \quad a = 1 \\
    f(t) &= \psi(2t) \quad ; \quad a = \frac{1}{2} \\
    f(t) &= \psi(4t) \quad ; \quad a = \frac{1}{4}
\end{align*}
\]

It is clear from the diagrams that, for a sinusoid \( \sin(\omega t) \), the scale factor \( a \) is related (inversely) to the radian frequency \( \omega \). Similarly, with wavelet analysis, the scale is related to the frequency of the signal. We’ll return to this topic later.

**Shifting**

Shifting a wavelet simply means delaying (or hastening) its onset. Mathematically, delaying a function \( f(t) \) by \( k \) is represented by \( f(t - k) \):

\[
\begin{align*}
    \text{Wavelet function } \psi(t) &\quad \rightarrow \quad \text{Shifted wavelet function } \psi(t - k)
\end{align*}
\]

**Five Easy Steps to a Continuous Wavelet Transform**

The continuous wavelet transform is the sum over all time of the signal multiplied by scaled, shifted versions of the wavelet. This process produces wavelet coefficients that are a function of scale and position.
It's really a very simple process. In fact, here are the five steps of an easy recipe for creating a CWT:

1. Take a wavelet and compare it to a section at the start of the original signal.

2. Calculate a number, $C$, that represents how closely correlated the wavelet is with this section of the signal. The higher $C$ is, the more the similarity. Note that the results will depend on the shape of the wavelet you choose.

\[
\begin{array}{c}
\text{Signal} \\
\text{Wavelet} \\
C = 0.0102
\end{array}
\]

3. Shift the wavelet to the right and repeat steps 1 and 2 until you’ve covered the whole signal.

\[
\begin{array}{c}
\text{Signal} \\
\text{Wavelet} \\
C = 0.2247
\end{array}
\]

4. Scale (stretch) the wavelet and repeat steps 1 through 3.

5. Repeat steps 1 through 4 for all scales.
When you're done, you'll have the coefficients produced at different scales by different sections of the signal. The coefficients constitute the results of a regression of the original signal performed on the wavelets.

How to make sense of all these coefficients? You could make a plot on which the $x$-axis represents position along the signal (time), the $y$-axis represents scale, and the color at each $x$-$y$ point represents the magnitude of the wavelet coefficient $C$. These are the coefficient plots generated by the graphical tools.

These coefficient plots resemble a bumpy surface viewed from above. If you could look at the same surface from the side, you might see something like this:

The continuous wavelet transform coefficient plots are precisely the time-scale view of the signal we referred to earlier. It is a different view of signal data than the time-frequency Fourier view, but it is not unrelated.
Scale and Frequency

Notice that the scales in the coefficients plot (shown as y-axis labels) run from 1 to 31. Recall that the higher scales correspond to the most “stretched” wavelets. The more stretched the wavelet, the longer the portion of the signal with which it is being compared, and thus the coarser the signal features being measured by the wavelet coefficients.

Thus, there is a correspondence between wavelet scales and frequency as revealed by wavelet analysis:

• Low scale \( a \) ⇒ Compressed wavelet ⇒ Rapidly changing details ⇒ High frequency \( \omega \).
• High scale \( a \) ⇒ Stretched wavelet ⇒ Slowly changing, coarse features ⇒ Low frequency \( \omega \).

The Scale of Nature

It’s important to understand that the fact that wavelet analysis does not produce a time-frequency view of a signal is not a weakness but a strength of the technique.

Not only is time-scale a different way to view data, it is a very natural way to view data deriving from a great number of natural phenomena.
Consider a lunar landscape, whose ragged surface (simulated below) is a result of centuries of bombardment by meteorites whose sizes range from gigantic boulders to dust specks.

If we think of this surface in cross-section as a one-dimensional signal, then it is reasonable to think of the signal as having components of different scales — large features carved by the impacts of large meteorites, and finer features abraded by small meteorites.

Here is a case where thinking in terms of scale makes much more sense than thinking in terms of frequency. Inspection of the CWT coefficients plot for this signal reveals patterns among scales and shows the signal’s possibly fractal nature.

Even though this signal is artificial, many natural phenomena — from the intricate branching of blood vessels and trees, to the jagged surfaces of mountains and fractured metals — lend themselves to an analysis of scale.
What’s Continuous About the Continuous Wavelet Transform?

Any signal processing performed on a computer using real-world data must be performed on a discrete signal — that is, on a signal that has been measured at discrete time intervals. It is important to remember that the continuous wavelet transform is also operating in discrete time. So what exactly is “continuous” about it?

What’s “continuous” about the CWT, and what distinguishes it from the discrete wavelet transform (to be discussed in the following section), are the scales at which it operates.

Unlike the discrete wavelet transform, the CWT can operate at every scale, from that of the original signal up to some maximum scale which you determine by trading off your need for detailed analysis with available computational horsepower.

The CWT is also continuous in terms of shifting: during computation, the analyzing wavelet is shifted smoothly over the full domain of the analyzed function.
The Discrete Wavelet Transform

Calculating wavelet coefficients at every possible scale is a fair amount of work, and it generates an awful lot of data. What if we choose only a subset of scales and positions at which to make our calculations?

It turns out, rather remarkably, that if we choose scales and positions based on powers of two — so-called dyadic scales and positions — then our analysis will be much more efficient and just as accurate. We obtain just such an analysis from the discrete wavelet transform (DWT).

An efficient way to implement this scheme using filters was developed in 1988 by Mallat (see [Mal89]). The Mallat algorithm is in fact a classical scheme known in the signal processing community as a two-channel subband coder (see p. 1 of the book Wavelets and Filter Banks, by Strang and Nguyen).

This very practical filtering algorithm yields a fast wavelet transform — a box into which a signal passes, and out of which wavelet coefficients quickly emerge. Let's examine this in more depth.

One-Stage Filtering: Approximations and Details

For many signals, the low-frequency content is the most important part. It is what gives the signal its identity. The high-frequency content, on the other hand, imparts flavor or nuance. Consider the human voice. If you remove the high-frequency components, the voice sounds different, but you can still tell what's being said. However, if you remove enough of the low-frequency components, you hear gibberish.

It is for this reason that, in wavelet analysis, we often speak of approximations and details.
The approximations are the high-scale, low-frequency components of the signal. The details are the low-scale, high-frequency components. The filtering process, at its most basic level, looks like this:

The original signal, $S$, passes through two complementary filters and emerges as two signals.

Unfortunately, if we actually perform this operation on a real digital signal, we wind up with twice as much data as we started with. Suppose, for instance, that the original signal $S$ consists of 1000 samples of data. Then the approximation and the detail will each have 1000 samples, for a total of 2000.

To correct this problem, we introduce the notion of downsampling. This simply means throwing away every second data point. While doing this introduces aliasing (a type of error, see p. 91 of the book Wavelets and Filter Banks, by Strang and Nguyen) in the signal components, it turns out we can account for this later on in the process.

The process on the right, which includes downsampling, produces DWT coefficients.

To gain a better appreciation of this process, let’s perform a one-stage discrete wavelet transform of a signal. Our signal will be a pure sinusoid with high-frequency noise added to it.
Here is our schematic diagram with real signals inserted into it:

![Schematic Diagram](image)

The MATLAB code needed to generate $s$, $c_D$, and $c_A$ is:

```matlab
s = sin(20.*linspace(0, pi, 1000)) + 0.5.*rand(1,1000);
[cA, cD] = dwt(s,'db2');
```

where `db2` is the name of the wavelet we want to use for the analysis.

Notice that the detail coefficients $c_D$ consist mainly of the high-frequency noise, while the approximation coefficients $c_A$ contain much less noise than does the original signal.

```matlab
» [length(cA) length(cD)]
an s = 501 501
```

You may observe that the actual lengths of the detail and approximation coefficient vectors are slightly more than half the length of the original signal. This has to do with the filtering process, which is implemented by convolving the signal with a filter. The convolution “smears” the signal, introducing several extra samples into the result.
Multiple-Level Decomposition

The decomposition process can be iterated, with successive approximations being decomposed in turn, so that one signal is broken down into many lower-resolution components. This is called the wavelet decomposition tree.

Looking at a signal’s wavelet decomposition tree can yield valuable information.

Number of Levels

Since the analysis process is iterative, in theory it can be continued indefinitely. In reality, the decomposition can proceed only until the individual details consist of a single sample or pixel. In practice, you’ll select a suitable number of levels based on the nature of the signal, or on a suitable criterion such as entropy (see Chapter 6 for details).
Wavelet Reconstruction

We've learned how the discrete wavelet transform can be used to analyze, or decompose, signals and images. The other half of the story is how those components can be assembled back into the original signal with no loss of information. This process is called reconstruction, or synthesis. The mathematical manipulation that effects synthesis is called the inverse discrete wavelet transform (IDWT).

To synthesize a signal in our toolbox, we reconstruct it from the wavelet coefficients:

Where wavelet analysis involves filtering and downsampling, the wavelet reconstruction process consists of upsampling and filtering. Upsampling is the process of lengthening a signal component by inserting zeros between samples:

The wavelet toolbox includes commands, like `idwt` and `waverec`, that perform one-level or multi-level reconstruction, respectively, on the components of one-dimensional signals. These commands have their two-dimensional analogues, `idwt2` and `waverec2`.
Reconstruction Filters
The filtering part of the reconstruction process also bears some discussion, because it is the choice of filters that is crucial in achieving perfect reconstruction of the original signal.

That perfect reconstruction is even possible is noteworthy. Recall that the downsampling of the signal components performed during the decomposition phase introduces a distortion called aliasing. It turns out that by carefully choosing filters for the decomposition and reconstruction phases that are closely related (but not identical), we can “cancel out” the effects of aliasing. This was the breakthrough made possible by the work of Ingrid Daubechies.

A technical discussion of how to design these filters can be found in p. 347 of the book Wavelets and Filter Banks, by Strang and Nguyen. The low- and highpass decomposition filters (L and H), together with their associated reconstruction filters (L’ and H’), form a system of what are called quadrature mirror filters:

Reconstructing Approximations and Details
We have seen that it is possible to reconstruct our original signal from the coefficients of the approximations and details.
It is also possible to reconstruct the approximations and details themselves from their coefficient vectors. As an example, let’s consider how we would reconstruct the first-level approximation $A_1$ from the coefficient vector $cA_1$.

We pass the coefficient vector $cA_1$ through the same process we used to reconstruct the original signal. However, instead of combining it with the level-one detail $cD_1$, we feed in a vector of zeros in place of the details:

The process yields a reconstructed approximation $A_1$, which has the same length as the original signal $S$ and which is a real approximation of it.

Similarly, we can reconstruct the first-level detail $D_1$, using the analogous process:

The reconstructed details and approximations are true constituents of the original signal. In fact, we find when we combine them that:

$$A_1 + D_1 = S$$

Note that the coefficient vectors $cA_1$ and $cD_1$ — because they were produced by downsampling, contain aliasing distortion, and are only half the length of the original signal — cannot directly be combined to reproduce the signal. It is necessary to reconstruct the approximations and details before combining them.
Extending this technique to the components of a multi-level analysis, we find that similar relationships hold for all the reconstructed signal constituents. That is, there are several ways to reassemble the original signal:

\[
S = A_1 + D_1 = A_2 + D_2 + D_1 = A_3 + D_3 + D_2 + D_1
\]

**Relationship of Filters to Wavelet Shapes**

In the section “Reconstruction Filters” on page 1-21, we spoke of the importance of choosing the right filters. In fact, the choice of filters not only determines whether perfect reconstruction is possible, it also determines the shape of the wavelet we use to perform the analysis.

In fact, to construct a wavelet of some practical utility, you seldom start by drawing a waveform. Instead, it usually makes more sense to design the appropriate quadrature mirror filters and then use them to create the waveform. Let’s see how this is done by focusing on an example.

Consider the lowpass reconstruction filter (L') for the db2 wavelet.

The filter coefficients can be obtained from the dbaux command:

```matlab
» Lprime = dbaux(2)
Lprime =
 0.3415  0.5915  0.1585  -0.0915
```
If we reverse the order of this vector (see `wrev`), and then multiply every second sample by \(-1\), we obtain the highpass filter \(H'\):

\[
H' = \\
-0.0915 & -0.1585 & 0.5915 & -0.3415
\]

Next, upsample \(H'\) by two (see `dyadup`), inserting zeros in alternate positions:

\[
H_U = \\
-0.0915 & 0 & -0.1585 & 0 & 0.5915 & 0 & -0.3415 & 0
\]

Finally, convolve the upsampled vector with the original lowpass filter:

\[
H_2 = \text{conv}(H_U, L')
\]

If we iterate this process several more times, repeatedly upsampling and convolving the resultant vector with the four-element filter vector \(L'\), a pattern begins to emerge:
The curve begins to look progressively more like the db2 wavelet. This means that the wavelet’s shape is determined entirely by the coefficients of the reconstruction filters.

This relationship has profound implications. It means that you cannot choose just any shape, call it a wavelet, and perform an analysis. At least, you can’t choose an arbitrary wavelet waveform if you want to be able to reconstruct the original signal accurately. You are compelled to choose a shape determined by quadrature mirror decomposition filters.

The Scaling Function
We’ve seen the interrelation of wavelets and quadrature mirror filters. The wavelet function \( \psi \) is determined by the highpass filter, which also produces the details of the wavelet decomposition.

There is an additional function associated with some but not all wavelets. This is the so-called scaling function, \( \phi \). The scaling function is very similar to the wavelet function. It is determined by the lowpass quadrature mirror filters, and thus is associated with the approximations of the wavelet decomposition.

In the same way that iteratively upsampling and convolving the highpass filter produces a shape approximating the wavelet function, iteratively upsampling and convolving the lowpass filter produces a shape approximating the scaling function.

Multistep Decomposition and Reconstruction
A multistep analysis-synthesis process can be represented as:
This process involves three aspects: breaking up a signal to obtain the wavelet coefficients, modifying the wavelet coefficients, and reassembling the signal from the coefficients.

We've already discussed decomposition and reconstruction at some length. Of course, there is no point breaking up a signal merely to have the satisfaction of immediately reconstructing it. We perform wavelet analysis because the coefficients thus obtained have many known uses, de-noising and compression being foremost among them.

But wavelet analysis is still a new and emerging field. Many uncharted uses of the wavelet coefficients no doubt lie in wait. The Wavelet Toolbox can be a means of exploring possible uses and hitherto unknown applications of wavelet analysis. Explore the toolbox functions and see what you discover.
Wavelet Packet Analysis

The wavelet packet method is a generalization of wavelet decomposition that offers a richer range of possibilities for signal analysis.

In wavelet analysis, a signal is split into an approximation and a detail. The approximation is then itself split into a second-level approximation and detail, and the process is repeated. For an $n$-level decomposition, there are $n+1$ possible ways to decompose or encode the signal.

$$S = A_1 + D_1$$
$$= A_2 + D_2 + D_1$$
$$= A_3 + D_3 + D_2 + D_1$$

In wavelet packet analysis, the details as well as the approximations can be split. This yields $2^n$ different ways to encode the signal. This is the wavelet packet decomposition tree:

For instance, wavelet packet analysis allows the signal $S$ to be represented as $A1 + AAD3 + DAD3 + DD2$. This is an example of a representation that is not possible with ordinary wavelet analysis.
Choosing one out of all these possible encodings presents an interesting problem. In this toolbox, we use an entropy-based criterion to select the most suitable decomposition of a given signal. This means we look at each node of the decomposition tree and quantify the information to be gained by performing each split.

Simple and efficient algorithms exist for both wavelet packet decomposition and optimal decomposition selection. This toolbox uses an adaptive filtering algorithm, based on work by Coifman and Wickerhauser, with direct applications in optimal signal coding and data compression.

Such algorithms allow the **Wavelet Packet 1-D** and **Wavelet Packet 2-D** tools to include “Best Level” and “Best Tree” features that optimize the decomposition both globally and with respect to each node.
History of Wavelets

From an historical point of view, wavelet analysis is a new method, though its mathematical underpinnings date back to the work of Joseph Fourier in the nineteenth century. Fourier laid the foundations with his theories of frequency analysis, which proved to be enormously important and influential.

The attention of researchers gradually turned from frequency-based analysis to scale-based analysis when it started to become clear that an approach measuring average fluctuations at different scales might prove less sensitive to noise.

The first recorded mention of the term “wavelet” was in 1909, in a thesis by Alfred Haar.

The concept of wavelets in its present theoretical form was first proposed by Jean Morlet and the team at the Marseille Theoretical Physics Center working under Alex Grossmann in France.

The methods of wavelet analysis have been developed mainly by Y. Meyer and his colleagues, who have ensured the methods’ dissemination. The main algorithm dates back to the work of Stephane Mallat in 1988. Since then, research on wavelets has become international. Such research is particularly active in the United States, where it is spearheaded by the work of scientists such as Ingrid Daubechies, Ronald Coifman, and Victor Wickerhauser.
An Introduction to the Wavelet Families

Several families of wavelets that have proven to be especially useful are included in this toolbox. What follows is an introduction to these wavelet families. To explore wavelet families on your own, check out the Wavelet Display tool:

1. Type wavemenu from the MATLAB command line. The Wavelet Toolbox Main Menu appears.

2. Click on the Wavelet Display menu item. The Wavelet Display tool appears.

3. Select a family from the Wavelet menu at the top right of the tool.

4. Click the Display button. Pictures of the wavelets and their associated filters appear.

5. Obtain more information by clicking on the information buttons located at the right.
An Introduction to the Wavelet Families

**Haar**

Any discussion of wavelets begins with Haar, the first and simplest. Haar is discontinuous, and resembles a step function. It represents the same wavelet as Daubechies db1. See "Haar" on page 64 for more detail.

![Wavelet function ψ](image)

**Daubechies**

Ingrid Daubechies, one of the brightest stars in the world of wavelet research, invented what are called compactly-supported orthonormal wavelets — thus making discrete wavelet analysis practicable.

The names of the Daubechies family wavelets are written dbN, where N is the order, and db the “surname” of the wavelet. The db1 wavelet, as mentioned above, is the same as Haar. Here are the next nine members of the family:

![Wavelet functions](image)

You can obtain a survey of the main properties of this family by typing `waveinfo('db')` from the MATLAB command line. See “Daubechies Wavelets: dbN” on page 6-63 for more detail.
Biorthogonal

This family of wavelets exhibits the property of linear phase, which is needed for signal and image reconstruction. By using two wavelets, one for decomposition and the other for reconstruction instead of the same single one, interesting properties are derived.
An Introduction to the Wavelet Families

You can obtain a survey of the main properties of this family by typing `waveinfo('bior')` from the MATLAB command line. See “Biorthogonal Wavelet Pairs: biorNr.Nd” on page 6-67 for more detail.

**Coiflets**

Built by I. Daubechies at the request of R. Coifman. The wavelet function has $2^N$ moments equal to 0 and the scaling function has $2^N-1$ moments equal to 0. The two functions have a support of length $6^N-1$. You can obtain a survey of the main properties of this family by typing `waveinfo('coif')` from the MATLAB command line. See “Coiflet Wavelets: coifN” on page 6-66 for more detail.

![Coiflet Wavelets: coifN](image)

**Symlets**

The symlets are nearly symmetrical wavelets proposed by Daubechies as modifications to the `db` family. The properties of the two wavelet families are similar.

![Symlet Wavelets: symN](image)

You can obtain a survey of the main properties of this family by typing `waveinfo('sym')` from the MATLAB command line. See “Symlet Wavelets: symN” on page 6-65 for more detail.
Morlet

This wavelet has no scaling function, but is explicit.

You can obtain a survey of the main properties of this family by typing `waveinfo('morl')` from the MATLAB command line. See “Morlet Wavelet: morl” on page 6-72 for more detail.

Mexican Hat

This wavelet has no scaling function and is derived from a function that is proportional to the second derivative function of the Gaussian probability density function.

You can obtain a survey of the main properties of this family by typing `waveinfo('mexh')` from the MATLAB command line. See “Mexican Hat Wavelet: mexh” on page 6-71 for more information.
Meyer

The Meyer wavelet and scaling function are defined in the frequency domain.

You can obtain a survey of the main properties of this family by typing `waveinfo('meyer')` from the MATLAB command line. See “Meyer Wavelet: meyr” on page 6-69 for more detail.
Wavelets: A New Tool for Signal Analysis