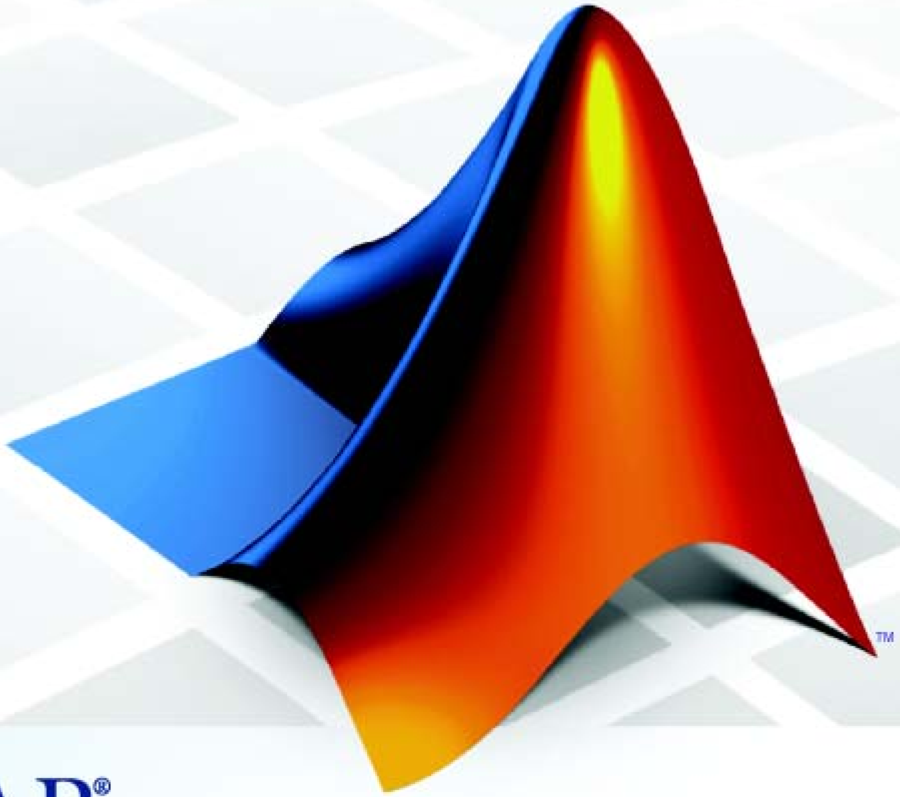
Wavelet Toolbox™ 4

User's Guide

Michel Misiti
Yves Misiti
Georges Oppenheim
Jean-Michel Poggi



MATLAB®

The MathWorks™
Accelerating the pace of engineering and science

How to Contact The MathWorks:

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup
`www.mathworks.com/contact_TS.html`	Technical support
`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

508-647-7000 (Phone)

508-647-7001 (Fax)

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

For contact information about worldwide offices, see the MathWorks Web site.

Wavelet Toolbox™ User's Guide

Trademarks

Patents

Revision History

March 1997	First printing	New for Version 1.0
September 2000	Second printing	Revised for Version 2.0 (Release 12)
June 2001	Online only	Revised for Version 2.1 (Release 12.1)
July 2002	Online only	Revised for Version 2.2 (Release 13)
June 2004	Online only	Revised for Version 3.0 (Release 14)
July 2004	Third printing	Revised for Version 3.0
October 2004	Online only	Revised for Version 3.0.1 (Release 14SP1)
March 2005	Online only	Revised for Version 3.0.2 (Release 14SP2)
June 2005	Fourth printing	Minor revision for Version 3.0.2
September 2005	Online only	Minor revision for Version 3.0.3 (Release R14SP3)
March 2006	Online only	Minor revision for Version 3.0.4 (Release 2006a)
September 2006	Online only	Revised for Version 3.1 (Release 2006b)
March 2007	Online only	Revised for Version 4.0 (Release 2007a)
September 2007	Online only	Revised for Version 4.1 (Release 2007b)
October 2007	Fifth printing	Revised for Version 4.1
March 2008	Online only	Revised for Version 4.2 (Release 2008a)
October 2008	Online only	Revised for Version 4.3 (Release 2008b)
March 2009	Online only	Revised for Version 4.4 (Release 2009a)
September 2009	Online only	Minor revision for Version 4.4.1 (Release 2009b)

Acknowledgments

The authors wish to express their gratitude to all the colleagues who directly or indirectly contributed to the making of the Wavelet Toolbox™ software.

Specifically

- For the wavelet questions to Pierre-Gilles Lemarié-Rieusset (Evry) and Yves Meyer (ENS Cachan)
- For the statistical questions to Lucien Birgé (Paris 6), Pascal Massart (Paris 11) and Marc Lavielle (Paris 5)
- To David Donoho (Stanford) and to Anestis Antoniadis (Grenoble), who give generously so many valuable ideas

Colleagues and friends who have helped us steadily are Patrice Abry (ENS Lyon), Samir Akkouche (Ecole Centrale de Lyon), Mark Asch (Paris 11), Patrice Assouad (Paris 11), Roger Astier (Paris 11), Jean Coursol (Paris 11), Didier Dacunha-Castelle (Paris 11), Claude Deniau (Marseille), Patrick Flandrin (Ecole Normale de Lyon), Eric Galin (Ecole Centrale de Lyon), Christine Graffigne (Paris 5), Anatoli Juditsky (Grenoble), Gérard Kerkyacharian (Paris 10), Gérard Malgouyres (Paris 11), Olivier Nowak (Ecole Centrale de Lyon), Dominique Picard (Paris 7), and Franck Tarpin-Bernard (Ecole Centrale de Lyon).

Several student groups have tested preliminary versions.

One of our first opportunities to apply the ideas of wavelets connected with signal analysis and its modeling occurred during a close and pleasant cooperation with the team “Analysis and Forecast of the Electrical Consumption” of Electricité de France (Clamart-Paris) directed first by Jean-Pierre Desbrosses, and then by Hervé Laffaye, and which included Xavier Brossat, Yves Deville, and Marie-Madeleine Martin.

Many thanks to those who tested and helped to refine the software and the printed matter and at last to The MathWorks group and specially to Roy Lurie, Jim Tung, Bruce Sesnovich, Jad Succari, Jane Carmody, and Paul Costa.

And finally, apologies to those we may have omitted.

About the Authors

Michel Misiti, Georges Oppenheim, and Jean-Michel Poggi are mathematics professors at Ecole Centrale de Lyon, University of Marne-La-Vallée and Paris 5 University. Yves Misiti is a research engineer specializing in Computer Sciences at Paris 11 University.

The authors are members of the “Laboratoire de Mathématique” at Orsay-Paris 11 University France. Their fields of interest are statistical signal processing, stochastic processes, adaptive control, and wavelets. The authors’ group, established more than 15 years ago, has published numerous theoretical papers and carried out applications in close collaboration with industrial teams. For instance:

- Robustness of the piloting law for a civilian space launcher for which an expert system was developed
- Forecasting of the electricity consumption by nonlinear methods
- Forecasting of air pollution

Notes by Yves Meyer

The history of wavelets is not very old, at most 10 to 15 years. The field experienced a fast and impressive start, characterized by a close-knit international community of researchers who freely circulated scientific information and were driven by the researchers’ youthful enthusiasm. Even as the commercial rewards promised to be significant, the ideas were shared, the trials were pooled together, and the successes were shared by the community.

There are lots of successes for the community to share. Why? Probably because the time is ripe. Fourier techniques were liberated by the appearance of windowed Fourier methods that operate locally on a time-frequency approach. In another direction, Burt-Adelson’s pyramidal algorithms, the quadrature mirror filters, and filter banks and subband coding are available. The mathematics underlying those algorithms existed earlier, but new computing techniques enabled researchers to try out new ideas rapidly. The numerical image and signal processing areas are blooming.

The wavelets bring their own strong benefits to that environment: a local outlook, a multiscaled outlook, cooperation between scales, and a time-scale analysis. They demonstrate that sines and cosines are not the only useful functions and that other bases made of weird functions serve to look at new foreign signals, as strange as most fractals or some transient signals.

Recently, wavelets were determined to be the best way to compress a huge library of fingerprints. This is not only a milestone that highlights the practical value of wavelets, but it has also proven to be an instructive process for the researchers involved in the project. Our initial intuition generally was that the proper way to tackle this problem of interweaving lines and textures was to use wavelet packets, a flexible technique endowed with quite a subtle sharpness of analysis and a substantial compression capability. However, it was a biorthogonal wavelet that emerged victorious and at this time represents the best method in terms of cost as well as speed. Our intuitions led one way, but implementing the methods settled the issue by pointing us in the right direction.

For wavelets, the period of growth and intuition is becoming a time of consolidation and implementation. In this context, a toolbox is not only possible, but valuable. It provides a working environment that permits experimentation and enables implementation.

Since the field still grows, it has to be vast and open. The Wavelet Toolbox product addresses this need, offering an array of tools that can be organized according to several criteria:

- Synthesis and analysis tools
- Wavelet and wavelet packets approaches
- Signal and image processing
- Discrete and continuous analyses
- Orthogonal and redundant approaches
- Coding, de-noising and compression approaches

What can we anticipate for the future, at least in the short term? It is difficult to make an accurate forecast. Nonetheless, it is reasonable to think that the pace of development and experimentation will carry on in many different fields. Numerical analysis constantly uses new bases of functions to encode its operators or to simplify its calculations to solve partial differential equations. The analysis and synthesis of complex transient signals touches musical instruments by studying the striking up, when the bow meets the cello string. The analysis and synthesis of multifractal signals, whose regularity (or rather irregularity) varies with time, localizes information of interest at its geographic location. Compression is a booming field, and coding and de-noising are promising.

For each of these areas, the Wavelet Toolbox software provides a way to introduce, learn, and apply the methods, regardless of the user's experience. It includes a command-line mode and a graphical user interface mode, each very capable and complementing to the other. The user interfaces help the novice to get started and the expert to implement trials. The command line provides an open environment for experimentation and addition to the graphical interface.

In the journey to the heart of a signal's meaning, the toolbox gives the traveler both guidance and freedom: going from one point to the other, wandering from a tree structure to a superimposed mode, jumping from low to high scale, and skipping a breakdown point to spot a quadratic chirp. The time-scale graphs of continuous analysis are often breathtaking and more often than not enlightening as to the structure of the signal.

Here are the tools, waiting to be used.

Yves Meyer
Professor, Ecole Normale Supérieure de Cachan and Institut de France

Notes by Ingrid Daubechies

Wavelet transforms, in their different guises, have come to be accepted as a set of tools useful for various applications. Wavelet transforms are good to have at one's fingertips, along with many other mostly more traditional tools.

Wavelet Toolbox software is a great way to work with wavelets. The toolbox, together with the power of MATLAB® software, really allows one to write complex and powerful applications, in a very short amount of time. The Graphic User Interface is both user-friendly and intuitive. It provides an excellent interface to explore the various aspects and applications of wavelets; it takes away the tedium of typing and remembering the various function calls.

Ingrid C. Daubechies
Professor, Princeton University, Department of Mathematics and Program in Applied and Computational Mathematics

Product Overview

Everywhere around us are signals that can be analyzed. For example, there are seismic tremors, human speech, engine vibrations, medical images, financial data, music, and many other types of signals. Wavelet analysis is a new and promising set of tools and techniques for analyzing these signals.

Wavelet Toolbox™ software is a collection of functions built on the MATLAB® technical computing environment. It provides tools for the analysis and synthesis of signals and images, and tools for statistical applications, using wavelets and wavelet packets within the framework of MATLAB.

The MathWorks™ provides several products that are relevant to the kinds of tasks you can perform with the toolbox. For more information about any of these products, see the products section of The MathWorks Web site.

Wavelet Toolbox software provides two categories of tools:

- Command-line functions
- Graphical interactive tools

The first category of tools is made up of functions that you can call directly from the command line or from your own applications. Most of these functions are M-files, series of statements that implement specialized wavelet analysis or synthesis algorithms. You can view the code for these functions using the following statement:

```
type function_name
```

You can view the header of the function, the help part, using the statement

```
help function_name
```

A summary list of the Wavelet Toolbox functions is available to you by typing

```
help wavelet
```

You can change the way any toolbox function works by copying and renaming the M-file, then modifying your copy. You can also extend the toolbox by adding your own M-files.

The second category of tools is a collection of graphical interface tools that afford access to extensive functionality. Access these tools from the command line by typing

```
wavemenu
```

Note The examples in this guide are generated using Wavelet Toolbox software with the DWT extension mode set to `'zpd'` (for zero padding), except when it is explicitly mentioned. So if you want to obtain exactly the same numerical results, type `dwtmode('zpd')`, before to execute the example code.

In most of the command-line examples, figures are displayed. To clarify the presentation, the plotting commands are partially or completely omitted. To reproduce the displayed figures exactly, you would need to insert some graphical commands in the example code.

Background Reading

Wavelet Toolbox™ software provides a complete introduction to wavelets and assumes no previous knowledge of the area. The toolbox allows you to use wavelet techniques on your own data immediately and develop new insights.

It is our hope that, through the use of these practical tools, you may want to explore the beautiful underlying mathematics and theory.

Excellent supplementary texts provide complementary treatments of wavelet theory and practice (see “References” on page 6-155). For instance:

- Burke-Hubbard [Bur96] is an historical and up-to-date text presenting the concepts using everyday words.
- Daubechies [Dau92] is a classic for the mathematics.
- Kaiser [Kai94] is a mathematical tutorial, and a physics-oriented book.
- Mallat [Mal98] is a 1998 book, which includes recent developments, and consequently is one of the most complete.
- Meyer [Mey93] is the “father” of the wavelet books.
- Strang-Nguyen [StrN96] is especially useful for signal processing engineers. It offers a clear and easy-to-understand introduction to two central ideas: filter banks for discrete signals, and for wavelets. It fully explains the connection between the two. Many exercises in the book are drawn from Wavelet Toolbox software.

The Wavelet Digest Internet site (http://www.wavelet.org/) provides much useful and practical information.

Installing Wavelet Toolbox™ Software

To install this toolbox on your computer, see the appropriate platform-specific MATLAB® installation guide. To determine if the Wavelet Toolbox™ software is already installed on your system, check for a subdirectory named `wavelet` within the main toolbox directory or folder.

Wavelet Toolbox software can perform signal or image analysis. For indexed images or truecolor images (represented by `m`-by-`n`-by-3 arrays of `uint8`), all wavelet functions use floating-point operations. To avoid `Out of Memory` errors, be sure to allocate enough memory to process various image sizes.

The memory can be real RAM or can be a combination of RAM and virtual memory. See your operating system documentation for how to configure virtual memory.

System Recommendations

While not a requirement, we recommend you obtain Signal Processing Toolbox™ and Image Processing Toolbox™ software to use in conjunction with the Wavelet Toolbox software. These toolboxes provide complementary functionality that give you maximum flexibility in analyzing and processing signals and images.

This manual makes no assumption that your computer is running any other MATLAB toolboxes.

Platform-Specific Details

Some details of the use of the Wavelet Toolbox software may depend on your hardware or operating system.

Windows Fonts

We recommend you set your operating system to use “Small Fonts.” Set this option by clicking the Display icon in your desktop’s Control Panel (accessible through the **Settings⇒Control Panel** submenu). Select the **Configuration** option, and then use the **Font Size** menu to change to **Small Fonts**. You’ll have to restart Windows® for this change to take effect.

Fonts for Non-Windows Platforms

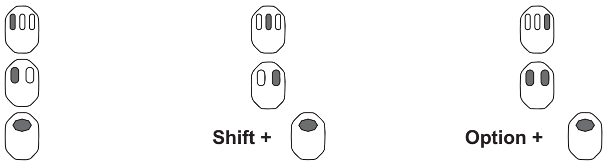
We recommend you set your operating system to use standard default fonts.

However, for all platforms, if you prefer to use large fonts, some of the labels in the GUI figures may be illegible when using the default display mode of the toolbox. To change the default mode to accept large fonts, use the `wtbxmngr` function. (For more information, see either the `wtbxmngr` help or its reference page.)

Mouse Compatibility

Wavelet Toolbox software was designed for three distinct types of mouse control.

Left Mouse Button	Middle Mouse Button	Right Mouse Button
Make selections. Activate controls.	Display cross-hairs to show position-dependent information.	Translate plots up and down, and left and right.



Note The functionality of the middle mouse button and the right mouse button can be inverted depending on the platform.

For more information, see “Using the Mouse” on page A-4.

Wavelet Applications

Wavelets have scale aspects and time aspects, consequently every application has scale and time aspects. To clarify them we try to untangle the aspects somewhat arbitrarily.

For scale aspects, we present one idea around the notion of local regularity. For time aspects, we present a list of domains. When the decomposition is taken as a whole, the de-noising and compression processes are center points.

Scale Aspects

As a complement to the spectral signal analysis, new signal forms appear. They are less regular signals than the usual ones.

The cusp signal presents a very quick local variation. Its equation is t^r with t close to 0 and $0 < r < 1$. The lower r the sharper the signal.

To illustrate this notion physically, imagine you take a piece of aluminum foil; The surface is very smooth, very regular. You first crush it into a ball, and then you spread it out so that it looks like a surface. The asperities are clearly visible. Each one represents a two-dimension cusp and analog of the one dimensional cusp. If you crush again the foil, more tightly, in a more compact ball, when you spread it out, the roughness increases and the regularity decreases.

Several domains use the wavelet techniques of regularity study:

- Biology for cell membrane recognition, to distinguish the normal from the pathological membranes
- Metallurgy for the characterization of rough surfaces
- Finance (which is more surprising), for detecting the properties of quick variation of values
- In Internet traffic description, for designing the services size

Time Aspects

Let’s switch to time aspects. The main goals are:

- Rupture and edges detection
- Study of short-time phenomena as transient processes

As domain applications, we get:

- Industrial supervision of gear-wheel
- Checking undue noises in craned or dented wheels, and more generally in nondestructive control quality processes
- Detection of short pathological events as epileptic crises or normal ones as evoked potentials in EEG (medicine)
- SAR imagery
- Automatic target recognition
- Intermittence in physics

Wavelet Decomposition as a Whole

Many applications use the wavelet decomposition taken as a whole. The common goals concern the signal or image clearance and simplification, which are parts of de-noising or compression.

We find many published papers in oceanography and earth studies.

One of the most popular successes of the wavelets is the compression of FBI fingerprints.

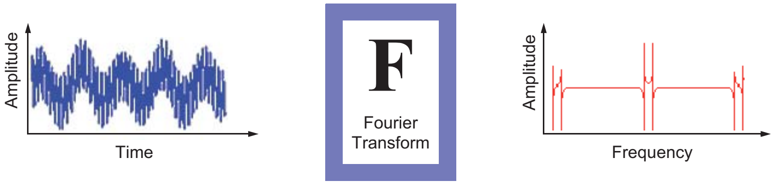
When trying to classify the applications by domain, it is almost impossible to sum up several thousand papers written within the last 15 years. Moreover, it is difficult to get information on real-world industrial applications from companies. They understandably protect their own information.

Some domains are very productive. Medicine is one of them. We can find studies on micro-potential extraction in EKGs, on time localization of His bundle electrical heart activity, in ECG noise removal. In EEGs, a quick transitory signal is drowned in the usual one. The wavelets are able to determine if a quick signal exists, and if so, can localize it. There are attempts to enhance mammograms to discriminate tumors from calcifications.

Another prototypical application is a classification of Magnetic Resonance Spectra. The study concerns the influence of the fat we eat on our body fat. The type of feeding is the basic information and the study is intended to avoid taking a sample of the body fat. Each Fourier spectrum is encoded by some of its wavelet coefficients. A few of them are enough to code the most interesting features of the spectrum. The classification is performed on the coded vectors.

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 time-based to frequency-based.



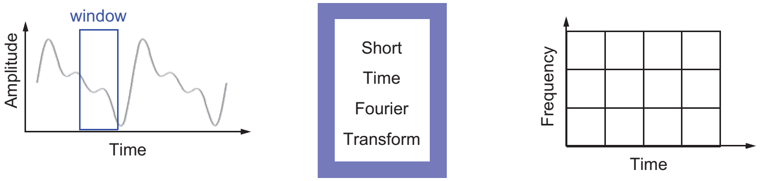
For many signals, Fourier analysis is extremely useful because the signal's frequency content is of great importance. So why do we need other techniques, like wavelet analysis?

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 the signal properties do not 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 nonstationary 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.

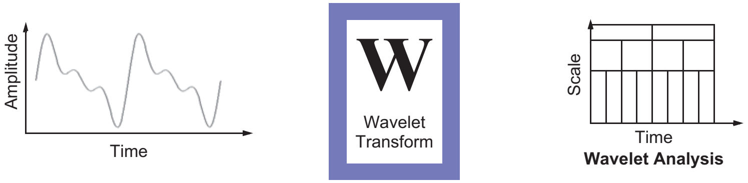


The STFT represents a sort of compromise between the time- and frequency-based views of a signal. It provides some information about both when and at what frequencies a signal event occurs. However, you can only obtain this information with limited precision, and that precision is determined by the size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

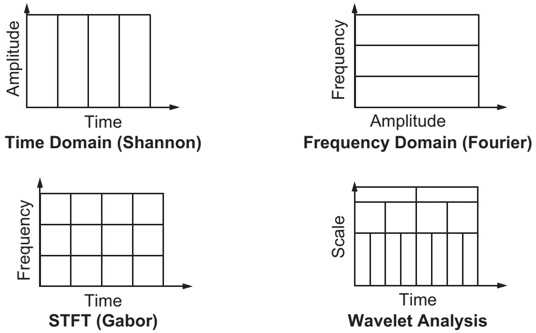
While the STFT compromise between time and frequency information can be useful, the drawback is that once you choose a particular size for the time window, that window is the same for all frequencies. Many signals require a more flexible approach — one where we can vary the window size to determine more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



Time Domain (Shannon)

Frequency Domain (Fourier)

STFT (Gabor)

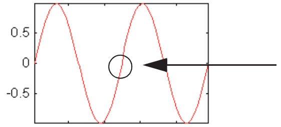
Wavelet Analysis

You may have noticed that wavelet analysis does not use a time-frequency region, but rather a time-*scale* region. For more information about the concept of scale and the link between scale and frequency, see "How to Connect Scale to Frequency?" on page 6-66.

What Can Wavelet Analysis Do?

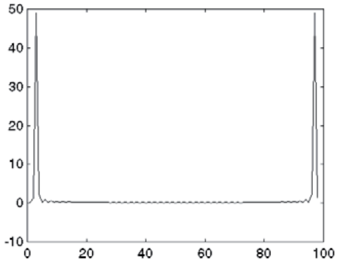
One major advantage afforded by wavelets is the ability to perform *local analysis* — that is, to analyze a localized area of a larger signal.

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.

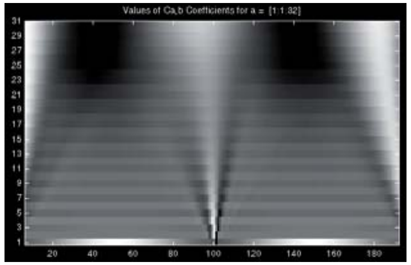


Sinusoid with a small discontinuity

A plot of the Fourier coefficients (as provided by the `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.



Fourier Coefficients



Wavelet Coefficients

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. Furthermore, 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.

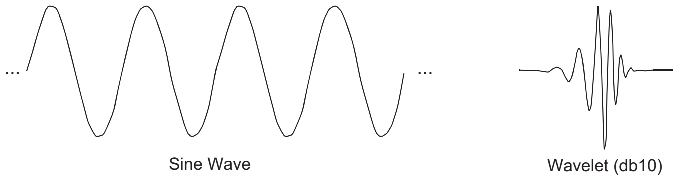
Indeed, in their brief history within the signal processing field, wavelets have already proven themselves to be an indispensable addition to the analyst's collection of tools and continue to enjoy a burgeoning popularity today.

What Is Wavelet Analysis?

Now that we know some situations when wavelet analysis is useful, it is worthwhile asking "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.



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 that 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.

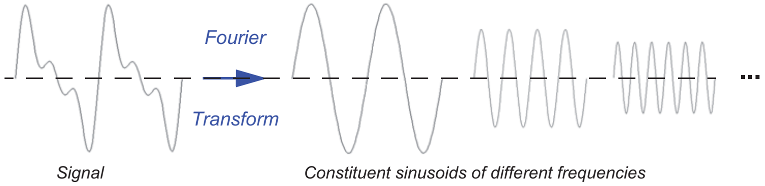
Continuous Wavelet Transform

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 frequency ω yield the constituent sinusoidal components of the original signal. Graphically, the process looks like

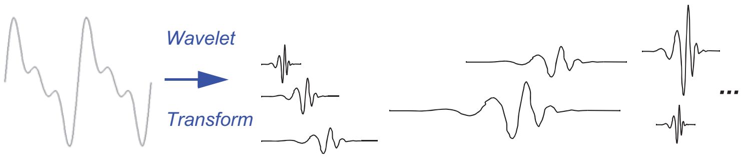


Similarly, the *continuous wavelet transform* (CWT) is defined as the sum over all time of the signal multiplied by scaled, shifted versions of the wavelet function ψ:

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

The results 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.



Signal

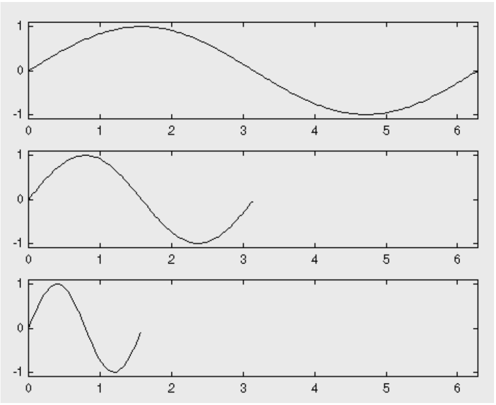
Constituent wavelets of different scales and positions

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.

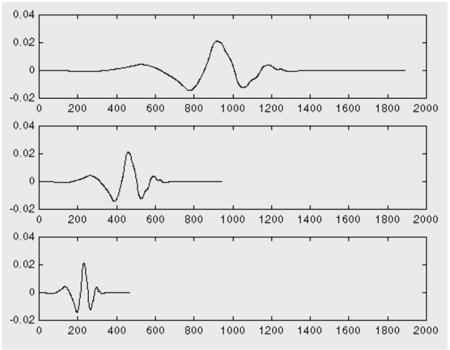


$f(t) = \sin(t)\ ;\quad a = 1$

$f(t) = \sin(2t)\ ;\quad a = \frac{1}{2}$

$f(t) = \sin(4t)\ ;\quad a = \frac{1}{4}$

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



$f(t) = \psi(t)\quad ;\quad a = 1$

$f(t) = \psi(2t)\quad ;\quad a = \frac{1}{2}$

$f(t) = \psi(4t)\quad ;\quad a = \frac{1}{4}$

It is clear from the diagrams that, for a sinusoid $\sin(\omega t)$, the scale factor a is related (inversely) to the radian frequency ω. 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 $f(t-k)$:

Wavelet function
$\psi(t)$

Shifted wavelet function
$\psi(t-k)$

Five Easy Steps to a Continuous Wavelet Transform

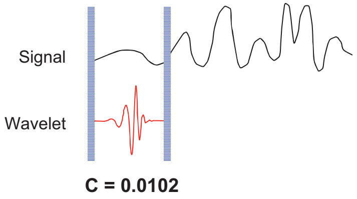
The continuous wavelet transform is the sum over all time of the signal multiplied by scaled, shifted versions of the wavelet. This process produces wavelet coefficients that are a function of scale and position.

It's really a very simple process. In fact, here are the five steps of an easy recipe for creating a CWT:

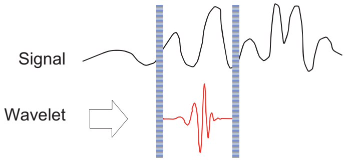
1 Take a wavelet and compare it to a section at the start of the original signal.

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. More precisely, if the signal energy and the wavelet energy are equal to one, C may be interpreted as a correlation coefficient.

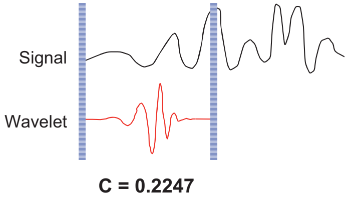
   Note that the results will depend on the shape of the wavelet you choose.



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



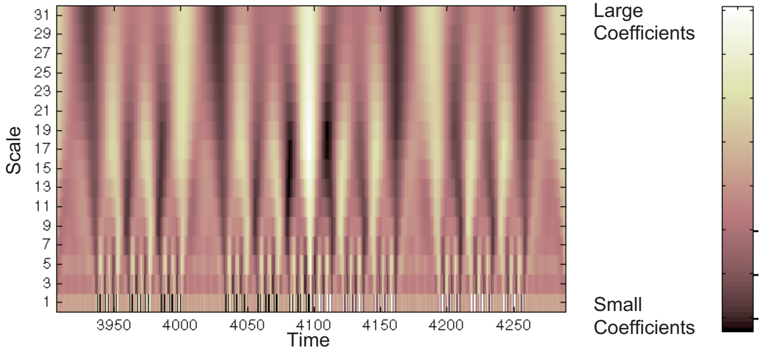
4 Scale (stretch) the wavelet and repeat steps 1 through 3.



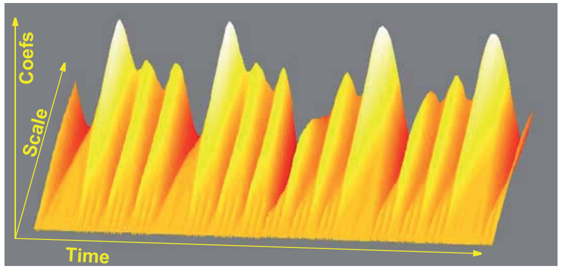
5 Repeat steps 1 through 4 for all scales.

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.



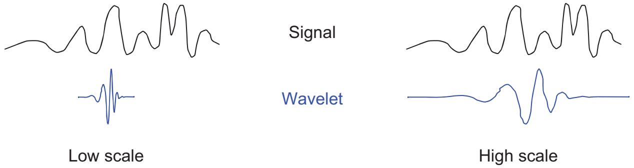
These coefficient plots resemble a bumpy surface viewed from above. If you could look at the same surface from the side, you might see something like this:



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 from 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 $a \Rightarrow$ Compressed wavelet $\Rightarrow$ Rapidly changing details $\Rightarrow$ High frequency ω.

- High scale $a \Rightarrow$ Stretched wavelet $\Rightarrow$ Slowly changing, coarse features $\Rightarrow$ Low frequency ω.

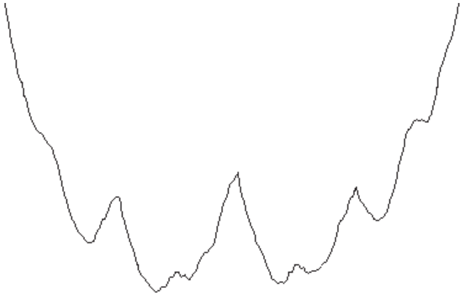
Scale of Nature

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.

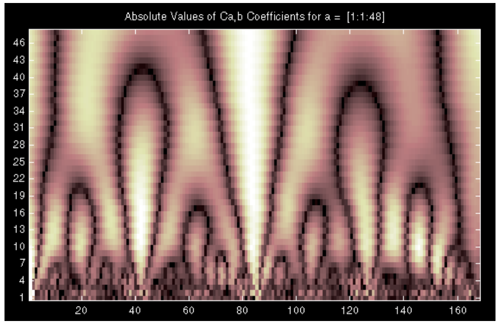
Not only is time-scale a different way to view data, it is a very natural way to view data deriving from a great number of natural phenomena.

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.

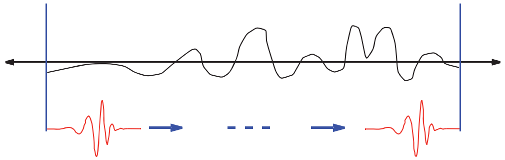
What's Continuous About the Continuous Wavelet Transform?

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. 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), is the set of scales and positions 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 that 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.



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 such an analysis from the *discrete wavelet transform* (DWT). For more information on DWT, see "Algorithms" on page 6-23.

An efficient way to implement this scheme using filters was developed in 1988 by Mallat (see [Mal89] in "References" on page 6-155). The Mallat algorithm is in fact a classical scheme known in the signal processing community as a *two-channel subband coder* (see page 1 of the book *Wavelets and Filter Banks*, by Strang and Nguyen [StrN96]).

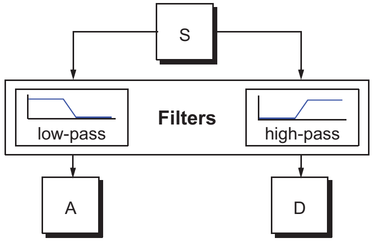
This very practical filtering algorithm yields a *fast wavelet transform* — a box into which a signal passes, and out of which wavelet coefficients quickly emerge. Let's examine this in more depth.

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.

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.

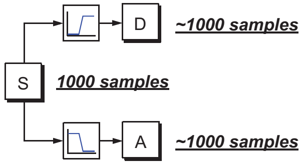
The filtering process, at its most basic level, looks like this.

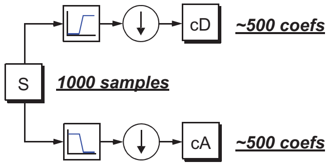


The original signal, S, passes through two complementary filters and emerges as two signals.

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 resulting signals will each have 1000 samples, for a total of 2000.

These signals A and D are interesting, but we get 2000 values instead of the 1000 we had. There exists a more subtle way to perform the decomposition using wavelets. By looking carefully at the computation, we may keep only one point out of two in each of the two 2000-length samples to get the complete information. This is the notion of *downsampling*. We produce two sequences called cA and cD.

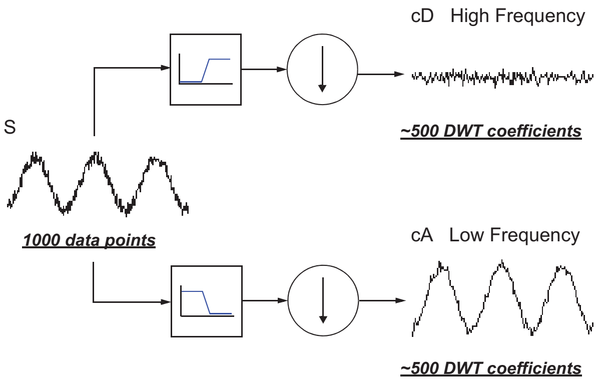




The process on the right, which includes downsampling, produces DWT coefficients.

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.



The MATLAB® code needed to generate s, cD, and cA is

```
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 cD are small and consist mainly of a high-frequency noise, while the approximation coefficients cA contain much less noise than does the original signal.

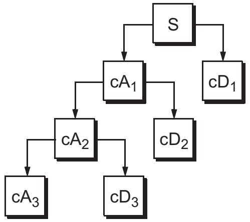
```
[length(cA) length(cD)]

ans =
   501   501
```

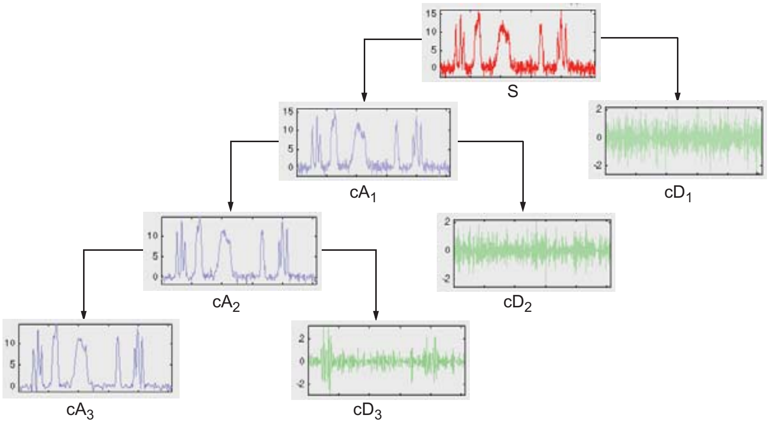
You may observe that the actual lengths of the detail and approximation coefficient vectors are slightly *more* than half the length of the original signal. This has to do with the filtering process, which is implemented by convolving the signal with a filter. The convolution "smears" the signal, introducing 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.



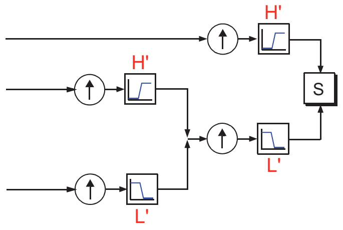
Number of Levels

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 "Choosing the Optimal Decomposition" on page 6-147).

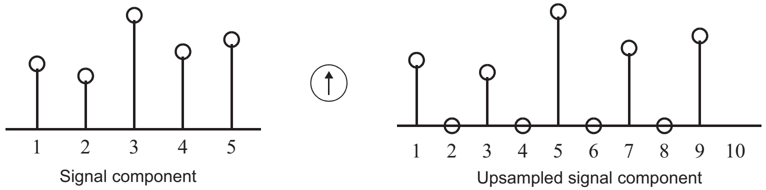
Wavelet Reconstruction

We've learned how the discrete wavelet transform can be used to analyze, or decompose, signals and images. This process is called *decomposition* or *analysis*. The other half of the story is how those components can be assembled back into the original signal without 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 using Wavelet Toolbox™ software, 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.



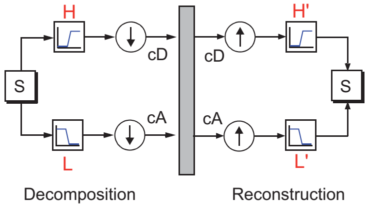
The toolbox includes commands, like `idwt` and `waverec`, that perform single-level or multilevel reconstruction, respectively, on the components of one-dimensional signals. These commands have their two-dimensional analogs, `idwt2` and `waverec2`.

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.

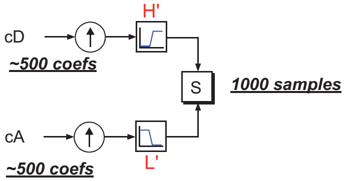
The downsampling of the signal components performed during the decomposition phase introduces a distortion called aliasing. It turns out that by carefully choosing filters for the decomposition and reconstruction phases that are closely related (but not identical), we can "cancel out" the effects of aliasing.

A technical discussion of how to design these filters is available on page 347 of the book *Wavelets and Filter Banks*, by Strang and Nguyen. The low- and high-pass decomposition filters (`L` and `H`), together with their associated reconstruction filters (`L'` and `H'`), form a system of what is called *quadrature mirror filters*:



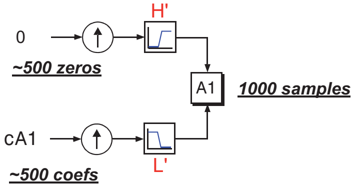
Reconstructing Approximations and Details

We have seen that it is possible to reconstruct our original signal from the coefficients of the approximations and details.



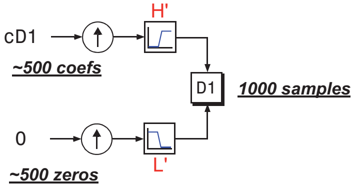
It is also possible to reconstruct the approximations and details themselves from their coefficient vectors. As an example, let's consider how we would reconstruct the first-level approximation `A1` from the coefficient vector `cA1`.

We pass the coefficient vector `cA1` through the same process we used to reconstruct the original signal. However, instead of combining it with the level-one detail `cD1`, we feed in a vector of zeros in place of the detail coefficients vector:



The process yields a reconstructed *approximation* `A1`, 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 `D1`, using the analogous process:

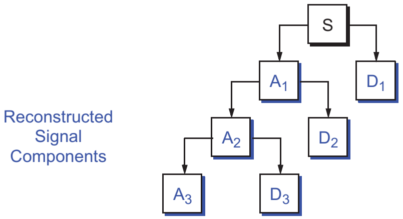


The reconstructed details and approximations are true constituents of the original signal. In fact, we find when we combine them that

$$A_1 + D_1 = S$$

Note that the coefficient vectors `cA1` and `cD1` — because they were produced by downsampling 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 multilevel analysis, we find that similar relationships hold for all the reconstructed signal constituents. That is, there are several ways to reassemble the original signal:



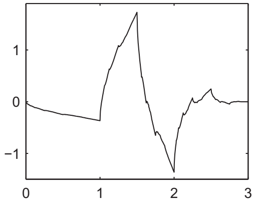
$$S = A_1 + D_1$$
$$= A_2 + D_2 + D_1$$
$$= A_3 + D_3 + D_2 + D_1$$

Relationship of Filters to Wavelet Shapes

In the section “Reconstruction Filters” on page 1-30, 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.

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 low-pass reconstruction filter (`L'`) for the `db2` wavelet.



db2 wavelet

The filter coefficients can be obtained from the `dbaux` command:

```
Lprime = dbaux(2)
```

```
Lprime =
    0.3415    0.5915    0.1585    0.0915
```

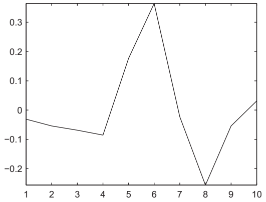
If we reverse the order of this vector (see `wrev`), and then multiply every even sample by –1, we obtain the high-pass filter `H'`:

```
Hprime =
    0.0915    0.1585    0.5915    0.3415
```

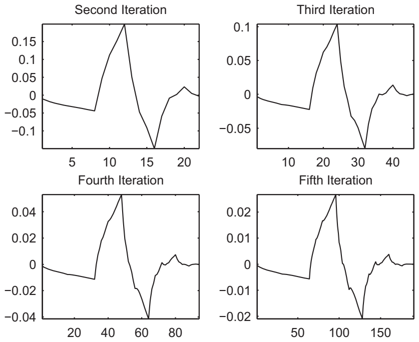
Next, upsample `Hprime` by two (see `dyadup`), inserting zeros in alternate positions:

```
HU =
     0.0915         0    0.1585         0    0.5915         0    0.3415
Finally, convolve the upsampled vector with the original low-pass
filter:
```

```
H2 = conv(HU,Lprime);
plot(H2)
```



If we iterate this process several more times, repeatedly upsampling and convolving the resultant vector with the four-element filter vector `Lprime`, 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.

Scaling Function

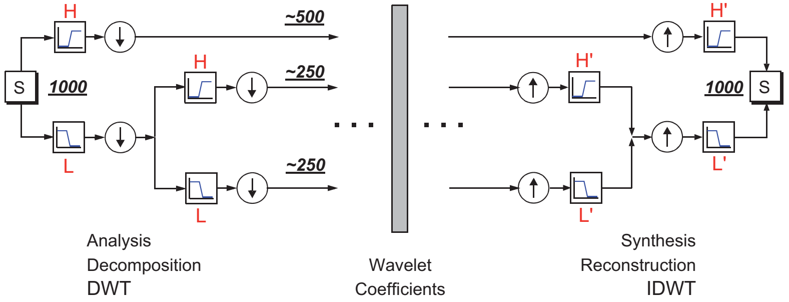
We've seen the interrelation of wavelets and quadrature mirror filters. The wavelet function ψ is determined by the high-pass filter, which also produces the details of the wavelet decomposition.

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

In the same way that iteratively upsampling and convolving the high-pass filter produces a shape approximating the wavelet function, iteratively upsampling and convolving the low-pass filter produces a shape approximating the scaling function.

Multistep Decomposition and Reconstruction

A multistep analysis-synthesis process can be represented as



This process involves two aspects: breaking up a signal to obtain 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 may modify the wavelet coefficients before performing the reconstruction step. 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. No doubt, many uncharted uses of the wavelet coefficients lie in wait. The 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.

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 what we now call a “wavelet” seems to be 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.

Barbara Burke Hubbard describes the birth, the history, and the seminal concepts in a very clear text. See *The World According to Wavelets,* A.K. Peters, Wellesley, 1996.

The wavelet domain is growing up very quickly. A lot of mathematical papers and practical trials are published every month.

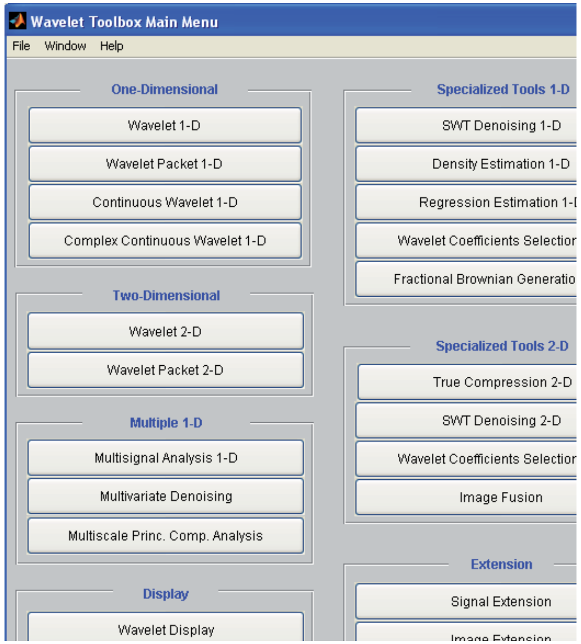
Introduction to the Wavelet Families

Several families of wavelets that have proven to be especially useful are included in this toolbox. What follows is an introduction to some wavelet families.

- “Haar” on page 1-41
- “Daubechies” on page 1-42
- “Biorthogonal” on page 1-43
- “Coiflets” on page 1-45
- “Symlets” on page 1-45
- “Morlet” on page 1-46
- “Mexican Hat” on page 1-46
- “Meyer” on page 1-47
- “Other Real Wavelets” on page 1-47
- “Complex Wavelets” on page 1-47

To explore all wavelet families on your own, check out the **Wavelet Display** tool:

1. Type `wavemenu` at the MATLAB® command line. The **Wavelet Toolbox Main Menu** appears.



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

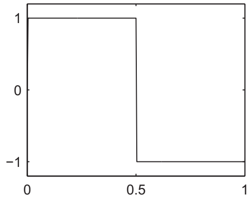
3 Select a family from the **Wavelet** menu at the top right of the tool.

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

5 Obtain more information by clicking the information buttons located at the right.

Haar

Any discussion of wavelets begins with Haar wavelet, the first and simplest. Haar wavelet is discontinuous, and resembles a step function. It represents the same wavelet as Daubechies `db1`. See “Haar” on page 6-73 for more detail.

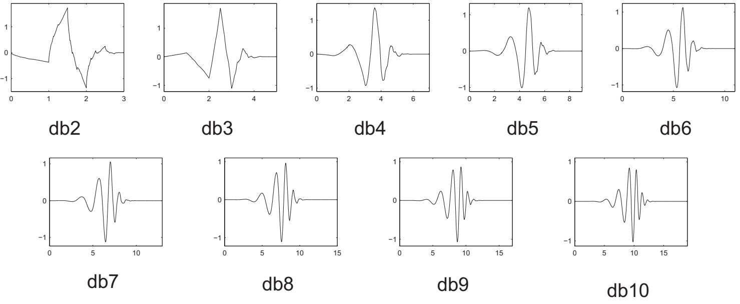


Wavelet function psi

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` wavelet. Here are the wavelet functions psi of the next nine members of the family:



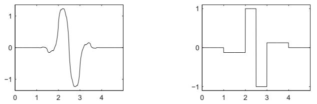
db2 db3 db4 db5 db6

db7 db8 db9 db10

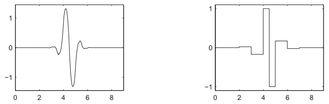
You can obtain a survey of the main properties of this family by typing `waveinfo('db')` from the MATLAB command line. See “Daubechies Wavelets: dbN” on page 6-72 for more detail.

Biorthogonal

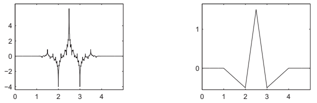
This family of wavelets exhibits the property of linear phase, which is needed for signal and image reconstruction. By using two wavelets, one for decomposition (on the left side) and the other for reconstruction (on the right side) instead of the same single one, interesting properties are derived.



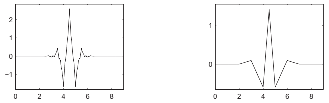
bior1.3



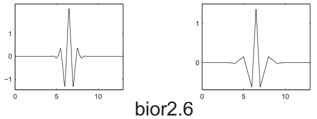
bior1.5



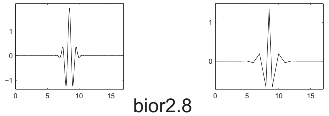
bior2.2



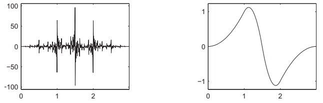
bior2.4



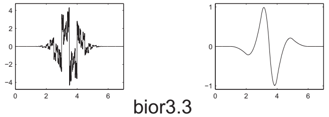
bior2.6



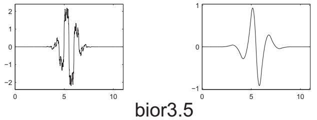
bior2.8



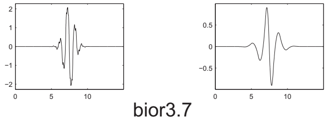
bior3.1



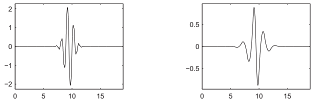
bior3.3



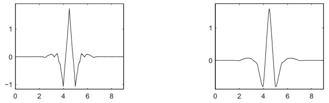
bior3.5



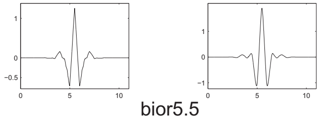
bior3.7



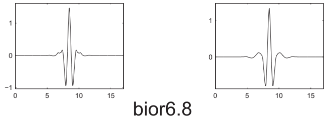
bior3.9



bior4.4



bior5.5

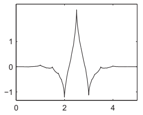


bior6.8

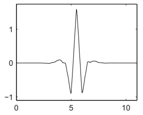
You can obtain a survey of the main properties of this family by typing `waveinfo('bior')` from the MATLAB command line. See “Biorthogonal Wavelet Pairs: biorNr.Nd” on page 6-76 for more detail.

Coiflets

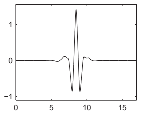
Built by I. Daubechies at the request of R. Coifman. The wavelet function has $2N$ moments equal to 0 and the scaling function has $2N$-1 moments equal to 0. The two functions have a support of length $6N$-1. You can obtain a survey of the main properties of this family by typing `waveinfo('coif')` from the MATLAB command line. See “Coiflet Wavelets: coifN” on page 6-75 for more detail.



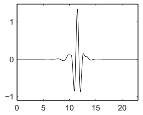
coif1



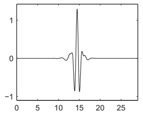
coif2



coif3



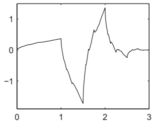
coif4



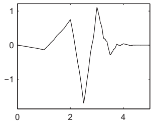
coif5

Symlets

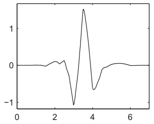
The symlets are nearly symmetrical wavelets proposed by Daubechies as modifications to the `db` family. The properties of the two wavelet families are similar. Here are the wavelet functions psi.



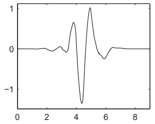
sym2



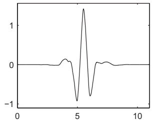
sym3



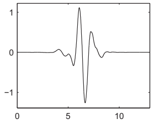
sym4



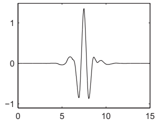
sym5



sym6



sym7

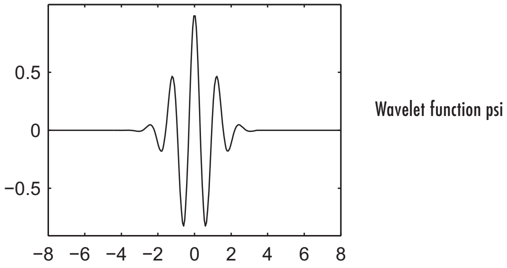


sym8

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 6-74 for more detail.

Morlet

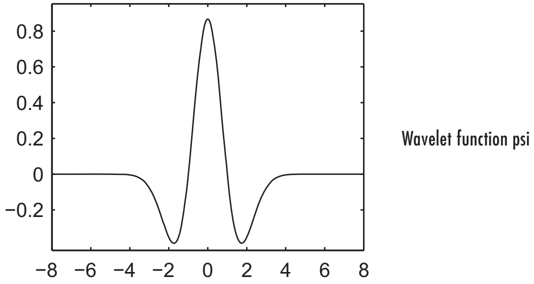
This wavelet has no scaling function, but is explicit.



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 6-81 for more detail.

Mexican Hat

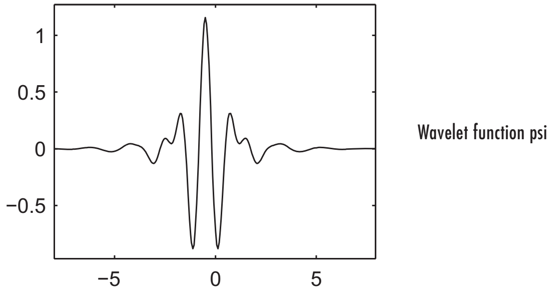
This wavelet has no scaling function and is derived from a function that is proportional to the second derivative function of the Gaussian probability density function.



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 6-80 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 6-78 for more detail.

Other Real Wavelets

Some other real wavelets are available in the toolbox:

- Reverse Biorthogonal
- Gaussian derivatives family
- FIR based approximation of the Meyer wavelet

See "Additional Real Wavelets" on page 6-82 for more information.

Complex Wavelets

Some complex wavelet families are available in the toolbox:

- Gaussian derivatives
- Morlet
- Frequency B-Spline
- Shannon

See "Complex Wavelets" on page 6-84 for more information.

One-Dimensional Continuous Wavelet Analysis

This section takes you through the features of continuous wavelet analysis using Wavelet Toolbox™ software.

The toolbox requires only one function for continuous wavelet analysis: `cwt`. You'll find full information about this function in its reference page.

In this section, you'll learn how to

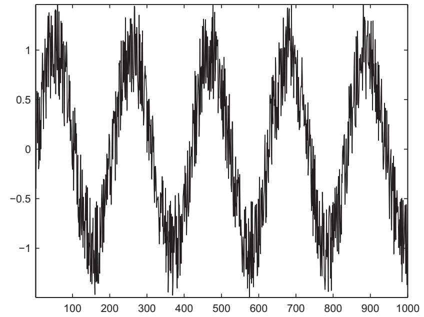
- Load a signal
- Perform a continuous wavelet transform of a signal
- Produce a plot of the coefficients
- Produce a plot of coefficients at a given scale
- Produce a plot of local maxima of coefficients across scales
- Select the displayed plots
- Switch from scale to pseudo-frequency information
- Zoom in on detail
- Display coefficients in normal or absolute mode
- Choose the scales at which analysis is performed

Since you can perform analyses either from the command line or using the graphical interface tools, this section has subsections covering each method.

The final subsection discusses how to exchange signal and coefficient information between the disk and the graphical tools.

Continuous Analysis Using the Command Line

This example involves a noisy sinusoidal signal.



1 Load a signal.

From the MATLAB® prompt, type

```
load noissin;
```

You now have the signal `noissin` in your workspace:

```
whos
```

Name	Size	Bytes	Class
noissin	1x1000	8000	double array

2 Perform a Continuous Wavelet Transform.

Use the `cwt` command. Type

```
c = cwt(noissin,1:48,'db4');
```

The arguments to cwt specify the signal to be analyzed, the scales of the analysis, and the wavelet to be used. The returned argument c contains the coefficients at various scales. In this case, c is a 48-by-1000 matrix with each row corresponding to a single scale.

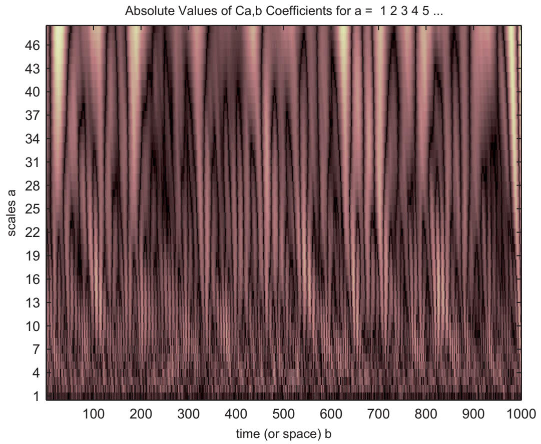
3 Plot the coefficients.

The cwt command accepts a fourth argument. This is a flag that, when present, causes cwt to produce a plot of the absolute values of the continuous wavelet transform coefficients.

The cwt command can accept more arguments to define the different characteristics of the produced plot. For more information, see the cwt reference page.

```
c = cwt(noissin,1:48,'db4','plot');
```

A plot appears.



Of course, coefficient plots generated from the command line can be manipulated using ordinary MATLAB graphics commands.

4 Choose scales for the analysis.

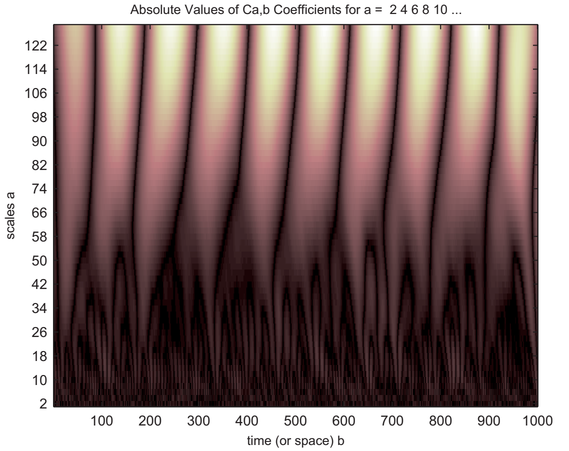
The second argument to cwt gives you fine control over the scale levels on which the continuous analysis is performed. In the previous example, we used all scales from 1 to 48, but you can construct any scale vector subject to these constraints:

- All scales must be real positive numbers.
- The scale increment must be positive.
- The highest scale cannot exceed a maximum value depending on the signal.

Let’s repeat the analysis using every other scale from 2 to 128. Type

```
c = cwt(noissin,2:2:128,'db4','plot');
```

A new plot appears:



This plot gives a clearer picture of what’s happening with the signal, highlighting the periodicity.

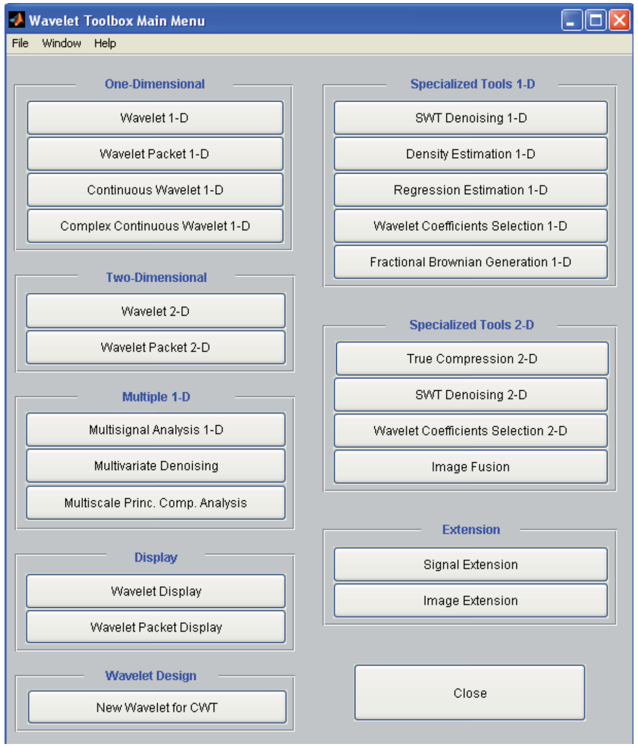
Continuous Analysis Using the Graphical Interface

We now use the **Continuous Wavelet 1-D** tool to analyze the same noisy sinusoidal signal we examined earlier using the command line interface in “Continuous Analysis Using the Command Line” on page 2-5.

1 Start the Continuous Wavelet 1-D Tool. From the MATLAB prompt, type

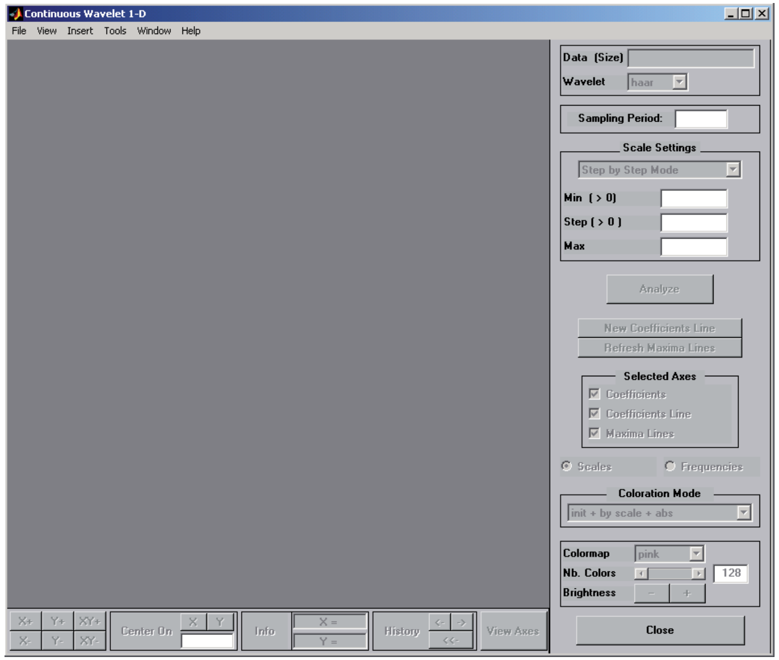
```
wavemenu
```

The **Wavelet Toolbox Main Menu** appears.



Click the **Continuous Wavelet 1-D** menu item.

The continuous wavelet analysis tool for one-dimensional signal data appears.



2 Load a signal.

Choose the **File > Load Signal** menu option.

When the **Load Signal** dialog box appears, select the demo MAT-file `noissin.mat`, which should reside in the MATLAB directory `toolbox/wavelet/wavedemo`. Click the **OK** button.

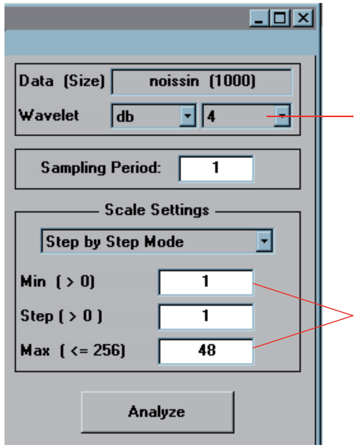
The noisy sinusoidal signal is loaded into the **Continuous Wavelet 1-D** tool.

The default value for the sampling period is equal to 1 (second).

3 Perform a Continuous Wavelet Transform.

To start our analysis, let's perform an analysis using the `db4` wavelet at scales 1 through 48, just as we did using command line functions in the previous section.

In the upper right portion of the **Continuous Wavelet 1-D** tool, select the `db4` wavelet and scales 1–48.



Data (Size) noissin (1000)

Wavelet db 4

Sampling Period: 1

Scale Settings

Step by Step Mode

Min (> 0) 1

Step (> 0) 1

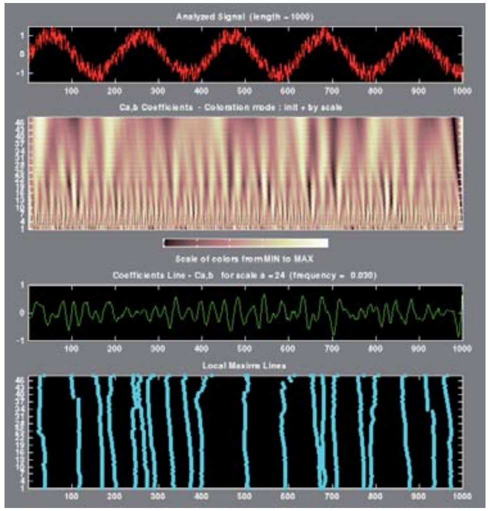
Max (<= 256) 48

Select db4

Select scales 1 to 48 in steps of 1

4 Click the **Analyze** button.

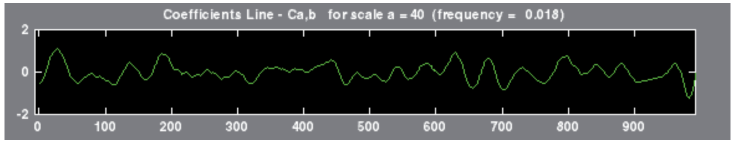
After a pause for computation, the tool displays the coefficients plot, the coefficients line plot corresponding to the scale a = 24, and the local maxima plot, which displays the chaining across scales (from a = 48 down to a = 1) of the coefficients local maxima.


Analyzed Signal (length = 1000)
Ca,b Coefficients - Coloration mode : init + by scale
Scale of colors from MIN to MAX
Coefficients Line - Ca,b for scale a = 24 (frequency = 0.030)
Local Maxima Lines
100 200 300 400 500 600 700 800 900 1000

5 View Wavelet Coefficients Line.

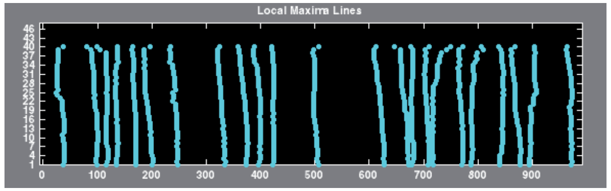
Select another scale a = 40 by clicking in the coefficients plot with the right mouse button. See step 9 to know, more precisely, how to select the desired scale.

Click the **New Coefficients Line** button. The tool updates the plot.

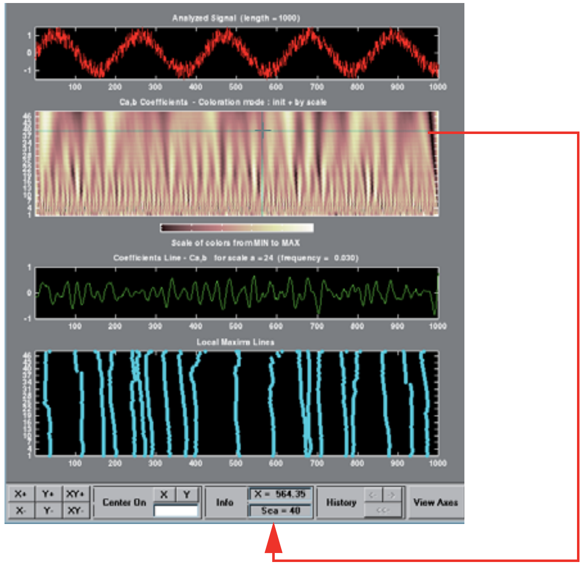


6 View Maxima Line.

Click the **Refresh Maxima Line** button. The local maxima plot displays the chaining across scales of the coefficients local maxima from a = 40 down to a = 1.

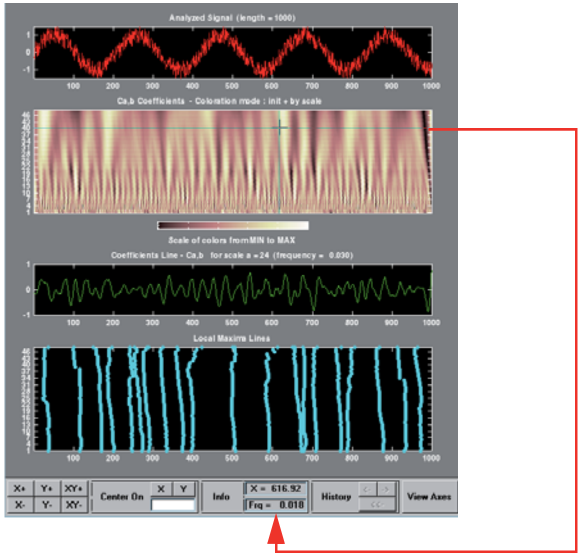


Hold down the right mouse button over the coefficients plot. The position of the mouse is given by the **Info** frame (located at the bottom of the screen) in terms of location (**X**) and scale (**Sca**).


Analyzed Signal (length = 1000)
Ca,b Coefficients - Coloration mode : init + by scale
Scale of colors from MIN to MAX
Coefficients Line - Ca,b for scale a = 24 (frequency = 0.030)
Local Maxima Lines
X+
Y+
XY+
X-
Y-
XY-
Center On
X
Y
Info
X = 564.35
Sca = 40
History
View Axes

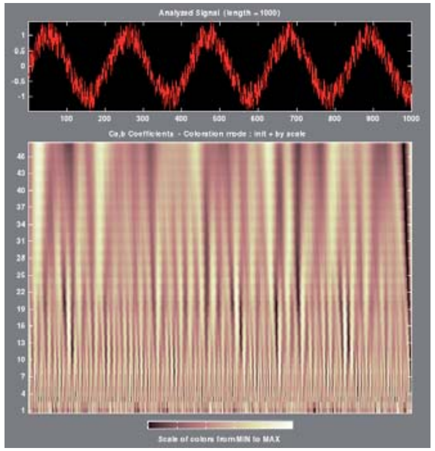
7 Switch from scale to Pseudo-Frequency Information.

Using the option button on the right part of the screen, select **Frequencies** instead of **Scales**. Again hold down the right mouse button over the coefficients plot, the position of the mouse is given in terms of location (**X**) and frequency (**Frq**) in Hertz.


Analyzed Signal (length = 1000)
Ca,b Coefficients - Coloration mode : init + by scale
Scale of colors from MIN to MAX
Coefficients Line - Ca,b for scale a = 24 (frequency = 0.030)
Local Maxima Lines
X+ Y+ XY+
X- Y- XY-
Center On X Y
Info
X = 616.92
Frq = 0.018
History
View Axes

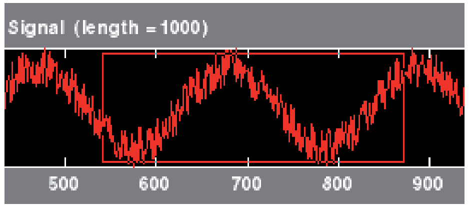
This facility allows you to interpret scale in terms of an associated pseudo-frequency, which depends on the wavelet and the sampling period. For more information on the connection between scale and frequency, see “How to Connect Scale to Frequency?” on page 6-66.

8 Deselect the last two plots using the check boxes in the **Selected Axes** frame.

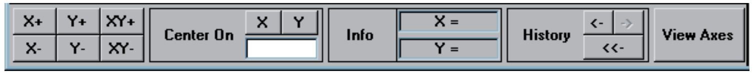

Analyzed Signal (length = 1000)
Ca,b Coefficients - Coloration mode : init + by scale
Scale of colors from MIN to MAX

9 Zoom in on detail.

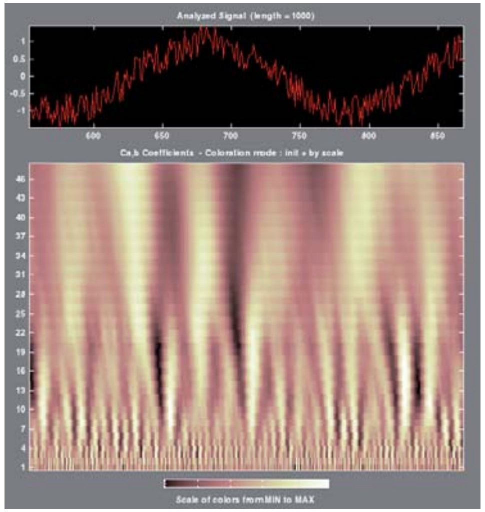
Drag a rubber band box (by holding down the left mouse button) over the portion of the signal you want to magnify.



10 Click the **X+** button (located at the bottom of the screen) to zoom horizontally only.



The **Continuous Wavelet 1-D** tool enlarges the displayed signal and coefficients plot (for more information on zooming, see “Connection of Plots” on page A-3).


Analyzed Signal (length = 1000)
1
0.5
0
-0.5
-1
600
650
700
750
800
850
Ca,b Coefficients - Coloration mode : init + by scale
46
43
40
37
34
31
28
25
22
19
16
13
10
7
4
1
Scale of colors from MIN to MAX

As with the command line analysis on the preceding pages, you can change the scales or the analyzing wavelet and repeat the analysis. To do this, just edit the necessary fields and click the **Analyze** button.

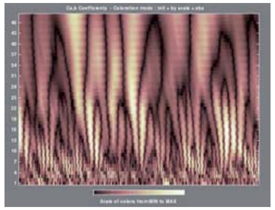
11 View normal or absolute coefficients.

The **Continuous Wavelet 1-D** tool allows you to plot either the absolute values of the wavelet coefficients, or the coefficients themselves.

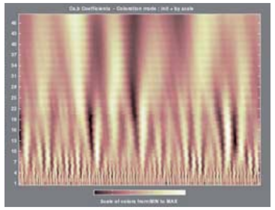
More generally, the coefficients coloration can be done in several different ways. For more details on the Coloration Mode, see “Controlling the Coloration Mode” on page A-7.

Choose either one of the absolute modes or normal modes from the **Coloration Mode** menu. In normal modes, the colors are scaled between the minimum and maximum of the coefficients. In absolute modes, the colors are scaled between zero and the maximum absolute value of the coefficients.

The coefficients plot is redisplayed in the mode you select.



Absolute Mode



Normal Mode

Importing and Exporting Information from the Graphical Interface

The Continuous Wavelet 1-D graphical interface tool lets you import information from and export information to disk.

You can

- Load signals from disk into the **Continuous Wavelet 1-D** tool.
- Save wavelet coefficients from the **Continuous Wavelet 1-D** tool to disk.

Loading Signals into the Continuous Wavelet 1-D Tool

To load a signal you’ve constructed in your MATLAB workspace into the **Continuous Wavelet 1-D** tool, save the signal in a MAT-file (with extension `mat` or other).

For instance, suppose you’ve designed a signal called `warma` and want to analyze it in the **Continuous Wavelet 1-D** tool.

```
save warma warma
```

The workspace variable warma must be a vector.

```
sizwarma = size(warma)

sizwarma =
           1         1000
```

To load this signal into the **Continuous Wavelet 1-D** tool, use the menu option **File > Load Signal**. A dialog box appears that lets you select the appropriate MAT-file to be loaded.

Note The first one-dimensional variable encountered in the file is considered the signal. Variables are inspected in alphabetical order.

Saving Wavelet Coefficients

The Continuous Wavelet 1-D tool lets you save wavelet coefficients to disk. The toolbox creates a MAT-file in the current directory with the extension wc1 and a name you give it.

To save the continuous wavelet coefficients from the present analysis, use the menu option **File > Save > Coefficients**.

A dialog box appears that lets you specify a directory and filename for storing the coefficients.

Consider the example analysis:

File > Example Analysis > with haar at scales [1:1:64] —> **Cantor curve**.

After saving the continuous wavelet coefficients to the file cantor.wc1, load the variables into your workspace:

```
load cantor.wc1 -mat
whos
```

Name	Size	Bytes	Class
coeff	64x2188	1120256	double array
scales	1x64	512	double array
wname	1x4	8	char array

Variables coefs and scales contain the continuous wavelet coefficients and the associated scales. More precisely, in the above example, coefs is a 64-by-2188 matrix, one row for each scale; and scales is the 1-by-64 vector 1:64. Variable wname contains the wavelet name.

One-Dimensional Complex Continuous Wavelet Analysis

This section takes you through the features of complex continuous wavelet analysis using the Wavelet Toolbox™ software and focuses on the differences between the real and complex continuous analysis.

You can refer to the section "One-Dimensional Continuous Wavelet Analysis" on page 2-4 if you want to learn how to

- Zoom in on detail
- Display coefficients in normal or absolute mode
- Choose the scales at which the analysis is performed
- Switch from scale to pseudo-frequency information
- Exchange signal and coefficient information between the disk and the graphical tools

Wavelet Toolbox software requires only one function for complex continuous wavelet analysis of a real valued signal: `cwt`. You'll find full information about this function in its reference page.

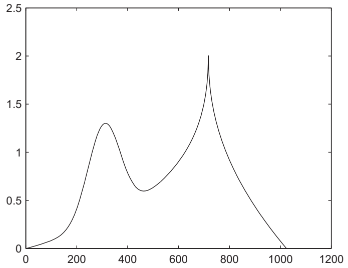
In this section, you'll learn how to

- Load a signal
- Perform a complex continuous wavelet transform of a signal
- Produce plots of the coefficients

Since you can perform analyses either from the command line or using the graphical interface tools, this section has subsections covering each method.

Complex Continuous Analysis Using the Command Line

This example involves a cusp signal.



1 Load a signal.

From the MATLAB® prompt, type

```
load cuspamax;
```

You now have the signal `cuspamax` in your workspace:

```
whos
```

Name	Size	Bytes	Class
caption	1x71	142	char array
cuspamax	1x1024	8192	double array

```
caption

caption =
       x = linspace(0,1,1024);
       y = exp(-128*((x-0.3).^2))-3*(abs(x-0.7).^0.4);
```

`caption` is a string that contains the signal definition.

2 Perform a Continuous Wavelet Transform.

Use the cwt command. Type

```
c = cwt(cuspamax,1:2:64,'cgau4');
```

The arguments to cwt specify the signal to be analyzed, the scales of the analysis, and the wavelet to be used. The returned argument c contains the coefficients at various scales. In this case, c is a complex 32-by-1024 matrix, each row of which corresponds to a single scale.

3 Plot the coefficients.

The cwt command accepts a fourth argument. This is a flag that, when present, causes cwt to produce four plots related to the complex continuous wavelet transform coefficients:

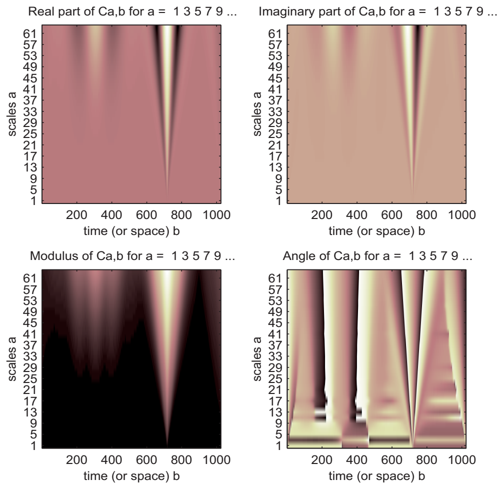
- Real and imaginary parts
- Modulus and angle

The cwt command can accept more arguments to define the different characteristics of the produced plots. For more information, see the cwt reference page.

Type

```
c = cwt(cuspamax,1:2:64,'cgau4','plot');
```

A plot appears:



Of course, coefficient plots generated from the command line can be manipulated using ordinary MATLAB graphics commands.

Complex Continuous Analysis Using the Graphical Interface

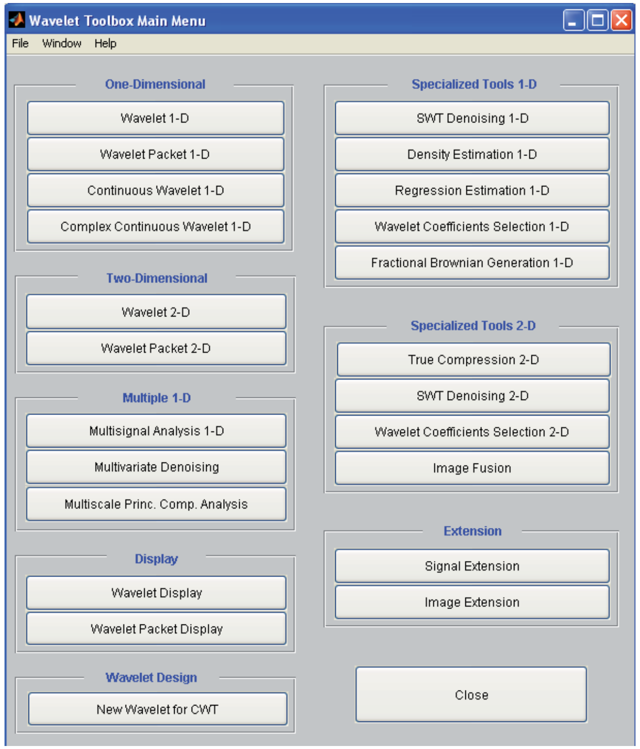
We now use the **Complex Continuous Wavelet 1-D** tool to analyze the same cusp signal we examined using the command line interface in the previous section.

1 Start the Complex Continuous Wavelet 1-D Tool.

From the MATLAB prompt, type

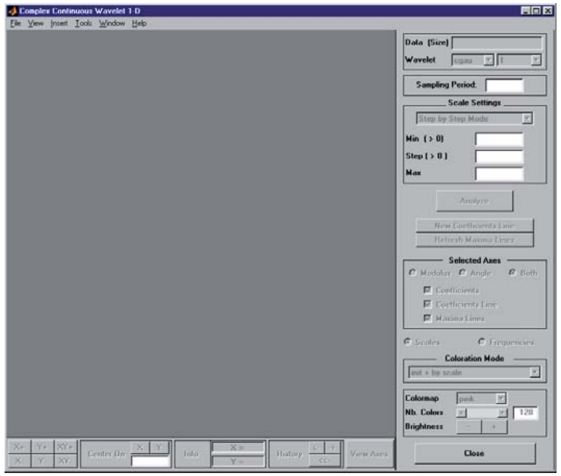
```
wavemenu
```

The **Wavelet Toolbox Main Menu** appears.



Click the **Complex Continuous Wavelet 1-D** menu item.

The continuous wavelet analysis tool for one-dimensional signal data appears.



2 Load a signal.

Choose the **File > Load Signal** menu option.

When the **Load Signal** dialog box appears, select the demo MAT-file `cuspamax.mat`, which should reside in the MATLAB directory `toolbox/wavelet/wavedemo`. Click the **OK** button.

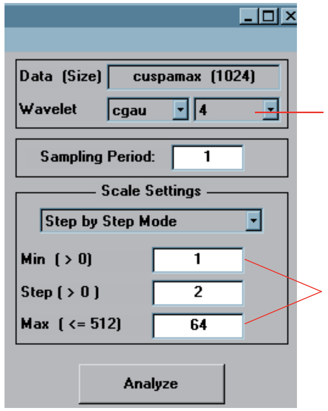
The cusp signal is loaded into the **Complex Continuous Wavelet 1-D** tool.

The default value for the sampling period is equal to 1 (second).

3 Perform a Complex Continuous Wavelet Transform

To start our analysis, let's perform an analysis using the `cgau4` wavelet at scales 1 through 64 in steps of 2, just as we did using command-line functions in “Complex Continuous Analysis Using the Command Line” on page 2-21.

In the upper-right portion of the **Complex Continuous Wavelet 1-D** tool, select the `cgau4` wavelet and scales 1–64 in steps of 2.



Data (Size) cuspamax (1024)

Wavelet cgau 4

Sampling Period: 1

Scale Settings

Step by Step Mode

Min (> 0) 1

Step (> 0) 2

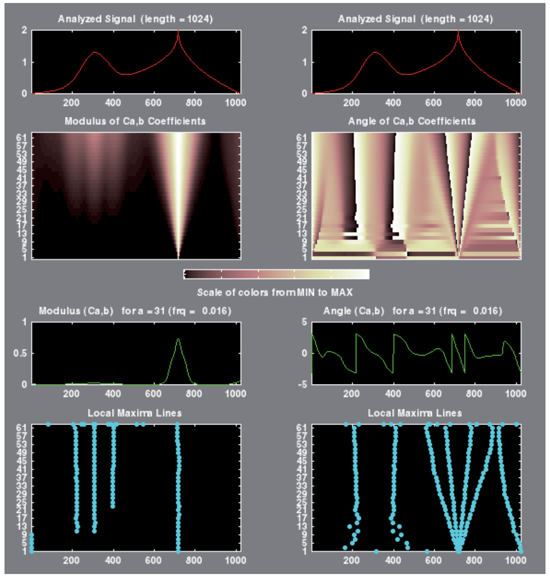
Max (<= 512) 64

Select cgau4

Select scales from 1 to 64 in steps of 2

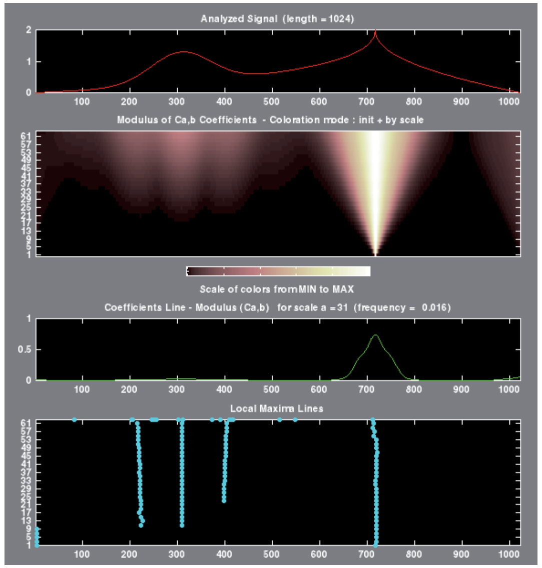
Click the **Analyze** button.

After a pause for computation, the tool displays the usual plots associated to the modulus of the coefficients on the left side, and the angle of the coefficients on the right side.



Each side has exactly the same representation that we found in “Continuous Analysis Using the Graphical Interface” on page 2-8.

Select the plots related to the modulus of the coefficients using the **Modulus** option button in the **Selected Axes** frame.



The figure now looks like the one in the real **Continuous Wavelet 1-D** tool.

Importing and Exporting Information from the Graphical Interface

To know how to import and export information from the Complex Continuous Wavelet Graphical Interface, see the corresponding paragraph in "One-Dimensional Continuous Wavelet Analysis" on page 2-4.

The only difference is that the variable `coefs` is a complex matrix (see "Saving Wavelet Coefficients" on page 2-18).

One-Dimensional Discrete Wavelet Analysis

This section takes you through the features of one-dimensional discrete wavelet analysis using the Wavelet Toolbox™ software.

The toolbox provides these functions for one-dimensional signal analysis. For more information, see the reference pages.

Analysis-Decomposition Functions

Function Name	Purpose
`dwt`	Single-level decomposition
`wavedec`	Decomposition
`wmaxlev`	Maximum wavelet decomposition level

Synthesis-Reconstruction Functions

Function Name	Purpose
`idwt`	Single-level reconstruction
`waverec`	Full reconstruction
`wrcoef`	Selective reconstruction
`upcoef`	Single reconstruction

Decomposition Structure Utilities

Function Name	Purpose
`detcoef`	Extraction of detail coefficients
`appcoef`	Extraction of approximation coefficients
`upwlev`	Recomposition of decomposition structure

De-noising and Compression

Function Name	Purpose
`ddencmp`	Provide default values for de-noising and compression
`wbmpen`	Penalized threshold for wavelet 1-D or 2-D de-noising
`wdcbm`	Thresholds for wavelet 1-D using BirgÐ©-Massart strategy
`wdencmp`	Wavelet de-noising and compression
`wden`	Automatic wavelet de-noising
`wthrmngr`	Threshold settings manager

In this section, you'll learn how to

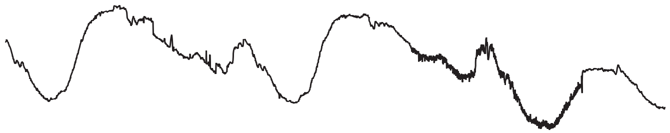
- Load a signal
- Perform a single-level wavelet decomposition of a signal
- Construct approximations and details from the coefficients
- Display the approximation and detail
- Regenerate a signal by inverse wavelet transform
- Perform a multilevel wavelet decomposition of a signal
- Extract approximation and detail coefficients
- Reconstruct the level 3 approximation
- Reconstruct the level 1, 2, and 3 details
- Display the results of a multilevel decomposition
- Reconstruct the original signal from the level 3 decomposition
- Remove noise from a signal
- Refine an analysis
- Compress a signal
- Show a signal's statistics and histograms

Since you can perform analyses either from the command line or using the graphical interface tools, this section has subsections covering each method.

The final subsection discusses how to exchange signal and coefficient information between the disk and the graphical tools.

One-Dimensional Analysis Using the Command Line

This example involves a real-world signal — electrical consumption measured over the course of 3 days. This signal is particularly interesting because of noise introduced when a defect developed in the monitoring equipment as the measurements were being made. Wavelet analysis effectively removes the noise.



1 Load a signal.

From the MATLAB® prompt, type

```
load leleccum;
```

Set the variables. Type

```
s = leleccum(1:3920);
l_s = length(s);
```

2 Perform a single-level wavelet decomposition of a signal.

Perform a single-level decomposition of the signal using the `db1` wavelet. Type

```
[cA1,cD1] = dwt(s,'db1');
```

This generates the coefficients of the level 1 approximation (`cA1`) and detail (`cD1`).

3 Construct approximations and details from the coefficients.

To construct the level 1 approximation and detail (A1 and D1) from the coefficients cA1 and cD1, type

```
A1 = upcoef('a',cA1,'db1',1,l_s);
D1 = upcoef('d',cD1,'db1',1,l_s);
```

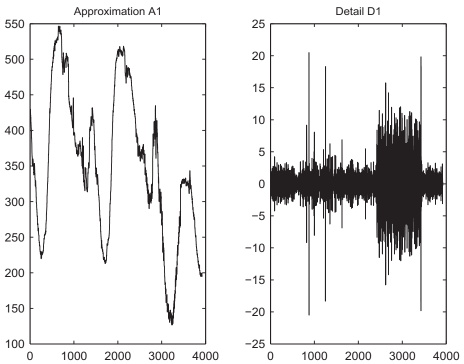
or

```
A1 = idwt(cA1,[],'db1',l_s);
D1 = idwt([],cD1,'db1',l_s);
```

4 Display the approximation and detail.

To display the results of the level-one decomposition, type

```
subplot(1,2,1); plot(A1); title('Approximation A1')
subplot(1,2,2); plot(D1); title('Detail D1')
```



5 Regenerate a signal by using the Inverse Wavelet Transform.

To find the inverse transform, type

```
A0 = idwt(cA1,cD1,'db1',l_s);
err = max(abs(s-A0))
```

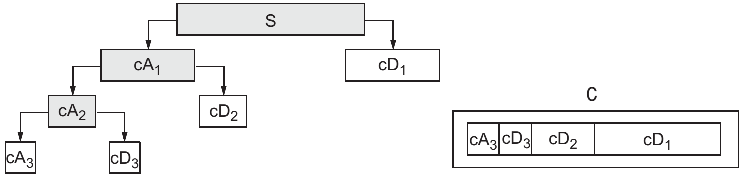
```
err =
     2.2737e-013
```

6 Perform a multilevel wavelet decomposition of a signal.

To perform a level 3 decomposition of the signal (again using the db1 wavelet), type

```
[C,L] = wavedec(s,3,'db1');
```

The coefficients of all the components of a third-level decomposition (that is, the third-level approximation and the first three levels of detail) are returned concatenated into one vector, C. Vector L gives the lengths of each component.



7 Extract approximation and detail coefficients.

To extract the level 3 approximation coefficients from C, type

```
cA3 = appcoef(C,L,'db1',3);
```

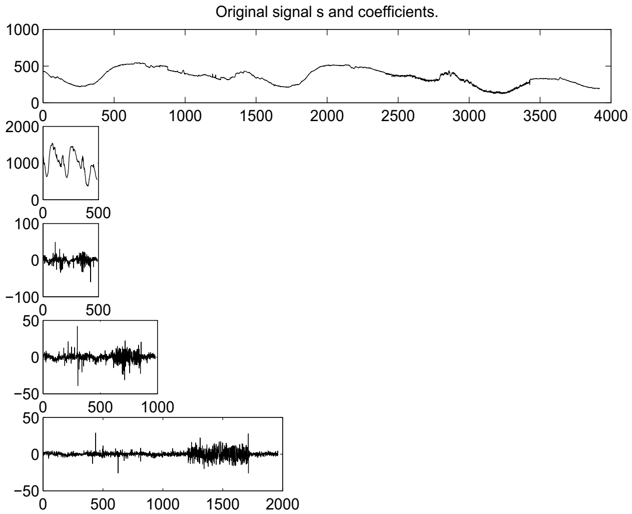
To extract the levels 3, 2, and 1 detail coefficients from C, type

```
cD3 = detcoef(C,L,3);
cD2 = detcoef(C,L,2);
cD1 = detcoef(C,L,1);
```

or

```
[cD1,cD2,cD3] = detcoef(C,L,[1,2,3]);
```

Results are displayed in the figure below, which contains the signal `s`, the approximation coefficients at level 3 (`cA3`), and the details coefficients from level 3 to 1 (`cD3`, `cD2` and `cD1`) from the top to the bottom.



8 Reconstruct the Level 3 approximation and the Level 1, 2, and 3 details.

To reconstruct the level 3 approximation from `C`, type

```
A3 = wrcoef('a',C,L,'db1',3);
```

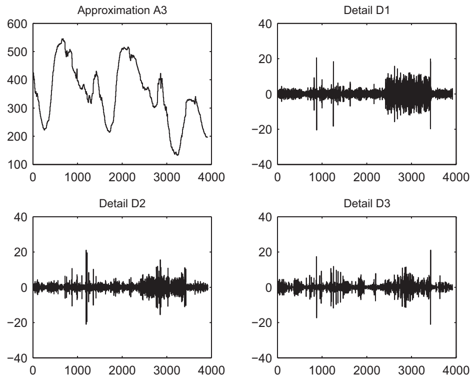
To reconstruct the details at levels 1, 2, and 3, from `C`, type

```
D1 = wrcoef('d',C,L,'db1',1);
D2 = wrcoef('d',C,L,'db1',2);
D3 = wrcoef('d',C,L,'db1',3);
```

9 Display the results of a multilevel decomposition.

To display the results of the level 3 decomposition, type

```
subplot(2,2,1); plot(A3);
title('Approximation A3')
subplot(2,2,2); plot(D1);
title('Detail D1')
subplot(2,2,3); plot(D2);
title('Detail D2')
subplot(2,2,4); plot(D3);
title('Detail D3')
```



10 Reconstruct the original signal from the Level 3 decomposition.

To reconstruct the original signal from the wavelet decomposition structure, type

```
A0 = waverec(C,L,'db1');

err = max(abs(s-A0))

err =
    4.5475e-013
```

11 Crude de-noising of a signal.

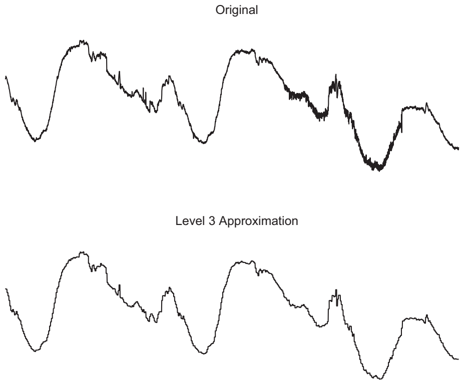
Using wavelets to remove noise from a signal requires identifying which component or components contain the noise, and then reconstructing the signal without those components.

In this example, we note that successive approximations become less and less noisy as more and more high-frequency information is filtered out of the signal.

The level 3 approximation, A3, is quite clean as a comparison between it and the original signal.

To compare the approximation to the original signal, type

```
subplot(2,1,1);plot(s);title('Original'); axis off
subplot(2,1,2);plot(A3);title('Level 3 Approximation');
axis off
```



Of course, in discarding all the high-frequency information, we've also lost many of the original signal's sharpest features.

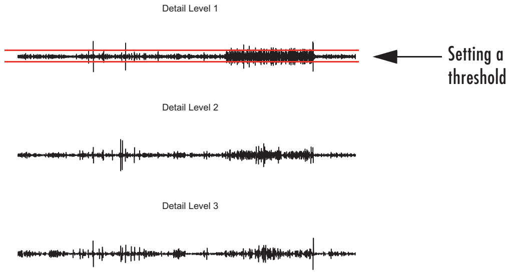
Optimal de-noising requires a more subtle approach called *thresholding*. This involves discarding only the portion of the details that exceeds a certain limit.

12 Remove noise by thresholding.

Let's look again at the details of our level 3 analysis.

To display the details D1, D2, and D3, type

```
subplot(3,1,1); plot(D1); title('Detail Level 1'); axis off
subplot(3,1,2); plot(D2); title('Detail Level 2'); axis off
subplot(3,1,3); plot(D3); title('Detail Level 3'); axis off
```



Most of the noise occurs in the latter part of the signal, where the details show their greatest activity. What if we limited the strength of the details by restricting their maximum values? This would have the effect of cutting back the noise while leaving the details unaffected through most of their durations. But there's a better way.

Note that cD1, cD2, and cD3 are just MATLAB vectors, so we could directly manipulate each vector, setting each element to some fraction of the vectors' peak or average value. Then we could reconstruct new detail signals D1, D2, and D3 from the thresholded coefficients.

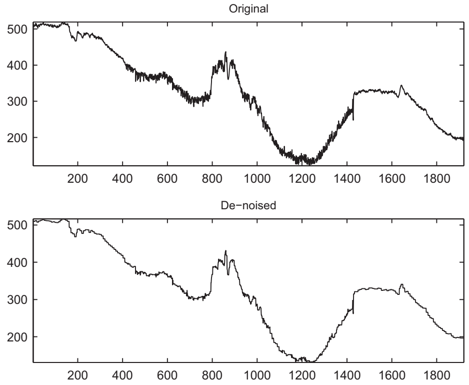
To denoise the signal, use the ddencmp command to calculate the default parameters and the wdencmp command to perform the actual de-noising, type

```
[thr,sorh,keepapp] = ddencmp('den','wv',s);
clean = wdencmp('gbl',C,L,'db1',3,thr,sorh,keepapp);
```

Note that wdencmp uses the results of the decomposition (C and L) that we calculated in step 6 on page 2-33. We also specify that we used the db1 wavelet to perform the original analysis, and we specify the global thresholding option 'gbl'. See ddencmp and wdencmp in the reference pages for more information about the use of these commands.

To display both the original and denoised signals, type

```
subplot(2,1,1); plot(s(2000:3920)); title('Original')
subplot(2,1,2); plot(clean(2000:3920)); title('De-noised')
```



We've plotted here only the noisy latter part of the signal. Notice how we've removed the noise without compromising the sharp detail of the original signal. This is a strength of wavelet analysis.

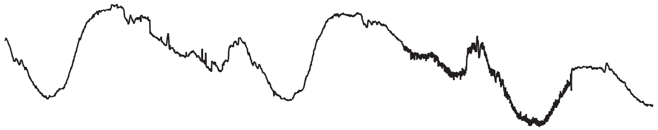
While using command line functions to remove the noise from a signal can be cumbersome, the software's graphical interface tools include an easy-to-use de-noising feature that includes automatic thresholding.

More information on the de-noising process can be found in the following sections:

- “Remove noise from a signal.” on page 2-49
- “De-Noising” on page 6-97
- “One-Dimensional Variance Adaptive Thresholding of Wavelet Coefficients” on page 2-158
- “One-Dimensional Variance Adaptive Thresholding of Wavelet Coefficients” on page 6-107

One-Dimensional Analysis Using the Graphical Interface

In this section, we explore the same electrical consumption signal as in the previous section, but we use the graphical interface tools to analyze the signal.

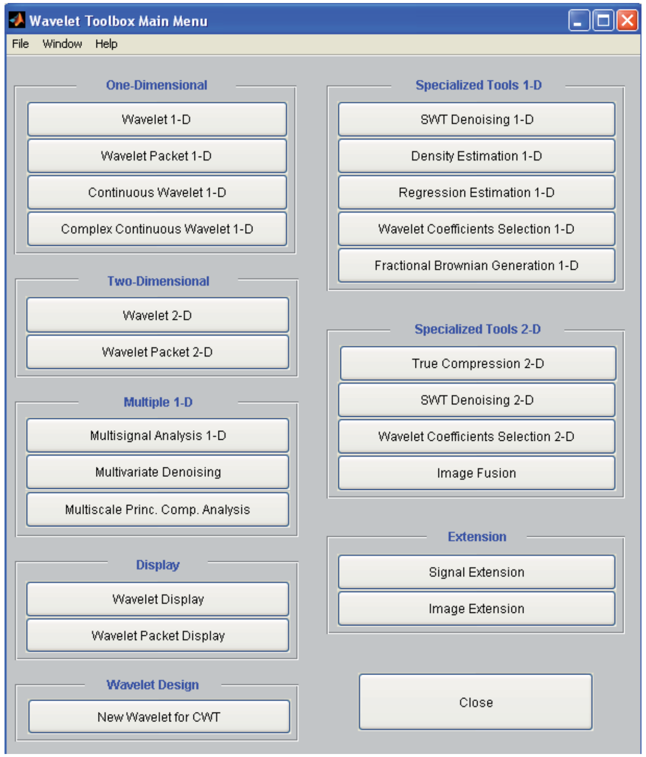


1 Start the 1-D Wavelet Analysis Tool.

From the MATLAB prompt, type

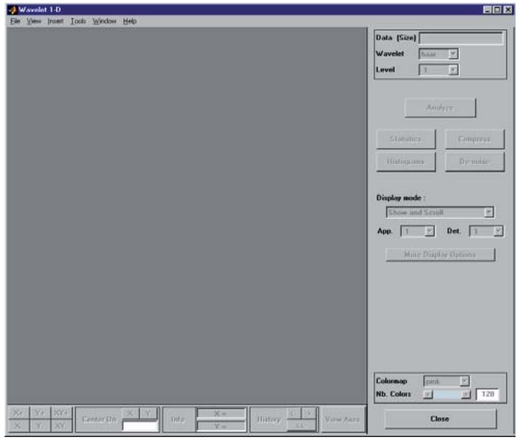
```
wavemenu
```

The **Wavelet Toolbox Main Menu** appears.



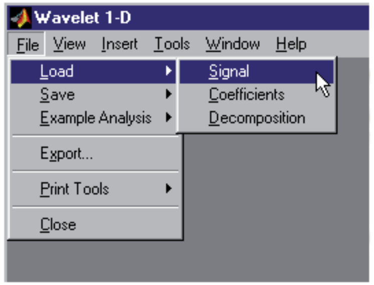
2 Click the **Wavelet 1-D** menu item.

The discrete wavelet analysis tool for one-dimensional signal data appears.

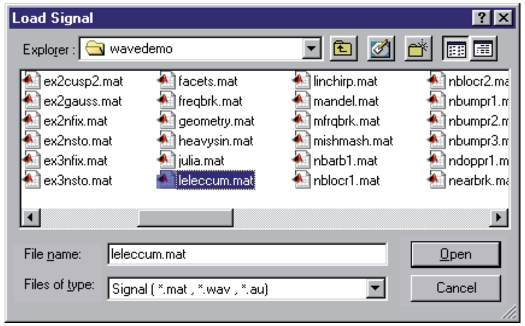


3 Load a signal.

From the **File** menu, choose the **Load > Signal** option.



When the **Load Signal** dialog box appears, select the demo MAT-file `leleccum.mat`, which should reside in the MATLAB directory `toolbox/wavelet/wavedemo`. Click the **OK** button.



Explorer : wavedemo

File name: leleccum.mat

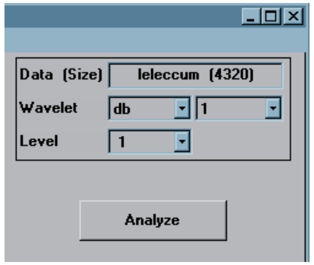
Files of type: Signal (*.mat , *.wav , *.au)

The electrical consumption signal is loaded into the **Wavelet 1-D** tool.

4 Perform a single-level wavelet decomposition.

To start our analysis, let's perform a single-level decomposition using the `db1` wavelet, just as we did using the command-line functions in "One-Dimensional Analysis Using the Command Line" on page 2-31.

In the upper right portion of the **Wavelet 1-D** tool, select the `db1` wavelet and single-level decomposition.



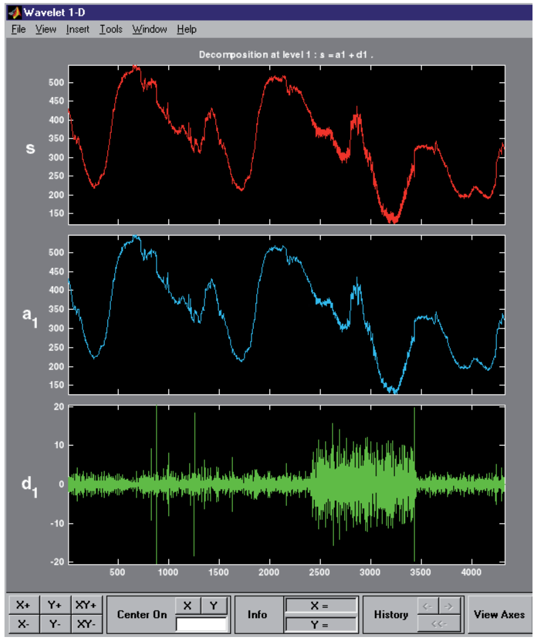
Data (Size) leleccum (4320)

Wavelet db 1

Level 1

Click the **Analyze** button.

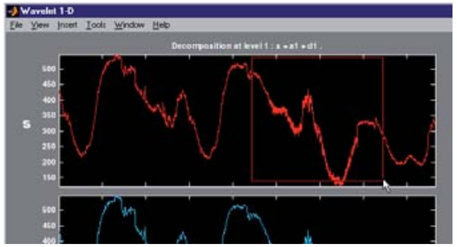
After a pause for computation, the tool displays the decomposition.



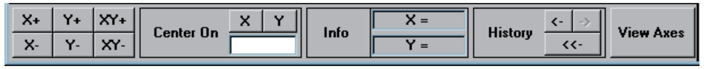
5 Zoom in on relevant detail.

One advantage of using the graphical interface tools is that you can zoom in easily on any part of the signal and examine it in greater detail.

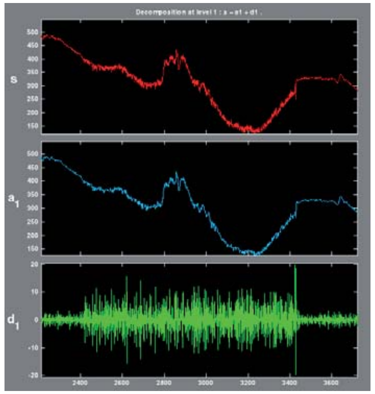
Drag a rubber band box (by holding down the left mouse button) over the portion of the signal you want to magnify. Here, we've selected the noisy part of the original signal.


Wavelet 1-D
File View Insert Tools Window Help
Decomposition at level 1 : s = a1 + d1 .
s

Click the **X+** button (located at the bottom of the screen) to zoom horizontally.



The **Wavelet 1-D** tool zooms all the displayed signals.

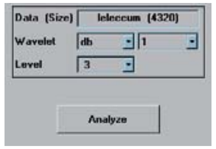


The other zoom controls do more or less what you'd expect them to. The **X-** button, for example, zooms out horizontally. The history function keeps track of all your views of the signal. Return to a previous zoom level by clicking the left arrow button.

6 Perform a multilevel decomposition.

Again, we'll use the graphical tools to emulate what we did in the previous section using command line functions. To perform a level 3 decomposition of the signal using the `db1` wavelet:

Select **3** from the **Level** menu at the upper right, and then click the **Analyze** button again.

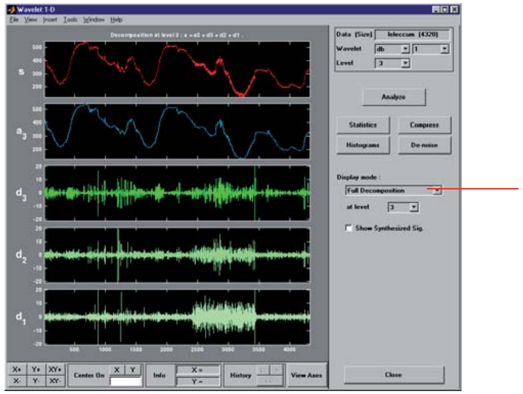


Data (Size) leleccum (4320)

Wavelet db 1

Level 3

After the decomposition is performed, you'll see a new analysis appear in the **Wavelet 1-D** tool.



Data (Size) leleccum (4320)

Wavelet db 1

Level 3

Decomposition at level 3 : s = a3 + d3 + d2 + d1 .

Display mode : Full Decomposition

at level 3

□ Show Synthesized Sig.

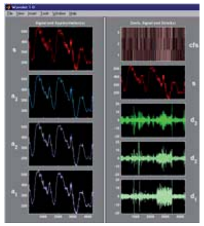
Select a view

Selecting Different Views of the Decomposition

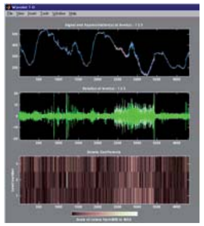
The **Display mode** menu (middle right) lets you choose different views of the wavelet decomposition.

The default display mode is called "Full Decomposition Mode." Other alternatives include:

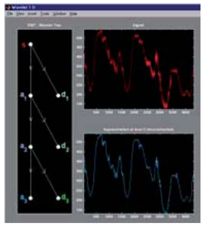
- "Separate Mode," which shows the details and the approximations in separate columns.
- "Superimpose Mode," which shows the details on a single plot superimposed in different colors. The approximations are plotted similarly.
- "Tree Mode," which shows the decomposition tree, the original signal, and one additional component of your choice. Click on the decomposition tree to select the signal component you'd like to view.
- "Show and Scroll Mode," which displays three windows. The first shows the original signal superimposed on an approximation you select. The second window shows a detail you select. The third window shows the wavelet coefficients.
- "Show and Scroll Mode (Stem Cfs)" is very similar to the "Show and Scroll Mode" except that it displays, in the third window, the wavelet coefficients as stem plots instead of colored blocks.



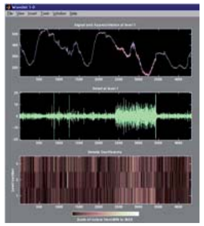
Separate Mode



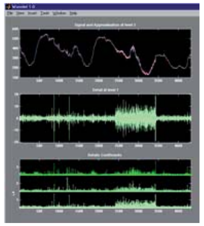
Superimpose Mode



Tree Mode



Show & Scroll Mode

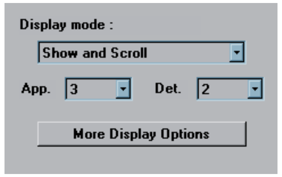


Show & Scroll Mode (Stem Cfs)

You can change the default display mode on a per-session basis. Select the desired mode from the **View > Default Display Mode** submenu.

Note The **Compression** and **De-noising** windows opened from the **Wavelet 1-D** tool will inherit the current coefficient visualization attribute (stems or colored blocks).

Depending on which display mode you select, you may have access to additional display options through the **More Display Options** button (for more information, see "More Display Options" on page A-19).



Display mode : Show and Scroll

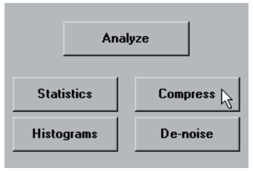
App. 3 Det. 2

These options include the ability to suppress the display of various components, and to choose whether or not to display the original signal along with the details and approximations.

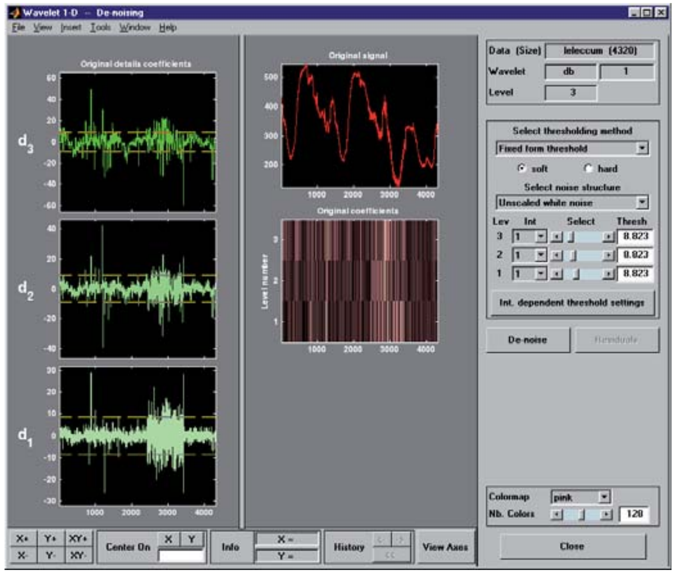
7 Remove noise from a signal.

The graphical interface tools feature a de-noising option with a predefined thresholding strategy. This makes it very easy to remove noise from a signal.

Bring up the de-noising tool: click the **De-noise** button, located in the middle right of the window, underneath the **Analyze** button.



The **Wavelet 1-D De-noising** window appears.



Data (Size) leleccum (4320)

Wavelet db 1

Level 3

Select thresholding method

Fixed form threshold

● soft ○ hard

Select noise structure

Unscaled white noise

Lev	Int	Select	Thresh
3	1		8.823
2	1		8.823
1	1		8.823

Int. dependent threshold settings

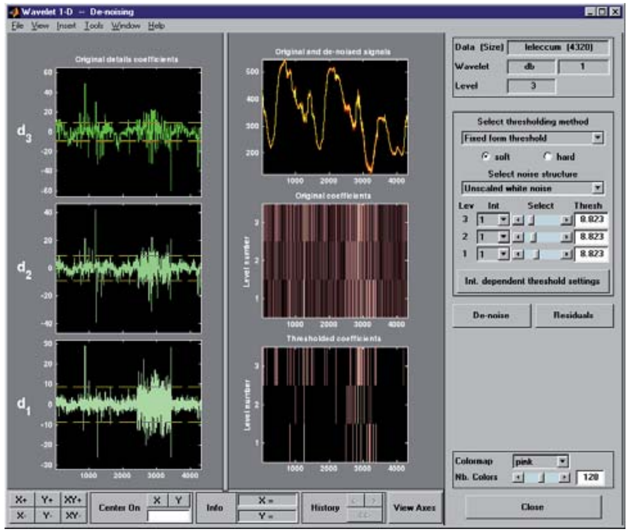
Colormap pink

Nb. Colors 128

While a number of options are available for fine-tuning the de-noising algorithm, we'll accept the defaults of soft fixed form thresholding and unscaled white noise.

Continue by clicking the **De-noise** button.

The de-noised signal appears superimposed on the original. The tool also plots the wavelet coefficients of both signals.



Data (Size) leleccum (4320)

Wavelet db 1

Level 3

Select thresholding method

Fixed form threshold

● soft ○ hard

Select noise structure

Unscaled white noise

Lev	Int	Select	Thresh
3	1		8.823
2	1		8.823
1	1		8.823

Int. dependent threshold settings

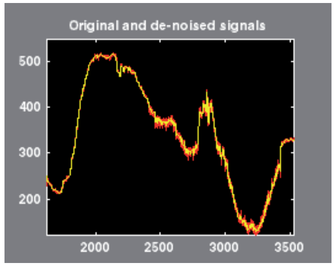
Colormap pink

Nb. Colors 128

Zoom in on the plot of the original and de-noised signals for a closer look.

Drag a rubber band box around the pertinent area, and then click the **XY+** button.

The **De-noise** window magnifies your view. By default, the original signal is shown in red, and the de-noised signal in yellow.



Dismiss the **Wavelet 1-D De-noising** window: click the **Close** button.

You cannot have the **De-noise** and **Compression** windows open simultaneously, so close the **Wavelet 1-D De-noising** window to continue. When the **Update Synthesized Signal** dialog box appears, click **No**. If you click **Yes**, the **Synthesized Signal** is then available in the **Wavelet 1-D** main window.

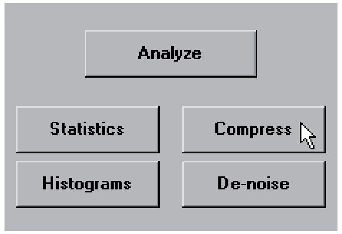
8 Refine the analysis.

The graphical tools make it easy to refine an analysis any time you want to. Up to now, we've looked at a level 3 analysis using `db1`. Let's refine our analysis of the electrical consumption signal using the `db3` wavelet at level 5.

Select 5 from the **Level** menu at the upper right, and select the `db3` from the **Wavelet** menu. Click the **Analyze** button.

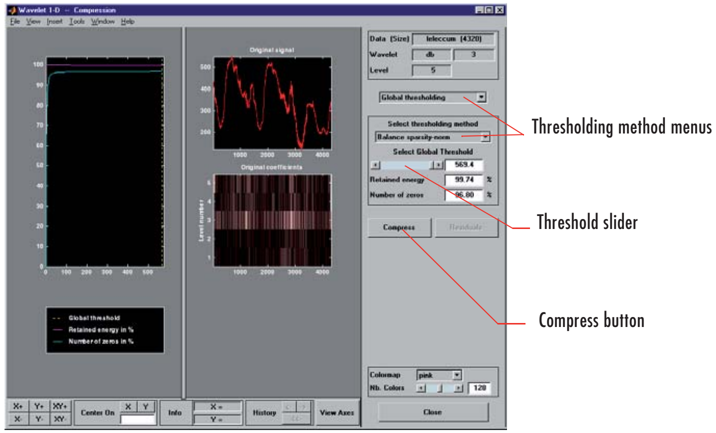
9 Compress the signal.

The graphical interface tools feature a compression option with automatic or manual thresholding.



Bring up the **Compression** window: click the **Compress** button, located in the middle right of the window, underneath the **Analyze** button.

The **Compression** window appears.

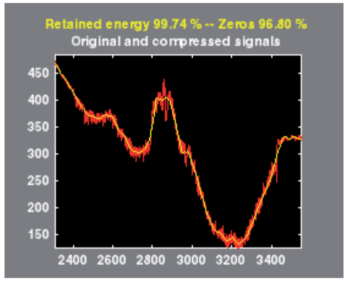


While you always have the option of choosing by level thresholding, here we'll take advantage of the global thresholding feature for quick and easy compression.

Note If you want to experiment with manual thresholding, choose the **By Level thresholding** option from the menu located at the top right of the **Wavelet 1-D Compression** window. The sliders located below this menu then control the level-dependent thresholds, indicated by yellow dotted lines running horizontally through the graphs on the left of the window. The yellow dotted lines can also be dragged directly using the left mouse button.

Click the **Compress** button, located at the center right.

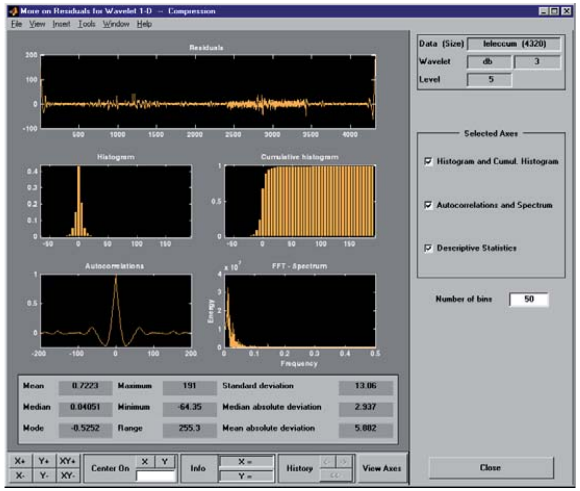
After a pause for computation, the electrical consumption signal is redisplayed in red with the compressed version superimposed in yellow. Below, we've zoomed in to get a closer look at the noisy part of the signal.



You can see that the compression process removed most of the noise, but preserved 99.74% of the energy of the signal. The automatic thresholding was very efficient, zeroing out all but 3.2% of the wavelet coefficients.

10 Show the residuals.

From the **Wavelet 1-D Compression** tool, click the **Residuals** button. The **More on Residuals for Wavelet 1-D Compression** window appears.


More on Residuals for Wavelet 1-D -- Compression

Data (Size): leleccum (4320)
Wavelet: db 3
Level: 5

Mean	0.7223	Maximum	191	Standard deviation	13.06
Median	0.04051	Minimum	-64.35	Median absolute deviation	2.937
Mode	-0.5252	Range	255.3	Mean absolute deviation	5.882

Selected Axes
☑ Histogram and Cumul. Histogram
☑ Autocorrelations and Spectrum
☑ Descriptive Statistics

Number of bins: 50

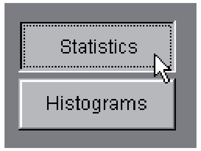
Displayed statistics include measures of tendency (mean, mode, median) and dispersion (range, standard deviation). In addition, the tool provides frequency-distribution diagrams (histograms and cumulative histograms), as well as time-series diagrams: autocorrelation function and spectrum. The same feature exists for the **Wavelet 1-D De-noising** tool.

Dismiss the **Wavelet 1-D Compression** window: click the **Close** button. When the **Update Synthesized Signal** dialog box appears, click **No**.

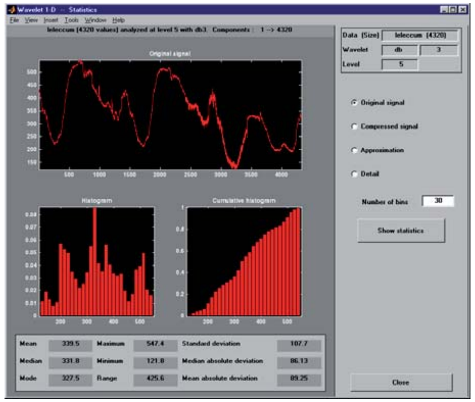
11 Show statistics.

You can view a variety of statistics about your signal and its components.

From the **Wavelet 1-D** tool, click the **Statistics** button.



The **Wavelet 1-D Statistics** window appears displaying by default statistics on the original signal.

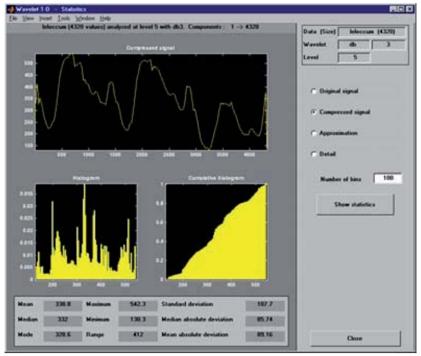


leleccum (4320 values) analyzed at level 5 with db3. Components : 1 --> 4320

Data (Size)	leleccum (4320)
Wavelet	db 3
Level	5
Number of bins	30

Mean	339.5	Maximum	547.4	Standard deviation	107.7
Median	331.8	Minimum	121.8	Median absolute deviation	86.13
Mode	327.5	Range	425.6	Mean absolute deviation	89.25

Select the synthesized signal or signal component whose statistics you want to examine. Click the appropriate option button, and then click the **Show Statistics** button. Here, we've chosen to examine the compressed signal using more 100 bins instead of 30, which is the default:



leleccum (4320 values) analyzed at level 5 with db3. Components : 1 --> 4320

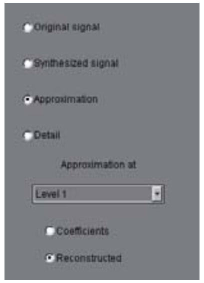
Data (Size)	leleccum (4320)
Wavelet	db 3
Level	5
Number of bins	100

Mean	338.8	Maximum	542.3	Standard deviation	107.7
Median	332	Minimum	130.3	Median absolute deviation	85.74
Mode	328.6	Range	412	Mean absolute deviation	89.16

Displayed statistics include measures of tendency (mean, mode, median) and dispersion (range, standard deviation).

In addition, the tool provides frequency-distribution diagrams (histograms and cumulative histograms). You can plot these histograms separately using the **Histograms** button from the **Wavelets 1-D** window.

Click the **Approximation** option button. A menu appears from which you choose the level of the approximation you want to examine.



Select Level 1 and again click the **Show Statistics** button. Statistics appear for the level 1 approximation.

Wavelet Applications

This chapter explores various applications of wavelets by presenting a series of sample analyses.

Introduction to Wavelet Analysis

Each example is followed by a discussion of the usefulness of wavelet analysis for the particular application area under consideration.

Use the graphical interface tools to follow along:

1 From the MATLAB® command line, type

```
wavemenu
```

2 Click on **Wavelets 1-D** (or another tool as appropriate).

3 Load the sample analysis by selecting the appropriate submenu item from **File⇒Example Analysis**.

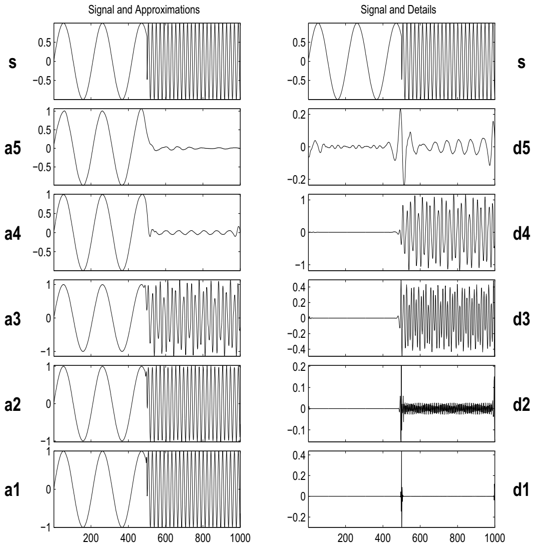
Feel free to explore on your own — use the different options provided in the graphical interface to look at different components of the signal, to compress or de-noise the signal, to examine signal statistics, or to zoom in and out on different signal features.

If you want, try loading the corresponding MAT-file from the MATLAB command line, and use Wavelet Toolbox™ functions to further investigate the sample signals. The MAT-files are located in the directory `toolbox/wavelet/wavedemo`.

There are also other signals in the `wavedemo` directory that you can analyze on your own.

Detecting Discontinuities and Breakdown Points I

The purpose of this example is to show how analysis by wavelets can detect the exact instant when a signal changes. The discontinuous signal consists of a slow sine wave abruptly followed by a medium sine wave.



Example Analysis
Frequency breakdown

MAT-file
`freqbrk.mat`

Wavelet
`db5`

Level
5

The first- and second-level details (`D`$_1$ and `D`$_2$) show the discontinuity most clearly, because the rupture contains the high-frequency part. Note that if we were *only* interested in identifying the discontinuity, `db1` would be a more useful wavelet to use for the analysis than `db5`.

The discontinuity is localized very precisely: only a small domain around time = 500 contains any large first- or second-level details.

Here is a noteworthy example of an important advantage of wavelet analysis over Fourier. If the same signal had been analyzed by the Fourier transform, we would not have been able to detect the instant when the signal's frequency changed, whereas it is clearly observable here.

Details $\mathtt{D}_3$ and $\mathtt{D}_4$ contain the medium sine wave. The slow sine is clearly isolated in approximation $\mathtt{A}_5$, from which the higher-frequency information has been filtered.

Discussion

The deterministic part of the signal may undergo abrupt changes such as a jump, or a sharp change in the first or second derivative. In image processing, one of the major problems is edge detection, which also involves detecting abrupt changes. Also in this category, we find signals with very rapid evolutions such as transient signals in dynamic systems.

The main characteristic of these phenomena is that the change is localized in time or in space.

The purpose of the analysis is to determine

- The site of the change (e.g., time or position)
- The type of change (a rupture of the signal, or an abrupt change in its first or second derivative)
- The amplitude of the change

The local aspects of wavelet analysis are well adapted for processing this type of event, as the processing scales are linked to the speed of the change.

Guidelines for Detecting Discontinuities

Short wavelets are often more effective than long ones in detecting a signal rupture. In the initial analysis scales, the support is small enough to allow fine analysis. The shapes of discontinuities that can be identified by the smallest wavelets are simpler than those that can be identified by the longest wavelets. Therefore, to identify

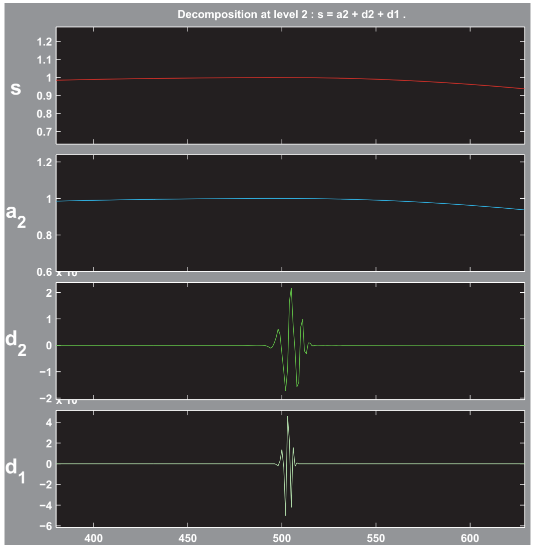
- A signal discontinuity, use the `haar` wavelet
- A rupture in the *j*-th derivative, select a sufficiently regular wavelet with at least *j* vanishing moments. (See "Detecting Discontinuities and Breakdown Points II" on page 3-6.)

The presence of noise, which is after all a fairly common situation in signal processing, makes identification of discontinuities more complicated. If the first levels of the decomposition can be used to eliminate a large part of the noise, the rupture is sometimes visible at deeper levels in the decomposition.

Check, for example, the sample analysis **File⇒Example Analysis⇒Basic Signals⇒ramp + white noise** (MAT-file `wnoislop`). The rupture is visible in the level-six approximation ($\mathtt{A}_6$) of this signal.

Detecting Discontinuities and Breakdown Points II

The purpose of this example is to show how analysis by wavelets can detect a discontinuity in one of a signal's derivatives. The signal, while apparently a single smooth curve, is actually composed of two separate exponentials that are connected at time = 500. The discontinuity occurs only in the second derivative, at time = 500.



Example Analysis
Second derivative breakdown

MAT-file
`scddvbrk.mat`

Wavelet
`db4`

Level
2

We have zoomed in on the middle part of the signal to show more clearly what happens around time = 500. The details are high only in the middle of the signal and are negligible elsewhere. This suggests the presence of high-frequency information — a sudden change or discontinuity — around time = 500.

Discussion

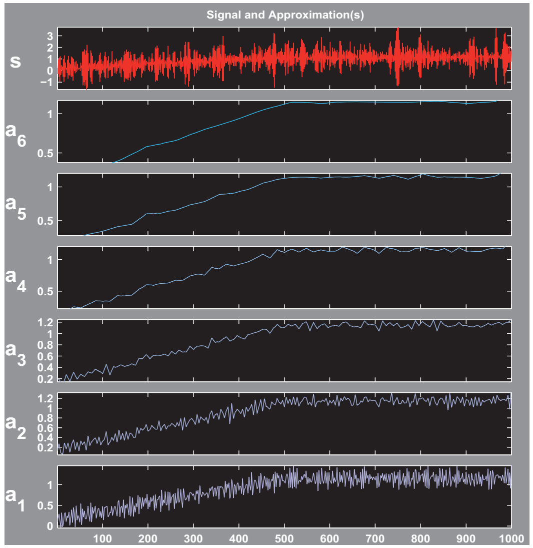
Regularity can be an important criterion in selecting a wavelet. We have chosen to use `db4`, which is sufficiently regular for this analysis. Had we chosen the `haar` wavelet, the discontinuity would not have been detected. If you try repeating this analysis using `haar` at level two, you'll notice that the details are equal to zero at time = 500.

Note that to detect a singularity, the selected wavelet must be sufficiently regular, which implies a longer filter impulse response.

See "Frequently Asked Questions" on page 6-61 and "Wavelet Families: Additional Discussion" on page 6-71 for a discussion of the mathematical meaning of regularity and a comparison of the regularity of various wavelets.

Detecting Long-Term Evolution

The purpose of this example is to show how analysis by wavelets can detect the overall trend of a signal. The signal in this case is a ramp obscured by “colored” (limited-spectrum) noise. (We have zoomed in along the *x*-axis to avoid showing edge effects.)



Example Analysis

Ramp + colored noise

MAT-file

`cnoislop.mat`

Wavelet

`db3`

Level

6

There is so much noise in the original signal, `s`, that its overall shape is not apparent upon visual inspection. In this level-6 analysis, we note that the trend becomes more and more clear with each approximation, `A1` to `A6`. Why is this?

The trend represents the slowest part of the signal. In wavelet analysis terms, this corresponds to the greatest scale value. As the scale increases, the resolution decreases, producing a better estimate of the unknown trend.

Another way to think of this is in terms of frequency. Successive approximations possess progressively less high-frequency information. With the higher frequencies removed, what’s left is the overall trend of the signal.

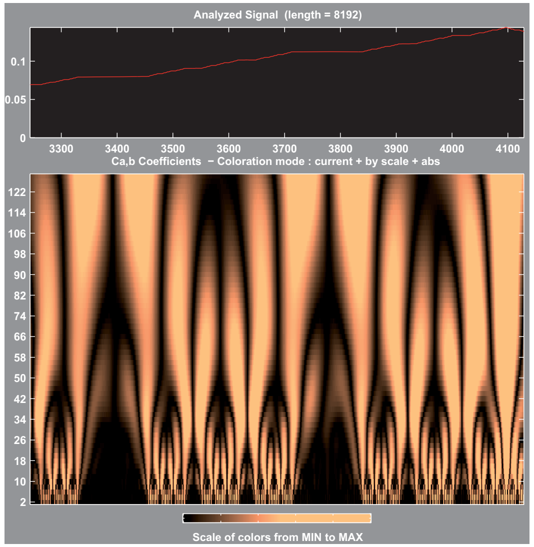
Discussion

Wavelet analysis is useful in revealing signal trends, a goal that is complementary to the one of revealing a signal hidden in noise. It’s important to remember that the trend is the slowest part of the signal. If the signal itself includes sharp changes, then successive approximations look less and less similar to the original signal.

Consider the demo analysis **File⇒Example Analysis⇒Basic Signals⇒Step signal** (MAT-file `wstep.mat`). It is instructive to analyze this signal using the **Wavelet 1-D** tool and see what happens to the successive approximations. Try it.

Detecting Self-Similarity

The purpose of this example is to show how analysis by wavelets can detect a self-similar, or fractal, signal. The signal here is the Koch curve — a synthetic signal that is built recursively.



Example Analysis
Koch curve
MAT-file
`vonkoch.mat`
Wavelet
`coif3`
Level
Continuous, 2:2:128

This analysis was performed with the **Continuous Wavelet 1-D** graphical tool. A repeating pattern in the wavelet coefficients plot is characteristic of a signal that looks similar on many scales.

Wavelet Coefficients and Self-Similarity

From an intuitive point of view, the wavelet decomposition consists of calculating a “resemblance index” between the signal and the wavelet. If the index is large, the resemblance is strong, otherwise it is slight. The indices are the wavelet coefficients.

If a signal is similar to itself at different scales, then the “resemblance index” or wavelet coefficients also will be similar at different scales. In the coefficients plot, which shows scale on the vertical axis, this self-similarity generates a characteristic pattern.

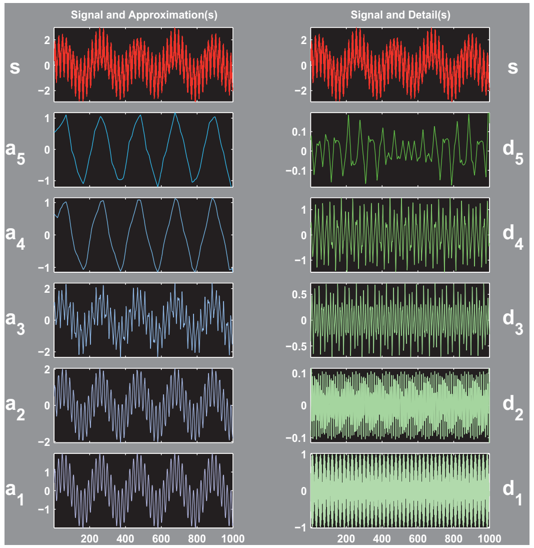
Discussion

The work of many authors and the trials that they have carried out suggest that wavelet decomposition is very well adapted to the study of the fractal properties of signals and images.

When the characteristics of a fractal evolve with time and become local, the signal is called a *multifractal*. The wavelets then are an especially suitable tool for practical analysis and generation.

Identifying Pure Frequencies

The purpose of this example is to show how analysis by wavelets can effectively perform what is thought of as a Fourier-type function — that is, resolving a signal into constituent sinusoids of different frequencies. The signal is a sum of three pure sine waves.



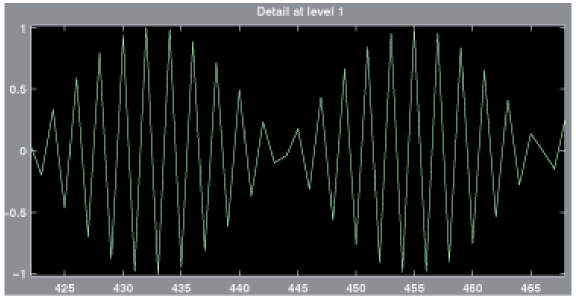
Example Analysis	Sum of sines
MAT-file	`sumsin.mat`
Wavelet	`db3`
Level	5

Discussion

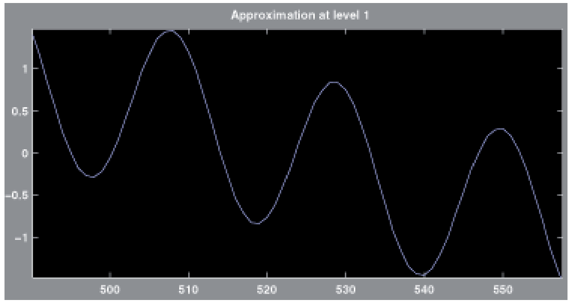
The signal is a sum of three sines: slow, medium, and rapid, which have periods (relative to the sampling period of 1) of 200, 20, and 2, respectively.

The slow, medium, and rapid sinusoids appear most clearly in approximation `A4`, detail `D4`, and detail `D1`, respectively. The slight differences that can be observed on the decompositions can be attributed to the sampling period.

Detail `D1` contains primarily the signal components whose period is between 1 and 2 (i.e., the rapid sine), but this period is not visible at the scale that is used for the graph. Zooming in on detail `D1` (see below) reveals that each "belly" is composed of 10 oscillations, and this can be used to estimate the period. We indeed find that it is close to 2.



The detail `D3` and (to an even greater extent), the detail `D4` contain the medium sine frequencies. We notice that there is a breakdown between approximations `A3` and `A4`, from which the medium frequency information has been subtracted. We should therefore use approximations `A1` to `A3` to estimate the period of the medium sine. Zooming in on `A1` reveals a period of around 20.



Now only the period of the slow sine remains to be determined. Examination of approximation `A4` (see the figure in "Identifying Pure Frequencies" on page 3-12) shows that the distance between two successive maximums is 200.

This slow sine still is visible in approximation A5, but were we to extend this analysis to further levels, we would find that it disappears from the approximation and moves into the details at level 8.

Signal Component	Found In	Period	Frequency
Slow sine	Approximation A4	200	0.005
Medium sine	Detail D4	20	0.05
Rapid sine	Detail D1	2	0.5

This also can be obtained automatically using the scal2frq function, which associates pseudo-frequencies to scales for a given wavelet.

```
lev = [1:5]; a = 2.^lev;     % scales.
wname ='db3';
delta = 1;
f = scal2frq(a,wname,delta); % corresponding pseudo-frequencies.
per  = 1./f;                 % corresponding pseudo-periods.
```

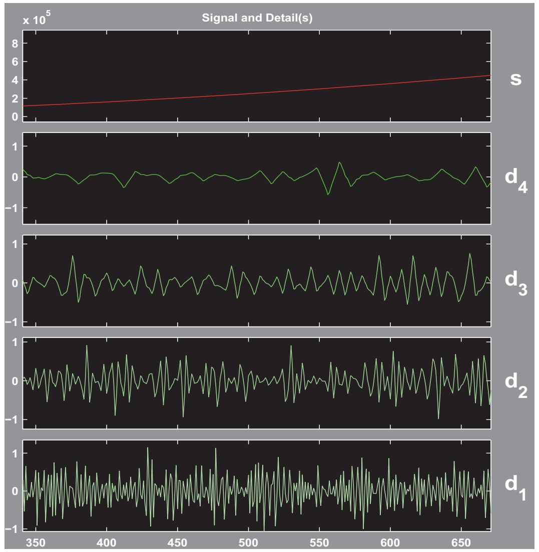
Leading to

Level	Scale	Pseudo-Period	Pseudo-Frequency
1	2	2.5	0.4
2	4	5	0.2
3	8	10	0.1
4	16	20	0.05
5	32	40	0.025

In summation, we have used wavelet analysis to determine the frequencies of pure sinusoidal signal components. We were able to do this because the different frequencies predominate at different scales, and each scale is taken into account by our analysis.

Suppressing Signals

The purpose of this example is to illustrate the property that causes the decomposition of a polynomial to produce null details, provided the number of *vanishing moments* of the wavelet (N for a Daubechies wavelet dbN) exceeds the degree of the polynomial. The signal here is a second-degree polynomial combined with a small amount of white noise.

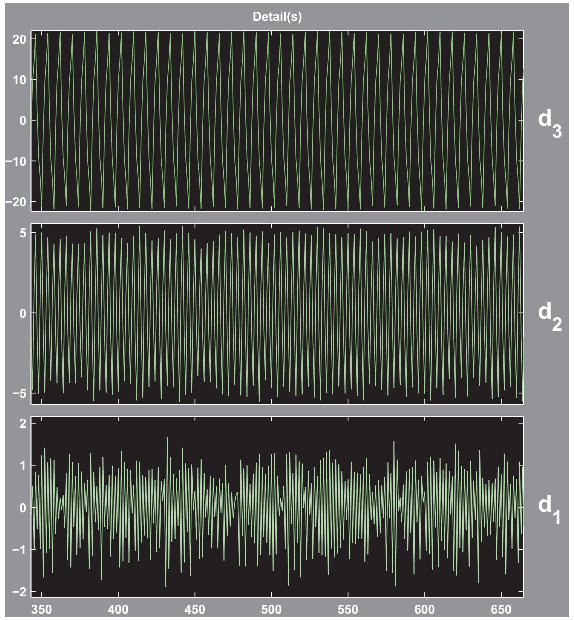


Example Analysis	Noisy polynomial
MAT-file	noispol.mat
Wavelet	db3
Level	4

Note that only the noise comes through in the details. The peak-to-peak magnitude of the details is about 2, while the amplitude of the polynomial signal is on the order of 10^5.

The db3 wavelet, which has three vanishing moments, was used for this analysis. Note that a wavelet of the Daubechies family with fewer vanishing moments would fail to suppress the polynomial signal. For more information, see the section “Daubechies Wavelets: dbN” on page 6-72.

Here is what the first three details look like when we perform the same analysis with db2.



The peak-to-peak magnitudes of the details D1, D2, and D3 are 2, 10, and 40, respectively. These are much higher detail magnitudes than those obtained using db3.

Discussion

For the db2 analysis, the details for levels 2 to 4 show a periodic form that is very regular, and that increases with the level. This is explained by the fact that the detail for level j takes into account primarily the fluctuations of the polynomial function around its mean value on dyadic intervals that are 2^j long. The fluctuations are periodic and very large in relation to the details of the noise decomposition.

On the other hand, for the db3 analysis, we find the presence of white noise thus indicating that the polynomial does not come into play in any of the details. The wavelet suppresses the polynomial part and analyzes the noise.

Suppressing part of a signal allows us to highlight the remainder.

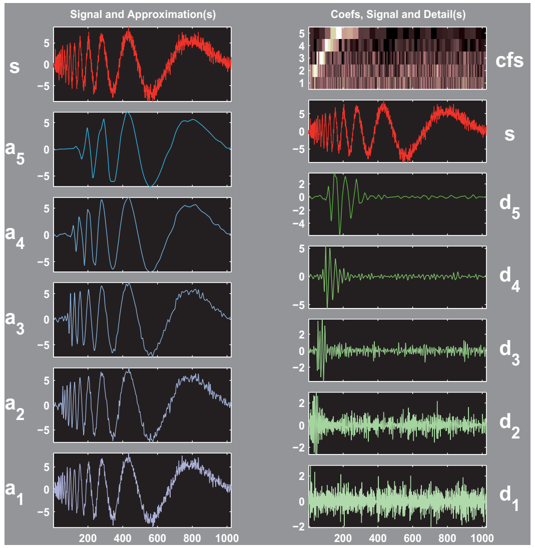
Vanishing Moments

The ability of a wavelet to suppress a polynomial depends on a crucial mathematical characteristic of the wavelet called its number of *vanishing moments*. A technical discussion of vanishing moments appears in the sections “Frequently Asked Questions” on page 6-61 and “Wavelet Families: Additional Discussion” on page 6-71. For the present discussion, it suffices to think of “moment” as an extension of “average.” Since a wavelet’s average value is zero, it has (at least) one vanishing moment.

More precisely, if the average value of $x^k\psi(x)$ is zero (where $\psi(x)$ is the wavelet function), for $k = 0, \ldots, n$, then the wavelet has $n + 1$ vanishing moments and polynomials of degree n are suppressed by this wavelet.

De-Noising Signals

The purpose of this example is to show how to de-noise a signal using wavelet analysis. This example also gives us an opportunity to demonstrate the automatic thresholding feature of the **Wavelet 1-D** graphical interface tool. The signal to be analyzed is a Doppler-shifted sinusoid with some added noise.

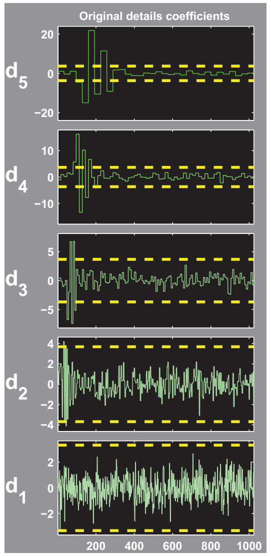


Example Analysis	Noisy Doppler
MAT-file	`noisdopp.mat`
Wavelet	`sym4`
Level	5

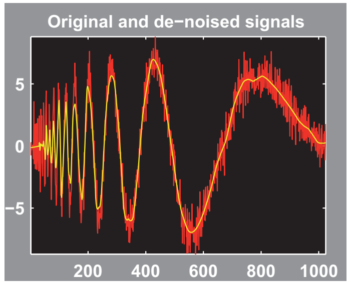
Discussion

We note that the highest frequencies appear at the start of the original signal. The successive approximations appear less and less noisy; however, they also lose progressively more high-frequency information. In approximation `A5`, for example, about the first 20% of the signal is truncated.

Click the **De-noise** button to bring up the **Wavelet 1-D De-Noising** window. This window shows each detail along with its automatically set de-noising threshold.



Click the **De-noise** button. On the screen, the original and de-noised signals appear superimposed in red and yellow, respectively.



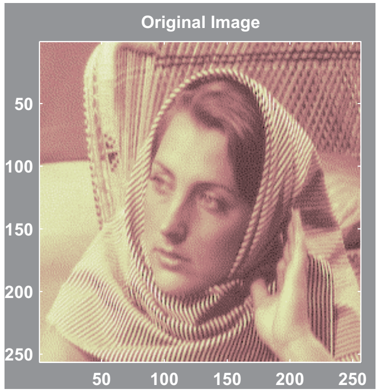
Note that the de-noised signal is flat initially. Some of the highest-frequency signal information was lost during the de-noising process, although less was lost here than in the higher level approximations A4 and A5.

For this signal, wavelet packet analysis does a better job of removing the noise without compromising the high-frequency information. Explore on your own: try repeating this analysis using the **Wavelet Packet 1-D** tool. Select the menu item **File⇒Example Analysis⇒noisdopp**.

De-Noising Images

The purpose of this example is to show how to de-noise an image using both a two-dimensional wavelet analysis and a two-dimensional stationary wavelet analysis. De-noising is one of the most important applications of wavelets.

The image to be de-noised is a noisy version of a piece of the following image.



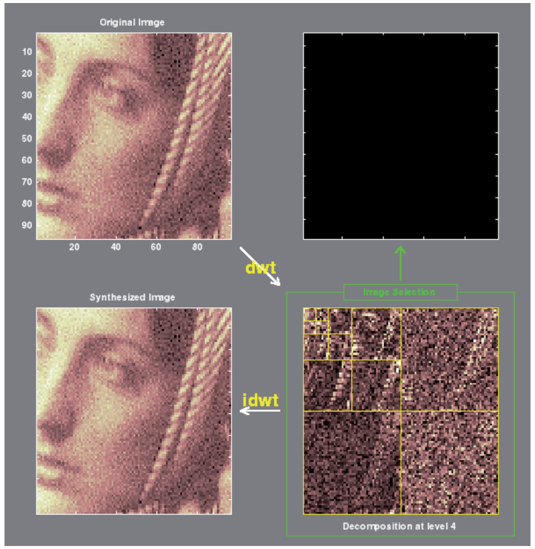
For this example, switch the extension mode to symmetric padding using the command

```
dwtmode('sym')
```

Open the **Wavelet 2-D** tool, select from the **File** menu the **Load Image** option, and select the MAT-file noiswom.mat, which should reside in the MATLAB® directory toolbox/wavelet/wavedemo.

The image is loaded into the **Wavelet 2-D** tool. Select the haar wavelet and select **4** from the level menu, and then click the **Analyze** button.

The analysis appears in the **Wavelet 2-D** window.

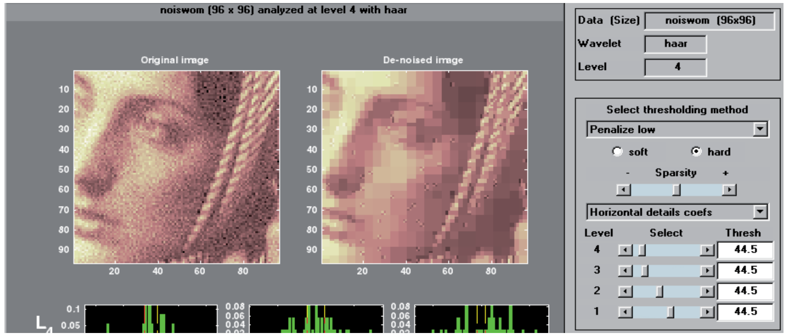


Example Analysis
Noisy Woman
MAT-file
`noiswom.mat`
Wavelet
`haar`
Level
4

Click the **De-noise** button (located at the middle right) to bring up the **Wavelet 2-D -- De-noising** window.

Discussion

The graphical tool provides automatically generated thresholds. From the **Select thresholding method** menu, select the item **Penalize low** and click the **De-noise** button.



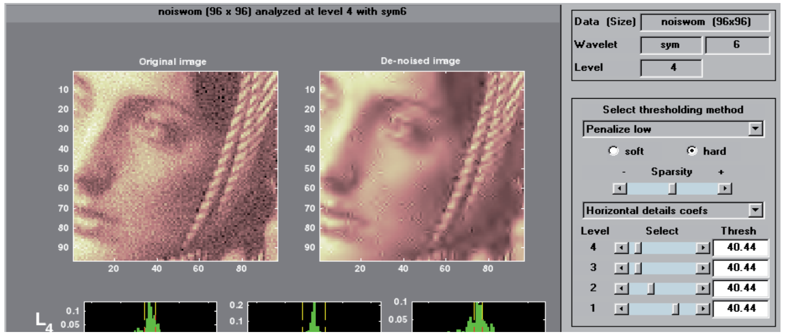
Data (Size) noiswom (96x96)
Wavelet haar
Level 4

Select thresholding method
Penalize low
○ soft ● hard
- Sparsity +
Horizontal details coefs

Level	Select	Thresh
4		44.5
3		44.5
2		44.5
1		44.5

The de-noised image exhibits some blocking effects. Let's try another wavelet. Click the **Close** button to go back to the **Wavelet 2-D** window. Select the `sym6` wavelet, and then click the **Analyze** button. Click the **De-noise** button to bring up the **Wavelet 2-D -- De-noising** window again.

From the **Select thresholding method** menu, select the item **Penalize low**, and click the **De-noise** button.



Data (Size) noiswom (96x96)
Wavelet sym 6
Level 4

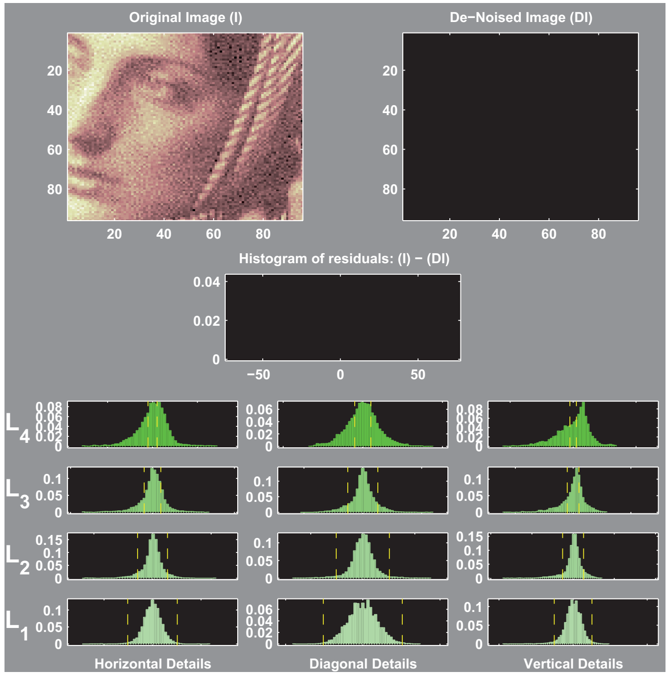
Select thresholding method
Penalize low
○ soft ● hard
- Sparsity +
Horizontal details coefs

Level	Select	Thresh
4		40.44
3		40.44
2		40.44
1		40.44

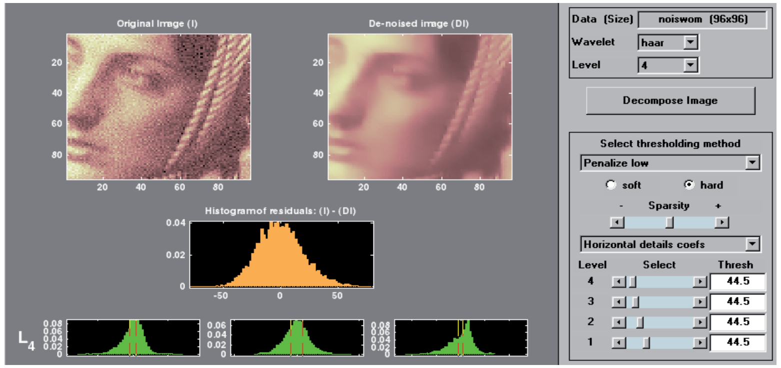
The de-noised image exhibits some ringing effects. Let's try another strategy based on the two-dimensional stationary wavelet analysis to de-noise images. The basic idea is to average many slightly different discrete wavelet analyses.

For more information, see the section “Discrete Stationary Wavelet Transform (SWT)” on page 6-44.

Click the **Close** button to go back to the **Wavelet 2-D** window and click the **Close** button again. Open the **SWT De-noising 2-D** tool, select from the **File** menu the **Load Image** option and select the MAT-file `noiswom.mat`. Select the `haar` wavelet and select **4** from the level menu, and then click the **Decompose Image** button.



The selected thresholding method is **Penalize low**. Use the **Sparsity** slider to adjust the threshold value close to 44.5 (the same as before to facilitate the comparison with the first trial), and then click the **De-noise** button.

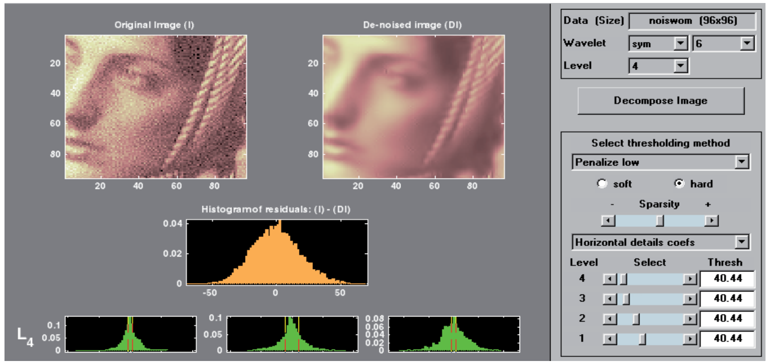


Data (Size) noiswom (96x96)
Wavelet haar
Level 4
Penalize low
○ soft ● hard
Horizontal details coefs

Level	Select	Thresh
4		44.5
3		44.5
2		44.5
1		44.5

The result is more satisfactory. It’s possible to improve it slightly.

Select the `sym6` wavelet and click the **Decompose Image** button. Use the **Sparsity** slider to adjust the threshold value close to 40.44 (the same as before to facilitate the comparison with the second trial), and then click the **De-noise** button.



Data (Size) noiswom (96x96)
Wavelet sym 6
Level 4
Penalize low
○ soft ● hard
Horizontal details coefs

Level	Select	Thresh
4		40.44
3		40.44
2		40.44
1		40.44

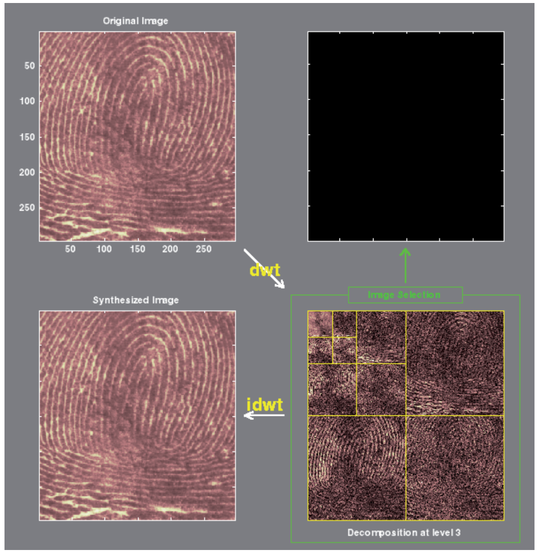
At the end of this example, turn back the extension mode to zero-padding using the command

```
dwtmode('zpd')
```

Compressing Images

The purpose of this example is to show how to compress an image using two-dimensional wavelet analysis. Compression is one of the most important applications of wavelets. The image to be compressed is a fingerprint.

For this example, open the **Wavelet 2-D** tool and select the menu item **File⇒Example Analysis⇒at level 3, with haar —> finger**.

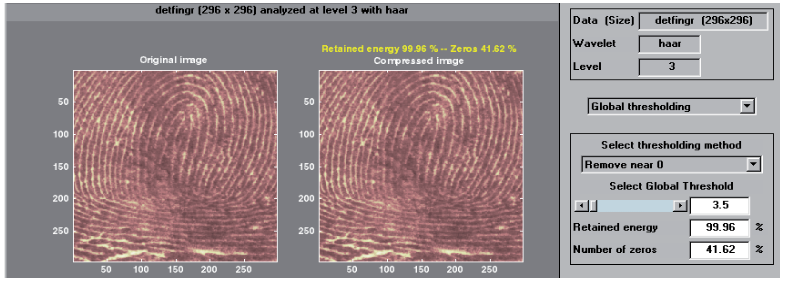


Example Analysis
Finger
MAT-file
`detfingr.mat`
Wavelet
`haar`
Level
3

The analysis appears in the **Wavelet 2-D** tool. Click the **Compress** button (located at the middle right) to bring up the **Wavelet 2-D Compression** window.

Discussion

The graphical tool provides an automatically generated threshold. From the **Select thresholding method** menu, select **Remove near 0**, setting the threshold to 3.5. Then, click the **Compress** button. Values under the threshold are forced to zero, achieving about 42% zeros while retaining almost all (99.96%) the energy of the original image.

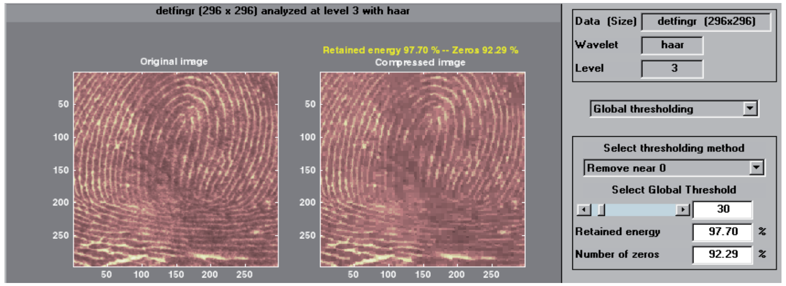

detfingr (296 x 296) analyzed at level 3 with haar

Retained energy 99.96 % -- Zeros 41.62 %

Data (Size) detfingr (296x296)
Wavelet haar
Level 3
Global thresholding
Select thresholding method: Remove near 0
Select Global Threshold 3.5
Retained energy 99.96 %
Number of zeros 41.62 %

The automatic thresholds usually achieve reasonable and various balances between the number of zeros and retained image energy. Depending on your data and your analysis criteria, you may find setting more or less aggressive thresholds achieves better results.

Here we've set the global threshold to around 30. This results in a compressed image consisting of about 92% zeros with 97.7% retained energy.


detfingr (296 x 296) analyzed at level 3 with haar

Retained energy 97.70 % -- Zeros 92.29 %

Data (Size) detfingr (296x296)
Wavelet haar
Level 3
Global thresholding
Select thresholding method: Remove near 0
Select Global Threshold 30
Retained energy 97.70 %
Number of zeros 92.29 %

Fast Multiplication of Large Matrices

This section illustrates matrix-vector multiplication in the wavelet domain.

- The problem is

  let *m* be a dense matrix of large size (*n*, *n*). We want to perform a large number, *L*, of multiplications of *m* by vectors *v*.
- The idea is

  Stage 1: (executed once) Compute the matrix approximation, *sm,* at a suitable level *k*. The matrix will be assimilated with an image.

  Stage 2: (executed *L* times) divided in the following three steps:

  a Compute vector approximation.

  b Compute multiplication in wavelet domain.

  c Reconstruct vector approximation.

It is clear that when *sm* is a sufficiently good approximation of *m*, the error with respect to ordinary multiplication can be small. This is the case in the first example below where *m* is a magic square. Conversely, when the wavelet representation of the matrix *m* is dense the error will be large (for example, if all the coefficients have the same order of magnitude). This is the case in the second example below where *m* is two-dimensional Gaussian white noise. The figure in Example 1 compares for $n = 512$, the number of floating point operations (flops) required by wavelet based method and by ordinary method versus *L*.

Example 1: Effective Fast Matrix Multiplication

```
n = 512;
lev = 5;
wav = 'db1';

% Wavelet based matrix multiplication by a vector:
% a  good  example
% Matrix is magic(512) Vector is (1:512)

m = magic(n);
v = (1:n)';
[Lo_D,Hi_D,Lo_R,Hi_R] = wfilters(wav);

% ordinary matrix multiplication by a vector.
p = m * v;

% The number of floating point operations used is 524,288

% Compute matrix approximation at level 5.
sm = m;
for i = 1:lev
    sm = dyaddown(conv2(sm,Lo_D),'c');
    sm = dyaddown(conv2(sm,Lo_D'),'r');
end

% The number of floating point operations used is 2,095,154

% The three steps:
% 1. Compute vector approximation.
% 2. Compute multiplication in wavelet domain.
% 3. Reconstruct vector approximation.

sv = v;
for i = 1:lev, sv = dyaddown(conv(sv,Lo_D)); end
sp = sm * sv;
for i = 1:lev, sp = conv(dyadup(sp),Lo_R); end
sp = wkeep(sp,length(v));

% Now, the number of floating point operations used is 9058
% Relative square norm error in percent when using wavelets.
rnrm = 100 * (norm(p-sp)/norm(p))

rnrm =
        2.9744e-06
```

Example 2: Ineffective Fast Matrix Multiplication

The commands used are the same as in Example 1, but applied to a new matrix *m*.

```
% Wavelet based matrix multiplication by a vector:
% a  bad  example with a randn matrix.
% Change the matrix m of example1 using:
randn('state',0);
m = randn(n,n);
```

Then, you obtain

```
% Relative square norm error in percent
rnrm = 100 * (norm(p-sp)/norm(p))

rnrm =
        99.2137
```

4

Wavelets in Action: Examples and Case Studies

This chapter presents examples of wavelet decomposition. Suggested areas for further exploration follow most examples, along with a summary of the topics addressed by that example. This chapter also includes a case study that examines the practical uses of wavelet analysis in greater detail, as well as a demonstration of the application of wavelets for fast multiplication of large matrices. An extended discussion of many of the topics addressed by the examples can be found in “Advanced Concepts” on page 6-1.

Illustrated Examples

Fourteen illustrated examples are included in this section, organized as shown:

Example	Equation	Signal Name	MAT-File
Example 1: A Sum of Sines on page 4-8	A sum of sines: $s_1(t) = \sin(3t) + \sin(0.3t) + \sin(0.03t)$	$s_1(t)$	`sumsin`
Example 2: A Frequency Breakdown on page 4-10	A frequency breakdown: $1 \le t \le 500, \quad s_2(t) = \sin(0.03t)$ $501 \le t \le 1000, \quad s_2(t) = \sin(0.3t)$	$s_2(t)$	`freqbrk`
Example 3: Uniform White Noise on page 4-12	A uniform white noise: on the interval $[-0.5 \quad 0.5]$	$b_1(t)$	`whitnois`
Example 4: Colored AR(3) Noise on page 4-14	A colored AR(3) noise: $b_2(t) = -1.5b_2(t-1) - 0.75b_2(t-2)$ $-0.125b_2(t-3) + b_1(t) + 0.5$	$b_2(t)$	`warma`
Example 5: Polynomial + White Noise on page 4-16	A polynomial + a white noise: on the interval $[1 \quad 1000]$ $s_3(t) = t^2 - t + 1 + b_1(t)$	$s_3(t)$	`noispol`
Example 6: A Step Signal on page 4-18	A step signal: $1 \le t \le 500, \quad s_4(t) = 0$ $501 \le t \le 1000, \quad s_4(t) = 20$	$s_4(t)$	`wstep`
Example 7: Two Proximal Discontinuities on page 4-20	Two proximal discontinuities: $1 \le t \le 499, \quad s_5(t) = 3t$ $500 \le t \le 510, \quad s_5(t) = 1500$ $511 \le t, \quad s_5(t) = 3t - 30$	$s_5(t)$	`nearbrk`
Example 8: A Second-Derivative Discontinuity on page 4-22	A second-derivative discontinuity: $t \in [-0.5 \quad 0.5] \subset R;$ $t < 0, f_3(t) = \exp(-4t^2)$ $t \ge 0, f_3(t) = \exp(-t^2)$ s_6 is f_3 sampled at 10^{-3}	$s_6(t)$	`scddvbrk`
Example 9: A Ramp + White Noise on page 4-24	A ramp + a white noise: $1 \le t \le 499, \quad s_7(t) = \frac{3t}{500} + b_1(t)$ $500 \le t \le 1000, \quad s_7(t) = 3 + b_1(t)$	$s_7(t)$	`wnoislop`
Example 10: A Ramp + Colored Noise on page 4-26	A ramp + a colored noise: $1 \le t \le 499, \quad s_8(t) = \frac{t}{500} + b_2(t)$ $500 \le t \le 1000, \quad s_8(t) = 1 + b_2(t)$	$s_8(t)$	`cnoislop`
Example 11: A Sine + White Noise on page 4-28	A sine + a white noise: $s_9(t) = \sin(0.03t) + b_1(t)$	$s_9(t)$	`noissin`

Example	Equation	Signal Name	MAT-File
Example 12: A Triangle + A Sine on page 4-30	A triangle + a sine: $1 \le t \le 500, \quad s_{10}(t) = \frac{t-1}{500} + \sin(0.3t)$ $501 \le t \le 1000, \quad s_{10}(t) = \frac{1000-t}{500} + \sin(0.3t)$	$s_{10}(t)$	`trsin`
Example 13: A Triangle + A Sine + Noise on page 4-32	A triangle + a sine + a noise: $501 \le t \le 1000,$ $s_{11}(t) = \frac{1000-t}{500} + \sin(0.3t) + b_1(t)$ $1 \le t \le 500, \; s_{11}(t) = \frac{t-1}{500} + \sin(0.3t) + b_1(t)$	$s_{11}(t)$	`wntrsin`
Example 14: A Real Electricity Consumption Signal on page 4-34	A real electricity consumption signal	—	`leleccum`

Please note that

- All the decompositions use Daubechies wavelets.
- The examples show the signal, the approximations, and the details.

The examples include specific comments and feature distinct domains — for instance, if the level of decomposition is 5,

- The left column contains the signal and the approximations A_5 to A_1.
- The right column contains the signal and the details D_5 to D_1.
- The approximation A_1 is located under A_2, A_2 under A_3, and so on; the same is true for the details.
- The abscissa axis represents the time; the unit for the ordinate axis for approximations and details is the same as that of the signal.

- When the approximations do not provide enough information, they are replaced by details obtained by changing wavelets.
- The examples include questions for you to think about:
  - What can be seen on the figure?
  - What additional questions can be studied?

Advice to the Reader

You should follow along and process these examples on your own, using either the graphical interface or the command line functions.

Use the graphical interface for immediate signal processing. To execute the analyses included in the figures,

1. To open the **Wavelet Toolbox Main Menu**, type

   ```
   wavemenu
   ```

2. Select the **Wavelet 1-D** menu option to open the **Wavelet 1-D** tool.
3. From the **Wavelet 1-D** tool, choose the **File⇒Example Analysis** menu option.
4. From the dialog box, select the sample analysis in question.

This triggers the execution of the examples.

When using the command line, follow the process illustrated in this M-file to conduct calculations:

```
% Load original 1-D signal.
    load sumsin; s = sumsin;

% Perform the decomposition of s at level 5, using coif3.
    w = 'coif3'
    [c,l] = wavedec(s,5,w);

% Reconstruct the approximation signals and detail signals at
% levels 1 to 5, using the wavelet decomposition structure [c,l].
for i = 1:5
    A(i,:) = wrcoef('a',c,l,w,i);
```

```
    D(i,:) = wrcoef('d',c,l,w,i);
end
```

Note This loop replaces 10 separate `wrcoef` statements defining approximations and details. The variable `A` contains the five approximations and the variable `D` contains the five details.

```
% Plots.
    t = 100:900;
    subplot(6,2,1); plot(t,s(t),'r');
    title('Orig. signal and approx. 1 to 5.');
    subplot(6,2,2); plot(t,s(t),'r');
    title('Orig. signal and details 1 to 5.');
    for i = 1:5,
        subplot(6,2,2*i+1); plot(t,A(5-i+1,t),'b');
        subplot(6,2,2*i+2); plot(t,D(5-i+1,t),'g');
    end
```

About Further Exploration

Tip 1. On all figures, visually check that for j = 0, 1, ..., $A_j = A_{j+1} + D_{j+1}$.

Tip 2. Don't forget to change wavelets. Test the shortest ones first.

Tip 3. Identify edge effects. They will create problems for a correct analysis. At present, there is no easy way to avoid them perfectly. You can use tools described in the section "Dealing with Border Distortion" on page 6-35 and see also the `dwtmode` reference page. They should eliminate or greatly reduce these effects.

Tip 4. As much as possible, conduct calculations manually to cross-check results with the values in the graphic representations. Manual calculations are possible with the `db1` wavelet.

For the sake of simplicity in the following examples, we use only the `haar` and `db` family wavelets, which are the most frequently used wavelets.

Example 1: A Sum of Sines

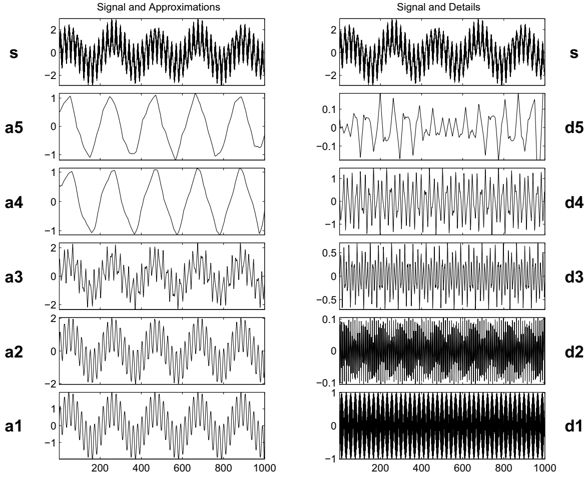
Analyzing wavelet: *db3*

Decomposition levels: *5*

The signal is composed of the sum of three sines: slow, medium, and rapid. With regard to the sampling period equal to 1, the periods are approximately 200, 20, and 2 respectively. We should, therefore, see this later period in D_1, the medium sine in D_4, and the slow sine in A_4. The slight differences that can be observed on the decompositions can be attributed to the sampling period. The scale of the approximation charts is 2, 4, or 10 times larger than that of the details. D_1 contains primarily the components whose period is situated between 1 and 2 (i.e., the rapid sine), but this period is not visible at the scale that is used for the graph. Zooming in on D_1 reveals that each “belly” is composed of 10 oscillations, and can be used to estimate the period. We find that the period is close to 2. D_2 is very small. This is also seen in the approximations: the first two resemble one another, since $A_1 = A_2 + D_2$.

The detail D_3 and, to an even greater extent, the detail D_4 contain the medium sine. We notice that there is a breakdown between approximations 3 and 4.

Approximations A_1 to A_3 can be used to estimate the period of the medium sine. Now, only the slow sine, which appears in A_4, remains to be determined. The distance between two successive maximums is equal to 200, which is the period of the slow sine. This latter sine is still visible in A_5, but will disappear from the approximation and move into the details at level 8.



Example 1: A Sum of Sines	
Addressed topics	• Detecting breakdown points • Detecting long-term evolution • Identifying pure frequencies • The effect of a wavelet on a sine • Details and approximations: a signal moves from an approximation to a detail • The level at which characteristics appear
Further exploration	• Compare with a Fourier analysis. • Change the frequencies. Analyze other linear combinations.

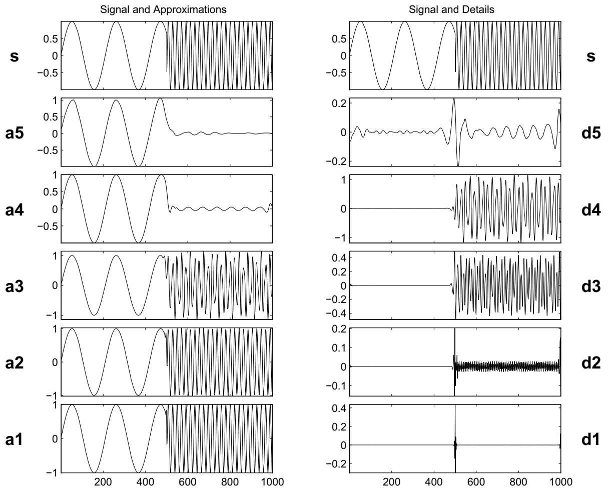
Example 2: A Frequency Breakdown

Analyzing wavelet: *db5*

Decomposition levels: *5*

The signal is formed of a slow sine and a medium sine, on either side of 500. These two sines are not connected in a continuous manner: D_1 and D_2 can be used to detect this discontinuity. It is localized very precisely: only a small domain around 500 contains large details. This is because the rupture contains the high-frequency part; the frequencies in the rest of the signal are not as high. It should be noted that if we are interested only in identifying the discontinuity, `db1` is more useful than `db5`.

D_3 and D_4 contain the medium sine as in the previous analysis. The slow sine appears clearly alone in A_5. It is more regular than in the s_1 analysis, since `db5` is more regular than `db3`. If the same signal had been analyzed by the Fourier transform, we would not have been able to detect the instant corresponding to the signal's frequency change, whereas it is clearly observable here.



Example 2: A Frequency Breakdown	
Addressed topics	• Suppressing signals • Detecting long-term evolution
Further exploration	• Compare to the signal s_1. • On a longer signal, select a deeper level of decomposition in such a way that the slow sinusoid appears into the details. • Compare with a Fourier analysis. • Compare with a windowed Fourier analysis.

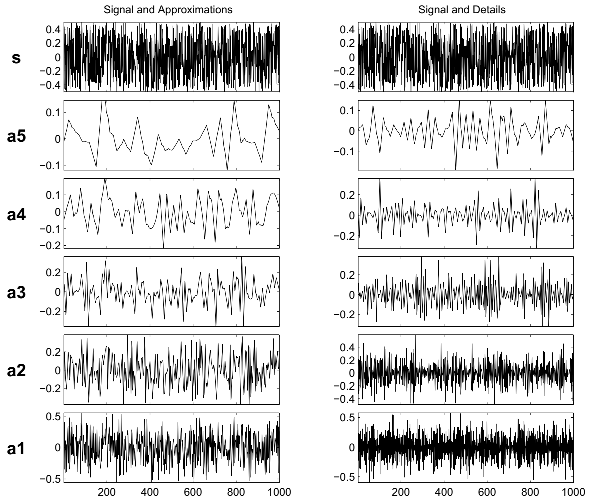
Example 3: Uniform White Noise

Analyzing wavelet: *db3*

Decomposition levels: *5*

At all levels we encounter noise-type signals that are clearly irregular. This is because all the frequencies carry the same energy. The variances, however, decrease regularly between one level and the next as can be seen reading the detail chart (on the right) and the approximations (on the left).

The variance decreases two-fold between one level and the next, i.e., variance(D_j) = variance(D_{j-1}) / 2. Lastly, it should be noted that the details and approximations are not white noise, and that these signals are increasingly interdependent as the resolution decreases. On the other hand, the wavelet coefficients are random, noncorrelated variables. This property is not evident on the reconstructed signals shown here, but it can be guessed at from the representation of the coefficients.



Example 3: Uniform White Noise	
Addressed topics	• Processing noise • The shapes of the decomposition values • The evolution of these shapes according to level; the correlation increases, the variance decreases
Further exploration	• Compare the frequencies included in the details with those in the approximations. • Study the values of the coefficients and their distribution. • On the continuous analysis, identify the chaotic aspect of the colors. • Replace the uniform white noise by a Gaussian white noise or other noise.

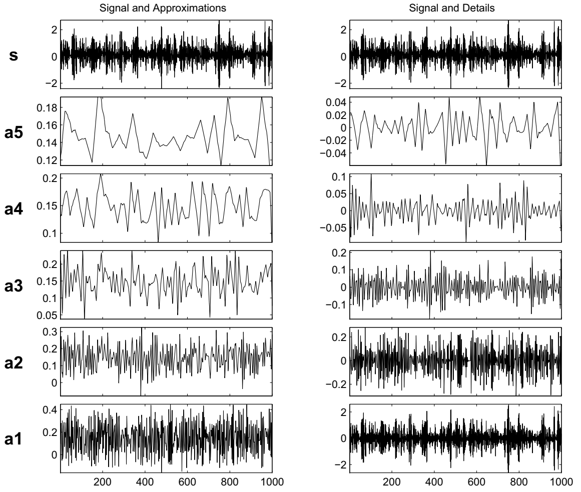
Example 4: Colored AR(3) Noise

Analyzing wavelet: *db3*

Decomposition levels: *5*

Note AR(3) means AutoRegressive model of order 3.

This figure can be examined in view of Example 3: Uniform White Noise on page 4-12, since we are confronted here with a nonwhite noise whose spectrum is mainly at the higher frequencies. Therefore, it is found primarily in D_1, which contains the major portion of the signal. In this situation, which is commonly encountered in practice, the effects of the noise on the analysis decrease considerably more rapidly than in the case of white noise. In A_3, A_4, and A_5, we encounter the same scheme as that in the analysis of b_1 (see the table in "Example 3: Uniform White Noise" on page 4-12), the noise from which b_2 is built using linear filtering. (b_1 and b_2 are defined explicitly in "Illustrated Examples" on page 4-2, Examples 3 and 4.)



Example 4: Colored AR(3) Noise	
Addressed topics	• Processing noise • The relative importance of different details • The relative importance of D_1 and A_1
Further exploration	• Compare the detail frequencies with those in the approximations. • Compare approximations A_3, A_4, and A_5 with those shown in Example 3: Uniform White Noise on page 4-12. • Replace AR(3) with an ARMA (AutoRegressive Moving Average) model noise. For instance, $b_3(t) = -1.5b_3(t-1) - 0.75b_3(t-2) - 0.125b_3(t-3) + b_1(t) - 0.7b_1(t-1)$ • Study an ARIMA (Integrated ARMA) model noise. For instance, $b_4(t) = b_4(t-1) + b_3(t)$ • Check that each detail can be modeled by an ARMA process.

Example 5: Polynomial + White Noise

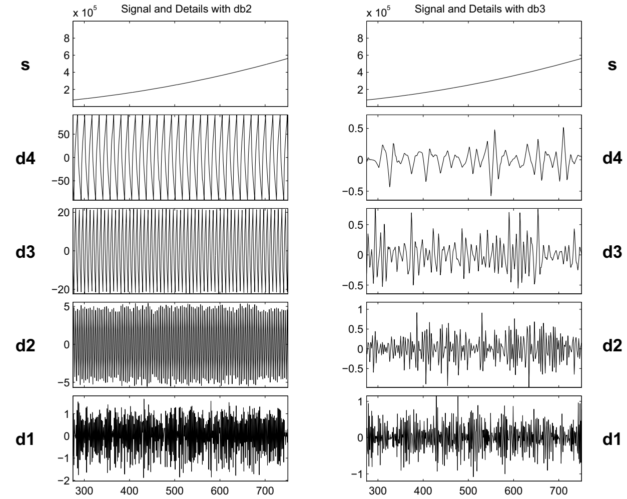
Analyzing wavelets: *db2* and *db3*

Decomposition levels: *4*

The purpose of this analysis is to illustrate the property that causes the decomposition by `dbN` of a `p`-degree polynomial to produce null details as long as `N` > `p`. In this case, `p`=2 and we examine the first four levels of details for two values of N: one is too small, `N`=2 on the left, and the other is sufficient, `N`=3 on the right. The approximations are left out since they differ very little from the signal itself.

For `db2` (on the left), we obtain the decomposition of $t^2 + b_1(t)$, since the $-t + 1$ part of the signal is suppressed by the wavelet. In fact, with the exception of level 1, where noise-generated irregularities can be seen, the details for levels 2 to 4 show a periodic form that is very regular, and which increases with the level. This is because the detail for level `j` takes into account that the fluctuations of the function around its mean value on dyadic intervals are long. The fluctuations are periodic and very large in relation to the details of the noise decomposition.

On the other hand, for `db3` (on the right) we again find the presence of white noise, thus indicating that the polynomial does not come into play in any of the details. The wavelet suppresses the polynomial part and analyzes the noise.



Example 5: Polynomial + White Noise	
Addressed topics	• Suppressing signals • Compare the results of the processing for the following wavelets: the short `db2` and the longer `db3`. • Explain the regularity that is visible in D_3 and D_4 in the analysis by `db2`.
Further exploration	• Increase noise intensity and repeat the analysis.

Example 6: A Step Signal

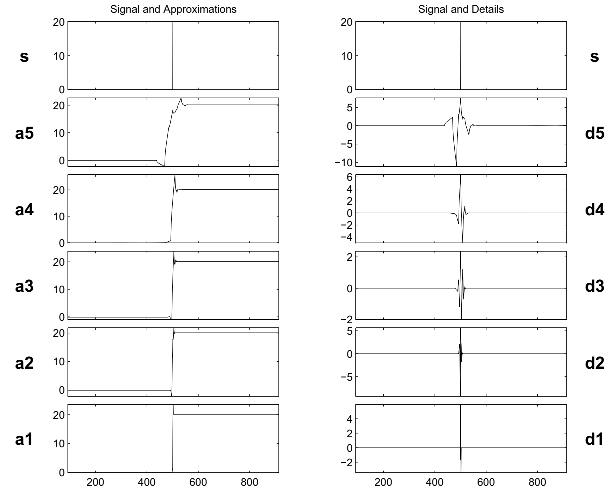
Analyzing wavelet: *db2*

Decomposition levels: *5*

In this case, we are faced with the simplest example of a rupture (i.e., a step). The time instant when the jump occurs is equal to 500. The break is detected at all levels, but it is obviously detected with greater precision in the higher resolutions (levels 1 and 2) than in the lower ones (levels 4 and 5). It is very precisely localized at level 1, where only a very small zone around the jump time can be seen.

It should be noted that the reconstructed details are primarily composed of the basic wavelet represented in the initial time.

Furthermore, the rupture is more precisely localized when the wavelet corresponds to a short filter.



Example 6: A Step Signal	
Addressed topics	• Detecting breakdown points • Suppressing signals • Detecting long-term evolution • Identifying the range width of the variations of details and approximations
Further exploration	• Use the coefficients of the FIR filter associated with the wavelet to check the values of D_1. • Replace the step by an impulse. • Add noise to the signal and repeat the analysis.

Example 7: Two Proximal Discontinuities

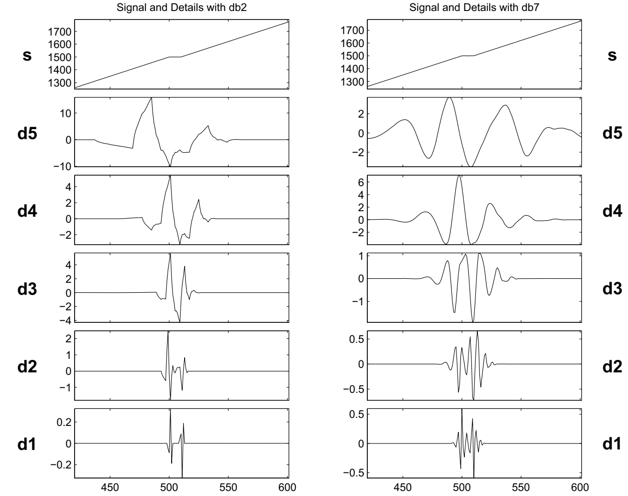
Analyzing wavelet: *db2* and *db7*

Decomposition levels: *5*

The signal is formed of two straight lines with identical slopes, extending across a very short plateau. On the initial signal, the plateau is in fact barely visible to the naked eye. Two analyses are thus carried out: one on a well localized wavelet with the short filter (`db2`, shown on the left side of the figure); and the other on a wavelet having a longer filter (`db7`, shown on the right side of the figure).

In both analyses, the plateau is detected clearly. With the exception of a fairly limited domain, D_1 is equal to zero. The regularity of the signal in the plateau, however, is clearly distinguished for `db2` (for which plateau beginning and end time are distinguished), whereas for `db7` both discontinuities are fused and only the entire plateau can be said to be visible.

This example suggests that the selected wavelets should be associated with short filters to distinguish proximal discontinuities of the first derivative. A look at the other detail levels again shows the lack of precision when detecting at low resolutions. The wavelet filters the straight line and analyzes the discontinuities.



Example 7: Two Proximal Discontinuities	
Addressed topics	• Detecting breakdown points
Further exploration	• Move the discontinuities closer together and further apart. • Add noise to the signal until the rupture is no longer visible. • Try using other wavelets, `haar` for instance.

Example 8: A Second-Derivative Discontinuity

Analyzing wavelets: *db1* and *db4*

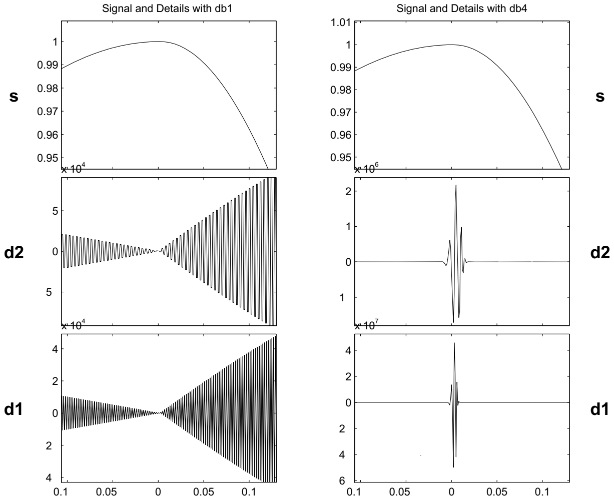
Decomposition levels: *2*

This figure shows that the regularity can be an important criterion in selecting a wavelet. The basic function is composed of two exponentials that are connected at 0, and the analyzed signal is the sampling of the continuous function with increments of 10^{-3}. The sampled signal is analyzed using two different wavelets: `db1`, which is insufficiently regular (shown on the left side of the figure); and `db4`, which is sufficiently regular (shown on the right side of the figure).

Looking at the figure on the left, notice that the singularity has not been detected in the extent that the details are equal to 0 at 0. The black areas correspond to very rapid oscillations of the details. These values are equal to the difference between the function and an approximation using a constant function. Close to 0, the slow decrease of the details absolute values followed by a slow increase is due to the fact that the function derivative is zero and continuous at 0. The value of the details is very small (close to 10^{-3} for `db1` and 10^{-6} for `db4`), since the signal is very smooth and does not contain any high frequency. This value is even smaller for `db4`, since the wavelet is more regular than `db1`.

However, with `db4` (right side of the figure), the discontinuity is well detected; the details are high only close to 0, and are 0 everywhere else. This is the only element that can be derived from the analysis. In this case, as a conclusion, notice that the selected wavelet must be sufficiently regular, which implies a longer filter impulse response to detect the singularity.

Note To produce the figure below you can use the One-Dimensional Wavelet GUI Tool. Type `wavemenu` at the MATLAB® prompt and click **Wavelet 1-D**. Then, select **File** > **Example Analysis** > **Basic Signals** > **with db1 at level 2 --->Second Derivative Breakdown** (and **... with db4 ...**). Detail values are very small, so to get the same shapes you must zoom the *y*-axis many times (close to 10^{-3} for `db1` and 10^{-6} for `db4`).



Example 8: A Second-Derivative Discontinuity	
Addressed topics	• Detecting breakdown points • Suppressing signals • Identifying a difficult discontinuity • Carefully selecting a wavelet to reveal an effect
Further exploration	• Calculate the detail values for the Haar wavelet. • Be aware of parasitic effects: rapid detail fluctuations may be artifacts. • Add noise to the signal until the rupture is no longer visible.

Example 9: A Ramp + White Noise

Analyzing wavelet: *db3*

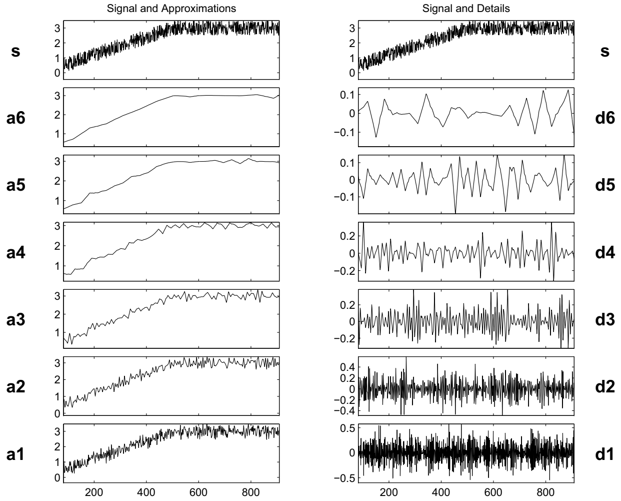
Decomposition levels: *6*

The signal is built from a trend plus noise. The trend is a slow linear rise from 0 to 3, up to t=500, and becoming constant afterwards. The noise is a uniform zero-mean white noise, varying between -0.5 and 0.5 (see the analyzed signal b_1).

Looking at the figure, in the chart on the right, we again find the decomposition of noise in the details. In the charts on the left, the approximations form increasingly precise estimates of the ramp with less and less noise. These approximations are quite acceptable from level 3, and the ramp is well reconstructed at level 6.

We can, therefore, separate the ramp from the noise. Although the noise affects all scales, its effect decreases sufficiently quickly for the low-resolution approximations to restore the ramp. It should also be noted that the breakdown point of the ramp is shown with good precision. This is due to the fact that the ramp is recovered at too low a resolution.

The uniform noise indicates that the ramp might be best estimated using half sums for the higher and lower portions of the signal. This approach is not applicable for other noises.



Example 9: A Ramp + White Noise	
Addressed topics	• Detecting breakdown points • Processing noise • Detecting long-term evolution • Splitting signal components • Identifying noises and approximations
Further exploration	• Compare with the white noise $b_1(t)$ shown in Example 3: Uniform White Noise on page 4-12. • Identify the number of levels needed to suppress the noise almost entirely. • Change the noise.

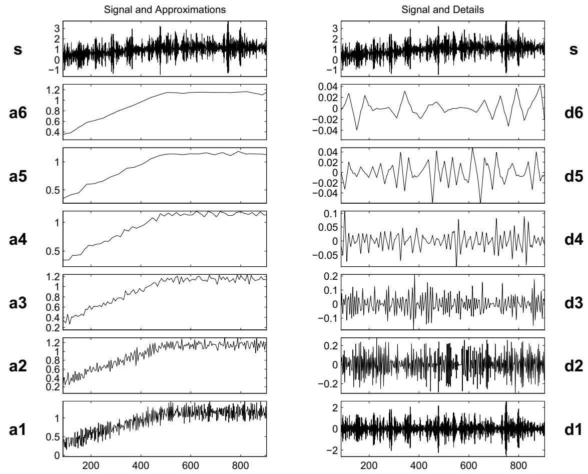
Example 10: A Ramp + Colored Noise

Analyzing wavelet: *db3*

Decomposition levels: *6*

The signal is built in the same manner as in “Example 9: A Ramp + White Noise” on page 4-24, using a trend plus a noise. The trend is a slow linear increase from 0 to 1, up to t=500. Beyond this time, the value remains constant. The noise is a zero mean AR(3) noise, varying between -3 and 3 (see the analyzed signal b_2). The scale of the noise is indeed six times greater than that of the ramp. At first glance, the situation seems a little bit less favorable than in the previous example, in terms of the separation between the ramp and the noise. This is actually a misconception, since the two signal components are more precisely separated in frequency.

Looking at the figure, the charts on the right show the detail decomposition of the colored noise. The charts on the left show a decomposition that resembles the one in the previous analysis. Starting at level 3, the curves provide satisfactory approximations of the ramp.



Example 10: A Ramp + Colored Noise	
Addressed topics	• Detecting breakdown points • Processing noise • Detecting long-term evolution • Splitting signal components
Further exploration	• Compare with the s_7(t) signal shown in Example 9: A Ramp + White Noise on page 4-24. • Identify the number of levels needed to suppress the noise almost entirely. • Identify the noise characteristics. Use the coefficients and the command line mode.

Example 11: A Sine + White Noise

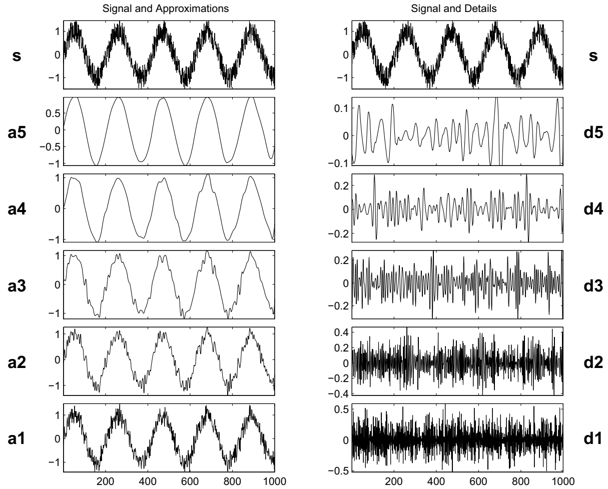
Analyzing wavelet: *db5*

Decomposition levels: *5*

The signal is formed of the sum of two previously analyzed signals: the slow sine with a period close to 200 and the uniform white noise b_1. This example is an illustration of the linear property of decompositions: the analysis of the sum of two signals is equal to the sum of analyses.

The details correspond to those obtained during the decomposition of the white noise.

The sine is found in the approximation A_5. This is a high enough level for the effect of the noise to be negligible in relation to the amplitude of the sine.



Example 11: A Sine + White Noise	
Addressed topics	• Processing noise • Detecting long-term evolution • Splitting signal components • Identifying the frequency of a sine
Further exploration	• Identify the noise characteristics. Use the coefficients and the command line mode.

Example 12: A Triangle + A Sine

Analyzing wavelet: *db5*

Decomposition levels: *6*

The signal is the sum of a sine having a period of approximately 20 and of a "triangle".

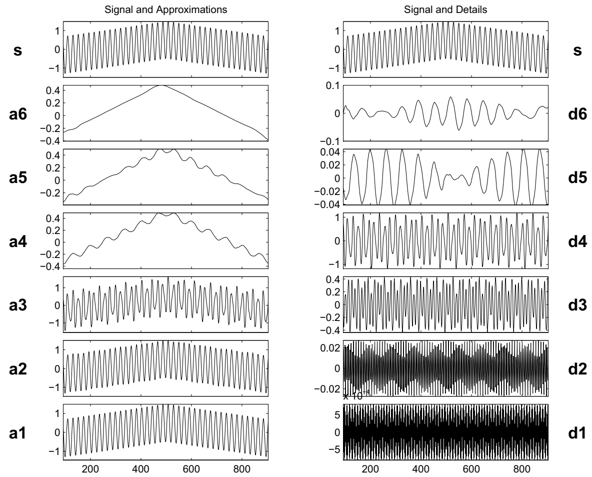
D_1 and D_2 are very small. This suggests that the signal contains no components with periods that are short in relation to the sampling period.

D_3 and especially D_4 can be attributed to the sine. The jump of the sine from A_3 to D_4 is clearly visible.

The details for the higher levels D_5 and D_6 are small, especially D_5.

D_6 exhibits some edge effects.

A_6 contains the triangle, which includes only low frequencies.



Example 12: A Triangle + A Sine	
Addressed topics	• Detecting long-term evolution • Splitting signal components • Identifying the frequency of a sine
Further exploration	• Try using sinusoids whose period is a power of 2.

Example 13: A Triangle + A Sine + Noise

Noise Analyzing wavelet: *db5*

Decomposition levels: *7*

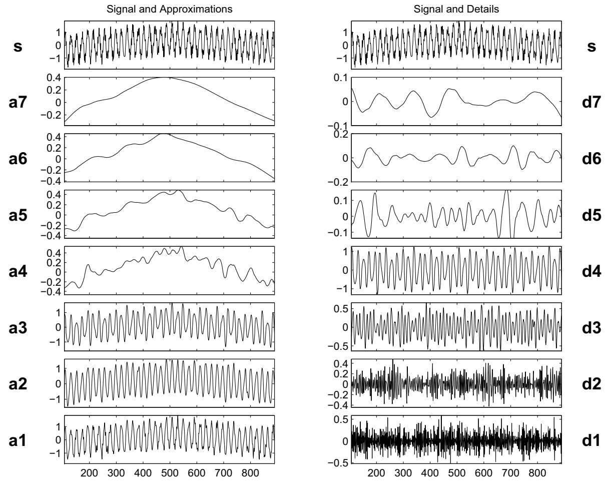
The signal examined here is the same as the previous signal plus a uniform white noise divided by 3. The analysis can, therefore, be compared to the previous analysis. All differences are due to the presence of the noise.

D_1 and D_2 are due to the noise.

D_3 and especially D_4 are due to the sine.

The higher level details are increasingly low, and originate in the noise.

A_7 contains a triangle, although it is not as well reconstructed as in the previous example.



Example 13: A Triangle + A Sine + Noise	
Addressed topics	• Detecting long-term evolution • Splitting signal components
Further exploration	• Increase the amplitude of the noise. • Replace the triangle by a polynomial. • Replace the white noise by an ARMA noise.

Example 14: A Real Electricity Consumption Signal

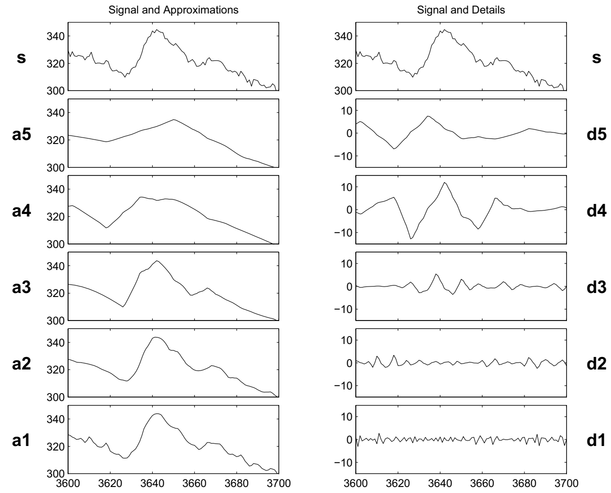
Analyzing wavelet: *db3*

Decomposition levels: *5*

The series presents a peak in the center, followed by two drops, a shallow drop, and then a considerably weaker peak.

The details for levels 1 and 2 are of the same order of magnitude and give a good expression of the local irregularities caused by the noise. The detail for level 3 presents high values in the beginning and at the end of the main peak, thus allowing us to locate the corresponding drops. The detail D_4 shows coarser morphological aspects for the series (i.e., three successive peaks). This fits the shape of the curve remarkably well, and includes the essential signal components for periods of less than 32 time-units. The approximations show this effect clearly: A_1 and A_2 bear a strong resemblance; A_3 forms a reasonably accurate approximation of the original signal. A look at A_4, however, shows that a considerable amount of information has been lost.

In this case, as a conclusion, the multiscale aspect is the most interesting and the most significant feature: the essential components of the electrical signal used to complete the description at 32 time-units (homogeneous to A_5) are the components with a period between 8 and 16 time-units.



Example 14: A Real Electricity Consumption Signal	
Addressed topics	• Detecting long-term evolution • Splitting signal components • Detecting breakdown points • Multiscale analysis
Further exploration	• Try the same analysis on various sections of the signal. Focus on a range other than the `[3600:3700]` shown here.

This signal is explored in much greater detail in “Case Study: An Electrical Signal” on page 4-36.

Case Study: An Electrical Signal

The goal of this section is to provide a statistical description of an electrical load consumption using the wavelet decompositions as a multiscale analysis.

Two problems are addressed. They both deal with signal extraction from the load curve corrupted by noise:

1. What information is contained in the signal, and what pieces of information are useful?
2. Are there various kinds of noises, and can they be distinguished from one another?

The context of the study is the forecast of the electrical load. Currently, short-term forecasts are based on the data sampled over 30 minutes. After eliminating certain components linked to weather conditions, calendar effects, outliers and known external actions, a SARIMA parametric model is developed. The model delivers forecasts from 30 minutes to 2 days. The quality of the forecasts is very high at least for 90% of all days, but the method fails when working with the data sampled over 1 minute.

Data and the External Information

The data consist of measurement of a complex, highly aggregated plant: the electrical load consumption, sampled minute by minute, over a 5-week period. This time series of 50,400 points is partly plotted at the top of the second plot in the “Analysis of the End of the Night Period” on page 4-39.

External information is given by electrical engineers, and additional indications can be found in several papers. This information, used to define reference situations for the purpose of comparison, includes these points:

- The load curve is the aggregation of hundreds of sensors measurements, thus generating measurement errors.
- Roughly speaking, 50% of the consumption is accounted for by industry, and the rest by individual consumers. The component of the load curve produced by industry has a rather regular profile and exhibits low-frequency changes. On the other hand, the consumption of individual consumers may be highly irregular, leading to high-frequency components.
- There are more than 10 million individual consumers.

- The fundamental periods are the weekly-daily cycles, linked to economic rhythms.
- Daily consumption patterns also change according to rate changes at different times (e.g., relay-switched water heaters to benefit from special night rates).
- Missing data have been replaced.
- Outliers have not been corrected.
- For the observations 2400 to 3400, the measurement errors are unusually high, due to sensors failures.

From a methodological point of view, the wavelet techniques provide a multiscale analysis of the signal as a sum of orthogonal signals corresponding to different time scales, allowing a kind of time-scale analysis.

Because of the absence of a model for the 1-minute data, the description strategy proceeds essentially by successive uses of various comparative methods applied to signals obtained by the wavelet decomposition.

Without modeling, it is impossible to define a signal or a noise effect. Nevertheless, we say that any repetitive pattern is due to signal and is meaningful.

Finally, it is known that two kinds of noise corrupt the signal: sensor errors and the state noise.

We shall not report here the complete analysis, which is included in the paper [MisMOP94] (see “References” on page 6-155). Instead, we illustrate the contribution of wavelet transforms to the local description of time series. We choose two small samples: one taken at midday, and the other at the end of the night.

In the first period, the signal structure is complex; in the second one, it is much simpler. The midday period has a complicated structure because the intensity of the electricity consumer activity is high and it presents very large changes. At the end of the night, the activity is low and it changes slowly.

For the local analysis, the decomposition is taken up to the level j = 5, because $2^5 = 32$ is very close to 30 minutes. We are then able to study the components of the signal for which the period is less than 30 minutes.

The analyzing wavelet used here is `db3`.

The results are described similarly for the two periods.

Analysis of the Midday Period

This signal (see “Example 14: A Real Electricity Consumption Signal” on page 4-34) is also analyzed more crudely in “Example 14: A Real Electricity Consumption Signal” on page 4-34.

The shape is a middle mode between 12:30 p.m. and 1:00 p.m., preceded and followed by a hollow off-peak, and next a second smoother mode at 1:15 p.m. The approximation A_5, corresponding to the time scale of 32 minutes, is a very crude approximation, particularly for the central mode: there is a peak time lag and an underestimation of the maximum value. So at this level, the most essential information is missing. We have to look at lower scales (4 for instance).

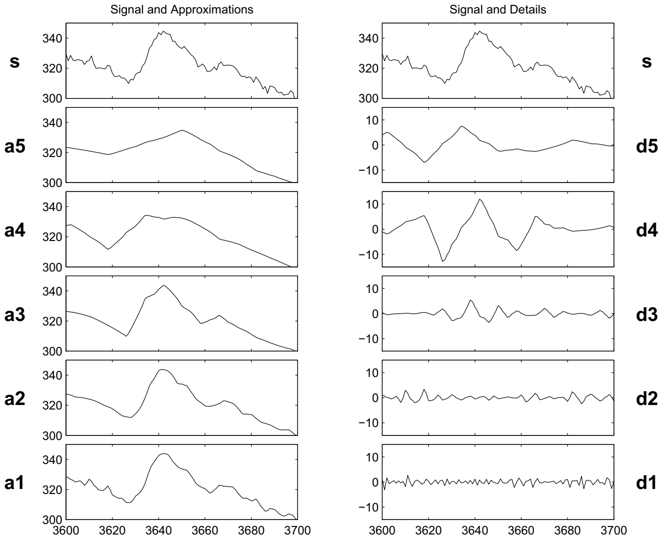
Let us examine the corresponding details.

The details D_1 and D_2 have small values and may be considered as local short-period discrepancies caused by the high-frequency components of sensor and state noises. In this bandpass, these noises are essentially due to measurement errors and fast variations of the signal induced by millions of state changes of personal electrical appliances.

The detail D_3 exhibits high values at times corresponding to the start and the end of the original middle mode. It allows time localization of the local minima.

The detail D_4 contains the main patterns: three successive modes. It is remarkably close to the shape of the curve. The ratio of the values of this level to the other levels is equal to 5. The detail D_5 does not bear much information. So the contribution of the level 4 is the highest one, both in qualitative and quantitative aspects. It captures the shape of the curve in the concerned period.

In conclusion, with respect to the approximation A_5, the detail D_4 is the main additional correction: the components of a period of 8 to 16 minutes contain the crucial dynamics.



Analysis of the End of the Night Period

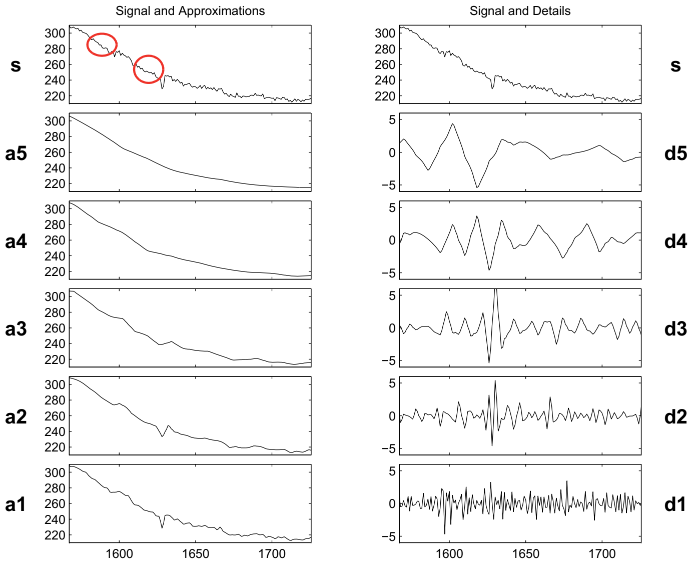
The shape of the curve during the end of the night is a slow descent, globally smooth, but locally highly irregular. One can hardly distinguish two successive local extrema in the vicinity of time t = 1600 and t = 1625. The approximation A_5 is quite good except at these two modes.

The accuracy of the approximation can be explained by the fact that there remains only a low-frequency signal corrupted by noises. The massive and simultaneous changes of personal electrical appliances are absent.

The details D_1, D_2, and D_3 show the kind of variation and have, roughly speaking, similar shape and mean value. They contain the local short period irregularities caused by noises, and the inspection of D_2 and D_3 allows you to detect the local minimum around t =1625.

The details D_4 and D_5 exhibit the slope changes of the regular part of the signal, and A_4 and A_5 are piecewise linear.

In conclusion, none of the time scales brings a significant contribution sufficiently different from the noise level, and no additional correction is needed. The retained approximation is A_4 or A_5.



All the figures in this paragraph are generated using the graphical user interface tools, but the user can also process the analysis using the command line mode. The following example corresponds to a command line equivalent for producing the figure below.

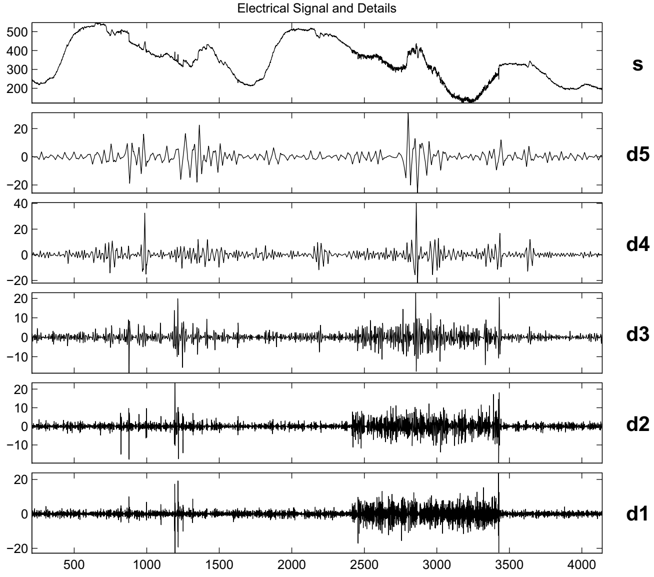
```
% Load the original 1-D signal, decompose, reconstruct details in
% original time and plot.
% Load the signal.
load leleccum; s = leleccum;

% Decompose the signal s at level 5 using the wavelet db3.
w = 'db3';
[c,l] = wavedec(s,5,w);
```

```
% Reconstruct the details using the decomposition structure.
for i = 1:5
   D(i,:) = wrcoef('d',c,l,w,i);
end
```

Note This loop replaces five separate `wrcoef` statements defining the details. The variable `D` contains the five details.

```
% Avoid edge effects by suppressing edge values and plot.
tt = 1+100:length(s)-100;
subplot(6,1,1); plot(tt,s(tt),'r');
title('Electrical Signal and Details');
for i = 1:5, subplot(6,1,i+1); plot(tt,D(5-i+1,tt),'g'); end
```



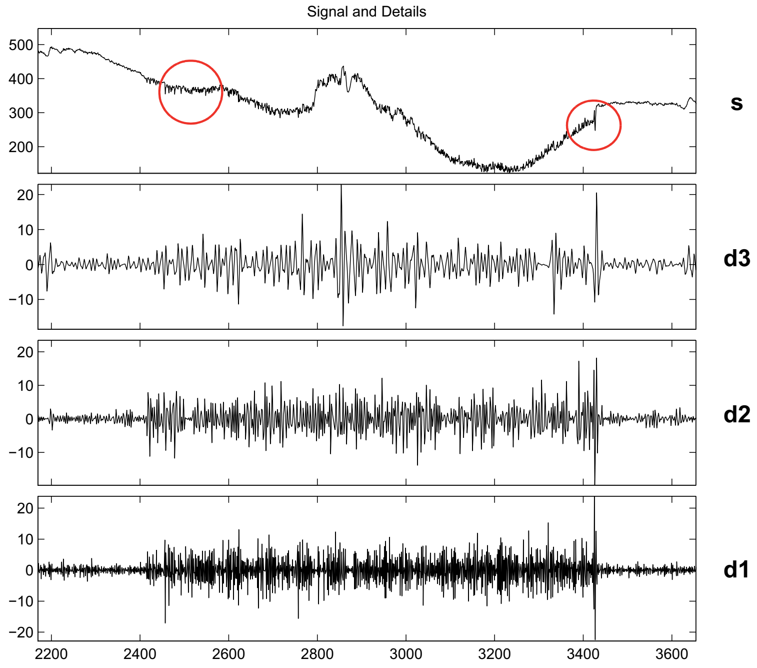
Suggestions for Further Analysis

Let us now make some suggestions for possible further analysis starting from the details of the decomposition at level 5 of 3 days.

Identify the Sensor Failure

Focus on the wavelet decomposition and try to identify the sensor failure directly on the details D_1, D_2, and D_3, and not the other ones. Try to identify the other part of the noise.

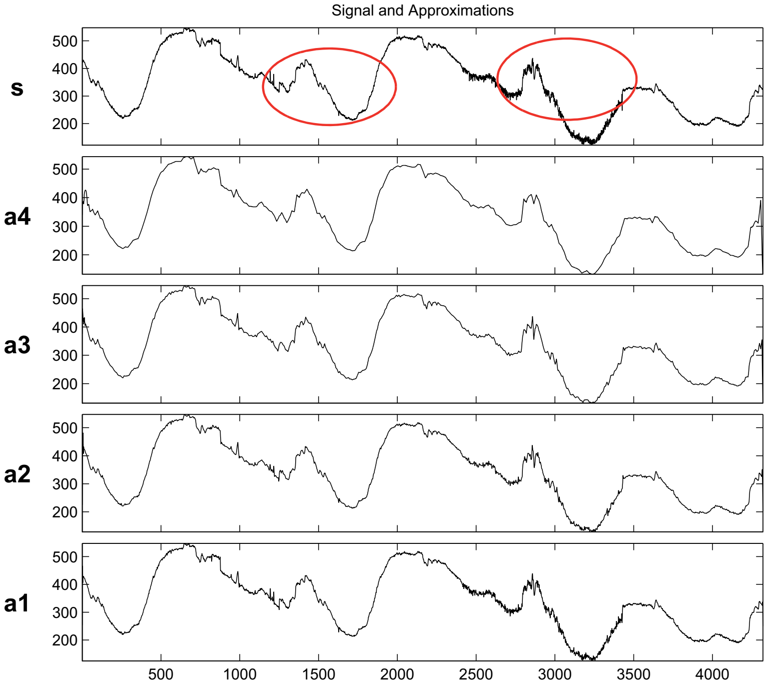
Indication: see figure below.



Suppress the Noise

Suppress measurement noise. Try by yourself and afterwards use the de-noising tools.

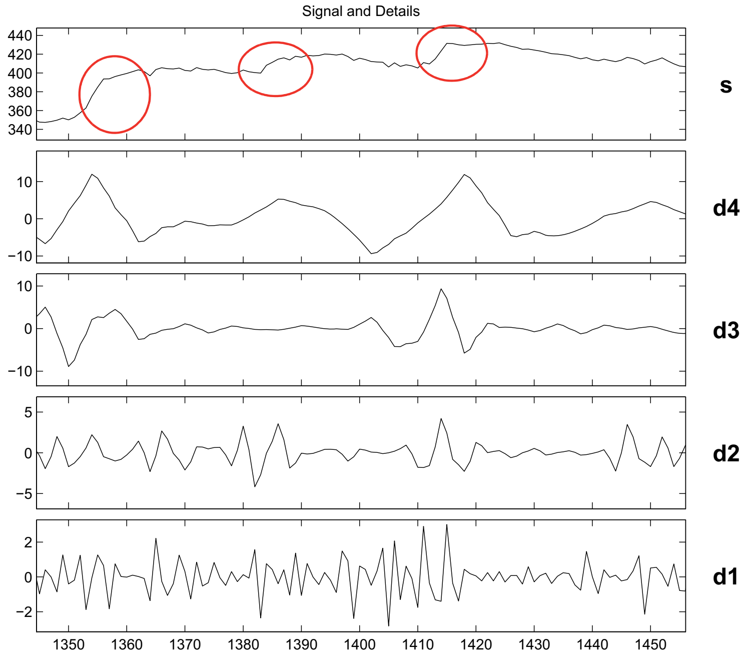
Indication: study the approximations and compare two successive days, the first without sensor failure and the second corrupted by failure (see figure below).



Identify Patterns in the Details

The idea here is to identify a pattern in the details typical of relay-switched water heaters.

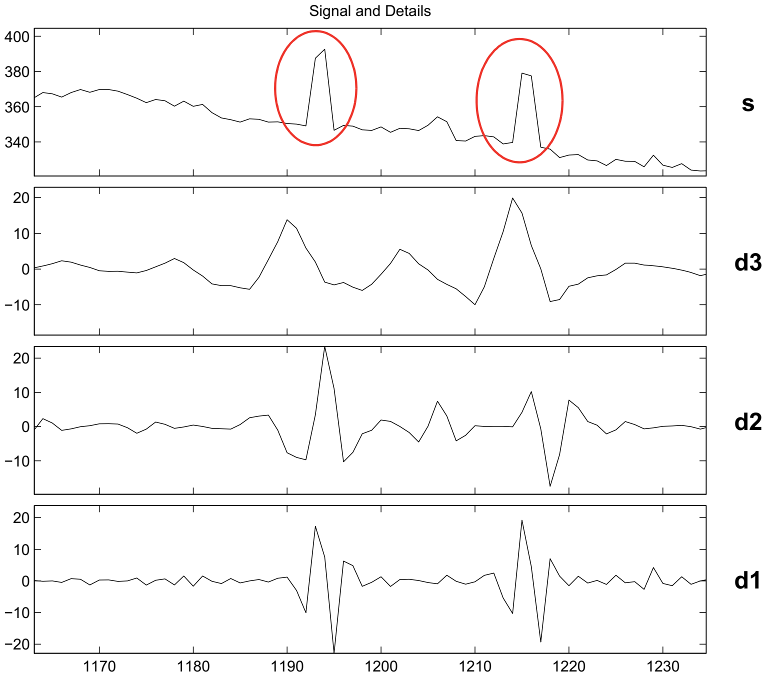
Indication: the figure below gives an example of such a period. Focus on details D_2, D_3, and D_4 around abscissa 1350, 1383, and 1415 to detect abrupt changes of the signal induced by automatic switches.



Locate and Suppress Outlying Values

Suppress the outliers by setting the corresponding values of the details to 0.

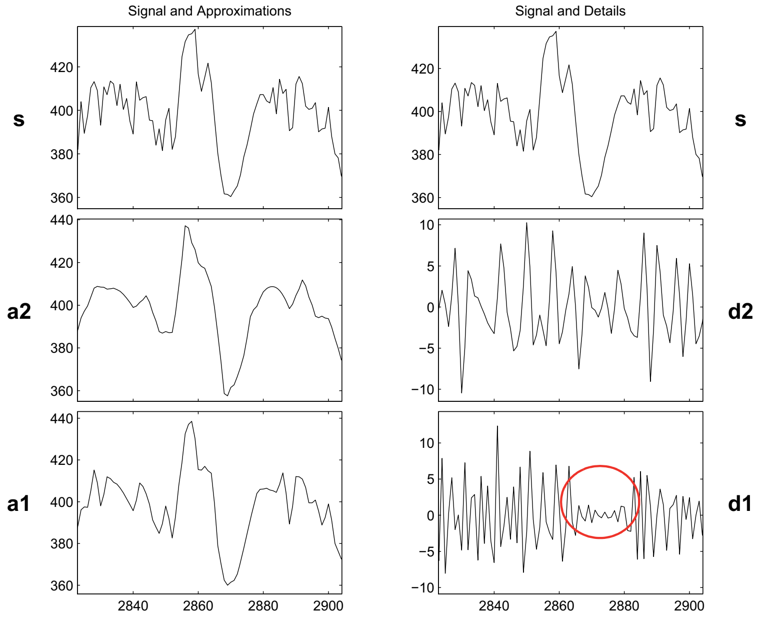
Indication: The figure below gives two examples of outliers around $t = 1193$ and $t = 1215$. The effect produced on the details is clear when focusing on the low levels. As far as outliers are concerned, D_1 and D_2 are synchronized with s, while D_3 shows a delayed effect.



Study Missing Data

Missing data have been crudely substituted (around observation 2870) by the estimation of 30 minutes of sampled data and spline smoothing for the intermediate time points. You can improve the interpolation by using an approximation and portions of the details taken elsewhere, thus implementing a sort of "graft."

Indication: see the figure below focusing around time 2870, and use the small variations part of D_1 to detect the missing data.



5

Using Wavelet Packets

6

Advanced Concepts

This chapter presents a more advanced treatment of wavelet methods, and focuses on real wavelets, except in the two sections dedicated to wavelet families.

Mathematical Conventions

This chapter and the reference pages use certain mathematical conventions.

General Notation	Interpretation
$a = 2^j, j \in Z$	Dyadic scale. j is the level, $1/a$ or 2^{-j} is the resolution.
$b = ka, k \in Z$	Dyadic translation
t	Continuous time
k or n	Discrete time
(i, j)	Pixel
s	Signal or image. The signal is a function defined on R or Z; the image is defined on R^2 or Z^2.
$\hat{f}$	Fourier transform of the function f or the sequence f
Continuous Time	
$L^2(R)$	Set of signals of finite energy
$\int_R s^2(x)dx$	Energy of the signal s
$\langle s, s' \rangle = \int_R s(x)s'(x)dx$	Scalar product of signals s and s'
$L^2(R^2)$	Set of images of finite energy
$\int_R \int_R s^2(x, y)\ dxdy$	Energy of the image s
$\langle s, s' \rangle = \int_R \int_R s(x, y)s'(x, y)dxdy$	Scalar product of images s and s'

General Notation	Interpretation (Continued)
Discrete Time	
$l^2(Z)$	Set of signals of finite energy
$\sum_{n \in Z} s^2(n)$	Energy of the signal s
$\langle s, s' \rangle = \sum_{n \in Z} s(n)s'(n)$	Scalar product of signals s and s'
$l^2(Z^2)$	Set of images of finite energy
$\sum_{n \in Z} \sum_{m \in Z} s^2(n, m)$	Energy of the image s
$\langle s, s' \rangle = \sum_{n \in Z} \sum_{m \in Z} s(n, m)s'(n, m)$	Scalar product of images s and s'

Wavelet Notation	Interpretation
A_j	j-level approximation or approximation at level j
D_j	j-level detail or detail at level j
ϕ	Scaling function
ψ	Wavelet
$\frac{1}{\sqrt{a}}\psi\left(\frac{x-b}{a}\right)$	Family associated with the one-dimensional wavelet, indexed by $a > 0$ and $b \in R$
$\frac{1}{\sqrt{a_1 a_2}}\psi\left(\frac{x_1 - b_1}{a_1}, \frac{x_2 - b_2}{a_2}\right), x = (x_1,x_2) \in R^2$	Family associated with the two-dimensional wavelet, indexed by $a_1 > 0, a_2 > 0, b_1 \in R, b_2 \in R$

Wavelet Notation	Interpretation (Continued)
$\phi_{j,k}(x) = 2^{-j/2}\phi(2^{-j}x - k), j \in Z, k \in Z$	Family associated with the one-dimensional scaling function for dyadic scales $a = 2^j, b = ka$; it should be noted that $\phi = \phi_{0,0}$.
$\psi_{j,k}(x) = 2^{-j/2}\psi(2^{-j}x - k), j \in Z, k \in Z$	Family associated with the one-dimensional ψ for dyadic scales $a = 2^j, b = ka$; it should be noted that $\psi = \psi_{0,0}$.
$(h_k), k \in Z$	Scaling filter associated with a wavelet
$(g_k), k \in Z$	Wavelet filter associated with a wavelet

General Concepts

This section presents a brief overview of wavelet concepts, focusing mainly on the orthogonal wavelet case. It includes the following sections:

- “Wavelets: A New Tool for Signal Analysis” on page 6-5
- “Wavelet Decomposition: A Hierarchical Organization” on page 6-5
- “Finer and Coarser Resolutions” on page 6-6
- “Wavelet Shapes” on page 6-6
- “Wavelets and Associated Families” on page 6-8
- “Wavelet Transforms: Continuous and Discrete” on page 6-12
- “Local and Global Analysis” on page 6-14
- “Synthesis: An Inverse Transform” on page 6-15
- “Details and Approximations” on page 6-15

Wavelets: A New Tool for Signal Analysis

Wavelet analysis consists of decomposing a signal or an image into a hierarchical set of approximations and details. The levels in the hierarchy often correspond to those in a dyadic scale.

From the signal analyst’s point of view, wavelet analysis is a decomposition of the signal on a family of analyzing signals, which is usually an orthogonal function method. From an algorithmic point of view, wavelet analysis offers a harmonious compromise between decomposition and smoothing techniques.

Wavelet Decomposition: A Hierarchical Organization

Unlike conventional techniques, wavelet decomposition produces a family of hierarchically organized decompositions. The selection of a suitable level for the hierarchy will depend on the signal and experience. Often the level is chosen based on a desired low-pass cutoff frequency.

At each level *j*, we build the *j*-level approximation A_j, or approximation at level *j*, and a deviation signal called the *j*-level detail D_j, or detail at level *j*. We can consider the original signal as the approximation at level 0, denoted by A_0. The words *approximation* and *detail* are justified by the fact that A_1 is an approximation of A_0 taking into account the *low frequencies* of A_0, whereas the detail D_1 corresponds to the *high frequency* correction. Among the figures presented in “Reconstructing Approximations and Details” on page 1-30, one of them graphically represents this hierarchical decomposition.

One way of understanding this decomposition consists of using an optical comparison. Successive images A_1, A_2, A_3 of a given object are built. We use the same type of photographic devices, but with increasingly poor resolution. The images are successive approximations; one detail is the discrepancy between two successive images. Image A_2 is, therefore, the sum of image A_4 and intermediate details D_4, D_3:

$$A_2 = A_3 + D_3 = A_4 + D_4 + D_3$$

Finer and Coarser Resolutions

The organizing parameter, the scale a, is related to level j by $a = 2^j$. If we define resolution as $1/a$, then the resolution increases as the scale decreases. The greater the resolution, the smaller and finer are the details that can be accessed.

j	10	9	...	2	1	0	-1	-2
Scale	1024	512	...	4	2	1	1/2	1/4
Resolution	$1/2^{10}$	$1/2^9$	...	1/4	1/2	1	2	4

From a technical point of view, the size of the revealed details for any j is proportional to the size of the domain in which the wavelet or analyzing function of the variable x, $\psi\left(\frac{x}{a}\right)$ is not too close to 0.

Wavelet Shapes

One-dimensional analysis is based on one scaling function ϕ and one wavelet ψ. Two-dimensional analysis (on a square or rectangular grid) is based on one scaling function $\phi(x_1, x_2)$ and three wavelets.

Figure 6-1 shows ϕ and ψ for each wavelet, except the Morlet wavelet and the Mexican hat, for which ϕ does not exist. All the functions decay quickly to zero. The Haar wavelet is the only noncontinuous function with three points of discontinuity (0, 0.5, 1). The ψ functions oscillate more than associated ϕ functions. `coif2` exhibits some angular points; `db6` and `sym6` are quite smooth. The Morlet and Mexican hat wavelets are symmetrical.

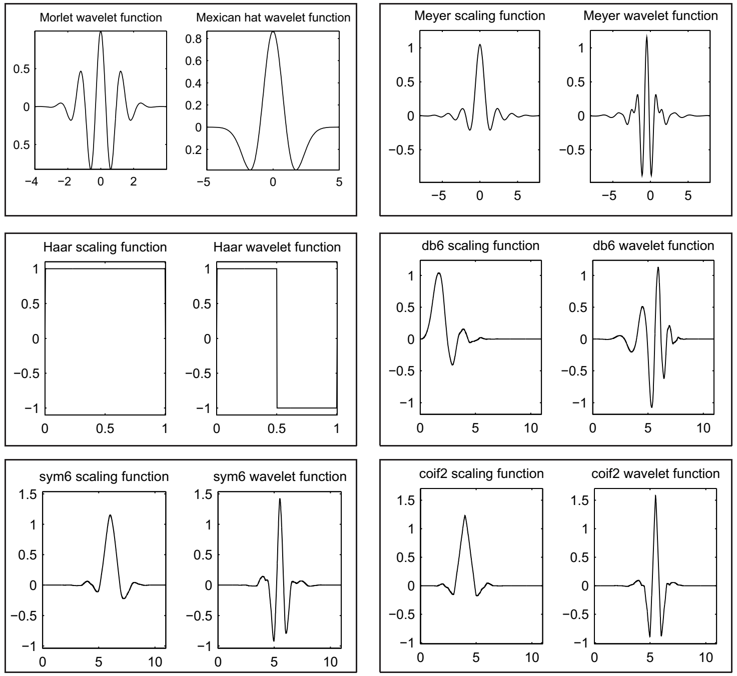


Figure 6-1: Various One-Dimensional Wavelets

Wavelets and Associated Families

In the one-dimensional context, we distinguish the wavelet ψ from the associated function ϕ, called the scaling function. Some properties of ψ and ϕ are

- The integral of ψ is zero, ($\int \psi(x)dx = 0$), and ψ is used to define the details.
- The integral of ϕ is 1, ($\int \phi(x)dx = 1$), and ϕ is used to define approximations.

The usual two-dimensional wavelets are defined as tensor products of one-dimensional wavelets: $\phi(x,y) = \phi(x)\phi(y)$ is the scaling function and $\psi_1(x,y) = \phi(x)\psi(y), \psi_2(x,y) = \psi(x)\phi(y), \psi_3(x,y) = \psi(x)\psi(y)$ are the three wavelets.

Figure 6-2 shows the four functions associated with the 2-D `coif2` wavelet.

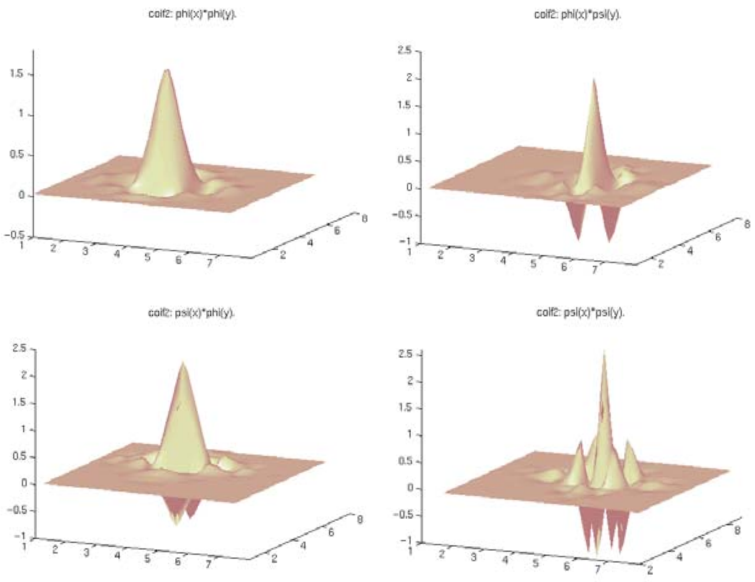


Figure 6-2: Two-Dimensional coif2 Wavelet

To each of these functions, we associate its doubly indexed family, which is used to:

- Move the basic shape from one side to the other, translating it to position b (see the following figure).
- Keep the shape while changing the one-dimensional time scale a ($a > 0$) (see Figure 6-4 on page 6-10).

So a wavelet family member has to be thought of as a function located at a position b, and having a scale a.

In one-dimensional situations, the family of translated and scaled wavelets associated with ψ is expressed as follows.

Translation	Change of Scale	Translation and Change of Scale
$\psi(x-b)$	$\frac{1}{\sqrt{a}}\psi\left(\frac{x}{a}\right)$	$\frac{1}{\sqrt{a}}\psi\left(\frac{x-b}{a}\right)$

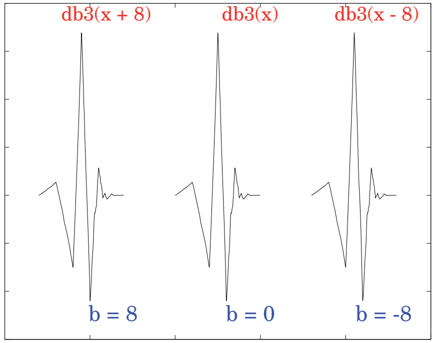


Figure 6-3: Translated Wavelets

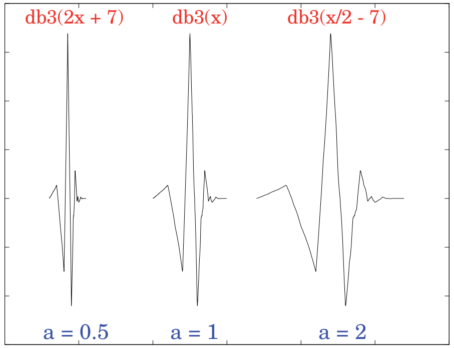


Figure 6-4: Time Scaled One-Dimensional Wavelet

In a two-dimensional context, we have the translation by vector (b_1,b_2) and a change of scale of parameter (a_1,a_2).

Translation and change of scale become:

$$\frac{1}{\sqrt{a_1 a_2}}\psi\left(\frac{x_1-b_1}{a_1},\frac{x_2-b_2}{a_2}\right) \text{ where } (x = (x_1,x_2) \in R^2)$$

In most cases, we will limit our choice of a and b values by using only the following discrete set (coming back to the one-dimensional context):

$$(j,k) \in Z^2 : a = 2^j, \qquad b = k2^j = ka$$

Let us define:

$$(j,k) \in Z^2 : \psi_{j,k} = 2^{-j/2}\psi(2^{-j}x-k),\ \phi_{j,k} = 2^{-j/2}\phi(2^{-j}x-k)$$

We now have a hierarchical organization similar to the organization of a decomposition; this is represented in the example of Figure 6-5, Wavelets Organization. Let $k = 0$ and leave the translations aside for the moment. The functions associated with $j = 0, 1, 2, 3$ for ϕ (expressed as $\phi_{j,0}$) and with $j = 1, 2, 3$ for ψ (expressed as $\psi_{j,0}$) are displayed in the following figure for the `db3` wavelet.

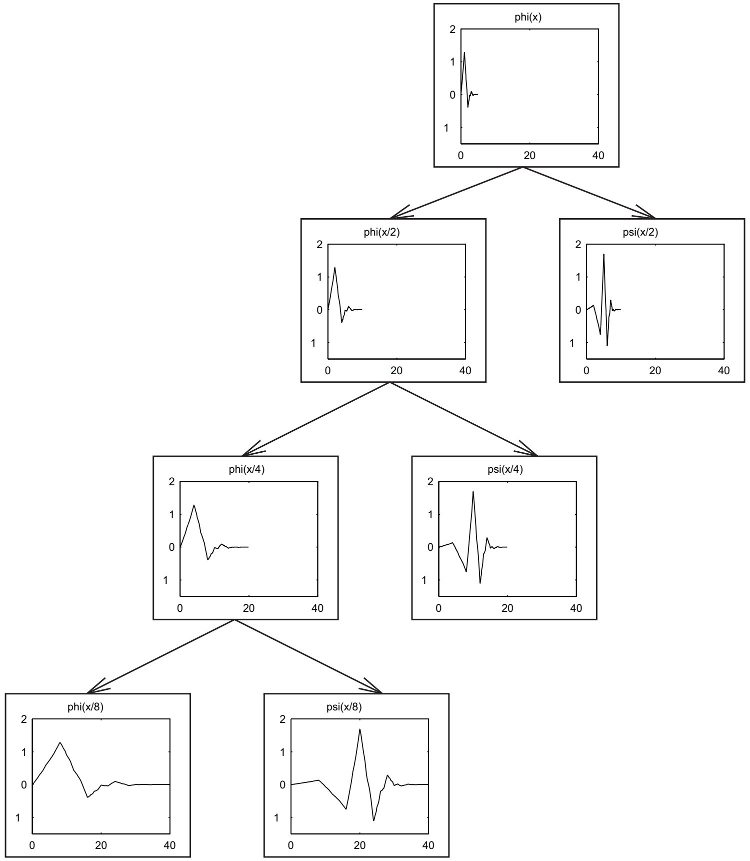


Figure 6-5: Wavelets Organization

In Figure 6-5, Wavelets Organization, the four-level decomposition is shown, progressing from the top to the bottom. We find $\phi_{0,0}$; then $2^{1/2}\phi_{1,0}$, $2^{1/2}\psi_{1,0}$; then $2\phi_{2,0}$, $2\psi_{2,0}$; then $2^{3/2}\phi_{3,0}$, $2^{3/2}\psi_{3,0}$. The wavelet is `db3`.

Wavelet Transforms: Continuous and Discrete

The wavelet transform of a signal *s* is the family *C*(*a*,*b*), which depends on two indices *a* and *b*. The set to which *a* and *b* belong is given below in the table. The studies focus on two transforms:

- Continuous transform
- Discrete transform

From an intuitive point of view, the wavelet decomposition consists of calculating a "resemblance index" between the signal and the wavelet located at position b and of scale a. If the index is large, the resemblance is strong, otherwise it is slight. The indexes *C*(*a*,*b*) are called coefficients.

We define the coefficients in the following table. We have two types of analysis at our disposal.

Continuous Time Signal **Continuous Analysis**	**Continuous Time Signal** **Discrete Analysis**
$C(a,b) = \int_R s(t)\frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right)dt$	$C(a,b) = \int_R s(t)\frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right)dt$
$a \in R^{+}-\{0\}, b \in R$	$a = 2^{j}, b = k2^{j}, (j,k) \in Z^{2}$

Next we will illustrate the differences between the two transforms, for the analysis of a fractal signal (see Figure 6-6).

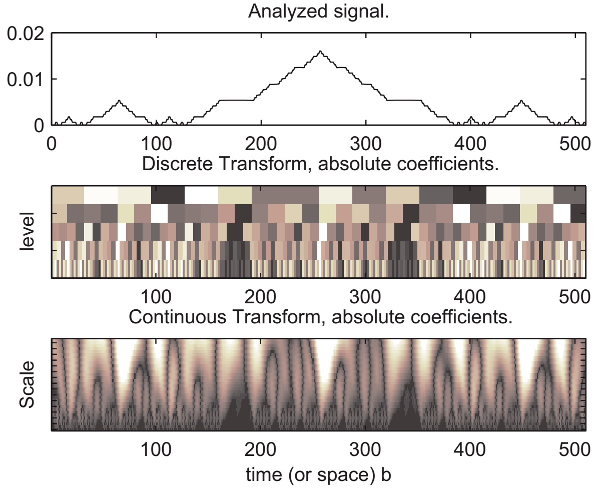


Figure 6-6: Continuous Versus Discrete Transform

Using a redundant representation close to the so-called continuous analysis, instead of a nonredundant discrete time-scale representation, can be useful for analysis purposes. The nonredundant representation is associated with an orthonormal basis, whereas the redundant representation uses much more scale and position values than a basis. For a classical fractal signal, the redundant methods are quite accurate.

- **Graphic representation of discrete analysis**: (in the middle of Figure 6-6, Continuous Versus Discrete Transform) time is on the abscissa and on the ordinate the scale *a* is dyadic: 2^1, 2^2, 2^3, 2^4, and 2^5 (from the bottom to the top), levels are 1, 2, 3, 4, and 5. Each coefficient of level k is repeated 2^k times.
- **Graphic representation of continuous analysis**: (at the bottom of Figure 6-6, Continuous Versus Discrete Transform) time is on the abscissa and on the ordinate the scale varies almost continuously between 2^1 and 2^5 by step 1 (from the bottom to the top). Keep in mind that when a scale is small, only small details are analyzed, as in a geographical map.

Local and Global Analysis

A small scale value permits us to perform a local analysis; a large scale value is used for a global analysis. Combining local and global is a useful feature of the method. Let us be a bit more precise about the local part and glance at the frequency domain counterpart.

Imagine that the analyzing function ϕ or ψ is zero outside of a domain U, which is contained in a disk of radius ρ: $\psi(u) = 0, \forall u \notin \mathrm{U}$. The wavelet ψ is localized. The signal s and the function ψ are then compared in the disk, taking into account only the t values in the disk. The signal values, which are located outside of the domain U, do not influence the value of the coefficient

$$\int_R s(t)\psi(t)dt \quad \text{and we get} \quad \int_R s(t)\psi(t)dt = \int_U s(t)\psi(t)dt$$

The same argument holds when ψ is translated to position b and the corresponding coefficient analyzes s around b. So this analysis is local.

The wavelets having a compact support are used in local analysis. This is the case for Haar and Daubechies wavelets, for example. The wavelets whose values are considered as very small outside a domain U can be used with caution, as if they were in fact actually zero outside U. Not every wavelet has a compact support. This is the case, for instance, of the Meyer wavelet.

The previous localization is temporal, and is useful in analyzing a temporal signal (or spatial signal if analyzing an image). The good spectral domain localization is a second type of a useful property. A result (linked to the Heisenberg uncertainty principle) links the dispersion of the signal f and the dispersion of its Fourier transform $\hat{f}$, and therefore of the dispersion of ψ and $\hat{\psi}$. The product of these dispersions is always greater than a constant c (which does not depend on the signal, but only on the dimension of the space). So it is impossible to reduce arbitrarily both time and frequency localization.

In the Fourier and spectral analysis, the basic function is $f(x) = \exp(i\omega x)$. This function is not a time localized function. The support is R. Its Fourier transform $\hat{f}$ is a generalized function concentrated at point ω.

The function f is very poorly localized in time, but $\hat{f}$ is perfectly localized in frequency. The wavelets generate an interesting "compromise" on the supports, and this compromise differs from that of complex exponentials, sine, or cosine.

Synthesis: An Inverse Transform

In order to be efficient and useful, a method designed for analysis also has to be able to perform synthesis. The wavelet method achieves this.

The analysis starts from s and results in the coefficients $C(a,b)$. The synthesis starts from the coefficients $C(a,b)$ and reconstructs s. Synthesis is the reciprocal operation of analysis.

For signals of finite energy, there are two formulas to perform the inverse wavelet transform:

- Continuous synthesis:

$$s(t) = \frac{1}{K_\psi}\int_{R^+}\int_R C(a,b)\frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right)\frac{da\ db}{a^2}$$

where K_ψ is a constant depending on ψ.

- Discrete synthesis:

$$s(t) = \sum_{j \in Z}\sum_{k \in Z} C(j,k)\psi_{j,k}(t).$$

Of course, the previous formulas need some hypotheses on the ψ function. More precisely, see "What Functions Are Candidates to Be a Wavelet?" on page 6-63 for the continuous synthesis formula and "Why Does Such an Algorithm Exist?" on page 6-28 for the discrete one.

Details and Approximations

The equations for continuous and discrete synthesis are of considerable interest and can be read in order to define the detail at level j:

1 Let us fix j and sum on k. A detail D_j is nothing more than the function

$$D_j(t) = \sum_{k \in Z} C(j,k)\psi_{j,k}(t).$$

2 Now, let us sum on j. The signal is the sum of all the details:

$$s = \sum_{j \in Z} D_j$$

The details have just been defined. Take a reference level called J. There are two sorts of details. Those associated with indices $j \le J$ correspond to the scales $a = 2^j \le 2^J$ which are the fine details. The others, which correspond to $j > J$, are the coarser details.

We group these latter details into

$$A_J = \sum_{j > J} D_j$$

which defines what is called an approximation of the signal s. We have just created the details and an approximation. They are connected. The equality

$$s = A_J + \sum_{j \le J} D_j$$

signifies that s is the sum of its approximation A_J and of its fine details. From the previous formula, it is obvious that the approximations are related to one another by

$$A_{J-1} = A_J + D_J$$

For an orthogonal analysis, in which the $\psi_{j,k}$ is an orthonormal family,

- A_J is orthogonal to D_J, $D_{J\text{-}1}$, $D_{J\text{-}2}$, ...
- s is the sum of the two orthogonal signals: A_J and $\sum_{j \le J} D_j$
- $D_j \perp D_k \quad \text{for} \quad j \ne k$
- A_J is an approximation of s. The quality (in energy) of the approximation of s by A_J is

  $$qual_J = \frac{\|A_J\|^2}{\|s\|^2}$$
- $qual_{J-1} = qual_J + \dfrac{\|D_J\|^2}{\|s\|^2} \ge qual_J$

The following table contains definitions of details and approximations.

Definition of the detail at level j	$D_j(t) = \sum_{k \in Z} C(j,k)\psi_{j,k}(t)$
The signal is the sum of its details	$s = \sum_{j \in Z} D_j$
The approximation at level J	$A_J = \sum_{j > J} D_j$
Link between $A_{J\text{-}1}$ and A_J	$A_{J\text{-}1} = A_J + D_J$
Several decompositions	$s = A_J + \sum_{j \le J} D_j$

From a graphical point of view, when analyzing a signal, it is always valuable to represent the different signals (s, A_j, D_j) and coefficients ($C(j,k)$).

Let us consider Figure 6-7. On the left side, **s** is the signal; **a5**, **a4**, **a3**, **a2**, and **a1** are the approximations at levels 5, 4, 3, 2, and 1. The best approximation is **a1**; the next one is **a2**, and so on. Noise oscillations are exhibited in **a1**, whereas **a5** is smoother.

On the right side, **cfs** represents the coefficients (for more information, see "Wavelet Transforms: Continuous and Discrete" on page 6-12), **s** is the signal, and **d5**, **d4**, **d3**, **d2**, and **d1** are the details at levels 5, 4, 3, 2, and 1.

The different signals that are presented exist in the same time grid. We can consider that the t index of detail $D_4(t)$ identifies the same temporal instant as that of the approximation $A_5(t)$ and that of the signal $s(t)$. This identity is of considerable practical interest in understanding the composition of the signal, even if the wavelet sometimes introduces dephasing.

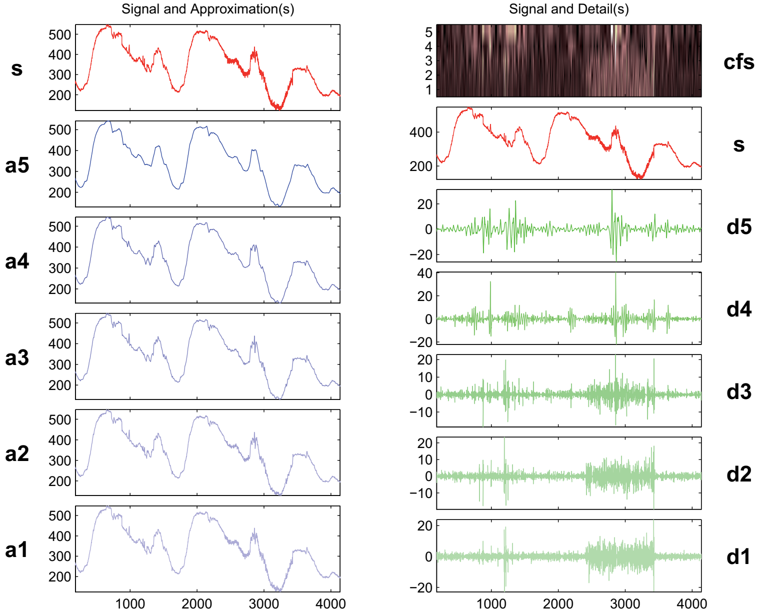


Figure 6-7: Approximations, Details, and Coefficients

Fast Wavelet Transform (FWT) Algorithm

In 1988, Mallat produced a fast wavelet decomposition and reconstruction algorithm [Mal89]. The Mallat algorithm for discrete wavelet transform (DWT) is, in fact, a classical scheme in the signal processing community, known as a two-channel subband coder using conjugate quadrature filters or quadrature mirror filters (QMFs).

- The decomposition algorithm starts with signal s, next calculates the coordinates of A_1 and D_1, and then those of A_2 and D_2, and so on.
- The reconstruction algorithm called the inverse discrete wavelet transform (IDWT) starts from the coordinates of A_J and D_J, next calculates the coordinates of $A_{J\text{-}1}$, and then using the coordinates of $A_{J\text{-}1}$ and $D_{J\text{-}1}$ calculates those of $A_{J\text{-}2}$, and so on.

This section addresses the following topics:

- "Filters Used to Calculate the DWT and IDWT"
- "Algorithms" on page 6-23
- "Why Does Such an Algorithm Exist?" on page 6-28
- "One-Dimensional Wavelet Capabilities" on page 6-32
- "Two-Dimensional Wavelet Capabilities" on page 6-33

Filters Used to Calculate the DWT and IDWT

For an orthogonal wavelet, in the multiresolution framework (see [Dau92] in Chapter 5, "Using Wavelet Packets"), we start with the scaling function ϕ and the wavelet function ψ. One of the fundamental relations is the twin-scale relation (dilation equation or refinement equation):

$$\frac{1}{2}\phi\left(\frac{x}{2}\right) = \sum_{n \in Z} w_n \phi(x-n)$$

All the filters used in DWT and IDWT are intimately related to the sequence

$$(w_n)_{n \in Z}$$

Clearly if ϕ is compactly supported, the sequence (w_n) is finite and can be viewed as a filter. The filter W, which is called the scaling filter (nonnormalized), is

- Finite Impulse Response (FIR)
- Of length $2N$
- Of sum 1
- Of norm $\frac{1}{\sqrt{2}}$
- A low-pass filter

For example, for the `db3` scaling filter,

```
load db3
db3
    db3 =
    0.2352   0.5706   0.3252  -0.0955  -0.0604   0.0249

sum(db3)
    ans =
         1.0000

norm(db3)
    ans =
         0.7071
```

From filter W, we define four FIR filters, of length $2N$ and of norm 1, organized as follows.

Filters	Low-Pass	High-Pass
Decomposition	Lo_D	Hi_D
Reconstruction	Lo_R	Hi_R

The four filters are computed using the following scheme.

$$\begin{tikzcd}
W \arrow[d] & \\
\mathrm{Lo\_R} = \dfrac{W}{\texttt{norm}(W)} \arrow[r] \arrow[d] & \mathrm{Lo\_D} = \texttt{wrev}(\mathrm{Lo\_R}) \\
\mathrm{Hi\_R} = \texttt{qmf}\ (\mathrm{Lo\_R}) \arrow[r] & \mathrm{Hi\_D} = \texttt{wrev}(\mathrm{Hi\_R})
\end{tikzcd}$$

where `qmf` is such that Hi_R and Lo_R are quadrature mirror filters (i.e., $\mathrm{Hi\_R}(k) = (-1)^{k}\, \mathrm{Lo\_R}(2N + 1 - k))$ for $k = 1, 2, ..., 2N$.

Note that `wrev` flips the filter coefficients. So Hi_D and Lo_D are also quadrature mirror filters. The computation of these filters is performed using `orthfilt`. Next, we illustrate these properties with the `db6` wavelet. The plots associated with the following commands are shown in Figure 6-8 on page 6-22.

```
% Load scaling filter.
load db6; w = db6;
subplot(421); stem(w); title('Original scaling filter');

% Compute the four filters.
[Lo_D,Hi_D,Lo_R,Hi_R] = orthfilt(w);
subplot(423); stem(Lo_D);
title('Decomposition low-pass filter Lo{\_}D');
subplot(424); stem(Hi_D);
title('Decomposition high-pass filter Hi{\_}D');
subplot(425); stem(Lo_R);
title('Reconstruction low-pass filter Lo{\_}R');
subplot(426); stem(Hi_R);
title('Reconstruction high-pass filter Hi{\_}R');

% High and low frequency illustration.
n = length(Hi_D);
freqfft = (0:n-1)/n;
nn = 1:n;
N = 10*n;
```

```
for k=1:N
    lambda(k) = (k-1)/N;
    XLo_D(k) = exp(-2*pi*j*lambda(k)*(nn-1))*Lo_D';
    XHi_D(k) = exp(-2*pi*j*lambda(k)*(nn-1))*Hi_D';
end
fftld = fft(Lo_D);
ffthd = fft(Hi_D);
subplot(427); plot(lambda,abs(XLo_D),freqfft,abs(fftld),'o');
title('Transfer modulus: lowpass (Lo{\_}D or Lo{\_}R')
subplot(428); plot(lambda,abs(XHi_D),freqfft,abs(ffthd),'o');
title('Transfer modulus: highpass (Hi{\_}D or Hi{\_}R')
```

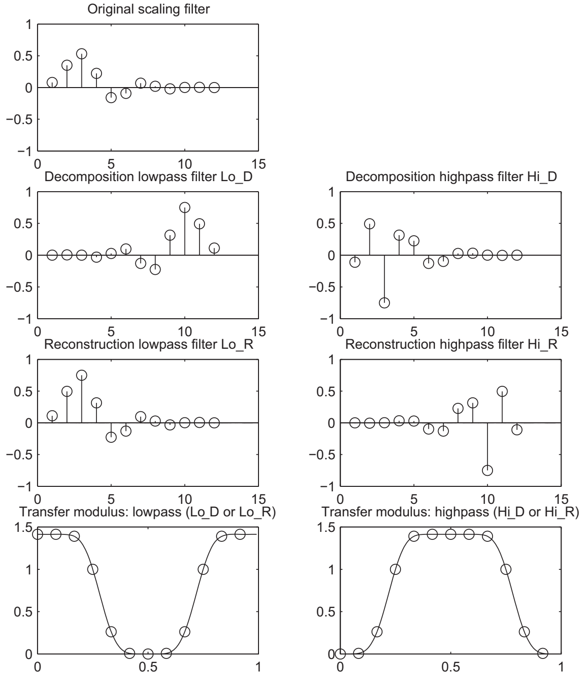
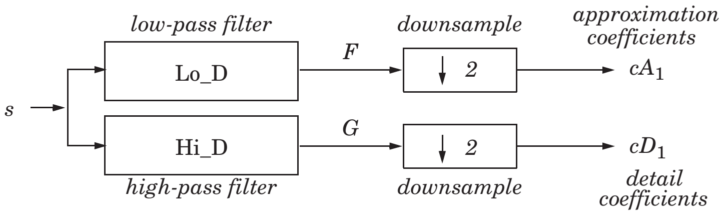


Figure 6-8: Four Wavelet Filters for db6

Algorithms

Given a signal s of length N, the DWT consists of $\log_2 N$ stages at most. Starting from s, the first step produces two sets of coefficients: approximation coefficients cA_1, and detail coefficients cD_1. These vectors are obtained by convolving s with the low-pass filter Lo_D for approximation, and with the high-pass filter Hi_D for detail, followed by dyadic decimation.

More precisely, the first step is



where $\boxed{\text{X}}$ Convolve with filter X.

$\boxed{\downarrow 2}$ Keep the even indexed elements (see `dyaddown`).

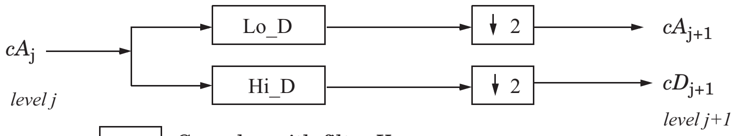
The length of each filter is equal to $2n$. If N= length (s), the signals F and G are of length N+ $2n$ - 1, and then the coefficients cA_1 and cD_1 are of length

$$\text{floor}\left(\frac{(N-1)}{2}+n\right)$$

The next step splits the approximation coefficients cA_1 in two parts using the same scheme, replacing s by cA_1 and producing cA_2 and cD_2, and so on.

One-Dimensional DWT

Decomposition Step



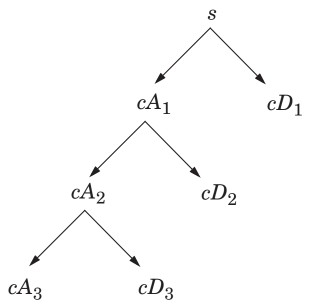
where $\boxed{X}$ Convolve with filter X.

$\boxed{\downarrow 2}$ Downsample.

Initialization $cA_0 = s.$

So the wavelet decomposition of the signal *s* analyzed at level *j* has the following structure: $[cA_j, cD_j, ..., cD_1]$.

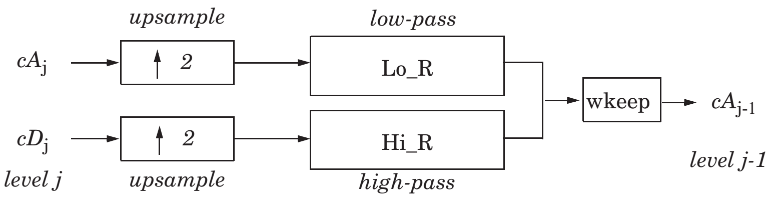
This structure contains for J = 3 the terminal nodes of the following tree.



- Conversely, starting from cA_j and cD_j, the IDWT reconstructs cA_{j-1}, inverting the decomposition step by inserting zeros and convolving the results with the reconstruction filters.

One-Dimensional IDWT

Reconstruction Step



where $\boxed{\uparrow 2}$ Insert zeros at odd-indexed elements.

$\boxed{X}$ Convolve with filter X.

$\boxed{\text{wkeep}}$ Take the central part of U with the convenient length.

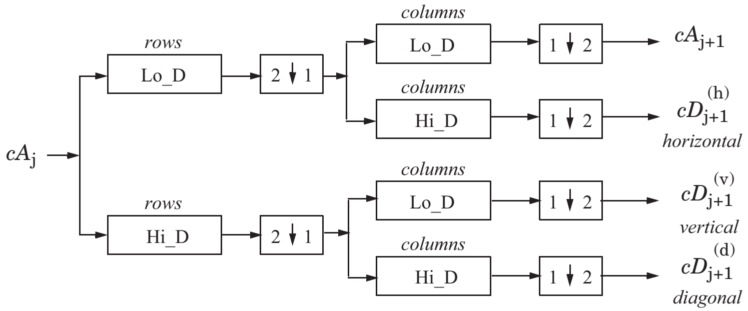
- For images, a similar algorithm is possible for two-dimensional wavelets and scaling functions obtained from one-dimensional wavelets by tensorial product.

  This kind of two-dimensional DWT leads to a decomposition of approximation coefficients at level *j* in four components: the approximation at level *j* + 1 and the details in three orientations (horizontal, vertical, and diagonal).

  The following charts describe the basic decomposition and reconstruction steps for images.

Two-Dimensional DWT

Decomposition Step



where $\boxed{2 \downarrow 1}$ Downsample columns: keep the even indexed columns.

$\boxed{1 \downarrow 2}$ Downsample rows: keep the even indexed rows.

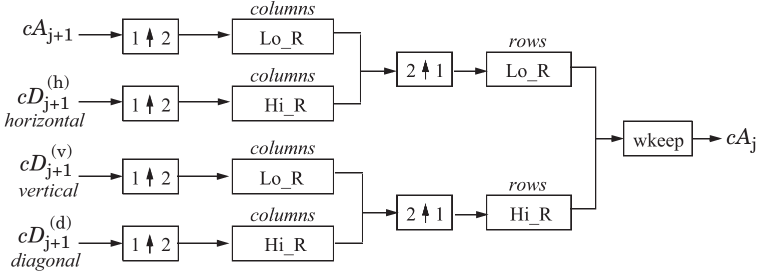
rows $\boxed{X}$ Convolve with filter X the rows of the entry.

columns $\boxed{X}$ Convolve with filter X the columns of the entry.

Initialization $CA_0 = s$ for the decomposition initialization.

Two-Dimensional IDWT

Reconstruction Step



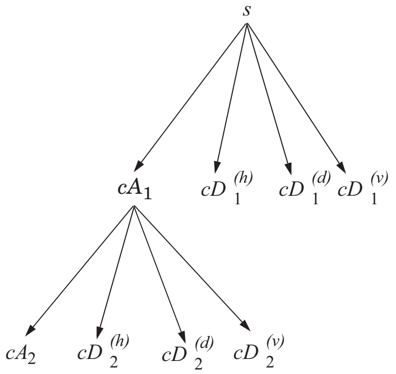
where $\boxed{2 \uparrow 1}$ Upsample columns: insert zeros at odd-indexed columns.

$\boxed{1 \uparrow 2}$ Upsample rows: insert zeros at odd-indexed rows.

rows $\boxed{X}$ Convolve with filter X the rows of the entry.

columns $\boxed{X}$ Convolve with filter X the columns of the entry.

So, for $J = 2$, the two-dimensional wavelet tree has the following form.



Finally, let us mention that, for biorthogonal wavelets, the same algorithms hold but the decomposition filters on one hand and the reconstruction filters on the other hand are obtained from two distinct scaling functions associated with two multiresolution analyses in duality.

In this case, the filters for decomposition and reconstruction are, in general, of different odd lengths. This situation occurs, for example, for "splines" biorthogonal wavelets used in the toolbox. By zero-padding, the four filters can be extended in such a way that they will have the same even length.

Why Does Such an Algorithm Exist?

The previous paragraph describes algorithms designed for finite-length signals or images. To understand the rationale, we must consider infinite-length signals. The methods for the extension of a given finite-length signal are described in "Dealing with Border Distortion" on page 6-35.

Let us denote h = Lo_R and g = Hi_R and focus on the one-dimensional case.

We first justify how to go from level j to level j+1, for the approximation vector. This is the main step of the decomposition algorithm for the computation of the approximations. The details are calculated in the same way using the filter g instead of filter h.

Let $(A_k^{(j)})_{k \in Z}$ be the coordinates of the vector A_j:

$$A_j = \sum_k A_k^{(j)} \phi_{j,k}$$

and $A_k^{(j+1)}$ the coordinates of the vector A_{j+1}:

$$A_{j+1} = \sum_k A_k^{(j+1)} \phi_{j+1,k}$$

$A_k^{(j+1)}$ is calculated using the formula

$$A_k^{(j+1)} = \sum_n h_{n-2k} A_n^{(j)}$$

This formula resembles a convolution formula.

The computation is very simple.

Let us define

$$\tilde{h}(k) = h(-k), \text{ and } F_k^{(j+1)} = \sum_n \tilde{h}_{k-n} A_n^{(j)}$$

The sequence $F^{(j+1)}$ is the filtered output of the sequence $A^{(j)}$ by the filter $\tilde{h}$.

We obtain

$$A_k^{(j+1)} = F_{2k}^{(j+1)}$$

We have to take the even index values of F. This is downsampling.

The sequence $A^{(j+1)}$ is the downsampled version of the sequence $F^{(j+1)}$.

The initialization is carried out using $A_k^{(0)} = s(k)$, where $s(k)$ is the signal value at time k.

There are several reasons for this surprising result, all of which are linked to the multiresolution situation and to a few of the properties of the functions $\phi_{j,k}$ and $\psi_{j,k}$.

Let us now describe some of them.

1. The family $(\phi_{0,k}, k \in Z)$ is formed of orthonormal functions. As a consequence for any j, the family $(\phi_{j,k}, k \in Z)$ is orthonormal.
2. The double indexed family $(\psi_{j,k}, j \in Z, k \in Z)$ is orthonormal.
3. For any j, the $(\phi_{j,k}, k \in Z)$ are orthogonal to $(\psi_{j',k}, j' \le j, k \in Z)$.
4. Between two successive scales, we have a fundamental relation, called the *twin-scale relation*.

Twin-Scale Relation for ϕ

$\phi_{1,0} = \sum_{k \in Z} h_k \phi_{0,k}$	$\phi_{j+1,0} = \sum_{k \in Z} h_k \phi_{j,k}$

This relation introduces the algorithm's h filter ($h_n = \sqrt{2} w_n$). For more information, see "Filters Used to Calculate the DWT and IDWT" on page 6-19.

5. We check that:
   a. The coordinate of $\phi_{j+1,0}$ on $\phi_{j,k}$ is h_k and does not depend on j.

b The coordinate of $\phi_{j+1,n}$ on $\phi_{j,k}$ is equal to $\langle \phi_{j+1,n}, \phi_{j,k} \rangle = h_{k-2n}$.

6 These relations supply the ingredients for the algorithm.

7 Up to now we used the filter h. The high-pass filter g is used in the twin scales relation linking the ψ and ϕ functions. Between two successive scales, we have the following twin-scale fundamental relation.

Twin-Scale Relation Between ψ and φ

$\psi_{1,0} = \sum_{k \in Z} g_k \phi_{0,k}$	$\psi_{j+1,0} = \sum_{k \in Z} g_k \phi_{j,k}$

8 After the decomposition step, we justify now the reconstruction algorithm by building it. Let us simplify the notation. Starting from A_1 and D_1, let us study $A_0 = A_1 + D_1$. The procedure is the same to calculate $A_j = A_{j+1} + D_{j+1}$.

Let us define α_n, δ_n, α_k^0 by

$$A_1 = \sum_n \alpha_n \phi_{1,n} \qquad D_1 = \sum_n \delta_n \psi_{1,n} \qquad A_0 = \sum_k \alpha_k^0 \phi_{0,k}$$

Let us assess the α_k^0 coordinates as

$$\alpha_k^0 = \langle A_0, \phi_{0,k} \rangle = \langle A_1 + D_1, \phi_{0,k} \rangle = \langle A_1, \phi_{0,k} \rangle + \langle D_1, \phi_{0,k} \rangle$$

$$= \sum_n \alpha_n \langle \phi_{1,n}, \phi_{0,k} \rangle + \sum_n \delta_n \langle \psi_{1,n}, \phi_{0,k} \rangle$$

$$= \sum_n \alpha_n h_{k-2n} + \sum_n \delta_n g_{k-2n}$$

We will focus our study on the first sum $\sum_n \alpha_n h_{k-2n}$; the second sum $\sum_n \delta_n g_{k-2n}$ is handled in a similar manner.

The calculations are easily organized if we note that (taking $k = 0$ in the previous formulas, makes things simpler)

$$\sum_n \alpha_n h_{-2n} = \ldots + \alpha_{-1} h_2 + \alpha_0 h_0 + \alpha_1 h_{-2} + \alpha_2 h_{-4} + \ldots$$

$$= \ldots + \alpha_{-1} h_2 + 0h_1 + \alpha_0 h_0 + 0h_{-1} + \alpha_1 h_{-2} + 0h_{-3} + \alpha_2 h_{-4} + \ldots$$

If we transform the (α_n) sequence into a new sequence $(\tilde{\alpha}_n)$ defined by

..., α_{-1}, 0, α_0, 0, α_1, 0, α_2, 0, ... that is precisely

$$\alpha_{2n} = \alpha_n, \alpha_{2n+1} = 0$$

Then

$$\sum_n \alpha_n h_{-2n} = \sum_n \tilde{\alpha}_n h_{-n}$$

and by extension

$$\sum_n \alpha_n h_{k-2n} = \sum_n \tilde{\alpha}_n h_{k-n}$$

Since

$$\alpha_k^0 = \sum_n \tilde{\alpha}_n h_{k-n} + \sum_n \tilde{\delta}_n g_{k-n}$$

the reconstruction steps are:

1 Replace the α and δ sequences by upsampled versions $\tilde{\alpha}$ and $\tilde{\delta}$ inserting zeros.

2 Filter by h and g respectively.

3 Sum the obtained sequences.

One-Dimensional Wavelet Capabilities

Basic One-Dimensional Objects.

	Objects	Description
Signal in original time	s	Original signal
	A_k, $0 \le k \le j$	Approximation at level k
	D_k, $1 \le k \le j$	Detail at level k
Coefficients in scale-related time	cA_k, $1 \le k \le j$	Approximation coefficients at level k
	cD_k, $1 \le k \le j$	Detail coefficients at level k
	$[cA_j, cD_j, ..., cD_1]$	Wavelet decomposition at level j, $j \ge 1$

Analysis-Decomposition Capabilities.

Purpose	Input	Output	M-File
Single-level decomposition	s	cA_1, cD_1	`dwt`
Single-level decomposition	cA_j	cA_{j+1}, cD_{j+1}	`dwt`
Decomposition	s	$[cA_j, cD_j, ..., cD_1]$	`wavedec`

Synthesis-Reconstruction Capabilities.

Purpose	Input	Output	M-File
Single-level reconstruction	cA_1, cD_1	s or A_0	`idwt`
Single-level reconstruction	cA_{j+1}, cD_{j+1}	cA_j	`idwt`
Full reconstruction	$[cA_j, cD_j, ..., cD_1]$	s or A_0	`waverec`
Selective reconstruction	$[cA_j, cD_j, ..., cD_1]$	A_l, D_m	`wrcoef`

Decomposition Structure Utilities. .

Purpose	Input	Output	M-File
Extraction of detail coefficients	$[cA_j, cD_j, ..., cD_1]$	cD_k, $1 \le k \le j$	`detcoef`
Extraction of approximation coefficients	$[cA_j, cD_j, ..., cD_1]$	cA_k, $0\le k \le j$	`appcoef`
Recomposition of the decomposition structure	$[cA_j, cD_j, ..., cD_1]$	$[cA_k, cD_k, ..., cD_1]$ $1 \le k \le j$	`upwlev`

To illustrate command-line mode for one-dimensional capabilities, see "One-Dimensional Analysis Using the Command Line" on page 2-31.

Two-Dimensional Wavelet Capabilities

Basic Two-Dimensional Objects.

	Objects	Description
Image in original resolution	s	Original image
	A_0	Approximation at level 0
	A_k, $1 \le k \le j$	Approximation at level k
	D_k, $1 \le k \le j$	Details at level k
Coefficients in scale-related resolution	cA_k, $1 \le k \le j$	Approximation coefficients at level k
	cD_k, $1 \le k \le j$	Detail coefficients at level k
	$[cA_\mathrm{j}, cD_\mathrm{j}, ..., cD_1]$	Wavelet decomposition at level j

D_k stands for $[D_k^{(h)}, D_k^{(v)}, D_k^{(d)}]$, the horizontal, vertical, and diagonal details at level k.

The same holds for cD_k, which stands for $[cD_k^{(h)}, cD_k^{(v)}, cD_k^{(d)}]$.

The two-dimensional M-files are the same as those for the one-dimensional case, but with a `2` appended on the end of the command.

For example, `idwt` becomes `idwt2`. For more information, see "One-Dimensional Wavelet Capabilities" on page 6-32.

To illustrate command-line mode for two-dimensional capabilities, see "Two-Dimensional Analysis Using the Command Line" on page 2-69.

Dealing with Border Distortion

Classically, the DWT is defined for sequences with length of some power of 2, and different ways of extending samples of other sizes are needed. Methods for extending the signal include zero-padding, smooth padding, periodic extension, and boundary value replication (symmetrization).

The basic algorithm for the DWT is not limited to dyadic length and is based on a simple scheme: convolution and downsampling. As usual, when a convolution is performed on finite-length signals, border distortions arise.

Signal Extensions: Zero-Padding, Symmetrization, and Smooth Padding

To deal with border distortions, the border should be treated differently from the other parts of the signal.

Various methods are available to deal with this problem, referred to as "wavelets on the interval" (see [CohDJV93] in "References" on page 6-155). These interesting constructions are effective in theory but are not entirely satisfactory from a practical viewpoint.

Often it is preferable to use simple schemes based on signal extension on the boundaries. This involves the computation of a few extra coefficients at each stage of the decomposition process to get a perfect reconstruction. It should be noted that extension is needed at each stage of the decomposition process.

Details on the rationale of these schemes are in Chapter 8 of the book *Wavelets and Filter Banks*, by Strang and Nguyen (see [StrN96] in "References" on page 6-155).

The available signal extension modes are as follows (see `dwtmode`):

- **Zero-padding** (`'zpd'`): This method is used in the version of the DWT given in the previous sections and assumes that the signal is zero outside the original support.

  The disadvantage of zero-padding is that discontinuities are artificially created at the border.

- **Symmetrization** (`'sym'`): This method assumes that signals or images can be recovered outside their original support by symmetric boundary value replication.

  It is the default mode of the wavelet transform in the toolbox.

  Symmetrization has the disadvantage of artificially creating discontinuities of the first derivative at the border, but this method works well in general for images.
- **Smooth padding of order 1** (`'spd'` or `'sp1'`): This method assumes that signals or images can be recovered outside their original support by a simple first-order derivative extrapolation: padding using a linear extension fit to the first two and last two values.

  Smooth padding works well in general for smooth signals.
- **Smooth padding of order 0** (`'sp0'`): This method assumes that signals or images can be recovered outside their original support by a simple constant extrapolation. For a signal extension this is the repetition of the first value on the left and last value on the right.
- **Periodic-padding (1)** (`'ppd'`): This method assumes that signals or images can be recovered outside their original support by periodic extension.

  The disadvantage of periodic padding is that discontinuities are artificially created at the border.

The DWT associated with these five modes is slightly redundant. But IDWT ensures a perfect reconstruction for any of the five previous modes whatever the extension mode used for DWT.

- **Periodic-padding (2)** (`'per'`): If the signal length is odd, the signal is first extended by adding an extra-sample equal to the last value on the right. Then a minimal periodic extension is performed on each side. The same kind of rule exists for images. This extension mode is used for SWT (1-D & 2-D).

This last mode produces the smallest length wavelet decomposition. But the extension mode used for IDWT must be the same to ensure a perfect reconstruction.

Before looking at an illustrative example, let us compare some properties of the theoretical Discrete Wavelet Transform versus the actual DWT.

The theoretical DWT is applied to signals that are defined on an infinite length time interval (Z). For an orthogonal wavelet, this transform has the following desirable properties:

1. Norm preservation

   Let cA and cD be the approximation and detail of the DWT coefficients of an infinite length signal X. Then the l^2-*norm* is preserved:

   $$\|X\|^2 = \|cA\|^2 + \|cD\|^2$$
2. Orthogonality

   Let A and D be the reconstructed approximation and detail. Then, A and D are orthogonal and

   $$\|X\|^2 = \|A\|^2 + \|D\|^2$$
3. Perfect reconstruction

   $X = A + D$

Since the DWT is applied to signals that are defined on a finite-length time interval, extension is needed for the decomposition, and truncation is necessary for reconstruction.

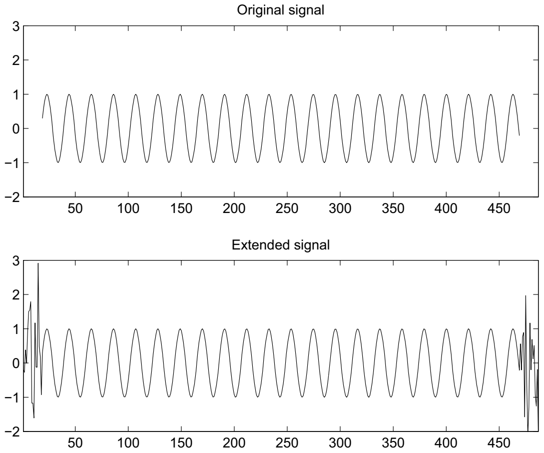
To ensure the crucial property **3** (perfect reconstruction) for arbitrary choices of

- The signal length
- The wavelet
- The extension mode

the properties **1** and **2** can be lost. These properties hold true for an extended signal of length usually larger than the length of the original signal. So only the perfect reconstruction property is always preserved. Nevertheless if the DWT is performed using the periodic extension mode (`'per'`) and if the length of the signal is divisible by 2^J, where J is the maximum level decomposition, the properties **1**, **2**, and **3** remain true.

It is interesting to notice that if arbitrary extension is used, and decomposition performed using the convolution-downsampling scheme, perfect reconstruction is recovered using `idwt` or `idwt2`. This point is illustrated below.

```
% Set initial signal and get filters.
x = sin(0.3*[1:451]); w = 'db9';
[Lo_D,Hi_D,Lo_R,Hi_R] = wfilters(w);
% In fact using a slightly redundant scheme, any signal
% extension strategy works well.
% For example use random padding.
```



```
lx = length(x); lf = length(Lo_D);
randn('seed',654);
ex = [randn(1,lf) x randn(1,lf)];
axis([1 lx+2*lf -2 3])
subplot(211), plot(lf+1:lf+lx,x), title('Original signal')
axis([1 lx+2*lf -2 3])
subplot(212), plot(ex), title('Extended signal')
axis([1 lx+2*lf -2 3])

% Decomposition.
la = floor((lx+lf-1)/2);
ar = wkeep(dyaddown(conv(ex,Lo_D)),la);
dr = wkeep(dyaddown(conv(ex,Hi_D)),la);
% Reconstruction.
```

```
xr = idwt(ar,dr,w,lx);

% Check perfect reconstruction.
err0 = max(abs(x-xr))

err0 =
    3.0464e-11
```

Now let us illustrate the differences between the first three methods both for 1-D and 2-D signals.

Zero-Padding.

Using the GUI we will examine the effects of zero-padding.

1 From the MATLAB® prompt, type

```
dwtmode('zpd')
```

2 From the MATLAB prompt, type `wavemenu`.

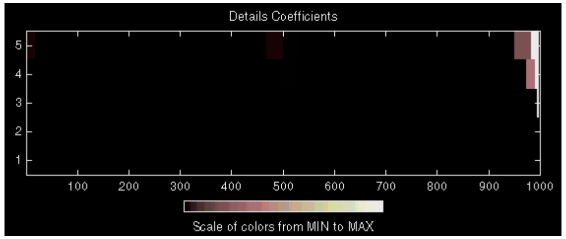
The **Wavelet Toolbox Main Menu** appears.

3 Click the **Wavelet 1-D** menu item.The discrete wavelet analysis tool for one-dimensional signal data appears.

4 From the **File** menu, choose the **Example Analysis** option and select **Basic Signals > with db2 at level 5 > Two nearby discontinuities**.

5 Select **Display Mode: Show and Scroll**.

The detail coefficients clearly show the signal end effects.



Symmetric Extension.

6 From the MATLAB prompt, type

```
dwtmode('sym')
```

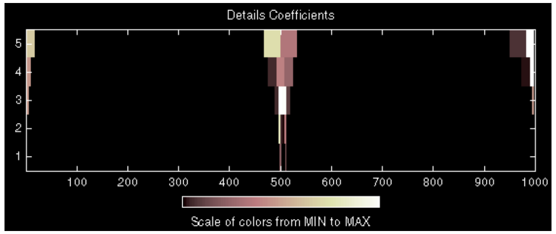
7 Click the **Wavelet 1-D** menu item.

The discrete wavelet analysis tool for one-dimensional signal data appears.

8 From the **File** menu, choose the **Example Analysis** option and select **Basic Signals > with db2 at level 5 > Two nearby discontinuities**.

9 Select **Display Mode: Show and Scroll**.

The detail coefficients show the signal end effects are present, but the discontinuities are well detected.



Smooth Padding.

10 From the MATLAB prompt, type

```
dwtmode('spd')
```

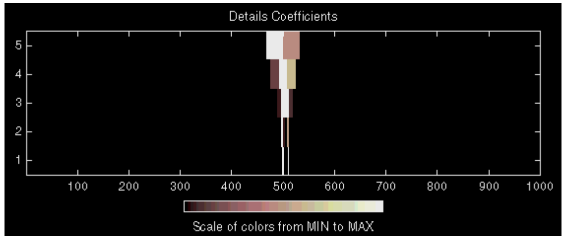
11 Click the **Wavelet 1-D** menu item.

The discrete wavelet analysis tool for one-dimensional signal data appears.

12 From the **File** menu, choose the **Example Analysis** option and select **Basic Signals > with db2 at level 5 --> Two nearby discontinuities**.

13 Select **Display Mode: Show and Scroll**.

The detail coefficients show the signal end effects are not present, and the discontinuities are well detected.

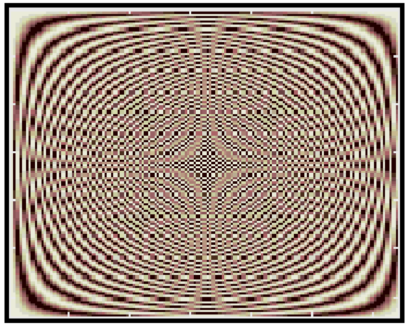


Let us now consider an image example.

Original Image.

1 From the MATLAB prompt, type

```
load geometry;
% X contains the loaded image and
% map contains the loaded colormap.
nbcol = size(map,1);
colormap(pink(nbcol));
image(wcodemat(X,nbcol));
```

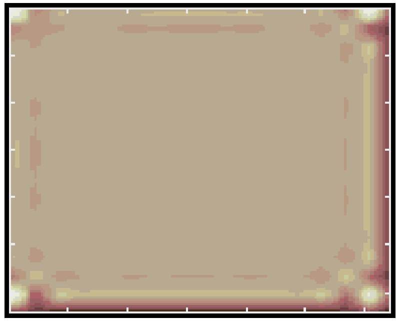


Zero-Padding.

Now we set the extension mode to zero-padding and perform a decomposition of the image to level 3 using the sym4 wavelet. Then we reconstruct the approximation of level 3.

2 From the MATLAB prompt, type

```
lev = 3; wname = 'sym4';
dwtmode('zpd')
[c,s] = wavedec2(X,lev,wname);
a = wrcoef2('a',c,s,wname,lev);
image(wcodemat(a,nbcol));
```

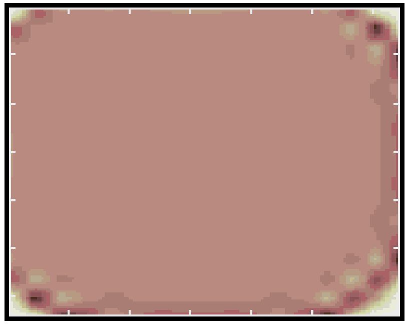


Symmetric Extension.

Now we set the extension mode to symmetric extension and perform a decomposition of the image again to level 3 using the sym4 wavelet. Then we reconstruct the approximation of level 3.

3 From the MATLAB prompt, type

```
dwtmode('sym')
[c,s] = wavedec2(X,lev,wname);
a = wrcoef2('a',c,s,wname,lev);
image(wcodemat(a,nbcol));
```

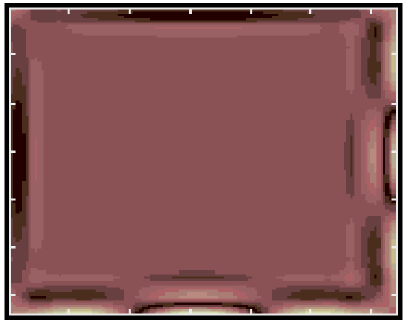


Smooth Padding.

Now set the extension mode to smooth padding and perform a decomposition of the image again to level 3 using the sym4 wavelet. Then reconstruct the approximation of level 3.

4 From the MATLAB prompt, type

```
dwtmode('spd')
[c,s] = wavedec2(X,lev,wname);
a = wrcoef2('a',c,s,wname,lev);
image(wcodemat(a,nbcol));
```

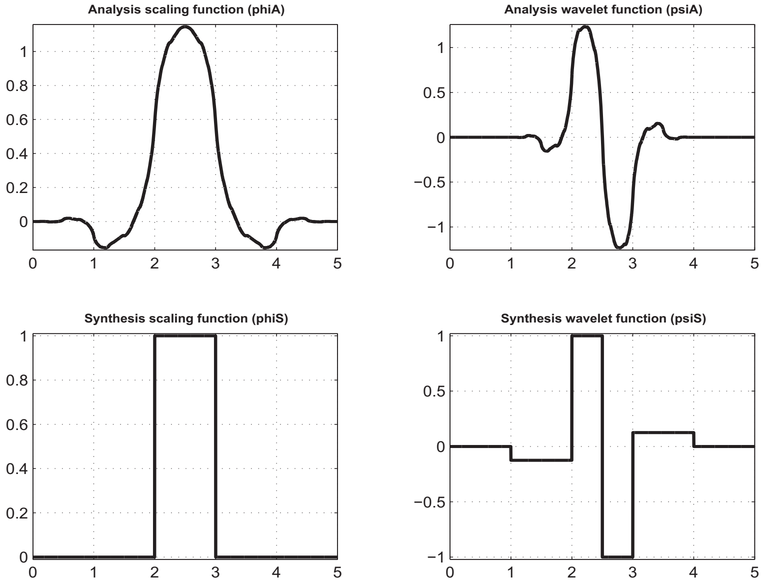


```
[LoDB,HiDB,LoRB,HiRB] = wfilters('bior1.3');
samewavelet =
isequal([LoDB,HiDB,LoRB,HiRB],[LoDN,-HiDN,LoRN,HiRN])

samewavelet =

     1

% Visualize the two times two pairs of scaling and wavelet
% functions.
bswfun(LoDN,HiDN,LoRN,HiRN,'plot');
```



Frequently Asked Questions

Continuous or Discrete Analysis?

When is continuous analysis more appropriate than discrete analysis? To answer this, consider the related questions: Do you need to know all values of a continuous decomposition to reconstruct the signal s exactly? Can you perform nonredundant analysis?

When the energy of the signal is finite, not all values of a decomposition are needed to exactly reconstruct the original signal, provided that you are using a wavelet that satisfies some admissibility condition (see [Dau92] pages 7, 24, and 27). Usual wavelets satisfy this condition. In which case, a continuous-time signal s is characterized by the knowledge of the discrete transform $C(j, k), (j,k) \in Z^2$.

In such cases, discrete analysis is sufficient and continuous analysis is redundant. When the signal is recorded in continuous time or on a very fine time grid, both analyses are possible. Which should be used? It depends; each one has its own advantages:

- Discrete analysis ensures space-saving coding and is sufficient for exact reconstruction.
- Continuous analysis is often easier to interpret, since its redundancy tends to reinforce the traits and makes all information more visible. This is especially true of very subtle information. Thus, the analysis gains in “readability” and in ease of interpretation what it loses in terms of saving space.

Why Are Wavelets Useful for Space-Saving Coding?

The family of functions $(\phi_{0,k};\psi_{j,l})\ j \le 0,\ (k, l) \in Z^2$ used for the analysis is an orthogonal basis, therefore leading to nonredundancy. The orthogonality properties are $\phi_{0,k} \perp \psi_{j',k'}$ as soon as $j' \le 0$, and $\psi_{j,k} \perp \psi_{j',k'}$ as soon as $(j,k) \ne (j',k')$.

Let us remember that for a one-dimensional signal, $u \perp v$ stands for

$$\int_R u(x)v(x)dx = 0$$

For biorthogonal wavelets, the idea is similar.

What Is the Advantage Having Zero Average and Sometimes Several Vanishing Moments?

When the wavelet's $k + 1$ moments are equal to zero ($\int_R t^j \psi(t) dt = 0$ for $j = 0, \ldots, k$) all the polynomial signals $s(t) = \sum_{0 \le j \le k} a_j t^j$ have zero wavelet coefficients.

As a consequence, the details are also zero. This property ensures the suppression of signals that are polynomials of a degree lower or equal to k.

What About the Regularity of a Wavelet ψ?

In theoretical and practical studies, the notion of regularity has been increasing in importance. Wavelets are tools used to study regularity and to conduct local studies. Deterministic fractal signals or Brownian motion trajectories are locally very irregular; for example, the latter are continuous signals, but their first derivative exists almost nowhere.

The definition of the concept of regularity is somewhat technical. To make things simple, we will define the regularity s of a signal f.

If the signal is s-time continuously differentiable at x_0 and s is an integer (≥ 0), then the regularity is s.

If the derivative of f of order m resembles $|x - x_0|^r$ locally around x_0, then $s = m + r$ with $0 < r < 1$.

The regularity of f in a domain is that of its least regular point.

The greater s, the more regular the signal.

The regularity of certain wavelets is known. The following table gives some indications for Daubechies wavelets.

ψ	***db*1 = Haar**	***db*2**	***db*3**	***db*4**	***db*5**	***db*7**	***db*10**
Regularity	Discontinuous	0.5	0.91	1.27	1.59	2.15	2.90

We have an asymptotic relation linking the size of the support of the Daubechies wavelets dbN and their regularity: when $N \to \infty$,

$$\text{length(support)} = 2N, \text{ regularity } s \approx \frac{N}{5}.$$

The functions are more regular at certain points than at others (see Figure 6-9 on page 6-63).

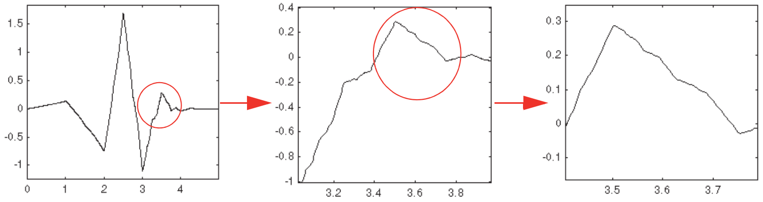


Figure 6-9: Zooming in on a *db*3 Wavelet

Selecting a regularity and a wavelet for the regularity is useful in estimations of the local properties of functions or signals. This can be used, for example, to make sure that a signal has a constant regularity at all points. Work by Donoho, Johnstone, Kerkyacharian, and Picard on function estimation and nonlinear regression is currently under way to adapt the statistical estimators to unknown regularity. See also the remarks by I. Daubechies (see [Dau92] page 301).

From a practical viewpoint, these questions arise in the world of finance in dealing with monetary and stock markets where detailed studies of very fast transactions are required.

Are Wavelets Useful in Fields Other Than Signal or Image Processing?

- From a theoretical viewpoint, wavelets are used to characterize large sets of mathematical functions and are used in the study of operators linked to partial differential equations.
- From a practical viewpoint, wavelets are used in several fields of numerical analysis, making certain complex calculations easier to handle or more precise.

What Functions Are Candidates to Be a Wavelet?

If a function f is continuous, has null moments, decreases quickly towards 0 when x tends towards infinity, or is null outside a segment of R, it is a likely candidate to become a wavelet.

More precisely, the admissibility condition for $\psi \in L^1(R) \cap L^2(R)$ is

$$\int_{R^-} \frac{|\hat{\psi}(s)|^2}{|s|} ds = \int_{R^+} \frac{|\hat{\psi}(s)|^2}{|s|} ds = K_\psi < +\infty$$

The family of shifts and dilations of ψ allows all finite energy signals to be reconstructed using the details in all scales. This allows only continuous analysis.

A wavelet satisfying only the admissibility condition is said to be crude.

In the toolbox, the ψ wavelet is usually associated with a scaling function ϕ. There are, however, some ψ wavelets for which we do not know how to associate a ϕ. In some cases we know how to prove that ϕ does not exist, for example, the Mexican hat wavelet.

Is It Easy to Build a New Wavelet?

For a minimal requirement on the wavelet properties, it is easy to build a new wavelet but not very interesting unless the new wavelet is adapted to a specific task. For example the paragraph “New Wavelet for CWT” on page 2-216 explains how to obtain wavelets adapted to a given pattern, which can then be used for an accurate pattern detection. If more interesting properties (like the existence of ϕ for example) are needed, then building the wavelet is more difficult. Let us mention that an interesting approach is the lifting method (see “Lifting Method for Constructing Wavelets” on page 6-51).

Very few wavelets have an explicit analytical expression. Notable exceptions are wavelets that are piecewise polynomials (Haar, Battle-Lemarié; see [Dau92] in “References” on page 6-155), Morlet, or Mexican hat.

Wavelets, even `db2`, `db3`, ..., are defined by functional equations. The solution is numerical, and is accomplished using a fairly simple algorithm.

The basic property is the existence of a linear relation between the two functions $\phi(x/2)$ and $\phi(x)$. Another relation of the same type links $\psi(x/2)$ to $\phi(x)$. These are the relations of the two scales, the twin-scale relations.

Indeed there are two sequences h and g of coefficients such that

$$h \in l^2(Z), g \in l^2(Z)$$

and

$$\frac{1}{2}\phi\left(\frac{x}{2}\right) = \frac{1}{\sqrt{2}} \sum_{n \in Z} h_n \phi(x-n)$$

$$\frac{1}{2}\psi\left(\frac{x}{2}\right) = \frac{1}{\sqrt{2}} \sum_{n \in Z} g_n \phi(x-n)$$

By rewriting these formulas using Fourier transforms (expressed using a hat) we obtain

$$\hat{\phi}(2\omega) = \frac{1}{\sqrt{2}}\hat{h}(\omega)\hat{\phi}(\omega) \qquad \hat{\psi}(2\omega) = \frac{1}{\sqrt{2}}\hat{g}(\omega)\hat{\phi}(\omega)$$

There are ϕ functions for which the h has a finite impulse response (FIR): there is only a finite number of nonzero h_n coefficients. The associated wavelets were built by I. Daubechies (see [Dau92] in “References” on page 6-155) and are used extensively in the toolbox. The reader can refer to page 164 and Chapter 10 of the book *Wavelets and Filter Banks*, by Strang and Nguyen (see [StrN96] in “References” on page 6-155).

What Is the Link Between Wavelet and Fourier Analysis?

Wavelet analysis complements the Fourier analysis for which there are several functions: `fft` in MATLAB® software and `spectrum` and `sptool` in Signal Processing Toolbox™ software.

Fourier analysis uses the basic functions $\sin(\omega t)$, $\cos(\omega t)$, and $\exp(i\omega t)$.

- In the frequency domain, these functions are perfectly localized. The functions are suited to the analysis and synthesis of signals with a simple spectrum, which is very well localized in frequency; for example, $\sin(\omega_1 t) + 0.5 \sin(\omega_2 t) - \cos(\omega_3 t)$.
- In the time domain, these functions are not localized. It is difficult for them to analyze or synthesize complex signals presenting fast local variations such as transients or abrupt changes: the Fourier coefficients for a frequency ω will depend on all values in the signal. To limit the difficulties involved, it is possible to “window” the signal using a regular function, which is zero or nearly zero outside a time segment $[-m, m]$.

We then build “a well localized slice” as I. Daubechies calls it (see page 2 of [Dau92] in “References” on page 6-155). The windowed-Fourier analysis coefficients are the doubly indexed coefficients:

$$G_s(\omega,t) = \int_R s(u)g(t-u)e^{-i\omega u}du$$

The analogy of this formula with that of the wavelet coefficients is obvious:

$$C(a,t) = \int_R s(u)\left(\frac{1}{\sqrt{a}}\right)\psi\left(\frac{t-u}{a}\right)\, du$$

The large values of a correspond to small values of ω.

The Fourier coefficient $G_s(\omega,t)$ depends on the values of the signal s on the segment $[t - m, t + m]$ with a constant width. If ψ, like g, is zero outside of $[-m, m]$, the $C(a,t)$ coefficients will depend on the values of the signal s on the segment $[t - am, t + am]$ of width $2am$, which varies as a function of a. This slight difference solves several difficulties, allowing a kind of time-windowed analysis, different at the various scales a.

The wavelets stay competitive, however, even in contexts considered favorable for the Fourier technique. I. Daubechies (see [Dau92] pages 3 to 6) gives an example of windowed-Fourier processing and complex Morlet wavelet processing, $\psi(t) = Ce^{-t^2/\alpha^2}(e^{i\pi t} - e^{-\pi^2\alpha^2/4})$ with $\alpha = 4$, of a signal composed mainly of the sum of two sines. This wavelet analysis gives good results.

How to Connect Scale to Frequency?

A common question is, what is the relationship between scale and frequency?

The answer can only be given in a broad sense, and it's better to speak about the pseudo-frequency corresponding to a scale.

A way to do it is to compute the center frequency F_c of the wavelet and to use the following relationship (see [Abr97] in "References" on page 6-155).

$$F_a = \frac{F_c}{a \cdot \Delta}$$

where

- a is a scale.
- Δ is the sampling period.
- F_c is the center frequency of a wavelet in Hz.
- F_a is the pseudo-frequency corresponding to the scale a, in Hz.

The idea is to associate with a given wavelet a purely periodic signal of frequency F_c. The frequency maximizing the `fft` of the wavelet modulus is F_c. The function `centfrq` can be used to compute the center frequency and it allows the plotting of the wavelet with the associated approximation based on the center frequency. Figure 6-10 on page 6-68 shows some examples generated using the `centfrq` function.

- Four real wavelets: Daubechies wavelets of order 2 and 7, coiflet of order 1, and the Gaussian derivative of order 4.
- Two complex wavelets: the complex Gaussian derivative of order 6 and a Shannon complex wavelet.

As you can see, the center frequency-based approximation captures the main wavelet oscillations. So the center frequency is a convenient and simple characterization of the leading dominant frequency of the wavelet.

If we accept to associate the frequency F_c to the wavelet function, then when the wavelet is dilated by a factor a, this center frequency becomes F_c / a. Lastly, if the underlying sampling period is Δ, it is natural to associate to the scale a the frequency

$$F_a = \frac{F_c}{a \cdot \Delta}$$

The function `scal2frq` computes this correspondence.

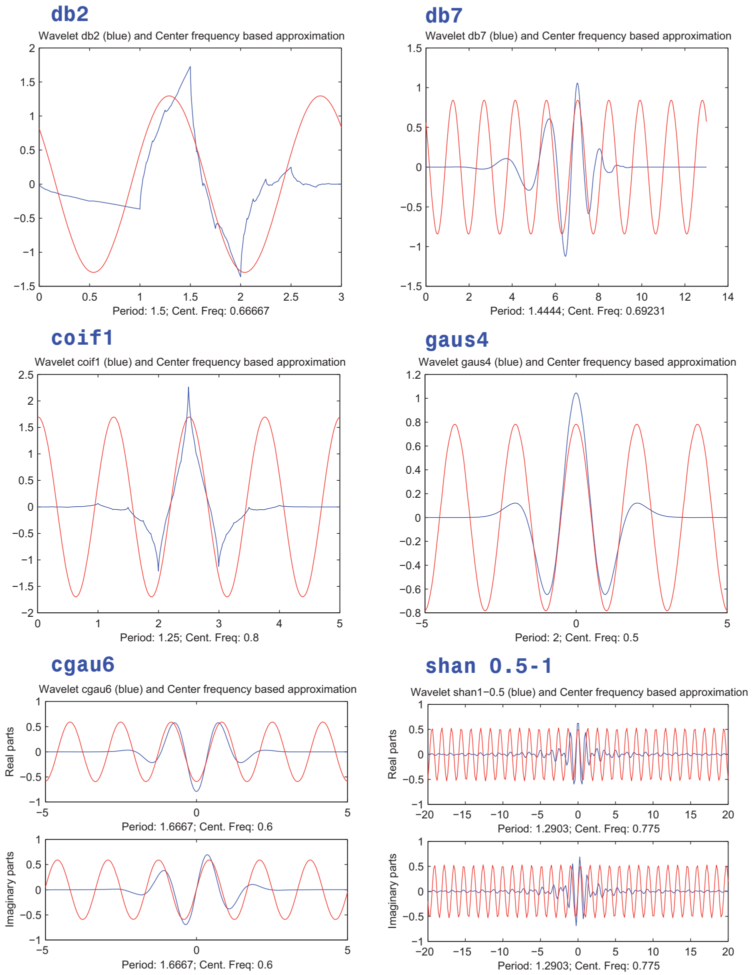


Figure 6-10: Center Frequencies for Real and Complex Wavelets

To illustrate the behavior of this procedure, consider the following simple test. We generate sine functions of sensible frequencies F_0. For each function, we shall try to detect this frequency by a wavelet decomposition followed by a translation of scale to frequency. More precisely, after a discrete wavelet decomposition, we identify the scale a^* corresponding to the maximum value of the energy of the coefficients. The translated frequency F^* is then given by

```
scal2frq(a_star,'wname',sampling_period)
```

The F^* values are close to the chosen F_0. The plots at the end of the example present the periods instead of the frequencies. If we change the F_0 values slightly, the results remain satisfactory.

For example:

```
% Set sampling period and wavelet name.
delta = 0.1; wname = 'coif3';

% Set scales.
amax = 7;
a = 2.^[1:amax];

% Compute associated pseudo-frequencies.
f = scal2frq(a,wname,delta);

% Compute associated pseudo-periods.
per = 1./f;

% Plot pseudo-periods versus scales.
subplot(211), plot(a,per)
title(['Wavelet: ',wname, ', Sampling period: ',num2str(delta)])
xlabel('Scale')
ylabel('Computed pseudo-period')

% For each scale 2^i:
% - generate a sine function of period per(i);
% - perform a wavelet decomposition;
% - identify the highest energy level;
% - compute the detected pseudo-period.
for i = 1:amax
    % Generate sine function of period
    % per(i) at sampling period delta.
    t = 0:delta:100;
    x = sin((t.*2*pi)/per(i));

    % Decompose x at level 9.
    [c,l] = wavedec(x,9,wname);

    % Estimate standard deviation of detail coefficients.
    stdc = wnoisest(c,l,[1:amax]);
    % Compute identified period.
    [y,jmax] = max(stdc);
    idper(i) = per(jmax);
end

% Compare the detected and computed pseudo-periods.
subplot(212), plot(per,idper,'o',per,per)
title('Detected vs computed pseudo-period')
xlabel('Computed pseudo-period')
ylabel('Detected pseudo-period')
```

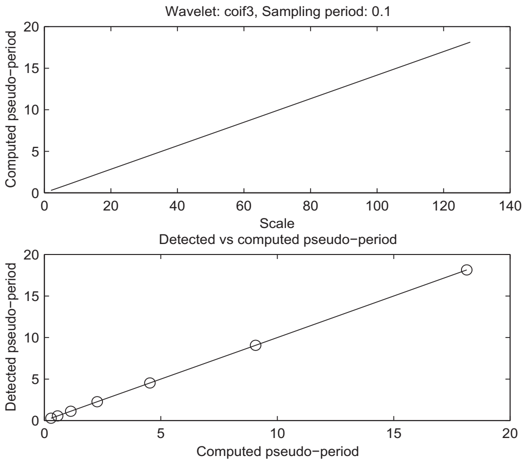


Figure 6-11: Detected Versus Computed Pseudo-Periods

Wavelet Families: Additional Discussion

There are different types of wavelet families whose qualities vary according to several criteria. The main criteria are:

- The support of ψ, $\hat{\psi}$ (and ϕ, $\hat{\phi}$): the speed of convergence to 0 of these functions ($\psi(t)$ or $\psi(\omega)$) when the time t or the frequency ω goes to infinity, which quantifies both time and frequency localizations
- The symmetry, which is useful in avoiding dephasing in image processing
- The number of vanishing moments for ψ or for ϕ (if it exists), which is useful for compression purposes
- The regularity, which is useful for getting nice features, like smoothness of the reconstructed signal or image, and for the estimated function in nonlinear regression analysis

These are associated with two properties that allow fast algorithm and space-saving coding:

- The existence of a scaling function ϕ
- The orthogonality or the biorthogonality of the resulting analysis

They may also be associated with these less important properties:

- The existence of an explicit expression
- The ease of tabulating
- The familiarity with use

Typing `waveinfo` in command-line mode displays a survey of the main properties of all wavelet families available in the toolbox.

Note that the ϕ and ψ functions can be computed using `wavefun`; the filters are generated using `wfilters`. We provide definition equations for several wavelets. Some are given explicitly by their time definitions, others by their frequency definitions, and still others by their filters.

The following table outlines the wavelet families included in the toolbox.

Wavelet Family Short Name	Wavelet Family Name
`'haar'`	Haar wavelet
`'db'`	Daubechies wavelets
`'sym'`	Symlets
`'coif'`	Coiflets
`'bior'`	Biorthogonal wavelets
`'rbio'`	Reverse biorthogonal wavelets
`'meyr'`	Meyer wavelet
`'dmey'`	Discrete approximation of Meyer wavelet
`'gaus'`	Gaussian wavelets
`'mexh'`	Mexican hat wavelet
`'morl'`	Morlet wavelet
`'cgau'`	Complex Gaussian wavelets
`'shan'`	Shannon wavelets
`'fbsp'`	Frequency B-Spline wavelets
`'cmor'`	Complex Morlet wavelets

Daubechies Wavelets: *dbN*

In *dbN*, *N* is the order. Some authors use 2*N* instead of *N*. More about this family can be found in [Dau92] pages 115, 132, 194, 242. By typing `waveinfo('db')`, at the MATLAB® command prompt, you can obtain a survey of the main properties of this family.

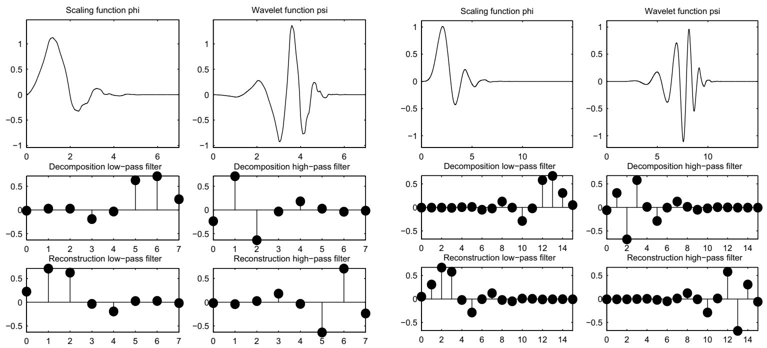


Figure 6-12: Daubechies Wavelets db4 on the Left and db8 on the Right

This family includes the Haar wavelet, written *db*1, the simplest wavelet imaginable and certainly the earliest. Using `waveinfo('haar')`, you can obtain a survey of the main properties of this wavelet.

Haar

$$\psi(x) = 1, \quad \text{if} \quad x \in [0, 0.5[$$

$$\psi(x) = -1, \quad \text{if} \quad x \in [0.5, 1[$$

$$\psi(x) = 0, \quad \text{if} \quad x \notin [0, 1[$$

$$\phi(x) = 1, \quad \text{if} \quad x \in [0, 1]$$

$$\phi(x) = 0, \quad \text{if} \quad x \notin [0, 1]$$

dbN

These wavelets have no explicit expression except for *db*1, which is the *Haar* wavelet. However, the square modulus of the transfer function of *h* is explicit and fairly simple.

- Let $P(y) = \sum_{k=0}^{N-1} C_k^{N-1+k} y^k$, where C_k^{N-1+k} denotes the binomial coefficients.

  Then

$$|m_0(\omega)|^2 = \left(\cos^2\left(\frac{\omega}{2}\right)\right)^N P\left(\sin^2\left(\frac{\omega}{2}\right)\right)$$

where $m_0(\omega) = \frac{1}{\sqrt{2}}\sum_{k=0}^{2N-1} h_k e^{-ik\omega}$

- The support length of ψ and ϕ is $2N$ - 1. The number of vanishing moments of ψ is N.
- Most *dbN* are not symmetrical. For some, the asymmetry is very pronounced.
- The regularity increases with the order. When N becomes very large, ψ and ϕ belong to $C^{\mu N}$ where μ is approximately equal to 0.2. Certainly, this asymptotic value is too pessimistic for small-order N. Note that the functions are more regular at certain points than at others.
- The analysis is orthogonal.

Symlet Wavelets: *symN*

In *symN*, *N* is the order. Some authors use 2*N* instead of *N*. Symlets are only near symmetric; consequently some authors do not call them symlets. More about symlets can be found in [Dau92], pages 194, 254-257. By typing `waveinfo('sym')` at the MATLAB command prompt, you can obtain a survey of the main properties of this family.

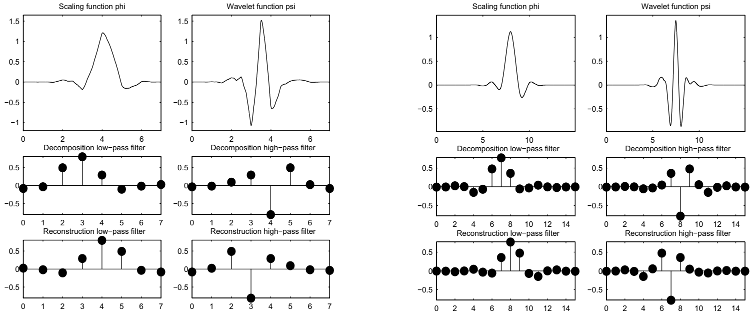


Figure 6-13: Symlets sym4 on the Left and sym8 on the Right

Daubechies proposes modifications of her wavelets that increase their symmetry can be increased while retaining great simplicity.

The idea consists of reusing the function m_0 introduced in the *dbN*, considering the $|m_0(\omega)|^2$ as a function *W* of $z = e^{i\omega}$.

Then we can factor *W* in several different ways in the form of $W(z) = U(z)\overline{U\left(\frac{1}{z}\right)}$ because the roots of *W* with modulus not equal to 1 go in pairs. If one of the roots is z_1, then $\frac{1}{z_1}$ is also a root.

- By selecting *U* such that the modulus of all its roots is strictly less than 1, we build Daubechies wavelets *dbN*. The *U* filter is a “minimum phase filter.”
- By making another choice, we obtain more symmetrical filters; these are symlets.

The symlets have other properties similar to those of the *dbN*s.

Coiflet Wavelets: *coifN*

In *coifN*, *N* is the order. Some authors use 2*N* instead of *N*. For the coiflet construction, see [Dau92] pages 258–259. By typing `waveinfo('coif')` at the MATLAB command prompt, you can obtain a survey of the main properties of this family.

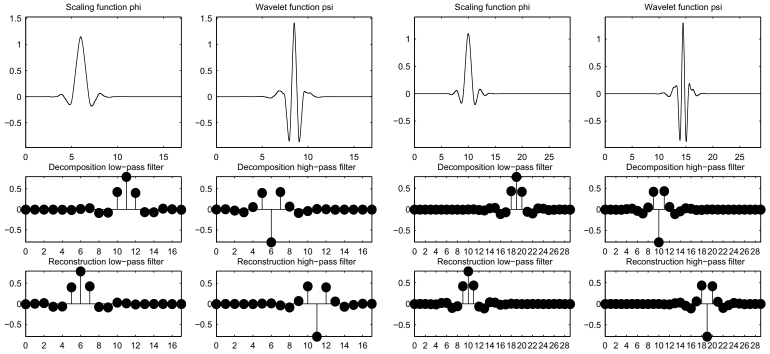


Figure 6-14: Coiflets coif3 on the Left and coif5 on the Right

Built by Daubechies at the request of Coifman, the function ψ has $2N$ moments equal to 0 and, what is more unusual, the function ϕ has $2N$-1 moments equal to 0. The two functions have a support of length $6N$-1.

The *coifN* ψ and ϕ are much more symmetrical than the *dbN*s. With respect to the support length, *coifN* has to be compared to *db3N* or *sym3N*. With respect to the number of vanishing moments of ψ, *coifN* has to be compared to *db2N* or *sym2N*.

If s is a sufficiently regular continuous time signal, for large j the coefficient $\langle s, \phi_{-j,k}\rangle$ is approximated by $2^{-j/2}s(2^{-j}k)$.

If s is a polynomial of degree d, $d \leq N$ - 1, then the approximation becomes an equality. This property is used, connected with sampling problems, when calculating the difference between an expansion over the $\phi_{j,k}$ of a given signal and its sampled version.

Biorthogonal Wavelet Pairs: *biorNr.Nd*

More about biorthogonal wavelets can be found in [Dau92] pages 259, 262–85 and in [Coh92]. By typing `waveinfo('bior')` at the MATLAB command prompt, you can obtain a survey of the main properties of this family, as well as information about Nr and Nd orders and associated filter lengths.

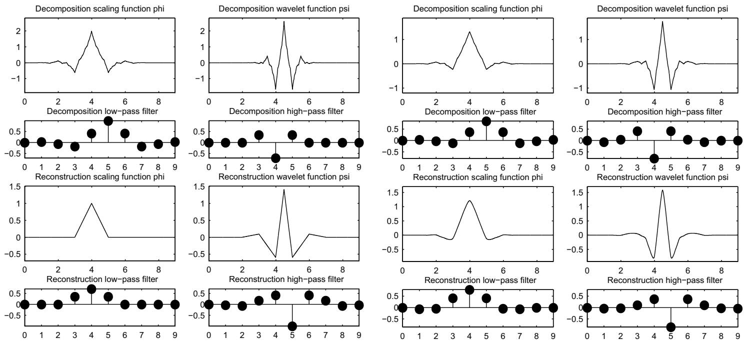


Figure 6-15: Biorthogonal Wavelets bior2.4 on the Left and bior4.4 on the Right

The new family extends the wavelet family. It is well known in the subband filtering community that symmetry and exact reconstruction are incompatible (except for the Haar wavelet) if the same FIR filters are used for reconstruction and decomposition. Two wavelets, instead of just one, are introduced:

- One, $\tilde{\psi}$, is used in the analysis, and the coefficients of a signal s are

  $$\tilde{c}_{j,k} = \int s(x)\tilde{\psi}_{j,k}(x)dx$$

- The other, ψ, is used in the synthesis

  $$s = \sum_{j,k} \tilde{c}_{j,k}\psi_{j,k}$$

In addition, the wavelets ψ and ψ are related by duality in the following sense:

$\int \tilde{\psi}_{j,k}(x)\psi_{j',k'}(x)dx = 0$ as soon as $j \neq j'$ or $k \neq k'$ and even

$\int \tilde{\phi}_{0,k}(x)\phi_{0,k'}(x)dx = 0$ as soon as $k \neq k'$

It becomes apparent, as Cohen pointed out in his thesis, that “the useful properties for analysis (e.g., oscillations, zero moments) can be concentrated on the $\tilde{\psi}$ function whereas the interesting properties for synthesis (regularity) are assigned to the ψ function. The separation of these two tasks proves very useful” (see [Coh92] page 110).

$\tilde{\psi}$, ψ can have very different regularity properties (see [Dau92] page 269).

The $\tilde{\psi}$, ψ, $\tilde{\phi}$, and ϕ functions are zero outside of a segment.

The calculation algorithms are maintained, and thus very simple.

The filters associated with m_0 and $\tilde{m}_0$ can be symmetrical. The functions used in the calculations are easier to build numerically than those used in the usual wavelets.

Meyer Wavelet: *meyr*

Both ψ and ϕ are defined in the frequency domain, starting with an auxiliary function ν (see [Dau92] pages 117, 119, 137, 152). By typing `waveinfo('meyr')` at the MATLAB command prompt, you can obtain a survey of the main properties of this wavelet.

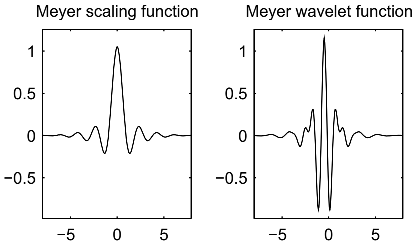


Figure 6-16: Meyer Wavelet

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

- Wavelet function

$$\hat{\psi}(\omega) = (2\pi)^{-1/2} e^{i\omega/2} \sin\left(\frac{\pi}{2}\nu\left(\frac{3}{2\pi}|\omega| - 1\right)\right) \qquad if \qquad \frac{2\pi}{3} \le |\omega| \le \frac{4\pi}{3}$$

$$\hat{\psi}(\omega) = (2\pi)^{-1/2} e^{i\omega/2} \cos\left(\frac{\pi}{2}\nu\left(\frac{3}{4\pi}|\omega| - 1\right)\right) \qquad if \qquad \frac{4\pi}{3} \le |\omega| \le \frac{8\pi}{3}$$

and $\hat{\psi}(\omega) = 0 \quad if \quad |\omega| \notin \left[\frac{2\pi}{3}, \frac{8\pi}{3}\right]$

where $\nu(a) = a^4(35 - 84a + 70a^2 - 20a^3) \qquad a \in [0,1]$

- Scaling function

$$\hat{\phi}(\omega) = (2\pi)^{-1/2} \qquad if \qquad |\omega| \le \frac{2\pi}{3}$$

$$\hat{\phi}(\omega) = (2\pi)^{-1/2} \cos\left(\frac{\pi}{2}\nu\left(\frac{3}{2\pi}|\omega| - 1\right)\right) \qquad if \qquad \frac{2\pi}{3} \le |\omega| \le \frac{4\pi}{3}$$

$$\hat{\phi}(\omega) = 0 \qquad if \qquad |\omega| > \frac{4\pi}{3}$$

By changing the auxiliary function, you get a family of different wavelets. For the required properties of the auxiliary function ν (see “References” on page 6-155 for more information). This wavelet ensures orthogonal analysis.

The function ψ does not have finite support, but ψ decreases to 0 when $x \to \infty$, faster than any inverse polynomial

$$\forall n \in N, \exists C_n \text{ such that } |\psi(x)| \le C_n(1 + |x|^2)^{-n}$$

This property holds also for the derivatives

$$\forall k \in N, \forall n \in N, \exists C_{k,n}, \text{ such that } \left|\psi^{(k)}x\right| \le C_{k,n}(1 + |x|^2)^{-n}$$

The wavelet is infinitely differentiable.

Note Although the Meyer wavelet is not compactly supported, there exists a good approximation leading to FIR filters, and then allowing DWT. By typing `waveinfo('dmey')` at the MATLAB command prompt, you can obtain a survey of the main properties of this pseudo-wavelet.

Battle-Lemarie Wavelets

See [Dau92] pages 146–148, 151.

These wavelets are not included in the toolbox, but we use the spline functions in the biorthogonal family.

There are two forms of the wavelet: one does not ensure the analysis to be orthogonal, while the other does. For *N*=1, the scaling functions are linear splines. For *N*=2, the scaling functions are quadratic B-spline with finite support. More generally, for an *N*-degree B-spline,

$$\hat{\phi}(\omega) = (2\pi)^{-1/2} e^{-i\kappa\omega/2} \left[\frac{\sin(\omega/2)}{\omega/2}\right]^{N+1}$$

with $\kappa = 0$ if *N* is odd, $\kappa = 1$ if *N* is even.

This formula can be used to build the filters. The twin scale relation is

$$\phi(x) = 2^{-2M}\sum_{j=0}^{2M+1} C_j^{2M+1}\phi(2x-M-1+j) \quad \text{if} \quad N = 2M$$

$$\phi(x) = 2^{-2M-1}\sum_{j=0}^{2M+2} C_j^{2M+2}\phi(2x-M-1+j) \text{ if } N = 2M+1$$

- For an even N, ϕ is symmetrical around, $x = 1/2$; ψ is antisymmetrical around $x = 1/2$. For an odd N, ϕ is symmetrical around $x = 0$; ψ is symmetrical around $x = 1/2$.
- The analysis becomes orthogonal if we transform the functions ψ and ϕ somewhat. For N=1, for instance, let

$$\widehat{\phi^{\perp}}(\omega) = 3^{1/2}(2\pi)^{-1/2}\frac{4\sin^2(\omega/2)}{\omega^2[1+2\cos^2(\omega/2)]^{1/2}}$$

- The supports of ψ and $\phi^{\perp}$ are not finite, but the decrease of the functions ψ and $\phi^{\perp}$ to 0 is exponential. The support of ϕ is compact. See [Dau92] p. 151.
- The ψ functions have derivatives up to order N-1.

Mexican Hat Wavelet: *mexh*

See [Dau92] page 75.

By typing `waveinfo('mexh')` at the MATLAB command prompt, you can obtain a survey of the main properties of this wavelet.

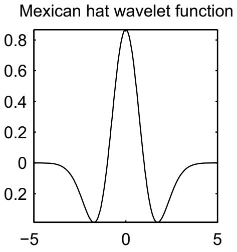


Figure 6-17: Mexican Hat

$$\psi(x) = \left(\frac{2}{\sqrt{3}}\pi^{-1/4}\right)(1-x^2)e^{-x^2/2}$$

This function is proportional to the second derivative function of the Gaussian probability density function.

As the ϕ function does not exist, the analysis is not orthogonal.

Morlet Wavelet: *morl*

See [Dau92] page 76.

By typing `waveinfo('morl')` at the MATLAB command prompt you can obtain a survey of the main properties of this wavelet.

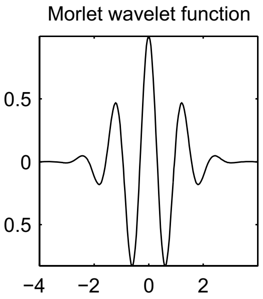


Figure 6-18: Morlet Wavelet

$$\psi(x) = Ce^{-x^2/2}\cos(5x)$$

The constant C is used for normalization in view of reconstruction.

The Morlet wavelet does not satisfy exactly the admissibility condition discussed in “What Functions Are Candidates to Be a Wavelet?” on page 6-63.

Additional Real Wavelets

Some other real wavelets are available in the toolbox.

Reverse Biorthogonal Wavelet Pairs: *rbioNr.Nd*

This family is obtained from the biorthogonal wavelet pairs previously described.

You can obtain a survey of the main properties of this family by typing `waveinfo('rbio')` from the MATLAB command line.

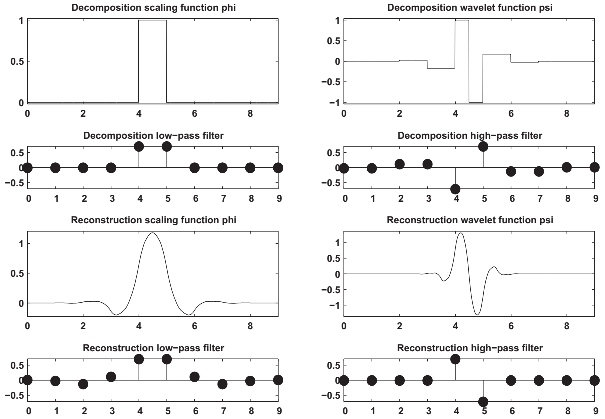


Figure 6-19: Reverse Biorthogonal Wavelet rbio1.5

Gaussian Derivatives Family: *gaus*

This family is built starting from the Gaussian function $f(x) = C_p e^{-x^2}$ by taking the p^{th} derivative of f.

The integer p is the parameter of this family and in the previous formula, C_p is such that

$$\|f^{(p)}\|^2 = 1 \text{ where } f^{(p)} \text{ is the } p^{th} \text{ derivative of } f.$$

You can obtain a survey of the main properties of this family by typing `waveinfo('gaus')` from the MATLAB command line.

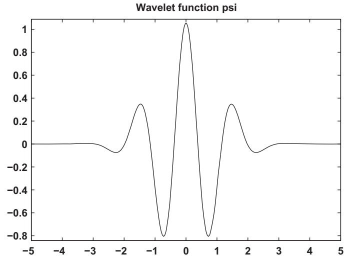


Figure 6-20: Gaussian Derivative Wavelet gaus8

FIR Based Approximation of the Meyer Wavelet: *dmey*

See [Abr97] page 268.

This wavelet is a FIR based approximation of the Meyer wavelet, allowing fast wavelet coefficients calculation using DWT.

You can obtain a survey of the main properties of this wavelet by typing `waveinfo('dmey')` from the MATLAB command line.

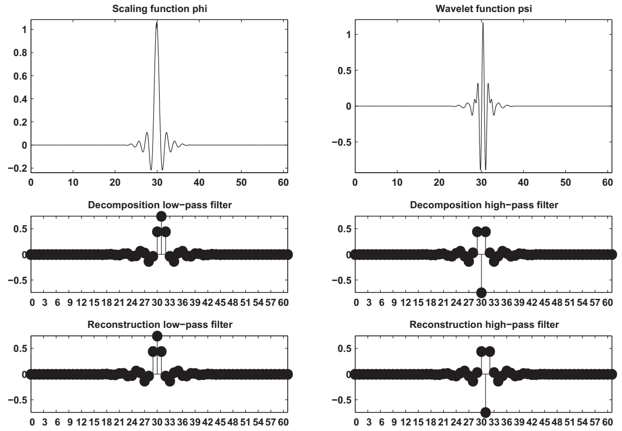


Figure 6-21: FIR Based Approximation of the Meyer Wavelet

Complex Wavelets

Some complex wavelet families are available in the toolbox.

Complex Gaussian Wavelets: *cgau*

This family is built starting from the complex Gaussian function

$f(x) = C_p e^{-ix} e^{-x^2}$ by taking the p^{th} derivative of f. The integer p is the parameter of this family and in the previous formula, C_p is such that $\|f^{(p)}\|^2 = 1$ where $f^{(p)}$ is the p^{th} derivative of f.

You can obtain a survey of the main properties of this family by typing `waveinfo('cgau')` from the MATLAB command line.

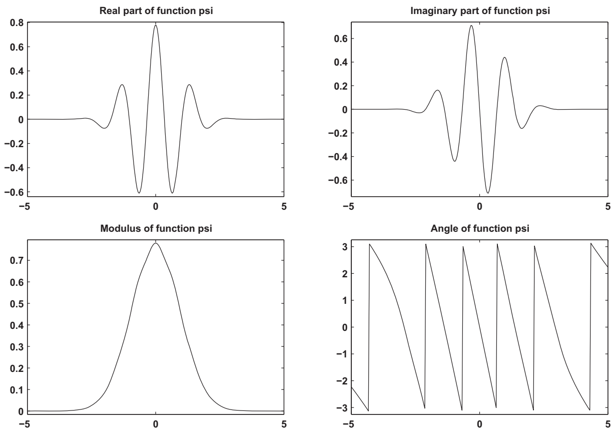


Figure 6-22: Complex Gaussian Wavelet cgau8

Complex Morlet Wavelets: *cmor*

See [Teo98] pages 62–65.

A complex Morlet wavelet is defined by

$$\psi(x) = \frac{1}{\sqrt{\pi f_b}} e^{2i\pi f_c x} e^{-\frac{x^2}{f_b}}$$

depending on two parameters:

- f_b is a bandwidth parameter.
- f_c is a wavelet center frequency.

You can obtain a survey of the main properties of this family by typing `waveinfo('cmor')` from the MATLAB command line.

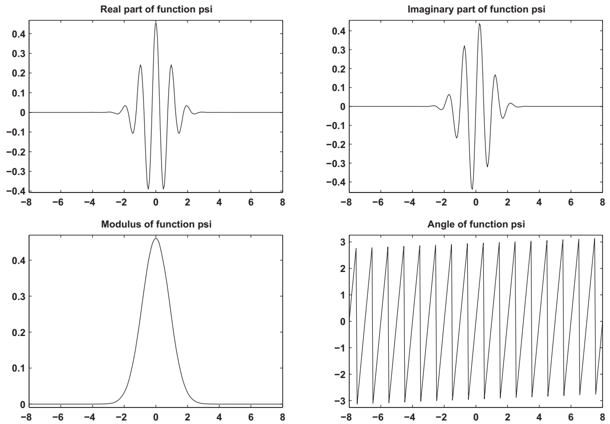


Figure 6-23: Complex Morlet Wavelet morl 1.5-1

Complex Frequency B-Spline Wavelets: *fbsp*

See [Teo98] pages 62–65.

A complex frequency B-spline wavelet is defined by

$$\psi(x) = \sqrt{f_b}\left(\mathrm{sinc}\left(\frac{f_b x}{m}\right)\right)^m e^{2i\pi f_c x}$$

depending on three parameters:

- m is an integer order parameter ($m \geq 1$).
- f_b is a bandwidth parameter.
- f_c is a wavelet center frequency.

You can obtain a survey of the main properties of this family by typing `waveinfo('fbsp')` from the MATLAB command line.

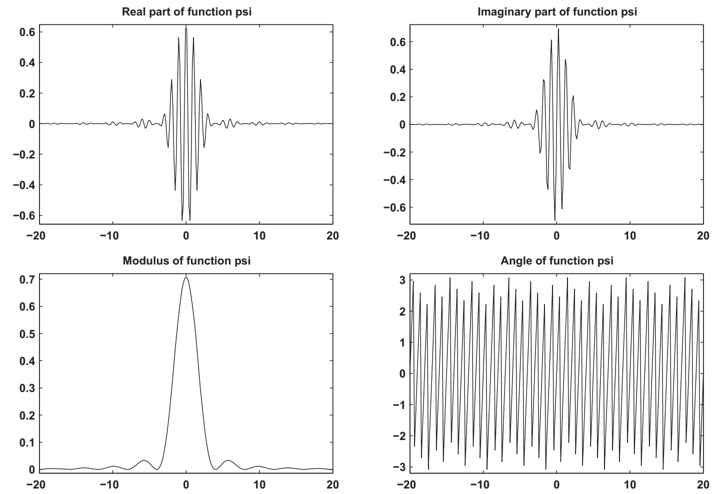


Figure 6-24: Complex Frequency B-Spline Wavelet fbsp 2-0.5-1

Complex Shannon Wavelets: *shan*

See [Teo98] pages 62–65.

This family is obtained from the frequency B-spline wavelets by setting m to 1.

A complex Shannon wavelet is defined by

$$\psi(x) = \sqrt{f_b}\,\mathrm{sinc}(f_b x)\, e^{2i\pi f_c x}$$

depending on two parameters:

- f_b is a bandwidth parameter.
- f_c is a wavelet center frequency.

You can obtain a survey of the main properties of this family by typing `waveinfo('shan')` from the MATLAB command line.

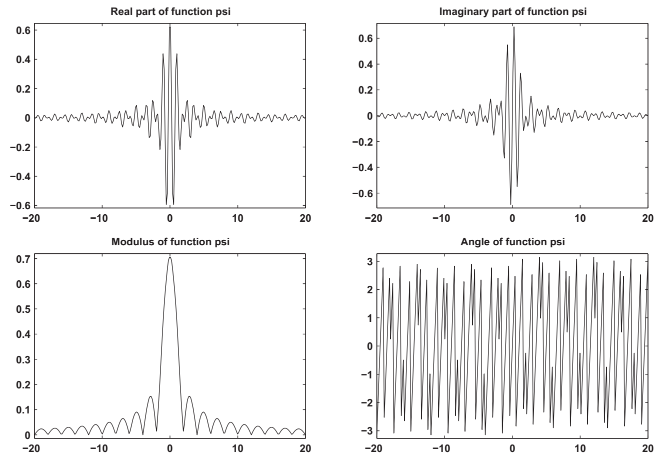


Figure 6-25: Complex Shannon Wavelet shan 0.5-1

Summary of Wavelet Families and Associated Properties (Part 1)

Property	morl	mexh	meyr	haar	*db*N	symN	coifN	biorNr.Nd
Crude	•	•						
Infinitely regular	•	•	•					
Arbitrary regularity					•	•	•	•
Compactly supported orthogonal				•	•	•	•	
Compactly supported biothogonal								•
Symmetry	•	•	•	•				•
Asymmetry					•			
Near symmetry						•	•	
Arbitrary number of vanishing moments					•	•	•	•
Vanishing moments for ϕ							•	
Existence of ϕ			•	•	•	•	•	•
Orthogonal analysis			•	•	•	•	•	
Biorthogonal analysis			•	•	•	•	•	•
Exact reconstruction	≈	•	•	•	•	•	•	•
FIR filters				•	•	•	•	•
Continuous transform	•	•	•	•	•	•	•	•
Discrete transform				•	•	•	•	•
Fast algorithm				•	•	•	•	•
Explicit expression	•	•		•				For splines

Crude wavelet — A wavelet is said to be crude when satisfying only the admissibility condition. See “What Functions Are Candidates to Be a Wavelet?” on page 6-63.

Regularity — See “What About the Regularity of a Wavelet y?” on page 6-62.

Orthogonal — See “Details and Approximations” on page 6-15.

Biorthogonal — See “Biorthogonal Wavelet Pairs: biorNr.Nd” on page 6-76.

Vanishing moments — See “Suppressing Signals” on page 6-92.

Exact reconstruction — See “Reconstruction Filters” on page 1-30.

Continuous — See “Continuous Wavelet Transform” on page 1-15.

Discrete — See “Discrete Wavelet Transform” on page 1-24.

FIR filters — See “Filters Used to Calculate the DWT and IDWT” on page 6-19.

Summary of Wavelet Families and Associated Properties (Part 2)

Property	rbioNr.Nd	gaus	dmey	cgau	cmor	fbsp	shan
Crude		•		•	•	•	•
Infinitely regular		•		•	•	•	•
Arbitrary regularity	•						
Compactly supported orthogonal							
Compactly supported biothogonal	•						
Symmetry	•	•	•	•	•	•	•
Asymmetry							
Near symmetry							
Arbitrary number of vanishing moments	•						
Vanishing moments for ϕ							
Existence of ϕ	•						
Orthogonal analysis							
Biorthogonal analysis	•						
Exact reconstruction	•	•	≈	•	•	•	•
FIR filters	•		•				
Continuous transform	•	•					
Discrete transform	•		•				
Fast algorithm	•		•				
Explicit expression	For splines	•		•	•	•	•
Complex valued				•	•	•	•
Complex continuous transform				•	•	•	•
FIR-based approximation			•				

Crude wavelet —A wavelet is said to be crude when satisfying only the admissibility condition. See “What Functions Are Candidates to Be a Wavelet?” on page 6-63.

Regularity — See “What About the Regularity of a Wavelet y?” on page 6-62.

Orthogonal — See “Details and Approximations” on page 6-15.

Biorthogonal — See “Biorthogonal Wavelet Pairs: biorNr.Nd” on page 6-76.

Vanishing moments — See “Suppressing Signals” on page 6-92.

Exact reconstruction — See “Reconstruction Filters” on page 1-30.

Continuous — See “Continuous Wavelet Transform” on page 1-15.

Discrete — See “Discrete Wavelet Transform” on page 1-24.

FIR filters — See “Filters Used to Calculate the DWT and IDWT” on page 6-19.

Wavelet Applications: More Detail

Chapter 3, "Wavelet Applications," and Chapter 4, "Wavelets in Action: Examples and Case Studies," illustrate wavelet applications with examples and case studies. This section reexamines some of the applications with additional theory and more detail.

Suppressing Signals

As shown in "Suppressing Signals" on page 3-15, by suppressing a part of a signal the remainder may be highlighted.

Let ψ be a wavelet with at least k+1 vanishing moments:

$$\text{for } j = 0, \ldots, k, \int_R x^j \psi(x) dx = 0$$

If the signal s is a polynomial of degree k, then the coefficients $C(a,b) = 0$ for all a and all b. Such wavelets automatically suppress the polynomials. The degree of s can vary with time x, provided that it remains less than k.

If s is now a polynomial of degree k on segment $[\alpha,\beta]$, then $C(a,b) = 0$ as long as the support of the function $\frac{1}{\sqrt{a}}\psi\left(\frac{x-b}{a}\right)$ is included in $[\alpha,\beta]$. The suppression is local. Effects will appear on the edges of the segment.

Likewise, let us suppose that, on $[\alpha,\beta]$ to which 0 belongs, we have the expansion $s(x) = [s(0) + xs'(0) + x^2 s^{(2)}(0) + \ldots + x^k s^{(k)}(0)] + g(x)$. The s and g signals then have the same wavelet coefficients. This is the technical meaning of the phrase "The wavelet suppresses a polynomial part of signal s." The signal g is the "irregular" part of the signal s. The ψ wavelet systematically suppresses the regular part and analyzes the irregular part. This effect is easily seen in details D_1 through D_4 in "Example 2: A Frequency Breakdown" in Chapter 4 (see the curves **d1**, **d2**, **d3**, and **d4**). The wavelet suppresses the slow sine wave, which is locally assimilated to a polynomial.

Another way of suppressing a component of the signal is to modify and force certain coefficients $C(a,b)$ to be equal to 0. Having selected a set E of indices, we stipulate that $\forall(a,b) \in E$, $C(a,b) = 0$. We then synthesize the signal using the modified coefficients.

Let us illustrate, with the following M-file, some features of wavelet processing using coefficients (resulting plots can be found in Figure 6-26 on page 6-94).

```
% Load original 1-D signal.
load sumsin; s = sumsin;

% Set the wavelet name and perform the decomposition
% of s at level 4, using coif3.
w = 'coif3'; maxlev = 4;
[c,l] = wavedec(s,maxlev,w);
newc = c;

% Force to zero the detail coefficients at levels 3 and 4.
newc = wthcoef('d',c,l,[3,4]);

% Force the detail coefficients at level 1 to zero on
% original time interval [400:600] and shrink otherwise.
% determine first and last index of
% level 1 coefficients.
k = maxlev+1;
first = sum(l(1:k-1))+1; last = first+l(k)-1;
indd1 = first:last;

% shrink by dividing by 3.
newc(indd1) = c(indd1)/3;

% find at level 1 indices of coefficients
% in the interval [400:600],
% note that time t in original grid corresponds to time
% t/2^k on the grid at level k. Here k=1.
indd1 = first+400/2:first+600/2;

% force it to zero.
newc(indd1) = zeros(size(indd1));

% Set to 4 a coefficient at level 2 corresponding roughly
% to original time t = 500.
k = maxlev; first = sum(l(1:k-1))+1;
newc(first+500/2^2) = 4;
% Synthesize modified decomposition structure.
synth = waverec(newc,l,w);
```

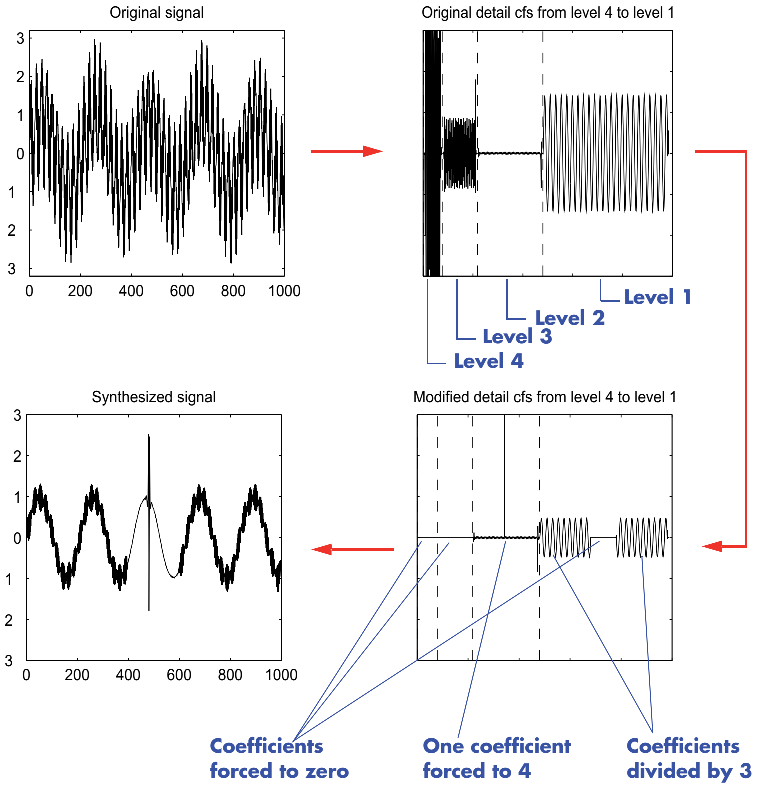


Figure 6-26: Suppress or Modify Signal Components, Acting on Coefficients

Simple procedures to select the set of indices E are used for de-noising and compression purposes (see “De-Noising” on page 6-97 and “Data Compression” on page 6-110).

Splitting Signal Components

Wavelet analysis is a linear technique: the wavelet coefficients of the linear combination of two signals $\alpha s^{(1)} + \beta s^{(2)}$ are equal to the linear combination of their wavelet coefficients $\alpha C_{j,k}^{(1)} + \beta C_{j,k}^{(2)}$. The same holds true for the corresponding approximations and details, for example $\alpha A_j^{(1)} + \beta A_j^{(2)}$ and $\alpha D_j^{(1)} + \beta D_j^{(2)}$.

Noise Processing

Let us first analyze noise as an ordinary signal. Then the probability characteristics correlation function, spectrum, and distribution need to be studied.

In general, for a one-dimensional discrete-time signal, the high frequencies influence the details of the first levels (the small values of j), while the low frequencies influence the deepest levels (the large values of j) and the associated approximations.

If a signal comprising only white noise is analyzed (for example, see “Example 3: Uniform White Noise” in Chapter 4), the details at the various levels decrease in amplitude as the level increases. The variance of the details also decreases as the level increases. The details and approximations are not white noise anymore, as color is introduced by the filters.

On the coefficients $C(j,k)$, where j stands for the scale and k for the time, we can add often-satisfied properties for discrete time signals:

- If the analyzed signal s is stationary, zero mean, and a white noise, the coefficients are uncorrelated.
- If furthermore s is Gaussian, the coefficients are independent and Gaussian.
- If s is a colored, stationary, zero mean Gaussian sequence, then the coefficients remain Gaussian. For each scale level j, the sequence of coefficients is a colored stationary sequence. It could be interesting to know how to choose the wavelet that would de-correlate the coefficients. This problem has not yet been resolved. Furthermore, the wavelet (if indeed it exists) most probably depends on the color of the signal. For the wavelet to be calculated, the color must be known. In most instances, this is beyond our reach.

- If s is a zero mean ARMA model stationary for each scale j, then $C(j,k)$, $k \in Z$ is also a stationary, zero mean ARMA process whose characteristics depend on j.
- If s is a noise whose
  - Correlation function ρ is known, we know how to calculate the correlations of $C(j,k)$ and $C(j,k')$.
  - Spectrum $\hat{\rho}$ is known, we know how to calculate the spectrum of $C(j,k)$, $k \in Z$ and the cross spectrum of two different levels j and j'.

These results are easily established, since they can be deduced from the fact that the $C(a,b)$ coefficients are calculated primarily by convolving ψ and s, and using conventional formulas. The quantity that comes into play is the self-reproduction function $U(a,b)$, which is obtained by analyzing the ψ wavelet as if it was a signal:

$$U(a,b) = \int_R \frac{1}{\sqrt{a}}\psi\left(\frac{x-b}{a}\right)\psi(x)dx$$

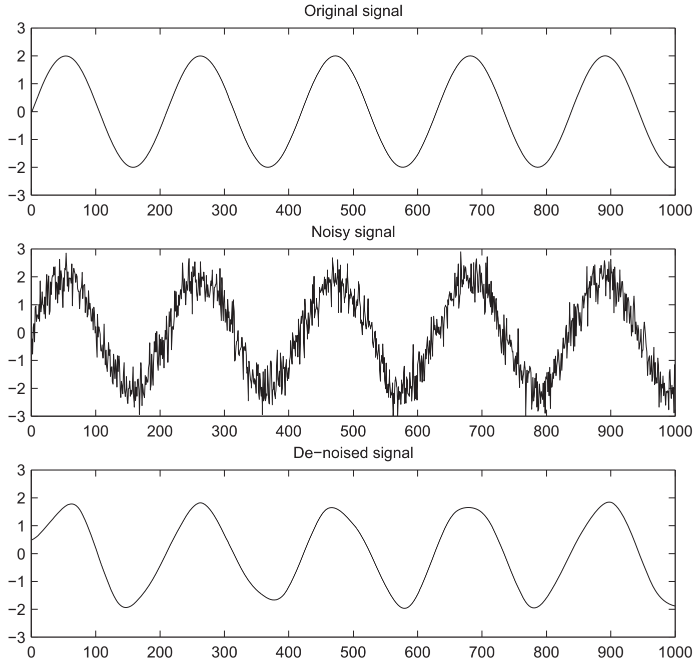
From the results for coefficients we deduce the properties of the details (and of the approximations), by using the formula

$$D_j(n) = \sum_{k \in Z} C(j,k)\psi_{j,k}(n)$$

where the $C(j,k)$ coefficients are random variables and the functions $\psi_{j,k}$ are not. If the support of ψ is finite, only a finite number of terms will be summed.

De-Noising

This section discusses the problem of signal recovery from noisy data. This problem is easy to understand looking at the following simple example, where a slow sine is corrupted by a white noise.


Original signal

Noisy signal

De−noised signal

Figure 6-27: Simple De-Noising Example

Basic One-Dimensional Model

The underlying model for the noisy signal is basically of the following form:

$$s(n) = f(n) + \sigma e(n)$$

where time n is equally spaced.

In the simplest model we suppose that *e(n)* is a Gaussian white noise $N(0,1)$ and the noise level σ is supposed to be equal to 1.

The de-noising objective is to suppress the noise part of the signal *s* and to recover *f*.

The method is efficient for families of functions *f* that have only a few nonzero wavelet coefficients. These functions have a sparse wavelet representation. For example, a smooth function almost everywhere, with only a few abrupt changes, has such a property.

From a statistical viewpoint, the model is a regression model over time and the method can be viewed as a nonparametric estimation of the function *f* using orthogonal basis.

De-Noising Procedure Principles

The general de-noising procedure involves three steps. The basic version of the procedure follows these steps:

1. Decompose

   Choose a wavelet, choose a level *N*. Compute the wavelet decomposition of the signal *s* at level *N*.

2. Threshold detail coefficients

   For each level from 1 to *N*, select a threshold and apply soft thresholding to the detail coefficients.

3. Reconstruct

   Compute wavelet reconstruction using the original approximation coefficients of level *N* and the modified detail coefficients of levels from 1 to *N*.

Two points must be addressed: how to choose the threshold, and how to perform the thresholding.

Soft or Hard Thresholding?

Thresholding can be done using the function

```
yt = wthresh(y,sorh,thr)
```

which returns soft or hard thresholding of input `y`, depending on the `sorh` option. Hard thresholding is the simplest method. Soft thresholding has nice mathematical properties and the corresponding theoretical results are available (For instance, see [Don95] in “References” on page 6-155).

Let us give a simple example.

```
y = linspace(-1,1,100);
thr = 0.4;
ythard = wthresh(y,'h',thr);
ytsoft = wthresh(y,'s',thr);
```

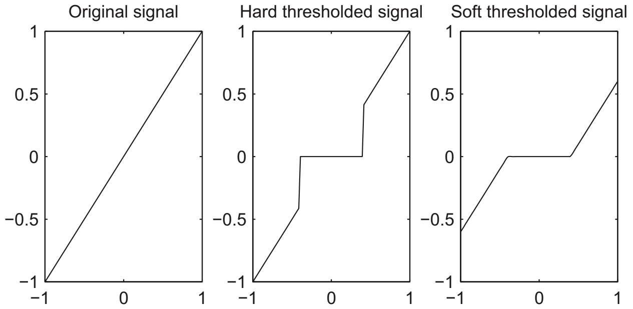


Figure 6-28: Hard and Soft Thresholding of the Signal *s* = *x*

Comment: Let *t* denote the threshold. The hard threshold signal is *x* if $|x| > t$, and is 0 if $|x| \leq t$. The soft threshold signal is sign $(x)(|x| - t)$ if $|x| > t$ and is 0 if $|x| \leq t$.

Hard thresholding can be described as the usual process of setting to zero the elements whose absolute values are lower than the threshold. Soft thresholding is an extension of hard thresholding, first setting to zero the elements whose absolute values are lower than the threshold, and then shrinking the nonzero coefficients toward 0 (see Figure 6-28).

As can be seen in the comment of Figure 6-28 on page 6-99, the hard procedure creates discontinuities at $x = \pm t$, while the soft procedure does not.

Threshold Selection Rules

According to the basic noise model, four threshold selection rules are implemented in the M-file `thselect`. Each rule corresponds to a `tptr` option in the command

```
thr = thselect(y,tptr)
```

which returns the threshold value.

Option	Threshold Selection Rule
'rigrsure'	Selection using principle of Stein's Unbiased Risk Estimate (SURE)
'sqtwolog'	Fixed form threshold equal to *sqrt*(2**log*(*length*(*s*)))
'heursure'	Selection using a mixture of the first two options
'minimaxi'	Selection using minimax principle

- Option `tptr = 'rigrsure'` uses for the soft threshold estimator a threshold selection rule based on Stein's Unbiased Estimate of Risk (quadratic loss function). You get an estimate of the risk for a particular threshold value *t*. Minimizing the risks in *t* gives a selection of the threshold value.
- Option `tptr = 'sqtwolog'` uses a fixed form threshold yielding minimax performance multiplied by a small factor proportional to *log*(*length*(*s*)).
- Option `tptr = 'heursure'` is a mixture of the two previous options. As a result, if the signal-to-noise ratio is very small, the SURE estimate is very noisy. So if such a situation is detected, the fixed form threshold is used.
- Option `tptr = 'minimaxi'` uses a fixed threshold chosen to yield minimax performance for mean square error against an ideal procedure. The minimax principle is used in statistics to design estimators. Since the de-noised signal can be assimilated to the estimator of the unknown regression function, the minimax estimator is the option that realizes the minimum, over a given set of functions, of the maximum mean square error.

Typically it is interesting to show how `thselect` works if `y` is a Gaussian white noise *N*(0,1) signal.

```
y = randn(1,1000);

thr = thselect(y,'rigrsure')
thr =
    2.0735

thr = thselect(y,'sqtwolog')
thr =
    3.7169

thr = thselect(y,'heursure')
thr =
    3.7169

thr = thselect(y,'minimaxi')
thr =
    2.2163
```

Because `y` is a standard Gaussian white noise, we expect that each method kills roughly all the coefficients and returns the result *f(x)* = 0. For Stein's Unbiased Risk Estimate and minimax thresholds, roughly 3% of coefficients are saved. For other selection rules, all the coefficients are set to 0.

We know that the detail coefficients vector is the superposition of the coefficients of *f* and the coefficients of *e*, and that the decomposition of *e* leads to detail coefficients, which are standard Gaussian white noises.

So minimax and SURE threshold selection rules are more conservative and would be more convenient when small details of function *f* lie near the noise range. The two other rules remove the noise more efficiently. The option `'heursure'` is a compromise. In this example, the fixed form threshold wins.

Recalling step 2 of the de-noise procedure, the function `thselect` performs a threshold selection, and then each level is thresholded. This second step can be done using `wthcoef`, directly handling the wavelet decomposition structure of the original signal *s*.

Dealing with Unscaled Noise and Nonwhite Noise

Usually in practice the basic model cannot be used directly. We examine here the options available to deal with model deviations in the main de-noising function `wden`.

The simplest use of `wden` is

```
sd = wden(s,tptr,sorh,scal,n,wav)
```

which returns the de-noised version `sd` of the original signal `s` obtained using the `tptr` threshold selection rule. Other parameters needed are `sorh`, `scal`, `n`, and `wav`. The parameter `sorh` specifies the thresholding of details coefficients of the decomposition at level `n` of `s` by the wavelet called `wav`. The remaining parameter `scal` is to be specified. It corresponds to threshold's rescaling methods.

Option	Corresponding Model
`'one'`	Basic model
`'sln'`	Basic model with unscaled noise
`'mln'`	Basic model with nonwhite noise

- Option `scal = 'one'` corresponds to the basic model.
- In general, you can ignore the noise level and it must be estimated. The detail coefficients cD_1 (the finest scale) are essentially noise coefficients with standard deviation equal to σ. The median absolute deviation of the coefficients is a robust estimate of σ. The use of a robust estimate is crucial for two reasons. The first one is that if level 1 coefficients contain *f* details, then these details are concentrated in a few coefficients if the function f is sufficiently regular. The second reason is to avoid signal end effects, which are pure artifacts due to computations on the edges.

  Option `scal = 'sln'` handles threshold rescaling using a single estimation of level noise based on the first-level coefficients.

- When you suspect a nonwhite noise *e*, thresholds must be rescaled by a level-dependent estimation of the level noise. The same kind of strategy as in the previous option is used by estimating σ_{lev} level by level.

  This estimation is implemented in M-file `wnoisest`, directly handling the wavelet decomposition structure of the original signal *s*.

  Option `scal = 'mln'` handles threshold rescaling using a level-dependent estimation of the level noise.

For a more general procedure, the `wdencmp` function performs wavelet coefficients thresholding for both de-noising and compression purposes, while directly handling one-dimensional and two-dimensional data. It allows you to define your own thresholding strategy selecting in

```
xd = wdencmp(opt,x,wav,n,thr,sorh,keepapp);
```

where

- `opt = 'gbl'` and `thr` is a positive real number for uniform threshold.
- `opt = 'lvd'` and `thr` is a vector for level dependent threshold.
- `keepapp = 1` to keep approximation coefficients, as previously and
- `keepapp = 0` to allow approximation coefficients thresholding.
- `x` is the signal to be de-noised and `wav`, `n`, `sorh` are the same as above.

De-Noising in Action

We begin with examples of one-dimensional de-noising methods with the first example credited to Donoho and Johnstone. You can use the following M-file to get the first test function using `wnoise`.

```
% Set signal to noise ratio and set rand seed.
sqrt_snr = 4; init = 2055615866;

% Generate original signal xref and a noisy version x adding
% a standard Gaussian white noise.
[xref,x] = wnoise(1,11,sqrt_snr,init);

% De-noise noisy signal using soft heuristic SURE thresholding
% and scaled noise option, on detail coefficients obtained
% from the decomposition of x, at level 3 by sym8 wavelet.
xd = wden(x,'heursure','s','one',3,'sym8');
```

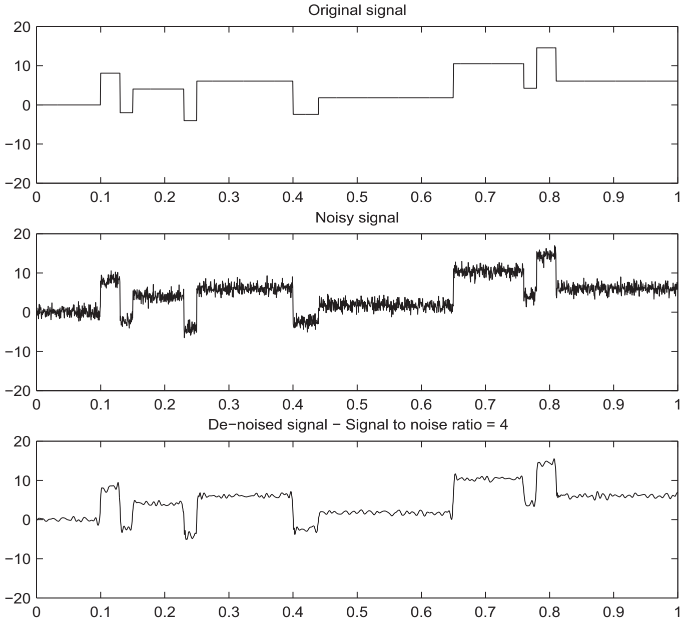


Figure 6-29: Blocks Signal De-Noising

Since only a small number of large coefficients characterize the original signal, the method performs very well (see Figure 6-29). If you want to see more about how the thresholding works, use the GUI (see “De-Noising Signals” on page 3-18).

As a second example, let us try the method on the highly perturbed part of the electrical signal studied above.

According to this previous analysis, let us use `db3` wavelet and decompose at level 3.

To deal with the composite noise nature, let us try a level-dependent noise size estimation.

```
% Load electrical signal and select part of it.
load leleccum; indx = 2000:3450;
x = leleccum(indx);

% Find first value in order to avoid edge effects.
deb = x(1);

% De-noise signal using soft fixed form thresholding
% and unknown noise option.
xd = wden(x-deb,'sqtwolog','s','mln',3,'db3')+deb;
```

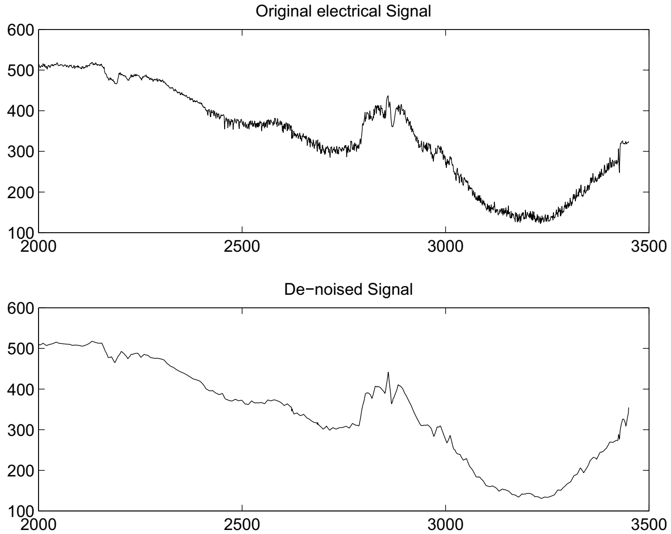


Figure 6-30: Electrical Signal De-Noising

The result is quite good in spite of the time heterogeneity of the nature of the noise after and before the beginning of the sensor failure around time 2450.

Extension to Image De-Noising

The de-noising method described for the one-dimensional case applies also to images and applies well to geometrical images. A direct translation of the one-dimensional model is

$$s(i,j) = f(i,j) + \sigma e(i,j)$$

where e is a white Gaussian noise with unit variance.

The two-dimensional de-noising procedure has the same three steps and uses two-dimensional wavelet tools instead of one-dimensional ones. For the threshold selection, `prod(size(s))` is used instead of `length(s)` if the fixed form threshold is used.

Note that except for the “automatic” one-dimensional de-noising case, de-noising and compression are performed using `wdencmp`. As an example, you can use the following M-file illustrating the de-noising of a real image.

```
% Load original image.
load  woman

% Generate noisy image.
init = 2055615866; randn('seed',init);
x = X + 15*randn(size(X));

% Find default values. In this case fixed form threshold
% is used with estimation of level noise, thresholding
% mode is soft and the approximation coefficients are
% kept.
[thr,sorh,keepapp] = ddencmp('den','wv',x);

% thr is equal to estimated_sigma*sqrt(log(prod(size(X))))
thr

thr =

  107.6428

% De-noise image using global thresholding option.
xd = wdencmp('gbl',x,'sym4',2,thr,sorh,keepapp);
% Plots.
colormap(pink(255)), sm = size(map,1);
subplot(221), image(wcodemat(X,sm)), title('Original Image')
subplot(222), image(wcodemat(x,sm)), title('Noisy Image')
subplot(223), image(wcodemat(xd,sm)), title('De-Noised Image')
```

The result shown below is acceptable.

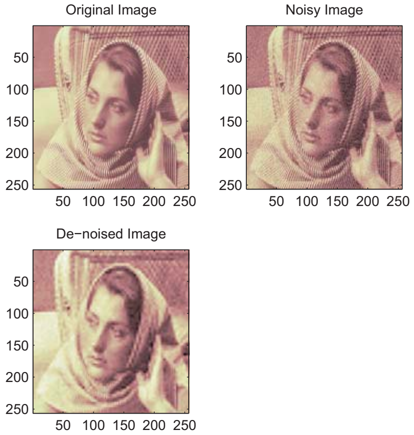


Figure 6-31: Image De-Noising

One-Dimensional Variance Adaptive Thresholding of Wavelet Coefficients

Local thresholding of wavelet coefficients, for one- or two-dimensional data, is a capability available from a lot of graphical interface tools throughout Wavelet Toolbox™ software (see “Using Wavelets” on page 2-1).

The idea is to define level by level time-dependent thresholds, and then increase the capability of the de-noising strategies to handle nonstationary variance noise models.

More precisely, the model assumes (as previously) that the observation is equal to the interesting signal superimposed on a noise (see “De-Noising” on page 6-97).

$$s(n) = f(n) + \sigma e(n)$$

But the noise variance can vary with time. There are several different variance values on several time intervals. The values as well as the intervals are unknown.

Let us focus on the problem of estimating the change points or equivalently the intervals. The algorithm used is based on an original work of Marc Lavielle about detection of change points using dynamic programming (see [Lav99] in “References” on page 6-155).

Let us generate a signal from a fixed-design regression model with two noise variance change points located at positions 200 and 600.

```
% Generate blocks test signal.
x = wnoise(1,10);

% Generate noisy blocks with change points.
init = 2055615866; randn('seed',init);
bb = randn(1,length(x));
cp1 = 200; cp2 = 600;
x = x + [bb(1:cp1),bb(cp1+1:cp2)/3,bb(cp2+1:end)];
```

The aim of this example is to recover the two change points from the signal x. In addition, this example illustrates how the GUI tools (see “Using Wavelets” on page 2-1) locate the change points for interval dependent thresholding.

Step 1. Recover a noisy signal by suppressing an approximation.

```
% Perform a single-level wavelet decomposition
% of the signal using db3.
wname = 'db3'; lev = 1;
[c,l] = wavedec(x,lev,wname);

% Reconstruct detail at level 1.
det = wrcoef('d',c,l,wname,1);
```

The reconstructed detail at level 1 recovered at this stage is almost signal free. It captures the main features of the noise from a change points detection viewpoint if the interesting part of the signal has a sparse wavelet representation. To remove almost all the signal, we replace the biggest values by the mean.

Step 2. To remove almost all the signal, replace 2% of biggest values by the mean.

```
x = sort(abs(det));
v2p100 = x(fix(length(x)*0.98));
ind = find(abs(det)>v2p100);
det(ind) = mean(det);
```

Step 3. Use the wvarchg function to estimate the change points with the following parameters:

- The minimum delay between two change points is d = 10.
- The maximum number of change points is 5.

```
[cp_est,kopt,t_est] = wvarchg(det,5)
cp_est =
    199   601

kopt =
    2

t_est =
      1024         0         0         0         0         0
       601      1024         0         0         0         0
       199       601      1024         0         0         0
       199       261       601      1024         0         0
       207       235       261       601      1024         0
       207       235       261       393       601      1024
```

Two change points and three intervals are proposed. Since the three interval variances for the noise are very different the optimization program detects easily the correct structure.

The estimated change points are close to the true change points: 200 and 600.

Step 4. (Optional) Replace the estimated change points.

For 2 ≤ i ≤ 6, t_est(i,1:i-1) contains the i-1 instants of the variance change points, and since kopt is the proposed number of change points; then

```
cp_est = t_est(kopt+1,1:kopt);
```

You can replace the estimated change points by computing

```
% cp_New = t_est(knew+1,1:knew); % where 1 ≤ knew ≤ 5
```

More About De-Noising

The de-noising methods based on wavelet decomposition appear mainly initiated by Donoho and Johnstone in the USA, and Kerkyacharian and Picard in France. Meyer considers that this topic is one of the most significant applications of wavelets (cf. [Mey93] page 173). This chapter and the corresponding M-files follow the work of the above mentioned researchers. More details can be found in Donoho's references in "References" on page 6-155 and in "More About the Thresholding Strategies" on page 6-125.

Data Compression

The compression features of a given wavelet basis are primarily linked to the relative scarceness of the wavelet domain representation for the signal. The notion behind compression is based on the concept that the regular signal component can be accurately approximated using the following elements: a small number of approximation coefficients (at a suitably chosen level) and some of the detail coefficients.

Like de-noising, the compression procedure contains three steps:

1 Decompose

Choose a wavelet, choose a level *N*. Compute the wavelet decomposition of the signal *s* at level *N*.

2 Threshold detail coefficients

For each level from 1 to *N*, a threshold is selected and hard thresholding is applied to the detail coefficients.

3 Reconstruct

Compute wavelet reconstruction using the original approximation coefficients of level *N* and the modified detail coefficients of levels from 1 to *N*.

The difference of the de-noising procedure is found in step **2**. There are two compression approaches available. The first consists of taking the wavelet expansion of the signal and keeping the largest absolute value coefficients. In this case, you can set a global threshold, a compression performance, or a relative square norm recovery performance.

Thus, only a single parameter needs to be selected. The second approach consists of applying visually determined level-dependent thresholds.

Let us examine two real-life examples of compression using global thresholding, for a given and unoptimized wavelet choice, to produce a nearly complete square norm recovery for a signal (see Figure 6-32 on page 6-112) and for an image (see Figure 6-33 on page 6-113).

```
% Load electrical signal and select a part.
load leleccum; indx = 2600:3100;
x = leleccum(indx);

% Perform wavelet decomposition of the signal.
n = 3; w = 'db3';
[c,l] = wavedec(x,n,w);

% Compress using a fixed threshold.
thr = 35;
keepapp = 1;
[xd,cxd,lxd,perf0,perfl2] =
                 wdencmp('gbl',c,l,w,n,thr,'h',keepapp);
```

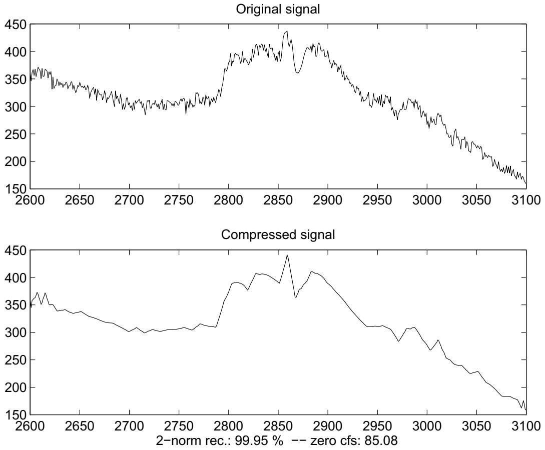


Figure 6-32: Signal Compression

The result is quite satisfactory, not only because of the norm recovery criterion, but also on a visual perception point of view. The reconstruction uses only 15% of the coefficients.

```
% Load original image.
load woman; x = X(100:200,100:200);
nbc = size(map,1);

% Wavelet decomposition of x.
n = 5; w = 'sym2'; [c,l] = wavedec2(x,n,w);

% Wavelet coefficients thresholding.
thr = 20;
keepapp =1;
[xd,cxd,lxd,perf0,perfl2] = 
                 wdencmp('gbl',c,l,w,n,thr,'h',keepapp);
```

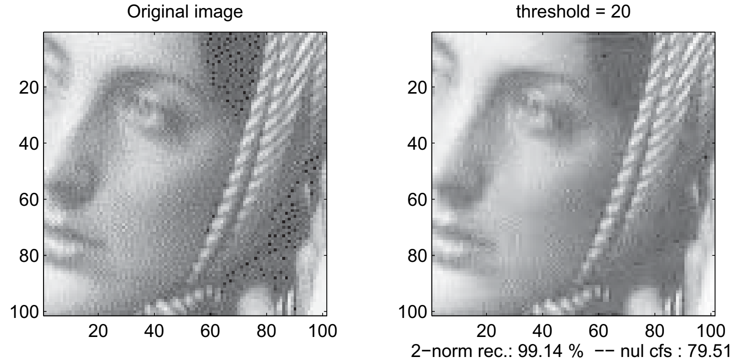


Figure 6-33: Image Compression

If the wavelet representation is too dense, similar strategies can be used in the wavelet packet framework to obtain a sparser representation. You can then determine the best decomposition with respect to a suitably selected entropy-like criterion, which corresponds to the selected purpose (de-noising or compression).

Compression Scores

When compressing using orthogonal wavelets, the *Retained energy* in percentage is defined by

$$\frac{100*(\text{vector-norm(coeffs of the current decomposition,2)})^2}{(\text{vector-norm(original signal,2)})^2}$$

When compressing using biorthogonal wavelets, the previous definition is not convenient. We use instead the *Energy ratio* in percentage defined by

$$\frac{100*(\text{vector-norm(compressed signal,2)})^2}{(\text{vector-norm(original signal,2)})^2}$$

and as a tuning parameter the *Norm cfs recovery* defined by

$$\frac{100\text{*(vector-norm(coeffs of the current decomposition,2))}^2}{\text{(vector-norm(coeffs of the original decomposition,2))}^2}$$

The *Number of zeros* in percentage is defined by

$$\frac{100\text{*(number of zeros of the current decomposition)}}{\text{(number of coefficients)}}$$

Function Estimation: Density and Regression

In this section we present two problems of functional estimation:

- Density estimation
- Regression estimation

Note According to the classical statistical notations, in this section, $\hat{g}$ denotes the estimator of the function g instead of the Fourier transform of g.

Density Estimation

The data are values $(X(i), 1 \le i \le n)$ sampled from a distribution whose density is unknown. We are looking for an estimate of this density.

What Is Density.

The well known histogram creates the information on the density distribution of a set of measures. At the very beginning of the 19th century, Laplace, a French scientist, repeating sets of observations of the same quantity, was able to fit a simple function to the density distribution of the measures. This function is called now the Laplace-Gauss distribution.

Density Applications.

Density estimation is a core part of reliability studies. It permits the evaluation of the life-time probability distribution of a TV set produced by a factory, the computation of the instantaneous availability, and of such other useful characteristics as the mean time to failure. A very similar situation occurs in survival analysis, when studying the residual lifetime of a medical treatment.

Density Estimators.

As in the regression context, the wavelets are useful in a nonparametric context, when very little information is available concerning the shape of the unknown density, or when you don't want to tell the statistical estimator what you know about the shape.

Several alternative competitors exist. The orthogonal basis estimators are based on the same ideas as the wavelets. Other estimators rely on statistical window techniques such as kernel smoothing methods.

We have theorems proving that the wavelet-based estimators behave at least as well as the others, and sometimes better. When the density *h(x)* has irregularities, such as a breakdown point or a breakdown point of the derivative *h'(x)*, the wavelet estimator is a good solution.

How to Perform Wavelet-Based Density Estimation.

The key idea is to reduce the density estimation problem to a fixed-design regression model. More precisely the main steps are as follows:

1. Transform the sample X into (Xb, Yb) data where the Xb are equally spaced, using a binning procedure. For each bin i, $Yb(i)$ = number of $X(j)$ within bin i.
2. Perform a wavelet decomposition of *Yb* viewed as a signal, using fast algorithm. Thus, the underlying *Xb* data is 1, 2, ..., *nb* where *nb* is the number of bins.
3. Threshold the wavelet coefficients according to one of the methods described for de-noising (see "De-Noising" on page 6-97).
4. Reconstruct an estimate *h1* of the density function *h* from the thresholded wavelet coefficients using fast algorithm (see "Fast Wavelet Transform (FWT) Algorithm" on page 6-19).
5. Postprocess the resulting function *h1*. Rescale the resulting function transforming 1, 2, ..., *nb* into *Xb* and interpolate *h1* for each bin to calculate $hest(X)$.

Steps **2** to **4** are standard wavelet-based steps. But the first step of this estimation scheme depends on *nb* (the number of bins), which can be viewed as a bandwidth parameter. In density estimation, *nb* is generally small with respect to the number of observations (equal to the length of *X*), since the binning step is a presmoother. A typical default value is *nb* = length(*X*) / 4.

For more information, you can refer for example to [AntP98], [HarKPT98], and [Ogd97] in “References” on page 6-155.

A More Technical Viewpoint.

Let us be a little more formal.

Let $X_1, X_2, \ldots, X_n$ be a sequence of independent and identically distributed random variables, with a common density function $h = h(x)$.

This density *h* is unknown and we want to estimate it. We have very little information on *h*.

For technical reasons we suppose that $\int h(x)^2 dx$ is finite. This allows us to express *h* in the wavelet basis.

We know that in the basis of functions ϕ and ψ with usual notations, *J* being an integer,

$$h = \sum_k a_{J,k}\phi_{J,k} + \sum_{j=-\infty}^{J}\sum_k d_{j,k}\psi_{j,k} = A_J + \sum_{j=-\infty}^{J} D_j$$

The estimator $\hat{h} = \hat{h}(x)$ will use some wavelet coefficients. The rationale for the estimator is the following.

To estimate *h*, it is sufficient to estimate the coordinates $a_{J,k}$ and the $d_{j,k}$.

We shall do it now.

We know the definition of the coefficients:

$$a_{J,k} = \int\phi_{J,k}(x)\ h(x)dx$$

and similarly

$$d_{j,k} = \int\psi_{j,k}(x)h(x)dx$$

The expression of the $a_{J,k}$ has a very funny interpretation. Because *h* is a density $\int\phi_{J,k}(x)h(x)dx$ is $E(\phi_{J,k}(X_i))$, the mean value of the random variable $\phi_{J,k}(X_i)$.

Usually such an expectation is estimated very simply by the mean value:

$$\hat{a}_{J,k} = \frac{1}{n}\sum_{i=1}^{n}\phi_{J,k}(X_i)$$

Of course the same kind of formula holds true for the $d_{j,k}$:

$$\hat{d}_{j,k} = \frac{1}{n}\sum_{i=1}^{n}\psi_{j,k}(X_i)$$

With a finite set of *n* observations, it is possible to estimate only a finite set of coefficients, those belonging to the levels from J-j_0 up to *J*, and to some positions *k*.

Besides, several values of the $d_{j,k}$ are not significant and are to be set to 0.

The values $d_{j,k}$, lower than a threshold *t*, are set to 0 in a very similar manner as the de-noising process and for almost the same reasons.

Inserting these expressions into the definition of *h*, we get an estimator:

$$\hat{h} = \sum_k \hat{a}_{J,k}\phi_{J,k} + \sum_{j=J-j_0}^{J}\sum_k \hat{d}_{j,k}1_{\{|\hat{d}_{j,k}|>t\}}\psi_{j,k}$$

This kind of estimator avoids the oscillations that would occur if all the detail coefficients would have been kept.

From the computational viewpoint, it is difficult to use a quick algorithm because the X_i values are not equally spaced.

Note that this problem can be overcome.

Let’s introduce the normalized histogram $\hat{H}$ of the values of *X*, having *nb* classes, where the centers of the bins are collected in a vector *Xb*, the frequencies of *Xi* within the bins are collected in a vector *Yb* and then

$\hat{H}(x) = \frac{Yb(r)}{n}$ on the r-th bin

We can write, using $\hat{H}$,

$$\hat{d}_{j,k} = \frac{1}{n}\sum_{i=1}^{n}\psi_{j,k}(X_i) \approx \frac{1}{n}\sum_{r=1}^{nb} Yb(r)\psi_{j,k}(Xb(r)) \approx c\int\psi_{j,k}(x)\hat{H}(x)dx$$

where $\frac{1}{c}$ is the length of each bin.

The signs $\approx$ occur because we lose some information when using histogram instead of the values X_i and when approximating the integral.

The last $\approx$ sign is very interesting. It means that $\hat{d}_{j,k}$ is, up to the constant *c*, the wavelet coefficient of the function $\hat{H}$ associated with the level *j* and the position *k*. The same result holds true for the $\hat{a}_{J,k}$.

So, the last $\approx$ sign of the previous equation shows that the coefficients $\hat{d}_{j,k}$ appear also to be (up to an approximation) wavelet coefficients — those of the decomposition of the sequence $\hat{H}$. If some of the coefficients at level *J* are known or computed, the Mallat algorithm computes the others quickly and simply.

And now we are able to finish computing $\hat{h}$ when the $\hat{d}_{j,k}$ and the $\hat{a}_{J,k}$ have been computed.

The trick is the transformation of irregularly spaced *X* values into equally spaced values by a process similar to the histogram computation, and that is called *binning*.

You can see the different steps of the procedure using the Density Estimation Graphical User Interface, by typing

```
wavemenu
```

and clicking the **Density Estimation 1-D** option.

Regression Estimation

What Is Regression?

The regression problem belongs to the family of the most common practical questions. The goal is to get a model of the relationship between one variable *Y* and one or more variables *X*. The model gives the part of the variability of *Y* taken in account or explained by the variation of *X*. A function *f* represents the central part of the knowledge. The remaining part is dedicated to the residuals, which are similar to a noise. The model is $Y = f(X) + e$.

Regression Models.

The simplest case is the linear regression $Y = aX+b+e$ where the function *f* is affine. A case a little more complicated occurs when the function belongs to a family of parametrized functions as $f(X) = cos(wX)$, the value of *w* being unknown. Statistics Toolbox™ software provides tools for the study of such models. When *f* is totally unknown, the problem of the nonlinear regression is said to be a nonparametric problem and can be solved either by using usual statistical window techniques or by wavelet based methods.

Regression Applications.

These regression questions occur in many domains. For example:

- Metallurgy, where you can try to explain the tensile strength by the carbon content
- Marketing, where the house price evolution is connected to an economical index
- Air-pollution studies, where you can explain the daily maximum of the ozone concentration by the daily maximum of the temperature

Two designs are distinguished: the fixed design and the stochastic design. The difference concerns the status of *X*.

Fixed-Design Regression.

When the *X* values are chosen by the designer using a predefined scheme, as the days of the week, the age of the product, or the degree of humidity, the design is a fixed design. Usually in this case, the resulting *X* values are equally spaced. When *X* represents time, the regression problem can be viewed as a de-noising problem.

Stochastic Design Regression.

When the *X* values result from a measurement process or are randomly chosen, the design is stochastic. The values are often not regularly spaced. This framework is more general since it includes the analysis of the relationship between a variable Y and a general variable X, as well as the analysis of the evolution of Y as a function of time X when X is randomized.

How to Perform Wavelet-Based Regression Estimation.

The key idea is to reduce a general problem of regression to a fixed-design regression model. More precisely the main steps are as follows:

1 Transform (X,Y) data into (Xb,Yb) data where the Xb are equally spaced, using a binning procedure. For each bin i,

$$,Yb(i)) = \frac{sum\{Y(j) \text{ such that } X(j) \text{ lies in bin } i\}}{number\{Y(j) \text{ such that } X(j) \text{ lies in bin } i\}}$$

with the convention $\frac{0}{0} = 0$.

2 Perform a wavelet decomposition of Yb viewed as a signal using fast algorithm. This last sentence means that the underlying Xb data is 1, 2, ..., nb where nb is the number of bins.

3 Threshold the wavelet coefficients according to one of the methods described for de-noising.

4 Reconstruct an estimate *f1* of the function *f* from the thresholded wavelet coefficients using fast algorithm.

5 Post-process the resulting function *f1*. Rescale the resulting function *f1* transforming 1, 2, ..., nb onto Xb and interpolate *f1* for each bin in order to calculate $fest(x)$.

Steps **2** to **4** are standard wavelet-based steps. But the first step of this estimation scheme depends on the number of bins, which can be viewed as a bandwidth parameter. Generally, the value of `nb` is not chosen too small with respect to the number of observations, since the binning step is a presmoother.

For more information, you can refer for example to [AntP98], [HarKPT98], and [Ogd97]. See “References” on page 6-155.

A More Technical Viewpoint.

The regression problem goes along the same lines as the density estimation. The main differences, of course, concern the model.

There is another difference with the density step: we have here two variables X and Y instead of one in the density scheme.

The regression model is $Y_i = f(X_i) + \varepsilon_i$ where $(\varepsilon_i)_{1 \le i \le n}$ is a sequence of independent and identically distributed (i.i.d.) random variables and where the (X_i) are randomly generated according to an unknown density h.

Also, let us assume that $(X_1, Y_1), \ldots, (X_n, Y_n)$ is a sequence of i.i.d. random variables.

The function f is unknown and we look for an estimator $\hat{f}$.

We introduce the function $g = f \cdot h$. So $f = \frac{g}{h}$ with the convention $\frac{0}{0} = 0$.

We could estimate g by a certain $\hat{g}$ and, from the density part, an $\hat{h}$, and then use $\hat{f} = \frac{\hat{g}}{\hat{h}}$. We choose to use the estimate of h given by the histogram suitably normalized.

Let us bin the X-values into nb bins. The l-th bin-center is called $Xb(l)$, the number of X-values belonging to this bin is $n(l)$. Then, we define $Yb(l)$ by the sum of the Y-values within the bin divided by $n(l)$.

Let’s turn to the f estimator. We shall apply the technique used for the density function. The coefficients of f, are estimated by

$$\hat{d}_{j,k} = \frac{1}{n}\sum_{i=1}^{n} Y_i \psi_{j,k}(X_i)$$

$$\hat{a}_{J,k} = \frac{1}{n}\sum_{i=1}^{n} Y_i \phi_{J,k}(X_i)$$

We get approximations of the coefficients by the following formula that can be written in a form proving that the approximated coefficients are also the wavelet decomposition coefficients of the sequence Yb:

$$\hat{d}_{j,k} \approx \frac{1}{n}\sum_{l=1}^{nb} Yb(l)\psi_{j,k}(Xb(l))$$

$$\hat{a}_{J,k} \approx \frac{1}{n}\sum_{l=1}^{nb} Yb(l)\phi_{J,k}(Xb(l))$$

The usual simple algorithms can be used.

You can see the different steps of the procedure using the Regression Estimation Graphical User Interface by typing `wavemenu`, and clicking the **Regression Estimation 1-D** option.

Available Methods for De-Noising, Estimation, and Compression Using GUI Tools

This section presents the predefined strategies available using the de-noising, estimation, and compression GUI tools.

One-Dimensional DWT and SWT De-Noising

Level-dependent or interval-dependent thresholding methods are available. Predefined thresholding strategies:

- Hard or soft (default) thresholding
- Scaled white noise, unscaled white noise (default) or nonwhite noise
- Thresholds values are
  - Donoho-Johnstone methods: Fixed-form (default), Heursure, Rigsure, Minimax
  - Birgé-Massart method: Penalized high, Penalized medium, Penalized low

    The last three choices include a sparsity parameter a $(a > 1)$.

    Using this strategy the defaults are $a = 6.25$, 2, and 1.5, respectively, and the thresholding mode is hard. Only scaled and unscaled white noise options are supported.

One-Dimensional DWT Compression

1. Level-dependent or interval-dependent hard thresholding methods are available. Predefined thresholding strategies are:
   - Birgé-Massart method: Scarce high (default), Scarce medium, Scarce low

     This method includes a sparsity parameter a $(1 < a < 5)$. Using this strategy the default is $a = 1.5$.
   - Empirical methods

     - Equal balance sparsity-norm

     - Remove near 0
2. Global hard thresholding methods with GUI-driven choice are available. Predefined thresholding strategies are:
   - Empirical methods

     - Balance sparsity-norm (default = equal)

     - Remove near 0

Two-Dimensional DWT and SWT De-Noising

Level-dependent and orientation-dependent (horizontal, vertical, and diagonal) thresholding methods are available. Predefined thresholding strategies are:

- Hard or soft (default) thresholding
- Scaled white noise, unscaled white noise (default) or nonwhite noise
- Thresholds values are:
  - Donoho-Johnstone method: Fixed form (default)
  - Birgé-Massart method: Penalized high, Penalized medium, Penalized low

    The last three choices include a sparsity parameter a ($a > 1$). See “One-Dimensional DWT and SWT De-Noising” on page 6-122.
  - Empirical method: Balance sparsity-norm, default = sqrt

Two-Dimensional DWT Compression

Level-dependent and orientation-dependent (horizontal, vertical, and diagonal) thresholding methods are available.

1. Level-dependent or interval-dependent hard thresholding methods are available. Predefined thresholding strategies are:
   - Birgé-Massart method: Scarce high (default); Scarce medium, Scarce low

     This method includes a sparsity parameter a ($1 < a < 5$), the default is $a = 1.5$.

Index