WAVELET-BASED SIGNAL PROCESSING
AND APPLICATIONS*

C. Sidney Burrus
Haitao Guo

This work is produced by OpenStax-CNX and licensed under the
Creative Commons Attribution License 4.0

This chapter gives a brief discussion of several areas of application. It is intended to show what areas and what tools are being developed and to give some references to books, articles, and conference papers where the topics can be further pursued. In other words, it is a sort of annotated bibliography that does not pretend to be complete. Indeed, it is impossible to be complete or up-to-date in such a rapidly developing new area and in an introductory book.

In this chapter, we briefly consider the application of wavelet systems from two perspectives. First, we look at wavelets as a tool for denoising and compressing a wide variety of signals. Second, we very briefly list several problems where the application of these tools shows promise or has already achieved significant success. References will be given to guide the reader to the details of these applications, which are beyond the scope of this book.

1 Wavelet-Based Signal Processing

To accomplish frequency domain signal processing, one can take the Fourier transform (or Fourier series or DFT) of a signal, multiply some of the Fourier coefficients by zero (or some other constant), then take the inverse Fourier transform. It is possible to completely remove certain components of a signal while leaving others completely unchanged. The same can be done by using wavelet transforms to achieve wavelet-based, wavelet domain signal processing, or filtering. Indeed, it is sometimes possible to remove or separate parts of a signal that overlap in both time and frequency using wavelets, something impossible to do with conventional Fourier-based techniques.

*Version 1.5: Jul 25, 2015 8:14 pm -0500
†http://creativecommons.org/licenses/by/4.0/
The classical paradigm for transform-based signal processing is illustrated in Figure 1 where the center "box" could be either a linear or nonlinear operation. The "dynamics" of the processing are all contained in the transform and inverse transform operation, which are linear. The transform-domain processing operation has no dynamics; it is an algebraic operation. By dynamics, we mean that a process depends on the present and past, and by algebraic, we mean it depends only on the present. An FIR (finite impulse response) filter such as is part of a filter bank is dynamic. Each output depends on the current and a finite number of past inputs (see ). The process of operating point-wise on the DWT of a signal is static or algebraic. It does not depend on the past (or future) values, only the present. This structure, which separates the linear, dynamic parts from the nonlinear static parts of the processing, allows practical and theoretical results that are impossible or very difficult using a completely general nonlinear dynamic system.

Linear wavelet-based signal processing consists of the processor block in Figure 1 multiplying the DWT of the signal by some set of constants (perhaps by zero). If undesired signals or noise can be separated from the desired signal in the wavelet transform domain, they can be removed by multiplying their coefficients by zero. This allows a more powerful and flexible processing or filtering than can be achieved using Fourier transforms. The result of this total process is a linear, time-varying processing that is far more versatile than linear, time-invariant processing. The next section gives an example of using the concentrating properties of the DWT to allow a faster calculation of the FFT.

2 Approximate FFT using the Discrete Wavelet Transform

In this section, we give an example of wavelet domain signal processing. Rather than computing the DFT from the time domain signal using the FFT algorithm, we will first transform the signal into the wavelet domain, then calculate the FFT, and finally go back to the signal domain which is now the Fourier domain.

Most methods of approximately calculating the discrete Fourier transform (DFT) involve calculating only a few output points (pruning), using a small number of bits to represent the various calculations, or approximating the kernel, perhaps by using cordic methods. Here we use the characteristics of the signal being transformed to reduce the amount of arithmetic. Since the wavelet transform concentrates the energy of many classes of signals onto a small number of wavelet coefficients, this can be used to improve the efficiency of the DFT [52], [56], [51], [57] and convolution [53].

2.1 Introduction

The DFT is probably the most important computational tool in signal processing. Because of the characteristics of the basis functions, the DFT has enormous capacity for the improvement of its arithmetic efficiency [21]. The classical Cooley-Tukey fast Fourier transform (FFT) algorithm has the complexity of $O(N \log_2 N)$. Thus the Fourier transform and its fast algorithm, the FFT, are widely used in many areas, including signal processing and numerical analysis. Any scheme to speed up the FFT would be very desirable.

http://cnx.org/content/m45101/1.5/
Although the FFT has been studied extensively, there are still some desired properties that are not provided by the classical FFT. Here are some of the disadvantages of the FFT algorithm:

1. Pruning is not easy. When the number of input points or output points are small compared to the length of the DWT, a special technique called pruning\[^{100}\] is often used. However, this often requires that the nonzero input data are grouped together. Classical FFT pruning algorithms do not work well when the few nonzero inputs are randomly located. In other words, a sparse signal may not necessarily give rise to faster algorithm.

2. No speed versus accuracy tradeoff. It is common to have a situation where some error would be allowed if there could be a significant increase in speed. However, this is not easy with the classical FFT algorithm. One of the main reasons is that the twiddle factors in the butterfly operations are unit magnitude complex numbers. So all parts of the FFT structure are of equal importance. It is hard to decide which part of the FFT structure to omit when error is allowed and the speed is crucial. In other words, the FFT is a single speed and single accuracy algorithm.

3. No built-in noise reduction capacity. Many real world signals are noisy. What people are really interested in are the DFT of the signals without the noise. The classical FFT algorithm does not have built-in noise reduction capacity. Even if other denoising algorithms are used, the FFT requires the same computational complexity on the denoised signal. Due to the above mentioned shortcomings, the fact that the signal has been denoised cannot be easily used to speed up the FFT.

2.2 Review of the Discrete Fourier Transform and FFT

The discrete Fourier transform (DFT) is defined for a length-N complex data sequence by

\[
X (k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N}, \quad k = 0, ..., N - 1
\]

where we use \( j = \sqrt{-1} \). There are several ways to derive the different fast Fourier transform (FFT) algorithms. It can be done by using index mapping \[^{21}\], by matrix factorization, or by polynomial factorization. In this chapter, we only discuss the matrix factorization approach, and only discuss the so-called radix-2 decimation in time (DIT) variant of the FFT.

Instead of repeating the derivation of the FFT algorithm, we show the block diagram and matrix factorization, in an effort to highlight the basic idea and gain some insight. The block diagram of the last stage of a length-8 radix-2 DIT FFT is shown in Figure 2. First, the input data are separated into even and odd groups. Then, each group goes through a length-4 DFT block. Finally, butterfly operations are used to combine the shorter DFTs into longer DFTs.

http://cnx.org/content/m45101/1.5/
The details of the butterfly operations are shown in Figure 3, where $W^i_N = e^{-j2\pi i/N}$ is called the twiddle factor. All the twiddle factors are of magnitude one on the unit circle. This is the main reason that there is no complexity versus accuracy tradeoff for the classical FFT. Suppose some of the twiddle factors had very small magnitude, then the corresponding branches of the butterfly operations could be dropped (pruned) to reduce complexity while minimizing the error to be introduced. Of course the error also depends on the value of the data to be multiplied with the twiddle factors. When the value of the data is unknown, the best way is to cutoff the branches with small twiddle factors.

The computational complexity of the FFT algorithm can be easily established. If we let $C_{FFT}(N)$ be the complexity for a length-N FFT, we can show

$$C_{FFT}(N) = O(N) + 2C_{FFT}(N/2), \quad (2)$$

where $O(N)$ denotes linear complexity. The solution to Equation (2) is well known:

$$C_{FFT}(N) = O(N\log_2 N). \quad (2)$$

This is a classical case where the divide and conquer approach results in very effective solution.
Figure 3: Butterfly Operations in a Radix-2 DIT FFT

The matrix point of view gives us additional insight. Let \( \mathbf{F}_N \) be the \( N \times N \) DFT matrix; i.e., \( \mathbf{F}_N (m, n) = e^{-j2\pi mn/N} \), where \( m, n \in \{0, 1, ..., N-1\} \). Let \( \mathbf{S}_N \) be the \( N \times N \) even-odd separation matrix; e.g.,

\[
\mathbf{S}_4 = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}.
\]

(3)

Clearly \( \mathbf{S}_N \mathbf{S}_N^T = \mathbf{I}_N \), where \( \mathbf{I}_N \) is the \( N \times N \) identity matrix. Then the DIT FFT is based on the following matrix factorization,

\[
\mathbf{F}_N = \mathbf{F}_N \mathbf{S}_N \mathbf{S}_N^T = \begin{bmatrix}
\mathbf{I}_{N/2} & \mathbf{T}_{N/2} \\
\mathbf{I}_{N/2} & -\mathbf{T}_{N/2}
\end{bmatrix}
\begin{bmatrix}
\mathbf{F}_{N/2} & 0 \\
0 & \mathbf{F}_{N/2}
\end{bmatrix} \mathbf{S}_N,
\]

(3)

where \( \mathbf{T}_{N/2} \) is a diagonal matrix with \( W_N^i, i \in \{0, 1, ..., N/2-1\} \) on the diagonal. We can visualize the above factorization as

\[
\begin{bmatrix}
\end{bmatrix}
\]

(3)

where we image the real part of DFT matrices, and the magnitude of the matrices for butterfly operations and even-odd separations. \( N \) is taken to be 128 here.

2.3 Review of the Discrete Wavelet Transform

In this section, we briefly review the fundamentals of the discrete wavelet transform and introduce the necessary notation for future sections. The details of the DWT have been covered in other chapters.
At the heart of the discrete wavelet transform are a pair of filters $h$ and $g$ — lowpass and highpass respectively. They have to satisfy a set of constraints Figure: Sinc Scaling Function and Wavelet [100], [101], [108]. The block diagram of the DWT is shown in Figure 4. The input data are first filtered by $h$ and $g$ then downsampled. The same building block is further iterated on the lowpass outputs.

![Building Block for the Discrete Wavelet Transform](image)

**Figure 4**: Building Block for the Discrete Wavelet Transform

The computational complexity of the DWT algorithm can also be easily established. Let $C_{DWT}(N)$ be the complexity for a length-$N$ DWT. Since after each scale, we only further operate on half of the output data, we can show

$$C_{DWT}(N) = O(N) + C_{DWT}(N/2),$$

which gives rise to the solution

$$C_{DWT}(N) = O(N).$$

The operation in Figure 4 can also be expressed in matrix form $W_N$; e.g., for Haar wavelet,

$$W_{Haar}^4 = \sqrt{2}/2 \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$  \tag{4}

The orthogonality conditions on $h$ and $g$ ensure $W_N^* W_N = I_N$. The matrix for multiscale DWT is formed by $W_N$ for different $N$; e.g., for three scale DWT,

$$\begin{bmatrix} W_{N/4} \\ I_{N/4} \\ I_{N/2} \end{bmatrix} \begin{bmatrix} W_{N/2} \\ I_{N/2} \end{bmatrix} W_N.$$  \tag{4}

We could further iterate the building block on some of the highpass outputs. This generalization is called the wavelet packets [27].

**2.4 The Algorithm Development**

The key to the fast Fourier transform is the factorization of $F_N$ into several sparse matrices, and one of the sparse matrices represents two DFTs of half the length. In a manner similar to the DIT FFT, the following
matrix factorization can be made:

\[
F_N = F_N W_N^T W_N = \begin{bmatrix}
A_{N/2} & B_{N/2} \\
C_{N/2} & D_{N/2}
\end{bmatrix} \begin{bmatrix}
F_{N/2} & 0 \\
0 & F_{N/2}
\end{bmatrix} W_N, \tag{4}
\]

where \(A_{N/2}, B_{N/2}, C_{N/2}, \) and \(D_{N/2}\) are all diagonal matrices. The values on the diagonal of \(A_{N/2}\) and \(C_{N/2}\) are the length-N DFT (i.e., frequency response) of \(h\), and the values on the diagonal of \(B_{N/2}\) and \(D_{N/2}\) are the length-N DFT of \(g\). We can visualize the above factorization as

where we image the real part of DFT matrices, and the magnitude of the matrices for butterfly operations and the one-scale DWT using length-16 Daubechies’ wavelets [31], [32]. Clearly we can see that the new twiddle factors have non-unit magnitudes.

\[\text{Figure 5: Last stage of a length-8 DWT based FFT.}\]

The above factorization suggests a DWT-based FFT algorithm. The block diagram of the last stage of a length-8 algorithm is shown in Figure 5. This scheme is iteratively applied to shorter length DFTs to get the full DWT based FFT algorithm. The final system is equivalent to a full binary tree wavelet packet transform [29] followed by classical FFT butterfly operations, where the new twiddle factors are the frequency response of the wavelet filters.
The detail of the butterfly operation is shown in Figure 6, where \( i \in \{0, 1, ..., N/2-1\} \). Now the twiddle factors are length-N DFT of \( h \) and \( g \). For well defined wavelet filters, they have well known properties; e.g., for Daubechies' family of wavelets, their frequency responses are monotone, and nearly half of which have magnitude close to zero. This fact can be exploited to achieve speed vs. accuracy tradeoff. The classical radix-2 DIT FFT is a special case of the above algorithm when \( h = [1, 0] \) and \( g = [0, 1] \). Although they do not satisfy some of the conditions required for wavelets, they do constitute a legitimate (and trivial) orthogonal filter bank and are often called the lazy wavelets in the context of lifting.

\[
\begin{align*}
A_{N/2}(i, i) & \\
C_{N/2}(i, i) & \\
B_{N/2}(i, i) & \\
D_{N/2}(i, i) & 
\end{align*}
\]

Figure 6: Butterfly Operations in a Radix-2 DIT FFT

2.5 Computational Complexity

For the DWT-based FFT algorithm, the computational complexity is on the same order of the FFT — \( O(N \log_2 N) \), since the recursive relation in (2) is again satisfied. However, the constant appearing before \( N \log_2 N \) depends on the wavelet filters used.

2.6 Fast Approximate Fourier Transform

The basic idea of the fast approximate Fourier transform (FAFT) is pruning; i.e., cutting off part of the diagram. Traditionally, when only part of the inputs are nonzero, or only part of the outputs are required, the part of the FFT diagram where either the inputs are zero or the outputs are undesired is pruned [100], so that the computational complexity is reduced. However, the classical pruning algorithm is quite restrictive, since for a majority of the applications, both the inputs and the outputs are of full length.

The structure of the DWT-based FFT algorithm can be exploited to generalize the classical pruning idea for arbitrary signals. From the input data side, the signals are made sparse by the wavelet transform [82], [79], [80], [32]; thus approximation can be made to speed up the algorithm by dropping the insignificant data. In other words, although the input signal are normally not sparse, DWT creates the sparse inputs for the butterfly stages of the FFT. So any scheme to prune the butterfly stages for the classical FFT can be used here. Of course, the price we have to pay here is the computational complexity of the DWT operations. In actual implementation, the wavelets in use have to be carefully chosen to balance the benefit of the pruning and the price of the transform. Clearly, the optimal choice depends on the class of the data we would encounter.

From the transform side, since the twiddle factors of the new algorithm have decreasing magnitudes, approximation can be made to speed up the algorithm by pruning the sections of the algorithm which correspond to the insignificant twiddle factors. The frequency response of the Daubechies' wavelets are shown in Figure 7. We can see that they are monotone decreasing. As the length increases, more and more
points are close to zero. It should be noted that those filters are not designed for frequency responses. They are designed for flatness at 0 and \( \pi \). Various methods can be used to design wavelets or orthogonal filter banks [85], [96], [108] to achieve better frequency responses. Again, there is a tradeoff between the good frequency response of the longer filters and the higher complexity required by the longer filters.

![Frequency response of Daubechies family of wavelets](image)

**Figure 7:** The Frequency Responses of Daubechies’ Family of Wavelets

### 2.7 Computational Complexity

The wavelet coefficients are mostly sparse, so the input of the shorter DFTs are sparse. If the implementation scales well with respect to the percentage of the significant input (e.g., it uses half of the time if only half of the inputs are significant), then we can further lower the complexity. Assume for \( N \) inputs, \( \alpha N \) of them are significant (\( \alpha \leq 1 \)), we have

\[
C_{F\text{AFT}}(N) = O(N) + 2\alpha C_{F\text{AFT}}(N/2). \tag{7}
\]

For example, if \( \alpha = \frac{1}{2} \), Equation (7) simplifies to

\[
C_{F\text{AFT}}(N) = O(N) + C_{F\text{AFT}}(N/2), \tag{7}
\]

which leads to

\[
C_{F\text{AFT}}(N) = O(N). \tag{7}
\]

So under the above conditions, we have a linear complexity approximate FFT. Of course, the complexity depends on the input data, the wavelets we use, the threshold value used to drop insignificant data, and
the threshold value used to prune the butterfly operations. It remains to find a good tradeoff. Also the implementation would be more complicated than the classical FFT.

### 2.8 Noise Reduction Capacity

It has been shown that the thresholding of wavelet coefficients has near optimal noise reduction property for many classes of signals [39]. The thresholding scheme used in the approximation in the proposed FAFT algorithm is exactly the hard thresholding scheme used to denoise the data. Soft thresholding can also be easily embedded in the FAFT. Thus the proposed algorithm also reduces the noise while doing approximation. If we need to compute the DFT of noisy signals, the proposed algorithm not only can reduce the numerical complexity but also can produce cleaner results.

### 2.9 Summary

In the past, the FFT has been used to calculate the DWT [109], [101], [108], which leads to an efficient algorithm when filters are infinite impulse response (IIR). In this chapter, we did just the opposite – using DWT to calculate FFT. We have shown that when no intermediate coefficients are dropped and no approximations are made, the proposed algorithm computes the exact result, and its computational complexity is on the same order of the FFT; i.e., \( O(N\log_2 N) \). The advantage of our algorithm is two fold. From the input data side, the signals are made sparse by the wavelet transform, thus approximation can be made to speed up the algorithm by *dropping* the insignificant data. From the transform side, since the twiddle factors of the new algorithm have decreasing magnitudes, approximation can be made to speed up the algorithm by *pruning* the section of the algorithm which corresponds to the insignificant twiddle factors. Since wavelets are an unconditional basis for many classes of signals [101], [80], [32], the algorithm is very efficient and has built-in denoising capacity. An alternative approach has been developed by Shenton, Mitra, Heute, and Hossen [99], [59] using subband filter banks.

### 3 Nonlinear Filtering or Denoising with the DWT

Wavelets became known to most engineers and scientists with the publication of Daubechies’ important paper [31] in 1988. Indeed, the work of Daubechies [32], Mallat [73], [74], [76], Meyer [81], [82], and others produced beautiful and interesting structures, but many engineers and applied scientist felt they had a "solution looking for a problem." With the recent work of Donoho and Johnstone together with ideas from Coifman, Beylkin and others, the field is moving into a second phase with a better understanding of why wavelets work. This new understanding combined with *nonlinear processing* not only solves currently important problems, but gives the potential of formulating and solving completely new problems. We now have a coherence of approach and a theoretical basis for the success of our methods that should be extraordinarily productive over the next several years. Some of the Donoho and Johnstone references are [41], [36], [39], [33], [40], [44], [43], [42], [22], [37], [38], [23] and related ones are [94], [84], [103], [102], [19]. Ideas from Coifman are in [29], [27], [28], [26], [25], [24], [13].

These methods are based on taking the discrete wavelet transform (DWT) of a signal, passing this transform through a threshold, which removes the coefficients below a certain value, then taking the inverse DWT, as illustrated in Figure 1. They are able to remove noise and achieve high compression ratios because of the "concentrating" ability of the wavelet transform. If a signal has its energy concentrated in a small number of wavelet dimensions, its coefficients will be relatively large compared to any other signal or noise that has its energy spread over a large number of coefficients. This means that thresholding or shrinking the wavelet transform will remove the low amplitude noise or undesired signal in the wavelet domain, and an inverse wavelet transform will then retrieve the desired signal with little loss of detail. In traditional Fourier-based signal processing, we arrange our signals such that the signals and any noise overlap as little as possible in the frequency domain and linear time-invariant filtering will approximately separate them. Where their Fourier spectra overlap, they cannot be separated. Using linear wavelet or other time-frequency
or time-scale methods, one can try to choose basis systems such that in that coordinate system, the signals overlap as little as possible, and separation is possible.

The new nonlinear method is entirely different. The spectra can overlap as much as they want. The idea is to have the amplitude, rather than the location of the spectra be as different as possible. This allows clipping, thresholding, and shrinking of the amplitude of the transform to separate signals or remove noise. It is the localizing or concentrating properties of the wavelet transform that makes it particularly effective when used with these nonlinear methods. Usually the same properties that make a system good for denoising or separation by nonlinear methods, makes it good for compression, which is also a nonlinear process.

3.1 Denoising by Thresholding

We develop the basic ideas of thresholding the wavelet transform using Donoho’s formulations [39], [41], [65]. Assume a finite length signal with additive noise of the form

\[ y_i = x_i + \epsilon n_i, \quad i = 1, ..., N \]  (7)

as a finite length signal of observations of the signal \( x \) that is corrupted by i.i.d. zero mean, white Gaussian noise \( n_i \) with standard deviation \( \epsilon \), i.e., \( n_i \sim N(0, 1) \). The goal is to recover the signal \( x \) from the noisy observations \( y \). Here and in the following, \( v \) denotes a vector with the ordered elements \( v_i \) if the index \( i \) is omitted. Let \( W \) be a left invertible wavelet transformation matrix of the discrete wavelet transform (DWT). Then Eq. (7) can be written in the transformation domain

\[ Y = X + N, \quad \text{or,} \quad Y_i = X_i + N_i, \]  (7)

where capital letters denote variables in the transform domain, i.e., \( Y = Wy \). Then the inverse transform matrix \( W^{-1} \) exists, and we have

\[ W^{-1}W = I. \]  (7)

The following presentation follows Donoho’s approach [41], [36], [39], [35], [65] that assumes an orthogonal wavelet transform with a square \( W \); i.e., \( W^{-1} = W^T \). We will use the same assumption throughout this section.

Let \( \hat{X} \) denote an estimate of \( X \), based on the observations \( Y \). We consider diagonal linear projections

\[ \Delta = \text{diag}(\delta_1, ..., \delta_N), \quad \delta_i \in \{0, 1\}, \quad i = 1, ..., N, \]  (7)

which give rise to the estimate

\[ \hat{x} = W^{-1} \hat{X} = W^{-1} \Delta Y = W^{-1} \Delta Wy. \]  (7)

The estimate \( \hat{X} \) is obtained by simply keeping or zeroing the individual wavelet coefficients. Since we are interested in the \( l_2 \) error we define the risk measure

\[ \mathcal{R} \left( \hat{X}, X \right) = \mathbb{E} \left[ \| \hat{x} - x \|_2 \right] = \mathbb{E} \left[ \| W^{-1} \left( \hat{X} - X \right) \|_2 \right] = \mathbb{E} \left[ \| \hat{x} - X \|_2 \right]. \]  (7)

Notice that the last equality in Eq. (7) is a consequence of the orthogonality of \( W \). The optimal coefficients in the diagonal projection scheme are \( \delta_i = 1_{X_i > \epsilon} \) i.e., only those values of \( Y \) where the corresponding elements of \( X \) are larger than \( \epsilon \) are kept, all others are set to zero. This leads to the ideal risk

\[ \mathcal{R}_{id} \left( \hat{X}, X \right) = \sum_{n=1}^{N} \min \left( X^2, \epsilon^2 \right). \]  (7)

\(^1\)It is interesting to note that allowing arbitrary \( \delta_i \in \mathbb{R} \) improves the ideal risk by at most a factor of 2[42]
The ideal risk cannot be attained in practice, since it requires knowledge of $X$, the wavelet transform of the unknown vector $x$. However, it does give us a lower limit for the $l_2$ error.

Donoho proposes the following scheme for denoising:

1. compute the DWT $Y = Wx$
2. perform thresholding in the wavelet domain, according to so-called hard thresholding
   
   $$\hat{X} = T_h(Y, t) = \begin{cases} Y, & |Y| \geq t \\ 0, & |Y| < t \end{cases}$$
   
   or according to so-called soft thresholding
   
   $$\hat{X} = T_S(Y, t) = \begin{cases} \text{sgn}(Y) (|Y| - t), & |Y| \geq t \\ 0, & |Y| < t \end{cases}$$

3. compute the inverse DWT $\hat{x} = W^{-1}\hat{X}$

This simple scheme has several interesting properties. It’s risk is within a logarithmic factor ($\log N$) of the ideal risk for both thresholding schemes and properly chosen thresholds $t(N, \epsilon)$. If one employs soft thresholding, then the estimate is with high probability at least as smooth as the original function. The proof of this proposition relies on the fact that wavelets are unconditional bases for a variety of smoothness classes and that soft thresholding guarantees (with high probability) that the shrinkage condition $|\hat{X}_i| < |X_i|$ holds. The shrinkage condition guarantees that $\hat{x}$ is in the same smoothness class as is $x$. Moreover, the soft threshold estimate is the optimal estimate that satisfies the shrinkage condition. The smoothness property guarantees an estimate free from spurious oscillations which may result from hard thresholding or Fourier methods. Also, it can be shown that it is not possible to come closer to the ideal risk than within a factor $\log N$. Not only does Donoho’s method have nice theoretical properties, but it also works very well in practice.

Some comments have to be made at this point. Similar to traditional approaches (e.g., low pass filtering), there is a trade-off between suppression of noise and oversmoothing of image details, although to a smaller extent. Also, hard thresholding yields better results in terms of the $l_2$ error. That is not surprising since the observation value $y_i$ itself is clearly a better estimate for the real value $x_i$ than a shrunk value in a zero mean noise scenario. However, the estimated function obtained from hard thresholding typically exhibits undesired, spurious oscillations and does not have the desired smoothness properties.

### 3.2 Shift-Invariant or Nondecimated Discrete Wavelet Transform

As is well known, the discrete wavelet transform is not shift invariant; i.e., there is no “simple” relationship between the wavelet coefficients of the original and the shifted signal\(^2\). In this section we will develop a shift-invariant DWT using ideas of a nondecimated filter bank or a redundant DWT [65], [66], [64]. Because this system is redundant, it is not a basis but will be a frame or tight frame (see Section: Overcomplete Representations, Frames, Redundant Transforms, and Adaptive Bases ). Let $X = Wx$ be the (orthogonal) DWT of $x$ and $S_R$ be a matrix performing a circular right shift by $R$ with $R \in \mathbb{Z}$. Then

$$X_s = Wx_s = W S_R x = W S_R W^{-1} X,$$

which establishes the connection between the wavelet transforms of two shifted versions of a signal, $x$ and $x_s$, by the orthogonal matrix $W S_R W^{-1}$. As an illustrative example, consider Figure 8.

---

\(^2\)Since we deal with finite length signals, we really mean circular shift.
The first and most obvious way of computing a shift invariant discrete wavelet transform (SIDWT) is simply computing the wavelet transform of all shifts. Usually the two band wavelet transform is computed as follows: 1) filter the input signal by a low-pass and a high-pass filter, respectively, 2) downsample each filter output, and 3) iterate the low-pass output. Because of the downsampling, the number of output values at each stage of the filter bank (corresponding to coarser and coarser scales of the DWT) is equal to the number of the input values. Precisely $N$ values have to be stored. The computational complexity is $O(N)$. Directly computing the wavelet transform of all shifts therefore requires the storage of $N^2$ elements and has computational complexity $O(N^2)$.

Beylkin [11], Shensa [98], and the Rice group\(^3\) independently realized that 1) there are only $N\log N$ different coefficient values among those corresponding to all shifts of the input signal and 2) those can be computed with computational complexity $N\log N$. This can be easily seen by considering one stage of the filter bank. Let

$$y = [y_0 \ y_1 \ y_2 \ ... \ y_N]^T = hx$$

(8)

where $y$ is the output of either the high-pass or the low-pass filter in the analysis filter bank, $x$ the input and the matrix $h$ describes the filtering operation. Downsampling of $y$ by a factor of two means keeping the even indexed elements and discarding the odd ones. Consider the case of an input signal shifted by one. Then the output signal is shifted by one as well, and sampling with the same operator as before corresponds to keeping the odd-indexed coefficients as opposed to the even ones. Thus, the set of data points to be further processed is completely different. However, for a shift of the input signal by two, the downsampled output signal differs from the output of the nonshifted input only by a shift of one. This is easily generalized for any odd and even shift and we see that the set of wavelet coefficients of the first stage of the filter bank for arbitrary shifts consists of only $2N$ different values. Considering the fact that only the low-pass component ($N$ values) is iterated, one recognizes that after $L$ stages exactly $LN$ values result. Using the same arguments as in the shift variant case, one can prove that the computational complexity is $O(N\log N)$. The derivation for the synthesis is analogous.

Mallat proposes a scheme for computing an approximation of the continuous wavelet transform [75] that turns out to be equivalent to the method described above. This has been realized and proved by Shensa

\(^3\)Those are the ones we are aware of.
Moreover, Shensa shows that Mallat’s algorithm exhibits the same structure as the so-called algorithm à trous. Interestingly, Mallat’s intention in [75] was not in particular to overcome the shift variance of the DWT but to get an approximation of the continuous wavelet transform. In the following, we shall refer to the algorithm for computing the SIDWT as the Beylkin algorithm since this is the one we have implemented. Alternative algorithms for computing a shift-invariant wavelet transform [70] are based on the scheme presented in [11]. They explicitly or implicitly try to find an optimal, signal-dependent shift of the input signal. Thus, the transform becomes shift-invariant and orthogonal but signal dependent and, therefore, nonlinear. We mention that the generalization of the Beylkin algorithm to the multidimensional case, to an M-band multiresolution analysis, and to wavelet packets is straightforward.

3.3 Combining the Shensa-Beylkin-Mallat-à trous Algorithms and Wavelet Denoising

It was Coifman who suggested that the application of Donoho’s method to several shifts of the observation combined with averaging yields a considerable improvement. This statement first lead us to the following algorithm: 1) apply Donoho’s method not only to “some” but to all circular shifts of the input signal 2) average the adjusted output signals. As has been shown in the previous section, the computation of all possible shifts can be effectively done using Beylkin’s algorithm. Thus, instead of using the algorithm just described, one simply applies thresholding to the SIDWT of the observation and computes the inverse transform.

Before going into details, we want to briefly discuss the differences between using the traditional orthogonal and the shift-invariant wavelet transform. Obviously, by using more than N wavelet coefficients, we introduce redundancy. Several authors stated that redundant wavelet transforms, or frames, add to the numerical robustness [32] in case of adding white noise in the transform domain; e.g., by quantization. This is, however, different from the scenario we are interested in, since 1) we have correlated noise due to the redundancy, and 2) we try to remove noise in the transform domain rather than considering the effect of adding some noise [65], [66].

3.4 Performance Analysis

The analysis of the ideal risk for the SIDWT is similar to that by Guo [50]. Define the sets A and B according to

\[
A = \{i \mid |X_i| \geq \epsilon \}
\]

\[
B = \{i \mid |X_i| < \epsilon \}
\]

and an ideal diagonal projection estimator, or oracle,

\[
\tilde{X} = \begin{cases} 
Y_i = X_i + N_i & i \in A \\
0 & i \in B.
\end{cases}
\]

The pointwise estimation error is then

\[
\hat{X}_i - X_i = \begin{cases} 
N_i & i \in A \\
-X_i & i \in B.
\end{cases}
\]

In the following, a vector or matrix indexed by A (or B) indicates that only those rows are kept that have indices out of A (or B). All others are set to zero. With these definitions and (7), the ideal risk for the

\[4\text{However, it should be noted that Mallat published his algorithm earlier.} \]

\[5\text{A similar remark can be found in [83], p. 53.} \]
SIDWT can be derived

\[
\mathcal{R}_{id} (\tilde{X}, X) = \mathbb{E} \left[ \| W^{-1} (\tilde{X} - X) \|_2^2 \right] = \mathbb{E} \left[ \| W^{-1} (N_A - X_B) \|_2^2 \right] = \mathbb{E} \left[ (N_A - X_B)^T W^{-1T} W^{-1} (N_A - X_B) \right] \tag{8}
\]

where \( \text{tr}(X) \) denotes the trace of \( X \). For the derivation we have used, the fact that \( N_A = \epsilon W_A n \) and consequently the \( N_A \) have zero mean. Notice that for orthogonal \( W \) the Eq. (8) immediately specializes to Eq. (7). Eq. (8) depends on the particular signal \( X_B \), the transform, \( W^{-1} \), and the noise level \( \epsilon \).

It can be shown that when using the SIDWT introduced above and the thresholding scheme proposed by Donoho (including his choice of the threshold) then there exists the same upper bound for the actual risk as for case of the orthogonal DWT. That is the ideal risk times a logarithmic (in \( N \)) factor. We give only an outline of the proof. Johnstone and Silverman state [62] that for colored noise an oracle chooses \( \delta_i = 1_{X_i \geq \epsilon_i} \), where \( \epsilon_i \) is the standard deviation of the \( i \)-th component. Since Donoho’s method applies uniform thresholding to all components, one has to show that the diagonal elements of \( C_{W^{-1}} \) (the variances of the components of \( N \)) are identical. This can be shown by considering the reconstruction scheme of the SIDWT. With these statements, the rest of the proof can be carried out in the same way as the one given by Donoho and Johnstone [41].

### 3.5 Examples of Denoising

The two examples illustrated in Figure 9 show how wavelet based denoising works. The first shows a chirp or doppler signal which has a changing frequency and amplitude. Noise is added to this chirp in (b) and the result of basic Donoho denoising is shown in (c) and of redundant DWT denoising in (d). First, notice how well the noise is removed and at almost no sacrifice in the signal. This would be impossible with traditional linear filters.

The second example is the Houston skyline where the improvement of the redundant DWT is more obvious.
Figure 9: Example of Noise Reduction using $\psi_{DS'}$
4 Statistical Estimation

This problem is very similar to the signal recovery problem; a signal has to be estimated from additive white Gaussian noise. By linearity, additive noise is additive in the transform domain where the problem becomes:

estimate $\hat{\theta}$ from $y = \theta + \epsilon z$, where $z$ is a noise vector (with each component being a zero mean variance one Gaussian random variable) and $\epsilon > 0$ is a scalar noise level. The performance measured by the mean squared error (by Parseval) is given by

$$R_c(\hat{\theta}, \theta) = E\| \hat{\theta}(y) - \theta \|_2^2.$$  (9)

It depends on the signal ($\theta$), the estimator $\hat{\theta}$, the noise level $\epsilon$, and the basis.

For a fixed $\epsilon$, the optimal minmax procedure is the one that minimizes the error for the worst possible signal from the coefficient body $\Theta$.

$$R^*_c(\Theta) = \inf_{\hat{\theta}} \sup_{\theta \in \Theta} R_c(\hat{\theta}, \theta).$$  (9)

Consider the particular nonlinear procedure $\hat{\theta}$ that corresponds to soft-thresholding of every noisy coefficient $y_i$:

$$T_\epsilon(x_i) = \text{sgn}(y_i)(|y_i| - \epsilon)_+.$$  (9)

Let $r_c(\theta)$ be the corresponding error for signal $\theta$ and let $r^*_c(U(\Theta))$ be the worst-case error for the coefficient body $\Theta$.

If the coefficient body is solid, orthosymmetric in a particular basis, then asymptotically ($\epsilon \to 0$) the error decays at least as fast in this basis as in any other basis. That is $r_c(\Theta)$ approaches zero at least as fast as $r_c(U\Theta)$ for any orthogonal matrix $U$. Therefore, unconditional bases are nearly optimal asymptotically. Moreover, for small $\epsilon$ we can relate this procedure to any other procedure as follows [36]:

$$R^*(\epsilon, \Theta) \leq r^*_c(\epsilon, \Theta) \leq O((\log(1/\epsilon)) \cdot R^*_c(\epsilon, \Theta), \quad \epsilon \to 0.$$  (9)

5 Signal and Image Compression

5.1 Fundamentals of Data Compression

From basic information theory, we know the minimum average number of bits needed to represent realizations of a independent and identically distributed discrete random variable $X$ is its entropy $H(X)$[30]. If the distribution $p(X)$ is known, we can design Huffman codes or use the arithmetic coding method to achieve this minimum [9]. Otherwise we need to use adaptive method [112].

Continuous random variables require an infinite number of bits to represent, so quantization is always necessary for practical finite representation. However, quantization introduces error. Thus the goal is to achieve the best rate-distortion tradeoff [61], [30], [20]. Text compression [9], waveform coding [61] and subband coding [109] have been studied extensively over the years. Here we concentrate on wavelet compression, or more general, transform coding. Also we concentrate on low bitrate.
5.2 Prototype Transform Coder

The simple three-step structure of a prototype transform coder is shown in Figure 10. The first step is the transform of the signal. For a length-$N$ discrete signal $f(n)$, we expand it using a set of orthonormal basis functions as

$$f(n) = \sum_{i=1}^{N} c_i \psi_i(n),$$

where

$$c_i = \langle f(n), \psi_i(n) \rangle.$$  \hspace{1cm} (10)

We then use the uniform scalar quantizer $Q$ as in Figure 11, which is widely used for wavelet based image compression [97], [91],

$$\hat{c}_i = Q(c_i).$$  \hspace{1cm} (10)

Denote the quantization step size as $T$. Notice in the figure that the quantizer has a dead zone, so if $|c_i| < T$, then $Q(c_i) = 0$. We define an index set for those insignificant coefficients.
\( \mathcal{I} = \{ i : |c_i| < T \} \). Let \( M \) be the number of coefficients with magnitudes greater than \( T \) (significant coefficients). Thus the size of \( \mathcal{I} \) is \( N - M \). The squared error caused by the quantization is

\[
\sum_{i=1}^{N} (c_i - \hat{c}_i)^2 = \sum_{i \in \mathcal{I}} c_i^2 + \sum_{i \notin \mathcal{I}} (c_i - \hat{c}_i)^2.
\]

(11)

Since the transform is orthonormal, it is the same as the reconstruction error. Assume \( T \) is small enough, so that the significant coefficients are uniformly distributed within each quantization bins. Then the second term in the error expression is

\[
\sum_{i \notin \mathcal{I}} (c_i - \hat{c}_i)^2 = MT^2 / 12.
\]

(11)

For the first term, we need the following standard approximation theorem [34] that relates it to the \( l_p \) norm of the coefficients,

\[
\| f \|_{p} = \left( \sum_{i=1}^{N} |c_i|^p \right)^{1/p}.
\]

(11)
Theorem 56 Let $\lambda = \frac{1}{p} > \frac{1}{2}$ then

$$\sum_{i \in I} c_i^2 \leq \frac{\|f\|_p^2}{2\lambda - 1} M^{1-2\lambda}$$

This theorem can be generalized to infinite dimensional space if $\|f\|_p^2 < +\infty$. It has been shown that for functions in a Besov space, $\|f\|_p^2 < +\infty$ does not depend on the particular choice of the wavelet as long as each wavelet in the basis has $q > \lambda - \frac{1}{2}$ vanishing moments and is $q$ times continuously differentiable [81]. The Besov space includes piece-wise regular functions that may include discontinuities. This theorem indicates that the first term of the error expression decreases very fast when the number of significant coefficient increases.

The bit rate of the prototype compression algorithm can also be separated in two parts. For the first part, we need to indicate whether the coefficient is significant, also known as the significant map. For example, we could use 1 for significant, and 0 for insignificant. We need a total of $N$ these indicators. For the second part, we need to represent the values of the significant coefficients. We only need $M$ values. Because the distribution of the values and the indicators are not known in general, adaptive entropy coding is often used [112].

Energy concentration is one of the most important properties for low bitrate transform coding. Suppose for the sample quantization step size $T$, we have a second set of basis that generate less significant coefficients. The distribution of the significant map indicators is more skewed, thus require less bits to code. Also, we need to code less number of significant values, thus it may require less bits. In the mean time, a smaller $M$ reduces the second error term as in (11). Overall, it is very likely that the new basis improves the rate-distortion performance. Wavelets have better energy concentration property than the Fourier transform for signals with discontinuities. This is one of the main reasons that wavelet based compression methods usually out perform DCT based JPEG, especially at low bitrate.

5.3 Improved Wavelet Based Compression Algorithms

The above prototype algorithm works well [77], [51], but can be further improved for its various building blocks [55]. As we can see from Figure 12, the significant map still has considerable structure, which could be exploited. Modifications and improvements use the following ideas:

- Insignificant coefficients are often clustered together. Especially, they often cluster around the same location across several scales. Since the distance between nearby coefficients doubles for every scale, the insignificant coefficients often form a tree shape, as we can see from Figure: Discrete Wavelet Transform of the Houston Skyline, using $\psi_{D8}$ with a Gain of $\sqrt{2}$ for Each Higher Scale. These so called zero-trees can be exploited [97], [91] to achieve excellent results.
- The choice of basis is very important. Methods have been developed to adaptively choose the basis for the signal [86], [117]. Although they could be computationally very intensive, substantial improvement can be realized.
- Special run-length codes could be used to code significant map and values [105], [106].
• Advanced quantization methods could be used to replace the simple scalar quantizer [63].
• Method based on statistical analysis like classification, modeling, estimation, and prediction also produces impressive result [72].
• Instead of using one fixed quantization step size, we can successively refine the quantization by using smaller and smaller step sizes. These embedded schemes allow both the encoder and the decoder to stop at any bit rate [97], [91].
• The wavelet transform could be replaced by an integer-to-integer wavelet transform, no quantization is necessary, and the compression is lossless [91].

Other references are: [39], [41], [36], [94], [49], [97], [3], [4], [97], [110], [91], [92], [20], [51].

6 Why are Wavelets so Useful?

The basic wavelet in wavelet analysis can be chosen so that it is smooth, where smoothness is measured in a variety of ways [79]. To represent \( f(t) \) with \( K \) derivatives, one can choose a wavelet \( \psi(t) \) that is \( K \) (or more) times continuously differentiable; the penalty for imposing greater smoothness in this sense is that the supports of the basis functions, the filter lengths and hence the computational complexity all increase. Besides, smooth wavelet bases are also the “best bases” for representing signals with arbitrarily many singularities [36], a remarkable property.

The usefulness of wavelets in representing functions in these and several other classes stems from the fact that for most of these spaces the wavelet basis is an unconditional basis, which is a near-optimal property.

To complete this discussion, we have to motivate the property of an unconditional basis being asymptotically optimal for a particular problem, say data compression [36]. Figure 13 suggests why a basis in which the coefficients are solid and orthosymmetric may be desired. The signal class is defined to be the interior of the rectangle bounded by the lines \( x = \pm a \) and \( y = \pm b \). The signal corresponding to point \( A \) is the worst-case signal for the two bases shown in the figure; the residual error (with \( n = 1 \)) is given by \( a\sin(\theta) + b\cos(\theta) \) for \( \theta \in \{0, \alpha\} \) and is minimized by \( \theta = 0 \), showing that the orthosymmetric basis is preferred. This result is really a consequence of the fact that \( a \neq b \) (which is typically the case why one uses transform coding—if \( a = b \), it turns out that the “diagonal” basis with \( \theta = \frac{\pi}{4} \) is optimal for \( n = 1 \)). The closer the coefficient
body is to a solid, orthosymmetric body with varying side lengths, the less the individual coefficients are correlated with each other and the greater the compression in this basis.

In summary, the wavelet bases have a number of useful properties:

1. They can represent smooth functions.
2. They can represent singularities.
3. The basis functions are local. This makes most coefficient-based algorithms naturally adaptive to inhomogeneities in the function.
4. They have the unconditional basis (or near optimal in a minimax sense) property for a variety of function classes implying that if one knows very little about a signal, the wavelet basis is usually a reasonable choice.

![Figure 13: Optimal Basis for Data Compression](http://cnx.org/content/m45101/1.5/)
7.2 Seismic and Geophysical Signal Processing

One of the exciting applications areas of wavelet-based signal processing is in seismic and geophysical signal processing. Applications of denoising, compression, and detection are all important here, especially with higher-dimensional signals and images. Some of the references can be found in [89], [118], [95], [2], [94], [67], [45], [54], [48], [47], [6], [16].

7.3 Medical and Biomedical Signal and Image Processing

Another exciting application of wavelet-based signal processing is in medical and biomedical signal and image processing. Again, applications of denoising, compression, and detection are all important here, especially with higher dimensional signals and images. Some of the references can be found in [5], [107], [60].

7.4 Application in Communications

Some applications of wavelet methods to communications problems are in [103], [68], [71], [116], [90].

7.5 Fractals

Wavelet-based signal processing has been combined with fractals and to systems that are chaotic [1], [78], [115], [33], [7], [8], [113], [114]. The multiresolution formulation of the wavelet and the self-similar characteristic of certain fractals make the wavelet a natural tool for this analysis. An application to noise removal from music is in [10].

Other applications are to the automatic target recognition (ATR) problem, and many other questions.

8 Wavelet Software

There are several software packages available to study, experiment with, and apply wavelet signal analysis. There are several MATLAB programs at the end of this book. MathWorks, Inc. has a Wavelet Toolbox [83]; Donoho’s group at Stanford has WaveTool; the Yale group has XWPL and WPLab [111]; Taswell at Stanford has WavBox[104], a group in Spain has Uvi-Wave; MathSoft, Inc. has S+WAVELETS; Aware, Inc. has WaveTool; and the DSP group at Rice has a MATLAB wavelet toolbox available over the internet at http://www-dsp.rice.edu. There is a good description and list of several wavelet software packages in [18]. There are several MATLAB programs in Appendix C of this book. They were used to create the various examples and figures in this book and should be studied when studying the theory of a particular topic.

References


http://cnx.org/content/m45101/1.5/


http://cnx.org/content/m45101/1.5/


http://cnx.org/content/m45101/1.5/


